Poster: HealthOS: A Platform for Integrating and Developing Pervasive Healthcare Applications

Jong Hyun Lim, Andong Zhan, Andreas Terzis
Johns Hopkins University
ljh@cs.jhu.edu, zad@cs.ljh.edu, terzis@cs.jhu.edu

Categories and Subject Descriptors: D.2.11 Software Engineering: Software Architectures

General Terms: Design, Management, Standardization

Keywords: Interoperability, Composability, Management, RESTful, Pervasive Healthcare Applications

1. THE HEALTHOS SYSTEM

Recently, an increasing number of pervasive healthcare applications have been developed as a way to overcome the shortcomings of the traditional clinical infrastructure. However, the nature of these applications, both closed and vertically-integrated, hinders integration with the existing infrastructure, increases the development cost, and fails to provide a unified management interface.

In response to these challenges, we propose HealthOS, a platform designed to develop pervasive healthcare applications. Figure 1 illustrates the target environment for HealthOS. As the figure suggests, HealthOS users can carry multiple healthcare-related devices in their living environments, each using their proprietary communication protocols and data formats. HealthOS collects, encrypts, and stores the data on either a local machine or in a secure cloud service. Upon request, HealthOS can translate the data into requested formats that different healthcare applications may require. From the perspective of an application developer, the attractiveness of the HealthOS platform lies in the need to implement only the analysis and representation logic of the application. We also envision the use of a *HealthStore* (similar to the Apple AppStore), in which pre-developed applications can be shared and reused. The various applications are handled using the unified management console in HealthOS. Furthermore, HealthOS can be adapted to mobile platforms.

Achieving our vision of HealthOS, requires implementing several modules for each device. First, a module is necessary to communicate with each device and translate the custom data that it produces. We call this module a HealthOS *driver*. Secondly, the formats used to present the data to different stakeholders (e.g., family members, healthcare professionals, etc.) may differ. Thus, HealthOS uses a *translator* module used for converting a device's proprietary data format to match well-defined, standardized medical data presentation formats used by major electronic medical record (EMR) systems [2, 6]. Both *drivers* and *translators* follow a component-based design in the sense that they provide and require well-defined interfaces for inter-module interaction and reuse.

Given that data can successfully be collected and interpreted using HealthOS, application developers can focus solely on properly analyzing and presenting the acquired data. To do so, HealthOS provides RESTful interfaces through which healthcare applications can access and manipulate the collected data [4]. Furthermore, we expose such interfaces in the form of software libraries.

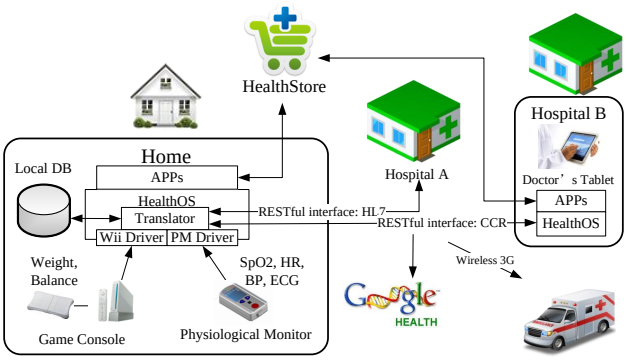

Figure 1: Target scenario for HealthOS. HealthOS (1) enables inter-operation between heterogeneous systems, (2) provides programming interfaces, and (3) unifies the management mechanism.

Since HealthOS deals with private sensitive information, it is required to encrypt medical data before storing them, not to mention when transmitting them as a way to prevent security threats [1]. However, encrypting data with per-user keys is not a scalable solution. To overcome this issue, HealthOS incorporates the CP-ABE mechanism to provide access policies for the data it manages [3]. Specifically, CP-ABE embeds the access policy of legitimate users within the keys so that only the users corresponding to the policy are able to decrypt the cipher text. Finally, HealthOS provides a unified interface that takes the role of managing the installed applications, controlling user access and extending the storage capabilities to publicly opened cloud services such as Google Health [5].

2. REFERENCES

[1] 104th Congress. Health Insurance Portability and Accountability Act of 1996. Public Law 104-191, 1996.

[2] ASTM Standard E2369, 05e1. Standard Specification for Continuity of Care Record (CCR), 2005.

[3] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[4] Roy Thomas Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000.

[5] Google. Google Health. `http://www.google.com/health`.

[6] Health Level 7. HL7 Version 3. `http://www.hl7.org/about/`, 2008.

MobiSys'11, June 28–July 1, 2011, Bethesda, Maryland, USA.
ACM 978-1-4503-0643-0/11/06.