ConFra: A Context Aware Human Machine Interface Framework for In-vehicle Infotainment Applications

Hemant Sharma, Dr. Roger Kuvedu-Libla, and Dr. A. K. Ramani

Abstract— **The omnipresent integration of computer technology in infotainment applications introduces new opportunities to make applications aware of the context in which they are used. Unlike existing context aware systems which isolate one characteristic such as road or driver workload and concentrate on it in exclusion of the others, this paper proposes a context awareness framework, *ConFra*, which brings the knowledge of application's context to Human Machine Interface and considers the driver, vehicle and environment as a whole.**
The approach in *ConFra* places context-aware design within a wider scope that takes into account information related to the driver, environment and vehicle at HMI and offloads infotainment applications from context related concerns. Such an approach aids in the design of safe in-vehicle context-aware systems.

***Index Terms*— Context Aware, In-vehicle Infotainment Systems, Human Machine Interface (HMI), *ConFra*.**

I. Introduction

With the continuous growth of wireless communication and entertainment, mobile hand-held devices, such as PDAs, media players and mobile phones, are becoming a powerful interface platform that allows automotive infotainment suppliers to create a whole range of next generation infotainment applications that are more intelligent and supportive to the driver and passengers of a vehicle compared to most nowadays applications. A pervasive computing environment [1] supposes that (1) applications are deployed on devices ranging from traditional computers to handheld devices, (2) that they are aware of contextual information on the user, the device itself and its environment, and (3) that they can be adapted to this dynamic context accordingly. For example, in such an in-vehicle infotainment environment applications may have to be adapted to the varying device capabilities, such as memory or display size. Furthermore, applications may have to be moved while active from one device to another, in order to provide the driver with the best infotainment experience.

Current interactive systems for mobile in-vehicle infotainment scenarios should be prepared to face up and accommodate the continuous and diverse variability inherent to mobility, in which the environmental constraints play a key role. We are referring to the well-known *Context- Awareness problem* [2], proposed about a decade ago. To tackle context-awareness, some kind of software infrastructure is needed to provide proactive adaptive capacities, able to evolve as the contextual constraints vary [3].

Recent trends in automotive software design indicate that the use of prefabricated building blocks for software development is on the rise. The prefabricated artifacts are the off-the-shelf (COTS) software infrastructure and domain-specific service components that one can acquire from different vendors and integrate them to deploy large-scale software applications. Vehicle Navigation is a good example of such an application on In-vehicle Infotainment systems. These COTS may not have support for context awareness.

In this paper we present *ConFra*, a context aware human machine interface framework for in-vehicle infotainment applications that can provide context awareness at HMI. Each infotainment application dynamically encodes in an application profile the way context should influence the HMI organization and information presentation to end user. *ConFra* uses this profile information to reduce the resources available to the application in the current context to a subset of 'plausible' ones. Each application also encodes the constraints and QoS needs into the context profile that *ConFra* applies to select the most suitable context among the plausible ones. *ConFra* builds on the following assumptions:

- The existence of a shared repository to refer to context elements and conditions, resource names and characteristics, and non-functional requirements.
- The integration with an existing context discovery protocol for pervasive resources on which *ConFra* relies for HMI display and information queries.

Hemant Sharma is Software Engineer at Delphi Delco Electronics Europe GmbH, Bad Salzdetfurth, Germany. (e-mail: hemant.sharma @ delphi.com).

Dr. Roger Kuvedu-Libla is EMC-Competency-Leader at Delphi Delco Electronics Europe GmbH, Bad Salzdetfurth, Germany. (e-mail: roger.kuvedu.libla @ delphi.com).

Dr. A. K. Ramani, is Professor at School of Computer Science, Devi Ahilya University, Indore, INDIA. (e-mail: ramani.scs@dauniv.ac.in).

The paper is structured as follows: In the following section an overview of related research is provided. Section 3 describes the architecture of *ConFra*. Section 4, gives an overview of context management subsystem of *ConFra*. In section 5, we describe the Context Infrastructure for *ConFra*. Section 6 describes an example context aware HMI based on *ConFra*. In section 7, we elaborate the future activities and finally conclude the paper.

