A Distributed Architecture for Reasoning about a Higher-Level Context

Waltenegus Dargie and Thomas Hamann

***Abstract*— This paper presents a distributed architecture for reasoning about a higher-level context as an abstraction of a dynamic real-world situation. Reasoning about a higher-level context entails dealing with data acquired from sensors, which can be inexact, incomplete, and/or uncertain. Inexact sensing arises mostly due to the inherent limitation of sensors to capture a real world phenomenon precisely. Incompleteness is caused by the absence of a mechanism to capture certain real-world aspects; and uncertainty stems from the lack of knowledge about the reliability of the sensing sources, such as their sensing range, accuracy, and resolution. The proposed architecture enables the modelling of an interactive computing environment with facts and beliefs to manipulate data from a variety of sensors with different sensing specifications. It will be shown how the architecture enables to reason about the whereabouts of a person in the absence of any location sensor. Depending on the available primitive contexts (temperature, relative humidity, sound pressure, light intensity and time) and the reliability of the sensors employed, the architecture enables to determine with different degree of uncertainty, whether a person is on a corridor or inside a room, a building, or outside a building.**

***Index Terms*—Context, Context-Aware Computing, Primitive Context, Higher-Level Context, Context Reasoning**

I. Introduction

The traditional model of human-computer interaction – in which a user enters a command and a computer executes the command [1] – puts a computing task at the centre of the user's attention. The user needs to provide all input information pertaining to the task a computer should accomplish, since the latter is entirely unaware of and quite unable to utilise background information surrounding the subject of the interaction. As long as the computing task is the only task which the user pays attention to, the degree of involvement the interaction demands of the user may not be a point of discussion. The problem comes when a user has to divide his attention between other impending activities such as driving, talking to other people, holding a presentation, attending to a child, and so on, while a computing task is in progress.

Manuscript received January 8, 2006.

W. Dargie and T. Hamann are with the Chair for Computer Networks at the Department of Computer Science, Dresden University of Technology, Hans-Grundig-Str. 25, 01062, Dresden, Germany. (Phone: +49-351-463-38352; fax: +49-351-46338251; e-mail: dargie@rn.inf.tu-dresden.de; hamann@rn.inf.tu-dresden.de).

This is even more pronounced when the user has to manage several devices at the same time: Whereas once it has been considered as a revolution to provide individuals with personal computers, it is now an everyday practice to carry with us several mobile computers and communicate with them on the spur of the moment.

The traditional computing model must therefore evolve to make, on the one hand, a computing task less obtrusive, and on the other hand, human-computer interaction more intuitive. In other words, the amount of explicit input a user has to supply for a computing task should be reduced, and interaction with computers should imitate human interactions. Otherwise, the increment in the ratio of mobile devices to a user causes additional distraction instead of helping a user in solving a problem.

II. Use of Context

Recently, context-aware computing has emerged promising the use of unused implicit information which can be vital for enabling mobile devices to establish a shared understanding [2]. The aim is to integrate computers into the environment, rather than taking them as distinct objects, so that they can seamlessly gather and employ surrounding information which enables them to provide useful services.

The need for establishing shared understanding has been clearly expressed by Weiser as follows:

> The idea […] first arose from contemplating the place of today's computer in actual activities of everyday life. In particular, anthropological studies of work life teach us that people primarily work in a world of shared situations and unexamined technological skills [3].

When people attend a meeting, for example, their eyes communicate to convey agreements or disagreements to what is said or not said; voices are whispered to exchange impromptu opinions; facial expressions reveal to the other participants fatigue, boredom, or disinterest. More importantly, speeches may not be grammatically correct or complete. Previous as well as unfolding incidents enable the audiences to capture what cannot be expressed verbally. Speakers shift from one language to another and use words with multiple meanings, and still the other participants can follow. Likewise, implicit situational information can be vital to enable mobile devices to be responsive to their

environment.

III. Understanding Context

In the example of attending a meeting, it has been assumed that the context encompassing the interactions between participants is effortlessly recognized by all participants. Consequently, with the knowledge of this context, many activities unfold, some of which are certainly unpremeditated activities but consistent with the context. Some of the activities express the freedom (or flexibility) associated with the recognition of the context – for example, using incomplete or incorrect statements, or using words with multiple meanings. Other activities reflect the participants' adjustment of behaviour in compliance with the context of interaction – for example, participants whispering to exchange impromptu ideas.

