

Rethinking context frameworks

Enabling cross-domain, cross-device experiences

Daniel Stewart and Nitya Narasimhan

Abstract—Seamless mobility entails providing an optimal user-centric experience across multiple domains and devices. Context-awareness is integral to this vision. However, existing mechanisms are often tailored for specific domains (e.g., smart spaces) or target a few specific parameters (e.g., location-based services). We argue that such solutions tend to isolate context-aware frameworks rather than unify them. Instead, we advocate having a more “integrated” context framework that can span different domains and applications, support a diverse range of context parameters and that can be scaled (up or down) to suit the host platform constraints and fulfill the needs of applications for seamless mobility. In this paper, we seek to motivate a discussion on the need for such a framework, set realistic objectives on what we can achieve and identify the core challenges that need to be addressed to make such a solution viable.

Index Terms—context-awareness, pervasive computing, event frameworks, overlay experiences, cross-domain interactions

I. INTRODUCTION

SEAMLESS MOBILITY involves providing users an optimal experience as they move between devices or across spaces. Enabling seamless mobility requires mechanisms such as context-awareness that help provide *task continuity* with user mobility. The mobile device plays a key role in this vision. It is not only most representative of the user (personal vs. shared device) but is also a rich context source since it accompanies the user everywhere and is aware of who/what interactions and activities the user engages in. However, to provide a truly seamless experience, we need both to gather this context information from multiple sources and to share relevant context between them in a transparent manner. For instance, if we consider “music listening” as the activity, then the user’s context for listening to music (or to a particular music item) can be understood best only if we gather such context across the various devices on which that user potentially consumes music (e.g., PC, phone, portable music player, car) and then mine the aggregated data.

This motivates the need for a cross-device, cross-domain framework for context-awareness. Much research in context-awareness [1] focuses either on comprehensive middleware (e.g., ontology-driven) for specific domains like smart spaces, or on rich applications (e.g., location-based services) that use a finite set of measurable parameters (e.g., location, time,

presence). These approaches fall short when applied to our seamless mobility vision. In fact, dedicated solutions for well-defined domains or applications can create an isolating effect. They can blind applications to runtime context that is relevant, but unknown to the application designer. As a result, they often require users to perform the data transfer (or translation) across devices. This makes for poor user experiences.

II. AN INTEGRATED CONTEXT FRAMEWORK

Instead we advocate having a more “integrated” approach to developing context frameworks. We envision each device hosting a *local* context repository, with mechanisms to enable context sharing across device repositories. The goal is then to enable these devices (and applications) to leverage collective context to create richer (and more intelligent) applications.

We want such an integrated framework to span a wide range of devices from resource-poor mobiles to resource-rich PCs. The framework should also support *any kind of context* information that exists. This includes standard ideas of context (location, time, proximity) as well as new types of application- or device-created context (e.g., recent call history, last song played). We believe this vision is in keeping with previous definitions of context [2, 3, 4] that included such {physical, informational, activity, social} themes. In addition, such a framework should also enable other applications to see, understand and use this new context without having any *a priori* knowledge about the source application..

So, how do we collect and share context information across a wide range of resource-diverse devices? We can see design challenges arising in several areas:

- **Defining context.** We need a *context representation* format that is simple (supported by all devices) but also extensible. Note that not all devices view the same context ‘type’ in the same way – e.g., “location” can imply GPS on one device and cell ID on another. Thus, we not only need a way to accommodate various interpretations of a context type – but also to enable their usage as a single entity (e.g., accessible under a single query for location)
- **Gathering context.** Unfortunately, many applications are developed in “silos” (lock-in context) and share data only in tightly-coupled situations. Instead we need a *context repository* providing a shared pool that any application can access as desired. This has the added value of reducing redundancy (reuse, not re-create) and reinforcing

the value of individual context items.