II. Background

Context is any measurable and relevant information that can be used to characterize the situation of an entity (e.g. driver) [9]. Context is highly dynamic in space and time. Context could be considered as a 'setting' in which interactions unfold [10]. The setting has the dual role of creating and constraining interactions.

Drivers of vehicles operate in highly dynamic environments or contexts. Existing context aware systems use context such as task at hand, location, user preferences and device capabilities [11, 12, 13] to deliver relevant information to the user. The relevance of the information is relative to a particular circumstance or context.

With an increasing number of applications using mobile devices, mobile Human-Computer Interaction (HCI) research places more emphasis to human-device interaction in terms of developing mobile context-aware applications. Kjeldskov and Graham [4] have reviewed a number of mobile HCI research methods, noting a focus towards building systems.

There are already several different architectures and frameworks supporting development of context aware software systems such as [5] and [6]. However, the important aspect of designing contexts and adaptation logic into HMI for In-vehicle infotainment systems is an uninvestigated area of automotive research.

III. The ConFra Architecture

The *ConFra* HMI framework is designed as set of components organized to form a layered architecture as shown in Figure 1.

The framework has been developed using Model Driven Architecture (MDA) methodology and partitioned into independent subsystems based on functional area. The framework subsystems are organized as set of independent components, communicating via well defined interfaces, to make the framework scalable and flexible. The framework has a set of core components and some optional components. Core components provide the bare minimum functionality that is required for an HMI application. Optional component may be configured along with framework to provide additional functional interfaces.

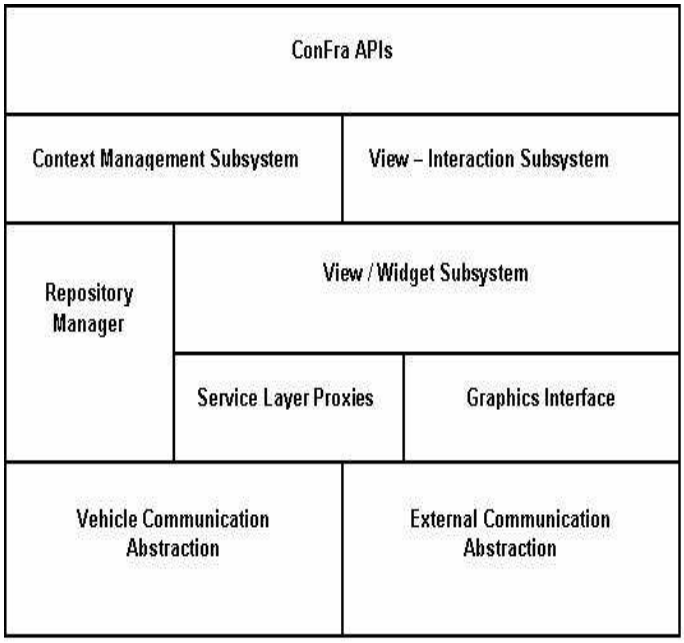

Figure 1: *ConFra* Architecture

In this paper, we focus on the architecture and functionality of *context management subsystem* of *ConFra*. In order to construct and provide context awareness *ConFra* uses contextual information that define the functional behavior of infotainment applications. *ConFra* supports the application functionality that belongs to the following functional groups:

A. *Information and Service Display:*

In order to reduce user workload the framework adjusts the set of offered information and functions according to detected and deduced environment of the user.

B. *Automated Command Execution*

Example of this group would be an off-board navigation application that detects the user has missed the intersection and automatically initiates rerouting to find new shortest path to destination.

C. *Context Storage*

Potential use of stored contextual information would be to enable application to autonomously extract user preferences from HMI interactions.

IV. The Context Management Subsystem

The *ConFra* uses components as modular building blocks for the design and deployment of adaptable HMI services. Component-based applications and services [7] are more flexible than monolithic applications: the component paradigm makes it possible to add, remove or replace a component at runtime when the context changes.