This assumption considers a context to be distinct from activities, and that it describes features of the environment wherein the activities take place. In reality, however, the distinction between a context and an activity is not always clear. This is because the scope and usefulness of a context are limited to a particular setting, particular instances of action, and particular parties to that action [4]. Moreover, among the set of activities enumerated earlier, some or all of them can be contexts to other activities which are affected by the occurrences of these activities. A meeting by itself is an activity triggered by the occurrence of a context which preceded it. Therefore, building a shared understanding requires the understanding of a context as a dynamic construct whose relevance is determined by episodes of use, social interaction, internal goals, and local influences [5].

IV. Context-Aware Features

Pascoe proposes three features required for a mobile device to establish a shared situation. These are: contextual sensing, contextual resource discovery, and context adaptation [6]. Contextual sensing refers to the ability of a system to augment the sensing capacity of a user by capturing a relevant context, so that when the context is detected, an associated action can be performed by the system; contextual resource discovery refers to the ability of a system to search and bind to resources which can be useful for increasing the interaction bandwidth with the user; and finally, context adaptation refers to the adjustment in behaviour of a system in accordance with the virtual as well as physical settings (as described by the captured context) wherein an interaction with the user is taking place.

To provide support for the design and implementation of such a system, Dey proposes a framework which has a set of features. These features are: context specification, separation of concern and context handling, context interpretation, transparent distributed communications, constant availability of context acquisition, context storage, and resource discovery [2].

Context specification refers to the process of specifying the system's behaviour and the context required to give rise to the observation of the behaviour. Here Dey identifies identity, location, activity, and time as basic context types. Separation of concern and context handling refers to the separation of context acquisition from context usage. Context interpretation refers to the provision of an appropriate abstraction to a piece of context so that it is meaningful to the system which uses (consumes) it. Transparent distributed communication refers to the possibility of acquiring a context in a transparent manner, possibly, from multiple sources which are not directly connected to the device on which a context-aware system is running. Constant availability of context sources is required to deliver every context which is specified by the developer at the time the system is designed. Context storage refers to the process of persisting contextual data for future reference. Finally, resource discovery, refers to the ability of a context-aware system to search and bind to context resources at runtime.

V. Revised Features of Context-Aware Computing

Some of the features of the framework of Dey are prohibitive in not allowing a system to capture dynamic real-world situations for the following reasons: (1) Specifying the behaviour of a context-aware system at design time prohibits a user from defining a user-specific behaviour which might not been foreseen by the developer; (2) determining at design time a context which causes certain behaviour to occur restricts a context-aware system from learning a new type of context which may equally cause the same behaviour to occur; (3) since a context-aware behaviour and the contexts associated with the behaviour can be unforeseen at design time, determining specific hardware and sensors for capturing a context may not be feasible; and (4) in a dynamic computing environment where the state of available resources and the resources themselves change over time, requiring a constant availability of sources for capturing specific context types can be unrealistic.

Subsequently, we propose a set of revised features, which we believe will improve the ability of mobile devices to establish in a more proactive manner a shared understanding of a pervasive computing environment.

A. Holistic and Conceptual Abstraction

Human-human interaction, as it is purported earlier, besides the expressiveness and the richness of the language employed, involves perception of a complex setting encompassing the subject of the interaction. Our brain presents the world to us not as a collection of raw data, but wholly, conceptually, and meaningfully [7]. Likewise, for a context-aware system to capture and abstract dynamic real-world situations, it is not merely enough to employ sensors and to make a piecemeal use of contextual elements. There is no doubt that employing sensors augments their sensing capacity thereby contributing to their awareness of the external world; awareness, however, is more than just sensing. Awareness means making sense of the sensed data. Hence, a context-aware system should be able

to perform various manipulations and comparisons on large amount of raw data acquired from sensors in order to transform them into a meaningful abstraction of the real world.

B. Managing Uncertainties

The taxonomy of neither Pascoe nor Dey considers the fact that context acquisition incorporates uncertainties. In reality, however, the sensory data from which a context is extracted can be incomplete, inexact, and, possibly, erroneous. Inexact sensing arises due, mostly, to the inherent limitation of sensors in precisely capturing a real world phenomenon; incompleteness arises due to the gap between what can be captured by employing sensors and what is needed to present the world to computers wholly, meaningfully, and conceptually. Erroneous sensing occurs due to physical and/or technical malfunction of the sensor or its communication link to the infrastructure. In other words, there are certain aspects which cannot be captured by employing sensors. Additionally, there is uncertainty due to the lack of knowledge about the reliability of the sensing sources, such as their sensing range, accuracy, and resolution. Therefore, a context-aware system should be able to deal with and to quantify the uncertainty associated with a context computing process.

