Platys: A Framework for Supporting Context-Aware Personal Agents

(Demonstration)

Pradeep K. Murukannaiah, Ricard Fogues, and Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
{pmuruka,rlopezf,singh}@ncsu.edu

ABSTRACT

A context-aware personal agent (CPA) adapts to the changing contexts of its user. Platys is an agent-oriented software engineering (AOSE) framework that supports the development and execution of CPAs. Specifically, the framework (1) facilitates modeling a CPA via cognitive constructs, simplifying development, and (2) delegates the concerns of context elicitation (from end users) and acquisition (from sensors) to a middleware, enhancing reusability and user experience as a user employs multiple CPAs.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements and Specifications—*Tools*

Keywords

Middleware; Context; Personal agent

1. INTRODUCTION

A context-aware personal agent (CPA) relies upon knowing its user's *context*, loosely referring to the user's whereabouts, actions, and interactions. CPAs arise in application domains such as healthcare, entertainment, and smart (physical or virtual) environments . Despite a growing body of research on context-aware computing [1], there is little practical support for systematically developing a CPA and putting it to use.

We describe and demonstrate[1] Platys, a framework that supports both development and execution of CPAs. Platys treats context as a cognitive construct in par with other cognitive constructs such as agents, and their goals and plans. Platys facilitates CPA development on the principled foundation of agent-oriented software engineering (AOSE) [2, 3, 5]. Figure 1 shows the major components of the Platys framework, which supports CPAs as follows.

[1]Video at `http://youtu.be/InwLFfmnCBQ`

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

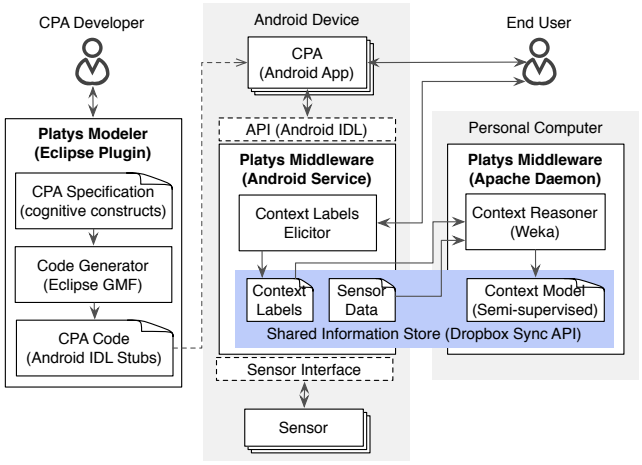


Figure 1: Platys framework includes a modeling tool and one or more middleware instances at run time.

Development: Platys *modeler* assists a developer in modeling a CPA via cognitive constructs and derive a high-level specification. In doing so, the developer can systematically follow an AOSE methodology. From the high-level specification, the modeler generates context acquisition code, which can be executed on a target platform.

Execution: Platys *middleware* provides runtime support for CPAs. Specifically, it hides from a CPA the nuances of reasoning about high-level context abstractions from low-level sensor data. Also, the middleware elicits a subjective context model from an end user and reuses it across CPAs (governing privacy policies), potentially enhancing the overall user experience (of employing CPAs).

2. PLATYS MODELER

Platys modeler is a tool for graphical modeling of a CPA. In addition to primitives such as goals, plans, and dependencies, Platys modeler supports contextual beliefs as a means of supporting runtime adaption. Specifically, Platys modeler supports the primitives described by the Xipho methodology [3]. Figure 2 shows an example model of *Ringer Manager Agent (RMA)*, a CPA that automatically configures the ringer mode of a user's smart phone.

Model validation: Platys modeler validates CPA models. Specifically, Platys verifies that conflicting goals, and de-