Figure 2 provides a functional overview of *context management subsystem* of *ConFra*. The subsystem is composed of *Context Infrastructure*, *Context Manager*, *Metadata Manager* and *Event Manager*.

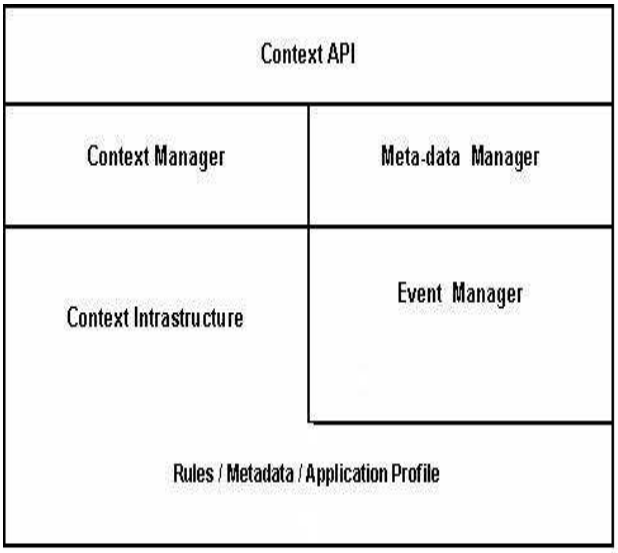

Figure 2: Context Management Subsystem Overview

Functionality of context subsystem is made available via a set of context APIs. Context Infrastructure shall, at minimum, contain the context rules, context metadata and context profiles from infotainment applications. Further details of context Infrastructure is provided in the next section of this paper.

The Metadata Manager (MM) and the Context Manager (CM) compose the high-level context framework facilities. Event Manager (EM) plays the crucial role of delivering the events relevant for triggering migration and binding policies.

A. Context Manager

CM is responsible for dynamically establishing the context of any HMI client application. In addition, other framework components can query CM to retrieve and modify the context objects of any HMI client at provision time.

B. Metadata Manager

MM supports the specification of the different kinds of infotainment metadata. It is in-charge of supporting the specification and update of application profile information, and of dynamically installing or enforcing policies for access control and interaction handling.

C. Event Manager

EM dispatches the registered events to interested policy subjects independent of subject migration during service provisioning. EM also permits to define aggregated events by composing several low level monitoring indicators.

For Instance, the *Communication Abstraction* subsystem of *ConFra* is capable of sensing when a PDA or BLUETOOTH mobile phone is granted connection to the infotainment system. In this scenario, Event Manager delivers the corresponding *ApplicationDetected* event to CM, which maintains the information about locally available resources to determine updated contexts, and to the other interested policy subjects.

Context Management Subsystem of *ConFra* provides a simple yet powerful dynamic model to represent context for different HMI requirements of infotainment applications. The context information is accessed by framework components through two complimentary interfaces: synchronous requests and asynchronous notifications. An infotainment HMI scenario can implement the different aspects of its context by only depending on context management subsystem of *ConFra*.

V. Context Infrastructure

The *context management* subsystem of *ConFra* uses variety of information, sources, profiles and rules to develop context awareness. This section discusses the context information building blocks of *ConFra*.

A. Context Domain Implementation

Figure 3 represents the elements of context domain for *ConFra*.