C. Belief Revision

The model of a computing environment should be updated to reflect the perceived change in the environment. This is useful for appropriately manipulating sensory data with the help of a timely knowledge of both the static and the dynamic properties of the computing environment. In other worlds, the facts and beliefs it has of entities to manipulate primitive contexts should reflect the reality. This could be possible only if the system accommodates a belief revision mechanism. Another reason for belief revision (model updating) is the detection of contradictory information in the model [8]. In situations involving imprecise knowledge of entities and relationship between them, it is possible to arrive at a conclusion which might turn out to be incorrect as soon as more reliable evidence becomes available.

D. Dealing with the Dynamics of Context Inputs

In a pervasive computing environment, due to the mobility of a user in time and in space, the availability of mechanisms for capturing interesting contexts may change significantly [9]. Different mechanisms may appear and/or existing mechanisms may fully or partially become unavailable, making it difficult for a designer of a context-aware system to list a priori the set of contextual states that may exist.

Cohen et al. [10] define a nonprocedural programming language called *iQL* for context composition. The language assists application developers to identify and bind to heterogeneous context data sources dynamically. An *iQL* programmer expresses requirements for data sources instead of identifying specific sources; a runtime system discovers appropriate data sources, binds to them, and rebinds when properties of data sources change.

Similarly, Dey proposes Context Widgets to allow application developers to specify the context types they are interested in without the need to know how the contexts are acquired. In both approaches, the required contexts remain the same; only the context sources or the context detection mechanisms may change.

Often, a real world situation can have multiple features, and at any given time, some primitive contexts may be able to reveal only a subset of these features. Besides, there can be other primitive contexts which can provide an indirect evidence of the occurrence of a context of interest (for example, the heart rate of a patient can be determined from a blood pressure sensor if there is no ECG available to directly measure the heart rate [11]). Hence, a context-aware system should allow dynamic specification of alternative primitive contexts to achieve a set goal rather than rigidly obliging application developers to specify input contexts at design time.

E. Dynamic Definition of Contextual States

A given context type may have different states (values), and only a part of these states (values) may be interesting for a particular user at any given time. Analogues to the fact that an object can be viewed from different perspectives, a given context can be a subject of interest in a variety of ways. Whether or not it is possible to capture all the desirable states of a context type depends on the sensors employed. Consequently, a context-aware system should be able to dynamically determine the possible states of a higher-level context based on the available primitive contexts.

The difference between the forth and the fifth feature should be clear. Dealing with the dynamics of context inputs refers to the characteristic of a context-aware system in copping with the constantly appearing and vanishing primitive context sources – we call this an input property. On the other hand, dynamic definition of contextual states refers to the capacity of a context-aware system in defining the states (values) for a given higher-level context. The scope and usefulness of these states are limited to a particular setting, particular instances of action, and particular parties to that action Thus, it is an output property.

VI. Related Work

Peltenon et al. [12] classify auditory scenes into predefined classes by employing two classification mechanisms: 1-NN classifier and Mel-frequency cepstral coefficients with Gaussian mixture models. The aim is to recognise a physical environment by using audio information only. The audio scene comprises several everyday outside and inside environments, such as streets, restaurants, offices, homes, cars, etc. The features to be extracted from the audio signal for the purpose of classification are time and frequency domain features as well as linear prediction coefficients. The classification systems are able to classify 17 indoor and outdoor scenes with an accuracy of 68.4% and analysis

duration of 30 seconds.

The limitation of this approach is the absence of a framework or architecture to generalise recognition of different types of higher-level contexts. Furthermore, the primitive contexts employed are extracted from a single context type: audio signal; therefore, it is hard to conclude that the scheme employs a variety of primitive context.

Korpipää et al. [13] propose a higher-level context recognition framework. It uses a naïve Bayesian classifier, where the child nodes of the network are atomic contexts and the parent node is the higher-level context to be captured. The context atoms are selected on the basis of their ability to describe some properties of the higher-level context. Other criteria include feasibility of measuring or recognising the chosen context as accurately and unambiguously as possible. Among its most important tasks, the framework manages uncertainty of sensor readings through the use of probability based inference and fuzzy membership.

Using the framework, a mobile device could recognise whether it is outdoors or indoors; whether a sound is of a car, a tap water, rock music, classical music, an elevator, a speech, or other sound. For the first two higher-level contexts, namely indoors and outdoors, 14 atomic contexts describing the environment's light, humidity, and temperature are used, while 47 audio-based atomic context atoms are used to describe the remaining seven higher-level contexts.