- **Storing context.** While context is often perceived as having real-time use, researchers [3, 5] also advocate storing context for different periods of time to create added value. Such *context histories* enable both “suggestive” behaviors (e.g., use recent history to suggest context relevant to current activity) and “predictive” behaviors (e.g., use historical activity patterns to predict context of interest, and potentially pre-provision devices with that information) that reduce the input and cognitive burden on users – particularly mobile ones.
- **Sharing context.** Cross-device, cross-domain sharing requires interaction protocols and network availability. However, many devices are intermittently connected to each other – either because they support wide-area networking (rely on coverage), or because they support local-area networking (rely on proximity). We need mechanisms that can *tolerate delays in context sharing* when correlating context related to a specific activity. Additionally, network usage will incur both monetary and resource costs (e.g., battery) that may be unacceptable to users. The framework will need to enable various trade-offs between cost and convenience in context-sharing.
- **Using context.** We need *simple APIs* with consistent (not necessarily identical) interfaces across all devices that will enable applications to perform simple operations – {create, modify and delete} context, {query} the repository, {register} for change notifications (including creation and deletion) and {associate} contextually-related items, irrespective of who created the items. Note that devices may elect to implement only a subset of these interfaces, and may choose to expose only a subset of these functions to the application.

III. EARLY THOUGHTS ON IMPLEMENTATION

These were some of the issues that influenced our initial design of an integrated framework [6] spanning the home (desktop PC, SetTop Box) and the mobile (phone) domains. Support for other devices (in other domains) is planned. We started by assuming the existence of one context repository per device. This repository was modeled along the lines of a **shared blackboard** – any device can add, view and modify the data contents. To remain lightweight, we favored a simple model where context items are represented by a type field, a list of attribute name-value pairs, and an optional payload blob. The type field is a dotted string (like a java package) signifying the kind of context element this object represents. Attribute names are defined by strings, with the corresponding values defined as common base types (strings, integers, floats, and booleans). If a payload exists, there is a MIME-type associated with it to indicate the encoding the payload. Context can be queried using the type field or any attribute name. However, the payload is an opaque entity that cannot be queried directly, but could be unpacked and used by any knowledgeable applications to extract richer data.

We also envision repositories being of two types – transient repositories whose contents are held in memory, and persistent repositories whose contents are written to a database or file. The transient repositories best represent “short term memory” for suggestive behaviors, while the persistent repositories reflect “long term memory” for predictive behaviors. Note that this distinction is transparent to the application; it only affects the scope of queries made through the existing interface. A resource-rich device could support both types locally; a resource-poor device typically supports only short-term memory locally, but can potentially connect up to a remote networked long-term memory store to sustain the illusion of a much larger context memory. In our current implementation, these behaviors are embodied by two components – the AppBus [7] running on the mobile phone, and the Home and Personal Events framework (HoPE) [8] running on a home device (desktop PC or set-top box). More details on these frameworks and on the design and usage of the integrated context framework can be found in [6].

IV. DISCUSSION

We would like to motivate a discussion on a few aspects of this work – its value to the pervasive computing community, its viability from a device and networks perspective, its utility from a usability and user experience viewpoint, extensibility to support enhancements to the basic framework (e.g., to accommodate existence of richer devices), design for reuse (i.e., can we design a core platform on which more comprehensive domain-specific or device-specific frameworks can be mapped) – and finally, identification of other pertinent issues or requirements (e.g., security and privacy).

REFERENCES

- [1] G. Chen and D. Kotz, “A survey of context-aware mobile computing research”, Technical Report TR2000-381, Dartmouth College, 2000.
- [2] B. Schilit, N. Adams and R. Want, “Context-aware computing applications”, in Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, 1994.
- [3] T. Prante, R. Stenzel, K. Petrovic and V. Bayon, “A Cross-Tool and Cross-Device approach to reduce compartmentalization when going back”, in Proceedings of the 1st International Workshop on Exploring Context Histories in Smart Environments (ECHISE), May 2005
- [4] A. Dey and G. Abowd, “Toward a better understanding of context and context-awareness”, in the Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands, April 3, 2000. Also Technical Report GIT-GVU-99-22.
- [5] D. Salber and G. Abowd, “The design and use of a generic context server”, in the Proceedings of the Perceptual User Interfaces Workshop (PUI '98), San Francisco, CA, November 5-6, 1998. pp. 63-66.
- [6] D. Stewart and N. Narasimhan, “The Integrated Context Framework”, Motorola Technical Report, Jan 2007 (in-progress)
- [7] C. Janssen, M. Pearce, S. Kollipara and N. Narasimhan, “AppBus: Mobile device application collaboration via short term memory”, Journal of Software, vol. 1., no. 2, pp 40-47, August 2006.
- [8] P. Eren, D. Stewart and J. Wodka, “Home and Person Focused Event Framework”, in the Proceedings of the International Symposium on Applications and the Internet Workshops (SAINT 2006), Phoenix, AZ, January 2006, pp. 23-27.