Koko: Engineering Affective Applications

Derek J. Sollenberger
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
djsollen@ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; H.1.2 [**Models and Principles**]: User/Machine Systems

General Terms

Design, Human Factors

Keywords

Koko, affect, emotion, affective social applications, booST, mobile

1. INTRODUCTION

Representing and reasoning about affect (i.e. emotion) is essential for producing believable characters and empathic interactions with users, both of which are necessary for effective entertainment and education. Leading applications of interest include pedagogical tools [1], military training simulations [2], and educational games [3].

Our work, like the research of the previously mentioned applications, is based on the concepts of appraisal theory. A fundamental concept of appraisal theory is that the environment of the agent is essential to determining the agent's affective state. As such, appraisal theory yields models of affect that are tied to a particular domain with a defined context. Therefore, each new problem domain requires a new instance of the model. A current and common practice has been to copy and edit a previous application (and, occasionally, to build from scratch) to meet the specifications of a new domain. This approach may be reasonable for research proofs-of-concept, but is not suitable for developing production applications.

Additionally, many affective applications use physical sensors that provide additional information about the user and the user's environment. The number and variety of sensors continues to increase and they are now available via a variety of public services (e.g., a weather service) and personal commercial devices (e.g., GPS in mobile phones). Current approaches require each affective application to interface with these sensors directly. This is not only tedious, but also nontrivial as the application must be adjusted whenever the set of sensors changes.

It is for these purposes that we introduce Koko. Koko is not a new model of emotions, but a middleware from which existing (and future) models of affect can operate within. It provides the means for both application developers and affect model designers to construct their respective software independently, while giving them access to new features that were before impossible. Further, Koko is intended to be used by affective models and applications that seek to recognize emotion in a human user. While it is possible to use Koko in systems that model emotion in virtual characters, many benefits, such as using physical sensors, most naturally apply when human users are involved.

Cite as: Koko: Engineering Affective Applications, Derek J. Sollenberger, Munindar P. Singh, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1423 – 1424

2. BENEFITS

Koko concentrates on providing two core benefits to affective application developers which are increasing developer productivity and providing an enhanced user experience. We briefly elaborate on the merits of each in the remainder of the section.

Developer Productivity. Koko separates the responsibility of developing an application from that of creating and maintaining the affect model. In Koko, the application logic and the affect model are treated as separate entities. By creating this separation we can in many cases completely shield one entity from the other and provide standardized interfaces at the required points of interaction.

Additionally, Koko avoids code duplication by identifying and separating modules for accessing affect models and various sensors, and then absorbing those modules into Koko. For example, by extracting the interfaces for physical sensors into the middleware, Koko enables each sensor to be leveraged through a common interface. Further, since the sensors are independent of the affect models and applications, a sensor that is used by one model or application can be used by any other model or application without the need for additional code.

Quality of User Experience. Abstracting affect models into Koko serendipitously serves another important purpose. It enables an enhanced user experience by providing data to both the application and its affect model that was previously unattainable, resulting in richer applications and more comprehensive models.

With current techniques it is simply not possible for separate applications to share affective information for their common users. This is because each application is independent and thereby unaware of other applications in use by that user. By contrast Koko-based applications—even those authored by different developers—can share common affective data through Koko. This reduces each application's overhead of initializing the user's affective state for each session, as well as provides insight into a user's affective state that would normally be outside the application's scope.

3. ARCHITECTURE

Existing affective applications tend to be monolithic where the affective model, external sensors, and application logic are all tightly coupled. As in any other software engineering endeavor, monolithic designs yield poor reusability and maintainability. Similar observations have led to other advances in software architecture [4]. Most pertinently, the idea of a knowledge base being separate from a solution strategy is motivated on such grounds.

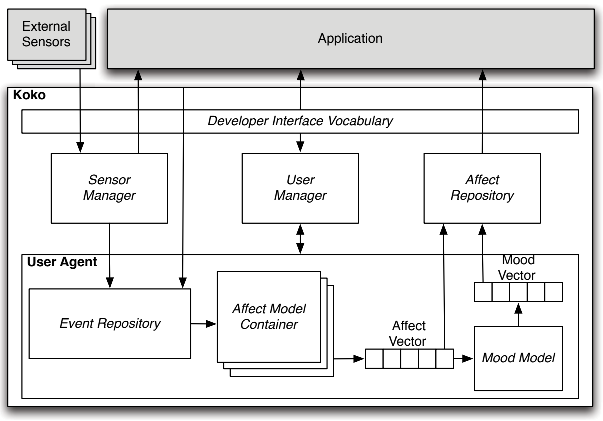

Figure 1: Koko architectural overview

Figure 1 shows Koko's general architecture using arrows to represent data flow. Next we will briefly discuss the key component of the middleware on which everything else is dependent upon: the user agents.