The framework provides support to quantify the uncertainty associated with a recognition process; an additional merit of the framework includes training the model with data to recognise new contextual states. A conspicuous limitation of the framework is its employment of a naïve Bayesian classifier, which assumes the absence of causal dependencies between the input primitive contexts. In most practical cases, however, this assumption does not hold true. For example, the temperature, light intensity, and relative humidity of a place are causally dependent on time, which, too, is a primitive context; likewise, the relative humidity of a place is causally dependent on the temperature of a place. Subsequently, it may not be possible to accurately reason about a complex real-world situation by employing a naïve Bayesian classifier alone.

VII. Summary of Our Contribution

The approaches above attempt to present to computers a portion of the real world conceptually and meaningfully by recognising a higher-level context as abstraction of or generalised from particular instances of atomic contexts. Our architecture builds upon the experience learned from these approaches while filling the gaps which were left unfilled by them. Its typical features are revised as follows:

- Capability to capture a dynamic real-world situation;
- Capability to measure and reflect to applications the uncertainty associated with a context;
- Capability to update and correct the belief of the system with regards to the conditional dependencies between a higher-level context and the primitive contexts it abstracts;
- Capability to deal with the dynamics of input context primitives; and,
- Capability to dynamically define contextual states.

VIII. Architecture

This section introduces the architecture we propose. It consists of Primitive Context Servers (PCS), Aggregators, a Knowledge Base (KB), an Empirical Ambient Knowledge component (EAK), and a Composer. We offer a conceptual treatment in this section; in section IX, the implementation of the architecture will be briefly discussed. Figure 1 shows our architecture.

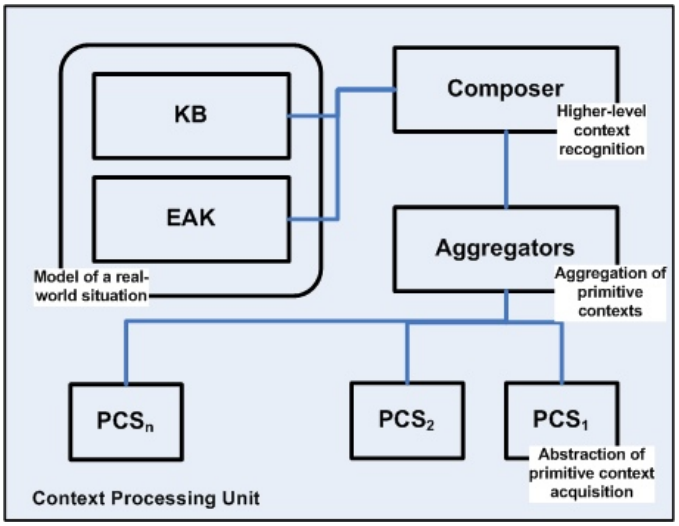

Figure 1: Architecture for higher-level context composition.

A. Primitive Context Server (PCS)

A PCS abstracts from other components, such as Aggregators, the details and complexities of extracting data from physical sensors. It transforms the raw sensory data into a meaningful context atom. In some cases, multiple features (context atoms) can be extracted from a single sensor – as an example an audio signal from which several audio features can be extracted; or as in the case of the DS1923 Dallas semiconductor humidity sensor from which both a relative humidity and a temperature value can be obtained. A primitive reveal a particular aspect of a real-world entity.

The primitive context provided by a PCS should be accompanied with a description of the sensing element, so that the components which employ the context can decide how to rate the context.

In addition to capturing and reporting atomic contexts, a PCS performs local administrative tasks such as tracking and monitoring the performance of the underlying sensing element. These activities are useful not only for ensuring smooth operations, but also for supporting quality checks.

One way of reducing uncertainty associated with imperfect sensing is setting objective metrics to evaluate the reliability of a sensing element. The metrics can be taken into account during data manipulations. Physical sensors are deployed with technical specifications, which include information about accuracy, sensing range, resolution, operation condition, and so on. This information is useful for rating the quality of data received from various Primitive Context Servers.

B. Aggregators

Often a piece of context is not sufficient to appropriately model a real-world situation. The real world is far too complex to be captured in complete detail by a single primitive context acquired from a single sensing element. Therefore, several aspects of entities should be included in the modelling. Moreover, the implication of inexact sensory data should also be taken into account.

Aggregation deals with the association, correlation, and combination of data from single or multiple sources to achieve a refined estimation [14]. The architecture supports two types of aggregators: homogeneous and heterogeneous aggregators. A homogeneous aggregator gathers and processes data from Primitive Context Sources delivering similar primitive contexts. This is necessary for dealing with inconsistency emanating from inexact sensing. Inconsistency may occur due, partly, to the fact that different sensors have different resolutions, sensing ranges, sensitivities, etc. As an example, a temperature aggregator aggregates temperature measurements from multiple temperature sensors monitoring the thermal condition of one and the same room.