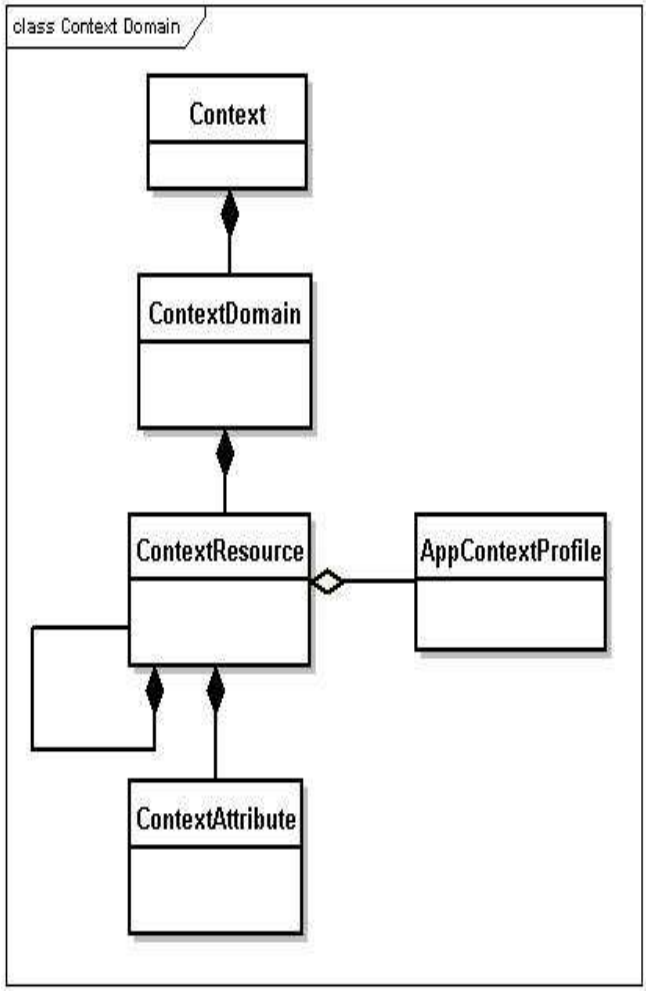

Figure 3: Context Domain Overview

The context in *ConFra* is made of several domains, each represented by a *ContextDomain* object and identified by a unique name. The purpose of context domains is to separate the different aspects of the execution context, and to allow each of these to use a custom implementation, be it for performance reason or for interoperability with existing infotainment applications in the system.

Finally, each context domain is modeled as a tree of named context resources, each being described by attributes (simple key, value pairs). This simple and generic model was chosen because of its generality, familiarity to context patterns [14] and because it does not impose a complex implementation

B. Context acquisition Framework

This framework is centered around the notion of *ContextInterface*, which represent the *ConFra* component responsible for the acquisition of context profile or contextual information. The interface methods are used to get the data that can be dynamically varying, requiring communication with lower layers or even directly with external environment. However, *ConFra* provides the *Context Manager* component in context subsystem to organize all the context interfaces. The default context domain implementation, which relies on this context acquisition framework, uses one context manager interface per HMI domain.

Figure 4 represents the context acquisition framework.

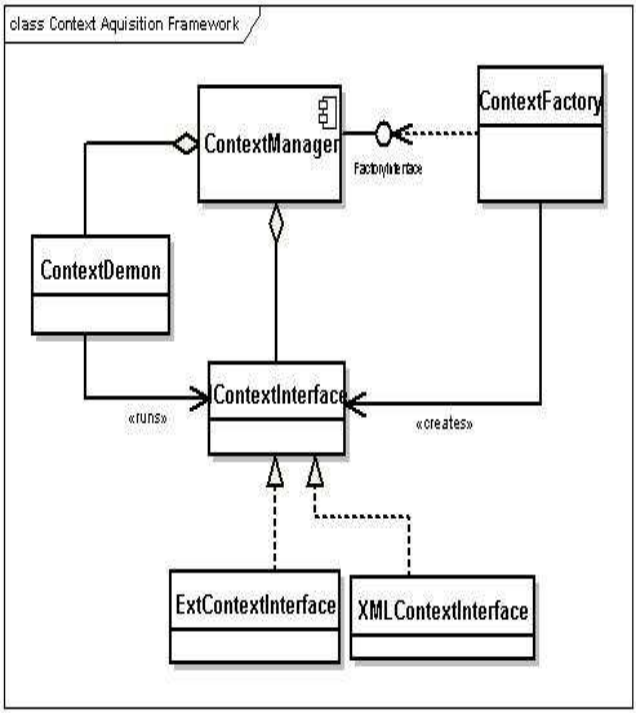

Figure 4: Context Acquisition Framework

C. Context Meta Data

Figure 5 represents the context metadata hierarchy for contexts in *ConFra*.

