2010 Fourth International Conference on Genetic and Evolutionary Computing

Towards A Context-aware Middleware in Smart Car Space

Jie Sun, YongPing Zhang, Jianbo Fan
School of Electronic and Information Engineering
Ningbo University of Technology
Ningbo, China
{sunjie, ypz}@nbut.cn, jbfan@163.com

***Abstract*—Cars have become an increasingly important and exciting test bed for ubiquitous computing. However, smart car space is still little addressed in the literature to enable high adaptation in a mobile and resource-limited space. This paper focuses on building a context-aware software infrastructure for smart car space in middleware framework. To support context-awareness, we embed capabilities of context modeling and context reasoning. We apply a three-layer manner, including sensor, context atom, and context situation, to efficiently acquire and process contexts. We implement rule-based reasoning in our prototype. We evaluate the performance, and show the proposed middleware is promising.**

Keywords- smart car space; context-aware; middleware; ubiquitous computing

I. Introduction

Ubiquitous computing [1, 2] applications have gotten involved in many areas, from household domains to working places, from mini-objects to cities. Recently, another novel application of ubiquitous computing in vehicles is attracting researchers [3, 4]. Intelligent transportation systems (ITS) [5] have made a rapid progress over the past two decades in both the transportation infrastructure and vehicles themselves [6, 7].

As new concepts, methods, tools, and devices continue to emerge in fields of automation, a car is more than computers with wheels, but a smart space. A smart space is more than smart applications and services. In pervasive computing community, there are a couple of work on smart spaces and their software infrastructure. Gaia project [20] considered an active space is analogous to traditional computing systems, and so built a component-based middleware operating system, GaiaOS. The Stanford Interactive Workspaces project [21] concentrates on task-oriented work and a prototype meta-operating system, iROS (Interactive Room Operating System), runs as a meta-OS to coordinate the various applications in the room. The Classroom 2000 project [12] develops Zen-star system to performs the tasks of capturing and synchronizing streams of information during each live session. Context toolkit [16] provides abstract components to easy the development of context aware application. The NIST Smart Space and Meeting Room projects [13] focus on developing tools for data formats, transport, distributed processing, and metadata in a meeting room. RCSM [22] enables to develop context-aware applications in that it represents an application as context sensitive interface and context-independent implementation. The Intelligent Room [23] uses the Java-based Metaglue agent infrastructure to build agents that can communicate with each other.

The smart car space is providing an increasingly important and exciting test bed for ubiquitous computing. However, most of current work on ITS and smart car concentrate on the new devices and services to help users to safe and comfortable driving. Little has been devoted to smart car space. This paper focuses on how to build a smart car space, which will provide appropriate information to the driver for safer and better driving, warn the driver passively when his car is in a hazardous situation or in possible hazardous situations. The paper is structured as follows. Section 2 is an overview of the architecture. Section 3 and 4 introduce two key components of the middleware in detail respectively. Section 5 is the performance evaluation. Section 6 summarizes the paper and discusses the further work.

II. An Overview of Architecture

Compared with other static spaces such as office, home, classroom and meeting room [10, 11, 12, 13], the smart car space is mobile when the car is driving, which means the environment context changes rapidly. For such a complicated and dynamic environment, the smart car space must provide support of high mobility and self-adapting computing framework that automates the configuration and reconfiguration of changeful environments.

The architecture is separated into two parts: the remote service center and smart car space, as shown in Figure 1.

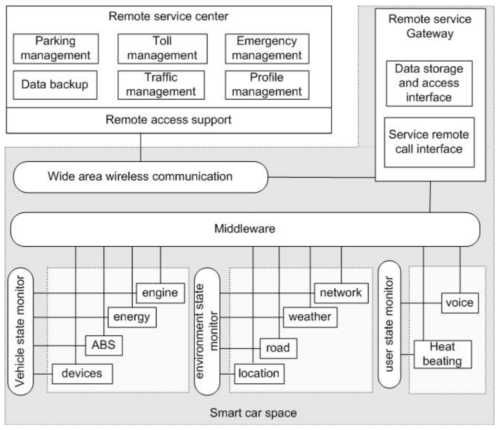

Figure 1. Illustration of the smart car space.

