

Hermes: a Context-Aware Application Development Framework for the Mobile Environment

Senaka Buthpitiya, Heng-Tze Cheng, Feng-Tso Sun, Martin Griss, Anind K. Dey
Carnegie Mellon University

1. INTRODUCTION

This paper describes a next generation software toolkit which provides a framework, including security and intelligibility support, for developing more powerful context-aware applications for the modern mobile environment.

Modern mobile computing environments, with reasonable connectivity, processing power and sensing capabilities on portable devices provide a means of achieving truly ubiquitous context-aware computing. To application developers, the mobile environment also presents a new and unique set of challenges:

- 1) **Limited resources:** Restrictions on communication, data storage, power, etc. that vary for each device in the ecosystem.
- 2) **Large number of information sources:** Fusing, filtering and regulating the flow of context information from various sources (sensors, online information sources, etc) bring further overheads to the context-aware application developer's task.
- 3) **Increased security risk:** Components/information sources of applications are spread out on physically separated devices, some of which may not be secure.
- 4) **Usability issues:** Context-aware applications are becoming extremely complex where the reasons and logic behind proactive application actions may not be intelligible to users causing users to mistrust applications and hinder adoption.
- 5) **Component and information reuse:** Context-aware applications should leverage the information derived by other applications which requires a uniform method of describing context information to allow dynamic application integration.

In previous work, many frameworks and toolkits for creating context-aware applications have been developed [1, 2] for relatively fixed environments and none of these address the specific challenges, opportunities and constraints presented by context-aware mobile devices and applications, especially those of security and intelligibility. The Context-Aware Computing Engine [1] is a heavyweight centralized framework for aggregating and transforming context information which assumes continuous connectivity between components. In [2], the authors describe a widget based toolkit for context-aware application development. Although these widgets are too heavy for a mobile environment and the toolkit lacks support for many aspects of a mobile environment, it provided the inspiration for this work.

2. ARCHITECTURE

The Hermes framework is designed around a concept of using context widgets as the basic building block. We define the widgets as light-weight program modules that are responsible for a single type of context information (e.g. location, activity) belonging to a single entity (any place, person or object). Widgets can exist on devices or in the cloud or on the telecommunication infrastructure. To reduce the overall *weight* of the widgets, functionality common across all widgets on a device are extracted into a common lower layer (Figure 1), i.e., the Shared Hermes Service Infrastructure (SHSI). A common SHSI layer also reduces communication costs and possibly provides efficient storage. Widgets connect to other widgets forming an ad-hoc widget network, which in most scenarios would have a tree like topology (Figure 1). The framework provides a protocol and ontology by

which a widget can discover other widgets that are able to provide the type of context information it requires and legibly exchange information.

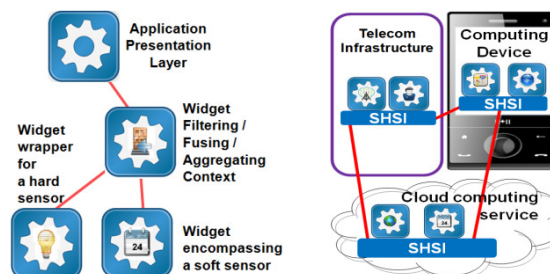


Figure 1. Left: a generic Widget-tree. Right: division of widgets across platforms

Security: Each widget provides security guarantees that allow the application as a whole to ensure a higher level of security. Widget security is focused on 1) authentication of other widgets, 2) ensuring widget integrity, and 3) securing communication between widgets. The protocol designed for this purpose operates without a central authentication server at run-time. It allows widgets to check each other's composition directly and match it against a verified design-time signature.

Intelligibility: A Widget can *explain* a piece of context information it produced by keeping track of the raw context information (and their source widgets) used to generate its output. Widgets could either generate an explanation directly or pass a request down the widget-tree; which returns fragments of explanation that are composed into a coherent explanation.

3. PROGRESS & FUTURE WORK

We have completed the design for the overall architecture, implemented prototype widgets and SHSI layers, and developed the security protocols. In parallel, we use the toolkit to build a variety of mobile context-aware applications. The *field experience* allows us to further refine the toolkit APIs.

We plan to investigate the possibility of designing a uniform and extendable model for representing context allowing widgets developed by various parties to connect and exchange context information without manual intervention or prior knowledge of other widgets. Development has just begun on intelligibility support, involving methods to allow developers to describe a widget's internal logic and techniques to concatenate explanations from various widgets.

4. REFERENCES

- [1] Carrizo, C., Hatakar, A., Memmott, L., Wood, M. 2008. Design of a Context Aware Computing Engine. In Proceedings of the IET 4th International Conference on Intelligent Environments (Seattle, WA, July 21 – 22, 2008). IE'08. IEEE Computer Society Press, 1-4.
- [2] Dey, A. K., Abowd, G. D., Salber, D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In Human-Computer Interaction. L. Erlbaum Associates Inc., 97-166.