A heterogeneous aggregator is similar to the aggregator proposed by Cohen et al. [10]. It searches and binds to various Primitive Context Servers each of which reports a different primitive context. The aim is to achieve completeness in modelling a real world situation.

C. Knowledge Base (KB)

A priori knowledge of entities (place, device, or person) is useful for appropriately manipulating primitive contexts. We distinguish between factual knowledge and knowledge based on beliefs. Facts reflect objective reality, while beliefs may not necessarily be true. In this subsection, we will discuss the usefulness of factual knowledge in modelling a real world situation.

Suppose we would like to model a lecture room. Descriptions such as size, volume and location are useful facts which do not change quickly. Moreover, the relation of the room to other physical places is additional useful knowledge. For example, a building subsumes a room; a room and a corridor are mutually exclusive places (in that one is not contained in the other). Given that a room and a corridor are complementary (mutually exclusive) concepts, it is possible to reason about a room, if all we have is only facts about a corridor. If we know that a person is inside a building that has only a room and a corridor, a necessary requirement to reason about his further whereabouts is knowledge of either a room or a corridor.

The KB thus comprises a collection of facts that constitute the vocabulary of an application domain and a list of assertions about individual named entities in terms of this vocabulary. The vocabulary consists of concepts, which denote sets of entities, and roles, which denote binary relationships between these entities. In addition to atomic concepts and roles, the KB allows the building of complex descriptions of concepts and roles.

D. Empirical Ambient Knowledge Component (EAK)

The KB accommodates facts only, however incomplete. There is little support to encode uncertain knowledge in it. Therefore, in our architecture we introduce the EAK in order to accommodate the inclusion of uncertain knowledge.

Empirical and heuristic knowledge of situations and people's perception of them is helpful to reason about dynamic situations. However, this type of knowledge is rather based on beliefs established on past experiences and observations that we do not represent it as an aggregation of facts, but as uncertain knowledge.

Suppose we want to model a lecture room by employing data from a variety of sensors describing different properties. The sensed data may refer to the temperature, relative humidity, light intensity, ambient sound pressure properties, air velocity, etc. To manipulate such data, am empirical ambient knowledge is required. For example: 'the temperature of a COLD ROOM in AUTUMN is between 15 and 20°C with a probability of 0.86'. Obviously, the empirical ambient knowledge as well as the sensed data incorporates uncertainties; hence cannot be taken as we would take facts.

Another example can be reasoning about people's thermal comfort inside a lecture room on the bases of the following empirical knowledge:

- Human thermal and humidity perception varies from season to season. A temperature ranging from 20 to 23.6° C is perceived as comfortable in winter, while in summer, the range from 22.8 to 26° C is perceived as comfortable. Likewise, a relative humidity ranging from 30 to 60% is perceived as comfortable in winter, while in summer, the range from 40 to 60% is perceived as comfortable [15].
- In winter a person is more sensitive to draught than in summer. Hence, the acceptable air velocity inside a room is below 0.15 m/s, while in summer it is in the order of 0.25 m/s [16].

The beliefs above may hold true for most places, but we may not be able to make sound conclusions by taking into account these statements alone. For these reasons, we represent empirical and ambient knowledge as beliefs, using conditional probabilities (or fuzzy membership functions, basic probability mass functions, etc).

E. Composer

Even when we have reliable sensory data, we may still not be able to capture a real world situation. This is because there is always a gap between what sensors can provide and what applications may need. Composition deals with a single higher-level context as an abstraction of more numerous but simple context atoms.

Once entities are modelled using the facts and beliefs stored in the KB and EAK, respectively, the composer accesses these components to retrieve useful knowledge which can serve as a reference to manipulate sensed data.

A necessary requirement for the design of a composer is that it should be robust to work in a highly dynamic and pervasive computing environment, where the state of available context sources as well as the available sources themselves may change quite significantly. One way to meet this challenge is to enrich the KB as well as the EAK with facts and beliefs such that if some facts or beliefs are missing, the composer should combine other facts and beliefs to arrive at a reliable conclusion.

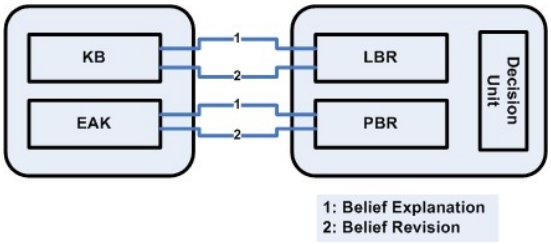

Figure 2: Belief explanation and revision

A composer has three components: the first component is logic based reasoning component (LBR), and manipulates the facts, relations, and assertions in the KB; the second component is a probabilistic reasoning (PR) component which should deal with imprecise, and possibly erroneous, data from sensors and in the EAK. The reasoning scheme may or may not be a probabilistic reasoning scheme, but for convenience of expression we continue to call such. Our own implementation is a probabilistic reasoning scheme based of Bayesian Networks.