The remote service center guarantees the seamless service access and encapsulates the problems brought by mobility. The smart car space provides self-adaption using a component-based middleware. The two parts interact with each other through remote service gateway wirelessly. Besides, the gateway supports automatically handover of network protocol when the driving car moves beyond the range of the current communication network and enters the range of another communication network.

1. The remote service center

The smart car is limited in resource and computation, so we implemented three fundamental services in the remote service center: data backup, traffic management and profile management. The smart space produces contexts in huge amount, which is impossible and unnecessary to store all of them in the car. We can store them in remote server and access them whenever necessary, just like the method of a small mobile device such as PDA. Three simple applications are built on the basis of the fundamental services: parking, toll and emergency management. Users can store their preference in the profile such as the contact person, so the emergency management service will call the person when the user is in emergency.

2. The middleware in smart car

The devices and software infrastructure are managed by a context-aware middleware, as shown in Figure 2. The presented platform includes four layers: network layer, broker layer, context infrastructure, and ser-vices layer.

(1) Network layer.

The smart car supports different communication approaches. A ZigBee wireless sensor network con-nects all mini-sensors. The smart vehicle network is a serial-bus system for the communicating between mechanical nodes (such as the engine and steering system). The WLAN 802.11a/b/g network supports the communication between digital devices. The CDMA 1xRTT network is responsible for wide-area communication.

(2) Broker layer.

The broker layer manages sensors, devices and other hardware. The sensor broker is responsible for discovery and registration of new sensors adding into the smart car. One broker manages one category of sensor. Sensors can transmit data via WLAN, serial port, Ethernet, and USB. The broker will assign a globally unique address or identity to a sensor, specify the updating frequency, and define the way for the sensor to transmit data and for the system to parse data. The device broker is responsible for discovering new devices and registering them for cooperation in the smart car. The ECU broker aims at managing processors and collecting specific contexts such as spare memories.

(3) Context infrastructure.

The context infrastructure has been implemented on the basis of Context Toolkit and consists of three parts: a) The context wrapper, which transforms sensor data into semantic context atoms; b) The context reasoner, which trains and recognizes context situations by aggregating various types of context atoms; c) The context storage, which is a repository for historical contexts and provides the advanced query services.

(4) Service layer.

Smart cars intend to create safer, more efficient and more convenient driving environment for drivers, so specific services should be developed. In a smart car, most services, such as slowing down when the distance to the front adjacent car is less than the safety limit, need to transfer signal via CAN to control a certain actuator. Each actuator is managed by an ECU. In order to execute the service, a message including the service API and parameters should be sent to the ECU that manages the actuator. The ECU will parse the message and send control signal to a relay, which will control the actuator to change its state.

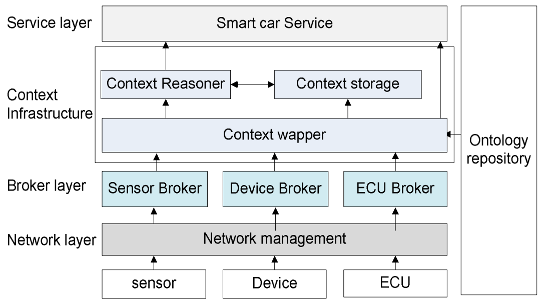

Figure 2. Architecture of the middleware.

III. CONTEXT WRAPPER

The ContextWrapper is responsible for extracting context from the physical or semantic sensors by polling the sensor periodically. The raw sensor data is numeric and meaningless to applications, so we must retrieve meaningful contexts from sensor data.