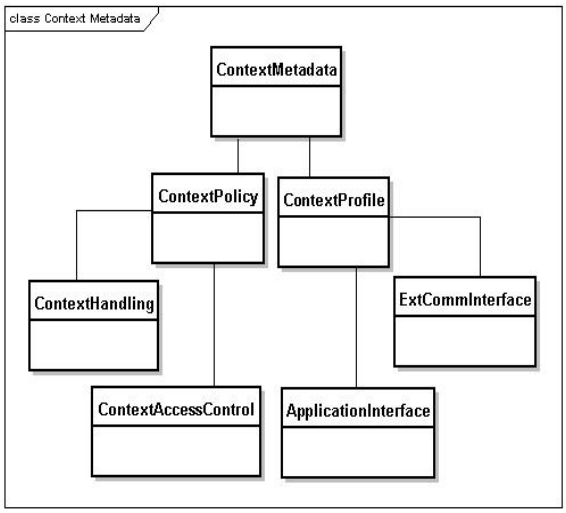

Figure 5: Context Metadata Overview

ConFra exploits two types of metadata: *context profiles* to describe the characteristics of any resource modeled in the system, and *context policies* to manage migration, binding and access control. *Context Profiles* describe the interfaces to applications, devices or vehicle network. *Context Policies* provide an abstraction, by listing context handling and access control methodologies.

D. Context Service Template

```
<?xml version="1.0" encoding="utf-8"?>
<ContextProfile xmlns="http://ConFra.delphi.com/context"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ConFra.delphi.com/context:\\CONTEXT\schema.
xsd">
  <AppParams>
    <Name>Media Player</Name>
    <ViewID>v_MpPlayList</ViewID>
    <ViewType>vt_ListSelect</ViewType>
  </AppParams>
  <ContextRules>
    <Rule1>v_MpPlayListDisplay.xml</Rule1>
    <Rule2>v_MpPlayListSelectRule.xml</Rule2>
  </ContextRules>
  <ContextParams>
    <Param>v_MpPlayListParam.xml</Param>
  </ContextParams>
</ContextProfile>
```

Figure 6: Context Service Template Example

Context-aware HMI acquire context information from the applications via *context manager* and adapt them accordingly. One common way in which HMIs are made context-aware is to specify rules that trigger actions when a specific contextual event happens. *ConFra* context-aware architecture makes it easy to establish context-aware behaviors by specifying IF-THEN rules [11, 12] in context service template. Figure 6 provides XML that presents an example of the context service template used for *ConFra*.

VI. Context Awareness for Media Player HMI

To test the context awareness support and context aware components built in *ConFra* framework we have developed HMI for *Media Player* application running on an *infotainment platform* on top of *ConFra*. The application setting assumed that the *Media Player* application has connections to media devices connected to the Infotainment system. Further, *the media player* has connection *to vehicle interface application* to get information such as speed and direction, time of a day, levels of noise and light. The higher level of contextual information is deduced based on basic contextual data. Based on deduced facts and rules within the *context management subsystem* the *context manager* recognizes that the vehicle is driven during the evening/night hours or in critical traffic conditions.

According to this information appropriate context profile is constructed which describes user interface for *Media Player* with night colours and necessary driving information widgets. Figure 7(a) shows the *Media Player* screen with the current song being played.

Figure 7(a): Media Player View

Following screenshot in figure 7(b) are taken at different levels of adaptation that the framework has performed autonomously in response to changing driver context.

Figure 7(b): Media Player View with Navigation Information

The navigation data is displayed on the screen with appropriate font size according to the speed of the user's vehicle. Also, appropriate zoom level for navigation pictogram is chosen with vehicle's location, the current street, is displayed from the view centre. In this manner the user is enabled to see the navigational information in front of him. Finally, the view contains remaining distance to maneuver as context reference.