Koko hosts an active computational entity or *agent* for each user. In particular, there is one agent for a human, who may use multiple Koko-based applications. Each agent has access to global resources such as sensors and messaging but operates autonomously with respect to other agents.

The affect model container manages the affect model(s) for each agent. Each application must specify exactly one affect model, whose instance is then managed by the container. The affect models take in information about the agent's environment and produce an affect vector, which contains the probabilities that the agent is in a certain emotional state. The affect models used within Koko follow a supervised machine learning approach [3] to modeling affect by populating predictive data structures with affective knowledge. By following this approach the affect models can be configured at runtime which enables us to maintain a domain-independent architecture with domain-dependent affect model instances. The affect models are built, executed, and maintained by the container using the Weka toolkit [5]. Further, the two standard affect models that Koko provides use Naive Bayes and decision trees as their underlying data structures.

4. CASE STUDY

The subject of our case study is a social, physical health application with affective capabilities, called booST. To operate, booST requires a mobile phone that is running the Android mobile operating system and is equipped with a GPS sensor. Notice, that while booST does take advantage of the fact that the sensor and application run on the same device this is not a restriction that is imposed by Koko.

The purpose of booST is to promote positive physical behavior in young adults by enhancing a social network with affective capabilities and interactive activities. As such, booST utilizes traditional social networking tools, such as MySpace and Facebook, to provide support for typical social functions such as maintaining a profile, managing a social circle, and sending and receiving messages. Where booST departs from traditional social applications is in the use of *energy levels* and *emotional status*.

Each user is assigned an *energy level* that is computed using simple heuristics from data retrieved from the GPS. Additionally, each user is assigned an *emotional status* generated from the affect vectors retrieved from Koko. The *emotional status* is represented as a number ranging from 1 to 10. The higher the number the happier the individual is reported to be. A user's *energy level* and *emotional status* are made available to both the user and members of the user's social circle.

To promote positive physical behavior, booST supports interactive physical activities among members of a user's social circle. The activities are classified as either competitive or cooperative. Both types of activities use the GPS to determine the user's progress toward achieving the activities goal. The difference between a competitive activity and a cooperative activity is that in a competitive activity the user to first reach the goal is the winner, whereas in a cooperative activity both parties must reach the goal in order for them to win.

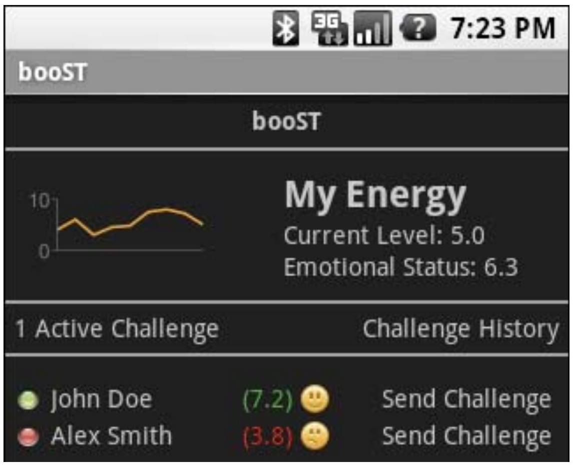
7:23 PM
booST
booST
My Energy
Current Level: 5.0
Emotional Status: 6.3
1 Active Challenge — Challenge History
John Doe (7.2) Send Challenge
Alex Smith (3.8) Send Challenge

Figure 2: booST buddy list

Koko's primary role in booST is to maintain the affect model that is used to generate the user's *emotional status*. booST simply provides the data about its environment, which in this case is the user's social interactions on Facebook and their participation in the booST activities. Koko passes this information to the appropriate user agent who processes the data and returns the appropriate *emotional status*.

5. REFERENCES

[1] C. Elliott, J. Rickel, and J. Lester. Lifelike pedagogical agents and affective computing: An exploratory synthesis. In *Artificial Intelligence Today, LNAI*, 1600:195–212, 1999.

[2] J. Gratch and S. Marsella. Fight the way you train: The role and limits of emotions in training for combat. *Brown Journal of World Affairs*, Vol X (1), Summer/Fall:63–76, 2003.

[3] S. McQuiggan and J. Lester. Modeling and evaluating empathy in embodied companion agents. *International Journal of Human-Computer Studies*, 65(4), 2007.

[4] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall, 1996.

[5] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.