Context is any information that can be used to characterize the situation of an entity [12]. According the degree of abstraction and semantics, we can divide context into context atom and context situation, as shown in Figure 3.

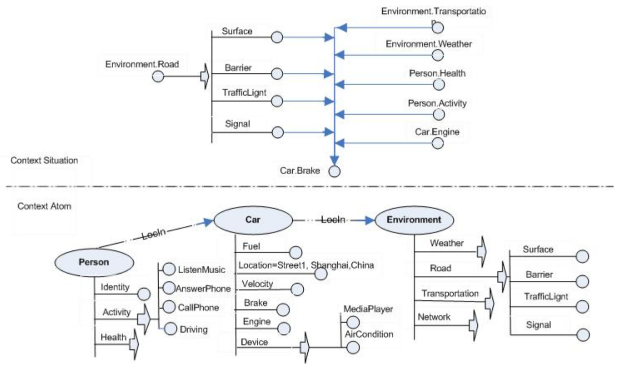

Figure 3. Contexts in smart car space.

A context atom is a semantic fact retrieved from a physical sensor and the fact cannot be divided to more trivial ones, such as Person (TOM) and Location (ROOM201). Each sensor corresponds to a context atom, but it is not true in reverse. Context atoms can be shared among different network nodes for that the communication protocol is based on HTTP and XML.

A context situation is the current state of the entities in the smart car space. The situations may be arbitrarily complex in the way of defining and recognizing them. We define the state of the smart car as a conjunction of context atoms, including environmental atoms, car atoms and person atoms. For example, a great deal of elements will lead the car to stop driving or slow down. They include environmental elements such as the traffic light turning to red and the signboard asking for a speed limit. If the smart car detects malfunction of the engine or illness of the driver, it will make a stop, too.

We apply ontology technology to enable sharing and interoperation between smart cars, for we believe there will be much cooperation between smart cars in the highway though Vehicular Ad Hoc Networks (VANET). The semantic label assigned to each atom comes from ontology repository. In our smart car space, we use three classes of ontology to provide agreed understanding of the contexts: ontology for environment context, ontology for car context, and ontology for driver context.

IV. CONTEXT REASONER

In order to make the car smart, we must enable it to analyze and deduce. ContextReasoner lays emphasis on deducing high-level context situations from low-level context atoms. As shown in Figure 4, the inference engine is composed of two parts: online rule-based reasoning and offline statistic-based learning.

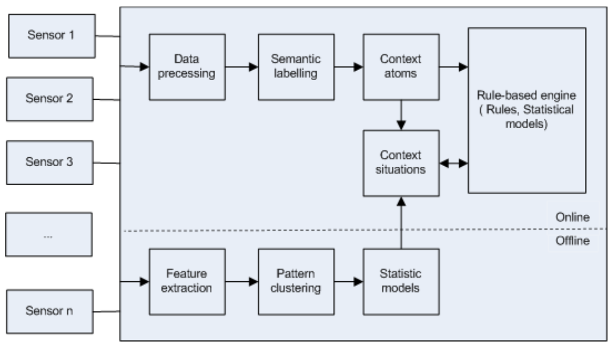

Figure 4. Flowchart of recognition of context situations.

We choose to implement rule-based reasoner by using first-order predicates. The structure of the first-order predicate has three fields: a subject, an object and a verb. For example, the physical location context "Tom sits in the driver seat" can be described as Location (Tom, DriverSeat). Rules are constructed with two properties: Action and Condition. Action is a statement representing what shall be done if the condition is true. Condition in a rule is a collection of context atoms. We use this reasoner to trigger situation transition and appropriate service, such as turning on the air-condition when it is too hot.

We use machine learning algorithm to recognize the current situation. The Kohonen Self-Organizing Map (KSOM)[15] clustering algorithm orders the sensory inputs by assigning map-units to each kind of input, and after a while, the resulting map is topologically ordered, i.e. similar inputs activate neighboring units. After a few iterations, the neurons start to organize themselves in a structured, topological way: different sensor inputs activate different neurons. A cluster corresponds to a situation. We assign a label to each situation.