The role of context subsystem of *ConFra* is to decrease the workload needed to operate *Media Player* and provide enough vehicle navigation information and therefore increase safety.

VII. Conclusion

Driving task is a highly complex behavior influenced by a large number of factors. Deployment of context aware infotainment systems in cars aims at improving driver's behavior. The development and deployment of In-vehicle infotainment applications motivate flexible and efficient HMI solutions with full context visibility and capable of proper handling of context modifications during application provisioning.

In this paper we presented the *context management subsystem* and *context infrastructure* of *ConFra*, the context aware HMI framework, to ease the creation of context-aware HMIs for In-vehicle infotainment applications. The subsystem is able to provide the necessary mechanisms to represent the different aspects of infotainment HMI context. The framework provides simple and efficient mechanism to present application and vehicle context at HMIs. The main features of *ConFra* are: (i) its simplicity from HMI development point of view, and (ii) flexible approach, as it uses context profile from applications.

The example presented here has shown that *ConFra* can simplify context aware HMI design and implementation, can provide effective service reconfiguration in response to runtime context changes, and can favor component reusability in different deployment conditions.

As part of our future work, we plan to extend the context management subsystem of *ConFra* to include:

- Support for context distribution, either at the level of the data acquisition framework, with interfaces gathering their raw data from infotainment applications, or more directly at the context domain level.
- support for storage, retrieval and querying of runtime context data, values and events so as to enable historical and statistical reasoning on the evolution of the context

REFERENCES

[1] Weiser, M.: The Computer for the Twenty-First Century. Scientific American (1991) pp.99-104.
[2] G. Chen and D. Kotz, A survey of context-aware mobile computing research, Technical Report TR2000-381, Computer Science Dep., Dartmouth College (2000).
[3] M. Sendín and J. Lorés, J., "Plasticity in Mobile Devices: a Dichotomic and Semantic View", Workshop Engineering Adaptive Web, supptd. by AH 2004, pp. 58-67, (2004).
[4] Kjeldskov, J. & Graham,C. (2003). A review of mobile HCI research methods. Mobile HCI 2003. Springer-Verlag, Berlin: 317-335
[5] Anind K. Dey, "Providing Architectural Support for Building Context-Aware Applications PhD thesis", College of Computing, Georgia Institute of Technology, December 2000.
[6] A. Chan, S. Chuang. MobiPADS "A Reflective Middleware for Context-Aware Mobile Computing", IEEE Transactions on Software Engineering, Vol. 29, No. 12, December 2003
[7] Szyperski, C.: Component Software: Beyond Object-Oriented Programming, 2nd edition. Addison-Wesley and ACM Press (2002)
[8] Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday, Experiences of developing and deploying a context-aware tourist guide: the GUIDE project, Proc. of MOBICOM'2000, Boston, ACM Press., 2000, pp. 20–31.
[9] Anind Dey and Gregory Abowd, Towards a better understanding of context and context-awareness, Conference on Human Factors in Computing Systems (CHI 2000):Workshop on theWhat,Who,Where,When and How of Context-Awareness (The Hague), April 2000.
[10] Paul Dourish, Where the action is: The foundation of embodied interaction, The MIT Press, 2001.
[11] David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste, Project Aura: Toward distraction-free pervasive computing, IEEE Pervasive computing (2002), 22–31.
[12] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy, Modeling context information in pervasive computing systems, 1st International Conference on Pervasive Computing (Zurich, Switzerland), Springer, August 26-28 2002, pp. 167–180.
[13] Patil, S. and J. Lai. Who Gets to Know What When: Configuring Privacy Preferences in an Awareness Application. In Proceedings of CHI 2005.
[14] E. Chung, J. I. Hong, J. Lin, M. K. Prabaker, J. A. Landay, and A. Liu. Development and Evaluation of Emerging Design Patterns for Ubiquitous Computing. In Proceedings of Designing Interactive Systems (DIS2004), 2004.