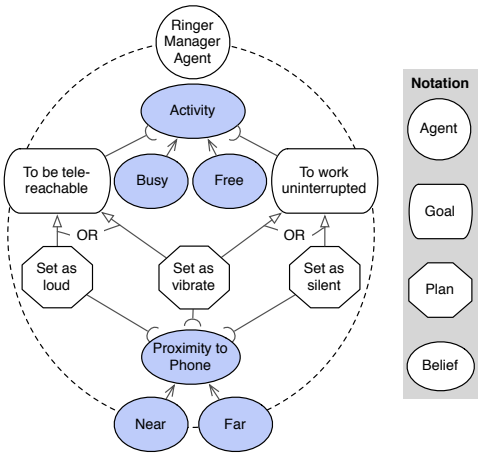
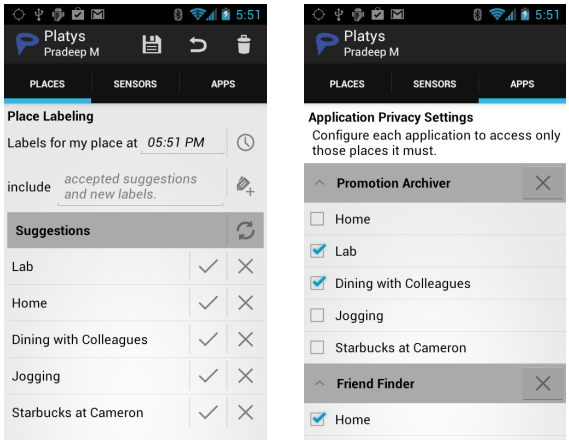


Figure 2: A goal model of the RMA.



(a) Context labeling. **(b) Configuring CPA.**

Figure 3: Platys middleware screenshots (Android).

composed goals and plans have a basis (contextual beliefs or resources) for runtime resolution.

Code generation: A valid CPA model can be specified as a set of contextual capabilities (i.e., a rule of the form $Activity = busy \wedge Proximity = Near \rightarrow Set\ as\ silent$ in Figure 2). Platys modeler translates such capabilities into code executable on a target platform (usually consisting of callbacks from the middleware).

3. PLATYS MIDDLEWARE

The Platys execution environment consists of (1) CPAs, relying upon knowing a user's contextual beliefs (high-level abstractions), (2) sensors, providing low-level contextual cues, and (3) Platys middleware, mapping sensor data to contextual beliefs. As shown in Figure 1, there can be multiple instances for each of these components, distributed across physical devices. However, there must be an instance of the middleware on each physical device a user employs. Mobile devices are ideal for sensing and hosting CPAs as they are always with a user, whereas computationally intensive components such as the context reasoner can be hosted on the user's personal computer.

Platys middleware performs the following tasks.

Provides an API for CPAs to query a user's context at the desired levels of abstraction (e.g., RMA needs to know if a user is *busy*). Instead of periodically polling the middleware, a CPA registers a callback with middleware, providing a stub the middleware can execute to invoke the CPA's contextual capability (e.g., *set as loud*). Figure 3b shows a screenshot from a middleware instance showing a CPA's capabilities and desired context abstractions.

Learns to recognize contexts from low-level sensor data. Platys seeks to learn a user's contexts in a user-centric manner (contextual beliefs, e.g., *busy* and *quiet*, are inherently subjective). The middleware elicits context labels from each end user as shown in Figure 3a. Further, the middleware employs *active learning* [4] for eliciting context labels, requiring a user to label only those contexts that the middleware is least confident of predicting (from historical sensor data and labels) correctly.

4. TECHNOLOGY

Platys modeler is an Eclipse plugin built on the Eclipse Graphical Modeling Framework (GMF). The plugin contains a graphical editor and the model validator. A stand alone Java module is used to auto-generate executable code (stubs in Android Interface Definition Language).

Platys middleware is implemented for (1) Android devices, where it runs as a service, and (2) personal computers, where it runs as an Apache daemon (requires Java; platform independent). The middleware instances communicate asynchronously via an application-private Dropbox folder (using Dropbox API), transparent to the user. The middleware employs GPS, Accelerometer, WiFi and Bluetooth sensors, and data from user interactions (email, text, and call logs). The reasoner employs Weka for machine learning. Importantly, the middleware is privacy preserving in that it does not require the user to reveal sensor data or context labels to a third party.

Acknowledgments

We thank the National Science Foundation for partial support under grant 0910868.

5. REFERENCES

[1] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Perv. Mob. Comput.*, 6(2):161–180, Apr. 2010.

[2] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *JAAMAS*, 8(3):203–236, May 2004.

[3] P. K. Murukannaiah and M. P. Singh. Xipho: Extending Tropos to Engineer Context-Aware Personal Agents. In *Proc. AAMAS*, 2014.

[4] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2012.

[5] M. Winikoff and L. Padgham. *Developing Intelligent Agent Systems: A Practical Guide*. Wiley, Chichester, UK, 2004.