The third component of the composer is a decision unit. When appropriate, it reconciles the outcomes of the LBR with and the PR to make consistent decisions. To demonstrate this, at times, the PR may not be able to decide because two or more propositions have equal posterior probabilities o the difference in posterior probabilities is not significant enough[1]. Consider the scenario of reasoning about the whereabouts of a person. Suppose the outcome of the PR indicates with equal probabilities that the person can be either on a CORRIDOR or inside a ROOM. We know from the KB that a CORRIDOR and a ROOM are mutually exclusive concepts, but a BUILDING subsumes both concepts. Hence, the decision unit will decide that a person is inside a BUILDING, instead of randomly selecting for either a COORIDOR or a ROOM.

Once a composer arrives at a conclusion regarding the situation of an entity and its relationship to other entities, it updates the KB and the EAK. The model of a computing environment should be updated to reflect the perceived change in the environment.

[1] This is a function of the tolerable error by the application or the user.

IX. Implementation

In this section we will illustrate the implementation of our architecture.

We installed various sensors and data loggers on multiple mobile devices belonging to a user. Among these sensors we hade: DS1971-F3 and Java powered DS1957B data loggers for storing secured profile information such as the user's name and password (used to monitor which user has logged on to a device) as well as address, billing information, and so forth; temperature loggers with different sensing parameters (DS1921G, DS1921Z-F5, and DS2422), a humidity and a temperature data logger (DS1923), and a light intensity data logger (PCE-172). The sensors have different accuracy, sensing range, and resolution.

At any give time, a user takes with him a random selection of the mobile devices. A Composer running on one of these devices interacts with Primitive Context Servers and Aggregators which are distributed across the mobile devices taken by the user to reason about various higher-level contexts.

A. Modelling Dynamic Situation

To setup the EAK, we model physical and virtual entities in terms of numerous primitive contexts. Since we cannot employ sensors to directly capture a higher-level context, the Composer manipulates various models, and obtains conceptual abstractions of a real world situation. Four steps are required to model a higher-level context: (1) the identification of a context of interest; (2) the identification of the various aspects of the context which can be captured by employing sensors; (3) the determination of the various states of each of the aspects; and (4) the establishment of relationships between the various aspects. Next, we will illustrate the modelling steps by identifying PLACE as a context of interest.

Step 1: We identified the following primitive contexts to model a PLACE: temperature, relative humidity, light intensity, sound pressure, and time. We chose these primitive contexts because of availability of sensors for capturing them. Moreover, each of them describes a particular physical property of a PLACE. However, we maintain the possibility of some of these primitive contexts may be unavailable at any given time, and we may not be able to foresee a priori which one of them might be unavailable. There can altogether be 15 possible combinations of available context inputs which can describe a PLACE with various degrees of certainty.

Step 2: We used fuzzy sets to define linguistic variables for the primitive contexts. The linguistic variables are results of empirical observations of various places. Our approach

resembles with earlier approaches of Mäntyjärvi et al. [17] and Korpipää et al. [13]; the essential difference is that we established conditional dependencies between *time* and the primitive contexts. By analysing the data were acquired from various rooms, corridors, and outdoors, we model the conditional dependencies between the primitive contexts and the higher-level context. As a result, the fuzzy set for *time* has three linguistic variables: *morning*, *noon*, and *afternoon*. Likewise, we established the following linguistic variables: For temperature: *very cold*, *cold*, *lukewarm*, *warm*, *hot*; for sound pressure*: noisy*, *loud*, *quiet*; for relative humidity: *dry*, *moderate*, *moist*; and for light intensity: *dim*, *visible*, *bright*, *very bright*. By using fuzzy modifiers, the members of a fuzzy set can be modified or new members can be introduced. As an example: *very noisy*, *more or less quite* can be generated from the fuzzy set of sound pressure.

Step 3. We employed a Bayesian Network (BN) to express causal dependencies between a PLACE and the primitive contexts – the nodes (random variables) of the BN represent the higher-level context as well as the primitive contexts, and the values of the random variables represent the linguistic variables (correspondingly, the term sets of the linguistic variables). Criteria for the choice of a BN were: ease of knowledge representation, wide availability of efficient and tractable algorithms, and robustness. Robustness refers to the capability of the network to reconfigure itself whenever nodes (Primitive Context Servers) arrive and depart.