In our middleware, we still support multiple kinds of reasoner including ontology reasoner and Bayesian reasoner. Ontology reasoner is a special instance of rule-based reasoner. The ontological reasoner consists of predefined rules, which can reason about OWL vocabularies and new concepts. When developing ontology reasoner, we impose emphasis on semantic mapping and knowledge base access. We build a context knowledge base, which provides a set of API's for other components to query, add, delete or modify context knowledge. The Context KB contains context ontologies and their instances. These instances may be specified by users in case of defined contexts or acquired from sensors. The context ontologies and their instances are preloaded into the context KB during system initiation, while the instances of sensed or deduced contexts are loaded during runtime. It is the major producer of new contexts. With the new contexts, the system is able to evolve and progress.

V. PERFORMANCE EVALUATION AND DISCUSSION

To determine performance of the middleware and smart car space, we use the following performance metrics: quality of context and performance of context reasoning.

1. Quality of context

We evaluate the quality of context from freshness and accuracy.

Freshness mostly depends on the frequency and latency time. Sensors will send their data to our middleware at their different frequency, such as 30 frames per second from the camera. As for latency, the temperature sensor will delay 11 seconds to respond to new temperature, while pressure and air quality sensors will report the data in 1ms. In atom layer, we will distinguish the difference between two sequential data. If they do not vary, the new data will be discarded. So the atom layer reduces the amount of context greatly, which reduce the computational complexity in consequence, with the guarantee of fresh contexts.

We have combined each sensor and hence each atom with a confidence, so the uncertainty from sensor layer will be taken into account in the beginning. Besides, the reasoning models will be checked carefully for correctness and performance. Thus the accuracy of context is guaranteed.

2. Efficiency of context reasoning

The rule set of rule-based reasoner consists of 20-100 forward chaining rules. We use the rule set to do adaptation. Bayesian networks structure was performed using the K2 algorithm. The K2 algorithm is a greedy search technique which starts from an empty network but with an initial ordering of the nodes. The reasoners are applied in different situations. We use Bayesian network to determine the current state of the smart space, so the accuracy is the most important issue; while we use rule-based reasoning to define the services that should be triggered in different state, so the respond time is the most interesting issue.

We design an experiment scenario to evaluate the middleware: two users enter a car and begin to drive at a speed of 60 km/h for 30 minutes. They can speak, have discussion, enjoy music, smoke, sleep, browse webpage, and make schedule. We repeat this scenario for 20 times.

Figure 5 shows the results of the experiments. The run time performance of context reasoning depends greatly on size of context information and complexity of rule sets. The result using small rule set greatly outperforms the one with a large rule set over identical context dataset. The CPU speed will have impact on performance too, but it is not our focus.

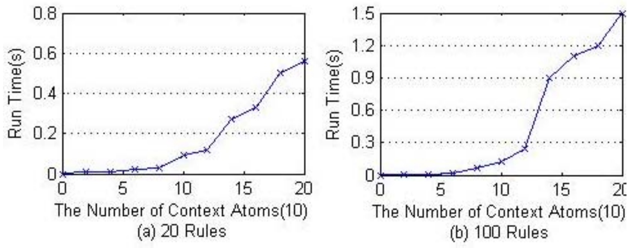

Figure 5. Run time performance of rule-based context reasoning.

Context reasoning is a computational intensive task. However, it is still feasible for non-time-critical applications, so that the delay of context reasoning (less than 2 seconds) is acceptable. For time-critical applications such as security and navigating systems, we need to control the scale of context dataset and the complexity of rule set. A tentative solution is to perform complex reasoning tasks in off-line manner.

VI. CONCLUSION AND FUTURE WORK

As a promising application area of ubiquitous computing, smart car space is becoming more and more important. The contribution of this paper is a context-aware middleware for smart car space, which is a component-based comprehensive framework with support of high adaptation.