X. Setting up a Bayesian Composer

A Bayesian Composer is set up by specifying the type of higher-level context to be recognised, the primitive contexts required for the computation, and the conditional dependencies between the various nodes of the network. A configuration file is obtained from the EAK. However, the particular topology of the Bayesian Composer at any given time depends on the available primitive contexts, whereas the network's configuration, i.e., the various states which can be captured depends on the reliability of the sensors available.

Figure 3 and 4 show two different topologies of the Bayesian Composer; each configuration displays the available primitive contexts and the corresponding adjustment in the state of the higher-level context to be recognised. In the figures, P stands for place, t stands for time, H stands for relative humidity, T stands for temperature, and L stands for light intensity. The various states of the nodes are as described in the previous section.

Once a Bayesian Network is established, the Composer begins to query available Primitive Context Servers; the data it obtains from them is mapped to appropriate linguistic variables in the EAK, and then the Composer computes posterior probabilities to determine the likely place the sensory data represents. A place with the highest posterior probability is declared to be the whereabouts of the user.

TABLE 1: Decision Reliability of a Bayesian Composer

Primitive context	**Higher-Level Context**
Temperature	({P = O, 0.6}, {P = I, 0.7}, {P = (O or I), 0.2}, {e = 0.2})
Relative Humidity	({P = O, 0.8}, {P = I, 0.6}, {P = (O or I), 0.1}, {e = 0.2})
Temperature, Relative Humidity	({P = O, 0.9}, {P = I, 0.8}, {P = (O or I), 0.05}, e = 0.1})
Temperature, Light Intensity	({P = O, 0.9}, {P = I, 0.8}, {P = (O or I), 0.1}, {e = 0.05})
Relative Humidity, Light Intensity	({P = O, 0.9}, {P = I, 0.9}, {e = 0.05})
Temperature, Relative Humidity, Light Intensity	({P = O, 0.9}, {P = R, 0.7}, {P = C, 0.6, {P = (C or R), 0.25}{P = (O or I), 0.0}, {e = 0.1})

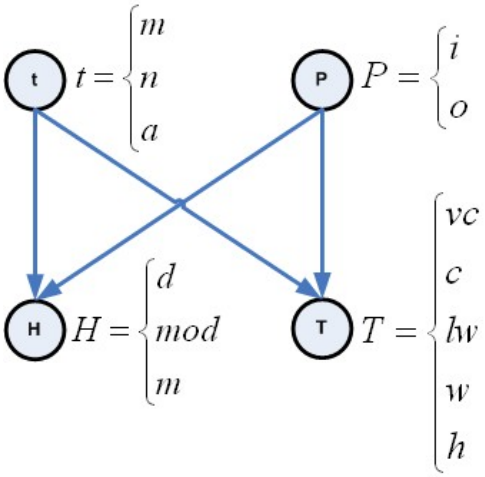

Figure 3: A Bayesian Composer with two parent nodes and two child nodes (relative humidity and temperature)

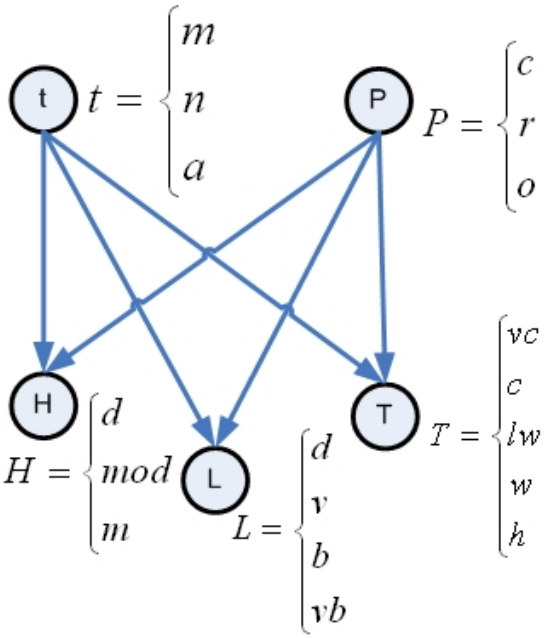

Figure 4: A Bayesian Composer with two parent nodes and three child nodes (relative humidity, temperature, and light intensity)

XI. Experiment Results

We employed the Bayesian Composer to reason about the whereabouts of a mobile user in the absence of any location sensor. Depending on the availability of primitive contexts, the Bayesian Composer could discriminate between ROOM, CORRIDOR and OUTDOORS with various levels of uncertainty. If there were no sufficient sensors to discriminate between a CORRIDOR and a ROOM, the Composer discriminated between INDOORS (correspondingly BUILDING) and OUTDOORS, since a BUILDING subsumes a ROOM and a CORRIDOR, which in turn is subsumed by INDOORS.

At any given time, the uncertainty associated with the outcome of the Composer depended on the diversity of the primitive contexts available and the reliability of the sensors delivering the primitive contexts.

The decisions of the Composer were classified as: *correct*, *error*, and *undecided*. Undecided referred to the inability of the Composer to distinguish between two or more states of a higher-level context because they had the same posterior probabilities, or the difference between the probabilities is not significant enough[2]. Error occurred when the Composer decided for the wrong state. 10 decisions were observed for each place and combination of input primitive contexts listed in table 1, which summarises the Composer's discriminating efficiency for time, temperature, light intensity and relative humidity primitive contexts.

XII. Conclusion

We proposed several criteria for a context-aware system to reason about dynamic real world situations on the basis of data from physical sensors. A distributed architecture was designed and implemented to accommodate the criteria proposed. The architecture consists of Primitive Context Servers, which abstract lower-level (primitive) context acquisition; Aggregators, which improve the quality of a primitive context, and which gather several primitive contexts describing the various aspects of a real-world situation; a Knowledge Base and an Empirical Ambient Knowledge component to encode facts and beliefs about entities – while the facts reflect objective reality, however incomplete, the beliefs incorporate ignorance; and a Composer to reason about a higher-level context.

We employed fuzzy sets to model beliefs, and a Bayesian Composer to manipulate these beliefs and sensed data in order to reason about the whereabouts of a person in the absence of any location sensor. The composer could discriminate between various places (INDOORS and OUTDOORS as well as between OUTDOORS, CORRIDOR, and ROOM) with different levels of uncertainties, depending on the variety and reliability of the sensors available.

[2] A difference in posterior probabilities below 15 percent was considered to be insufficient to discriminate between places; this limits the total error (error + undecided) below 5 per cent.

References

[1] W. Dargie, T. Loeffler, O. Droegehorn, and K. David, "Composition of Reusable Higher-Level Contexts," in Proceedings of 14th Wireless and Mobile Communication Summit, IST, 2005.

[2] A. Dey, "Providing architectural support for building context-aware applications", Doctoral Thesis, 2000.

[3] M. Weiser, "Ubiquitous Computing," *Computer* 26, 10 (Oct. 1993), 71-72.

[4] P. Dourish, "What we talk about when we talk about context," Personal Ubiquitous Comp. 8, 1, Feb. 2004.

[5] S. Greenberg, "Context as a Dynamic Construct. Human-Computer Interaction," Volume 16, pp. 257–268 (2001).

[6] J. Pascoe, "Adding generic contextual capabilities to wearable computers," in the Proceedings of the 2nd International Symposium on Wearable Computers (ISWC'98). IEEE, 1998.

[7] P. Yancey and P. Brand. "In His Image," Sondervan Publisher, 1987.

[8] J.P. Martines and S.C. Shapiro, "A model for belief revision," Journal of Artificial Intelligence. Elsevier Science Publishers, 1988.

[9] W. Dargie, O. Droegehorn, and K. David. 2004. Sharing of Context Information in Pervasive Computing. In Proc. of the 13th Mobile and Wireless Communication Summit. IST (June 2004).

[10] N.H Cohen, A. Purakayastha, J. Turek, L. Wong, and D. Yeh, "iQueue: A Pervasive Data Composition Framework," in the Proceedings of The 3rd International Conference on Mobile Data Management. IEEE, 2002.

[11] W.B. Heinzelman, A.L. Murphy,H.S. Carvalho, and M.A. Peri, "Middleware to Support Sensor Network Applications," Journal of. IEEE Networking, IEEE, 2003.

[12] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa, "Computational auditory scene recognition," in Proc. of International conference on acoustic speech and signal processing, 2002.

[13] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Kernen, and E-J. Malm, "Managing Context Information in Mobile Devices," Journal of Pervasive Computing, IEEE, 2003.

[14] D. Hall and J. Llians. (Editors), "Handbook of Multisensor Data Fusion," CRC Press, 2001.

[15] ASHRAE Handbook: Fundamentals, Ashrea, 2005.

[16] T. Malmstrom, J. Andersson, FR. Carrié, P. Wouters, and Ch. Delmotte, "Source Book for Energy Efficient Air Duct System in Europe," Airways Partners, 2002.

[17] J. Mäntyjärvi, J. Himberg, and P. Huuskonen, "Collaborative Context Recognition for Handheld Devices," in *Proceedings of the First IEEE international Conference on Pervasive Computing and Communications* (March 23 - 26, 2003). PERCOM. IEEE Computer Society, Washington, DC, 161.