Reliability is central feature in smart car. Besides taking uncertainty into account, our future work is to add controllability and reliability to the smart car space. There is much room remaining unexplored to help driver avoid the errors. We will concentrate on automatic acquisition and analysis of driver status. We are developing sensing methods to detect the physiology and psychology status of the driver.

ACKNOWLEDGMENT

This research is supported by Ningbo Natural Science Foundation (No. 2010A610108), NSF of Zhejiang Province, China (No.Y1080123), Foundation of MHRSS of China (No. 2009-416), and Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry(SRF for ROCS, SEM, No. 2009-1590).

REFERENCES

[1] Mark Weiser, “The Computer for the 21st Century,” Scientific American, Page(s): 94-100,1991.

[2] Want R, Schilit BN, Adams NI, Gold R, Petersen K, Goldberg D, Ellis JR, Weiser M., “An overview of the ParcTab ubiquitous computing experiment,” IEEE Personal Communications, 1995,2(6) Page(s):28-43.

[3] Gang Pan, Zhaohui Wu, Jie Sun, “Toward A Smart Space Inside Car”, The 9th International Conference on Ubiquitous Computing (ubicomp’07) LBR, Innsbruck, Austria, 2007.

[4] Gary E. Burnett, J. Mark Porter, “Ubiquitous computing within cars: designing controls for non-visual use,” International Journal of Human-Computer Studies, 55(4):521-531, 2001.

[5] Fei-Yue Wang, “Driving into the Future with ITS,” IEEE Intelligent Systems, Volume 21, Issue 3, Jan.-Feb. 2006, Page(s):94 – 95.

[6] Fei-Yue Wang, Daniel Zeng; Liuqing Yang, “Smart Cars on Smart Roads: An IEEE Intelligent Transportation Systems Society Update,” IEEE Pervasive Computing, Volume 5, Issue 4, Oct.-Dec. 2006, Page(s):68 - 69

[7] Moite. S, “How smart can a car be,” in Proceedings of the Intelligent Vehicles '92 Symposium, 29 June-1 July, 1992, Page(s):277 – 279.

[8] Renato Cerqueira, Christopher K. Hess, Manuel Romn, Roy H. Campbell, “Gaia: A Development Infrastructure for Active Spaces,” In Workshop on Application Models and Programming Tools for Ubiquitous Computing, 2001.

[9] Borchers, J.,Ringel, M.,Tyler, J.,Fox, A., “Stanford interactive workspaces: a framework for physical and graphical user interface prototyping,” IEEE Wireless Communications, Volume 9, Issue 6, pp.64- 69, 2002.

[10] Gregory D. Abowd, “Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment,” IBM Systems Journal, Special issue on Pervasive Computing, Volume 38, Number 4, pp. 508-530, 1999.

[11] Anind K. Dey, “Understanding and Using context,” Personal and Ubiquitous Computing, Volume 5, Issue 1, pp.4-7, 2001.

[12] V. Stanford, J. Garofolo, O. Galibert, M. Michel, C. Laprun, “The NIST Smart Space and Meeting Room Projects: Signals, Acquisition, Annotation and Metrics,” in Proceedings of ICASSP 2003 in special session on smart meeting rooms, invited paper, 2003.

[13] Stephen S. Yaua, Fariaz Karim, “A context-sensitive middleware for dynamic integra-tion of mobile devices with network infrastructures,” Journal of Parallel and Distributed Computing, Volume 64 , Issue 2, pp. 301–317, 2004.

[14] Peters, S., Shrobe, H.E, “Using semantic networks for knowledge representation in an intelligent environment,” in Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom’03), pp.323 – 329, 2003.

[15] Kristof Van Laerhoven, Ozan Cakmakci, What Shall We Teach Our Pants?, in Proceedings of the 4th IEEE International Symposium on Wearable Computers table of contents, 77, 2000.