Koko: Architecture and Methodology for Engineering Social Affective Applications

(Extended Abstract)

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.2.10 [**Software Engineering**]: Design—*methodologies*; 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, mood, affective social applications

1. INTRODUCTION

Representing and reasoning about affect (e.g. emotion and mood) 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.

Cite as: Koko: Architecture and Methodology for Engineering Social Affective Applications, (Extended Abstract), 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. 1137–1138

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.

2. BENEFITS

Koko concentrates on providing three core benefits to affective application developers, which are outlined below.

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.

Affective Social Applications. Importantly, Koko enables the development of affective social applications by introducing expressive communicative acts into agent-oriented software engineering. Using the multiagent environment provided by Koko, agents are

able to communicate affective information through the exchange of expressives. The communication of affective information is well known in the philosophy of language as an expressive communicative act [4]. However, expressive acts are a novelty in both agent-oriented software engineering and virtual agent systems, which have traditionally focused on other communicative acts.

Further, Koko enables us to create an affective, social methodology (ASM) centered on expressive communicative acts, the first such methodology to our knowledge. Using this methodology application developers can construct applications which are both social and affective.

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 [5]. Most pertinently, the idea of a knowledge base being separate from a solution strategy is motivated on such grounds.

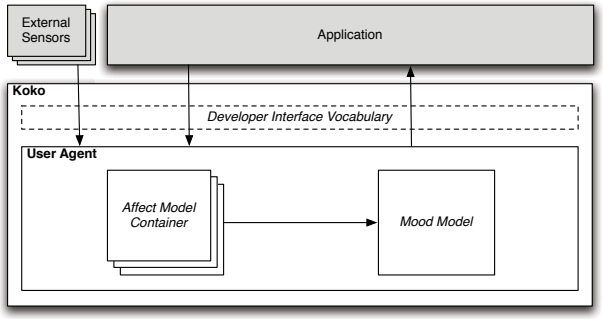

Figure 1: Koko basic architectural overview

Figure 1 provides a basic overview of Koko's architecture using arrows to represent data flow. Next, we briefly discuss the three components of Koko on which everything else is dependent upon: the user agents and their associated affect and mood models.

Koko hosts an active computational entity or *agent* for each user. In particular, there is one agent per 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 Koko to maintain a domain-independent architecture with domain-dependent affect model instances.

We define *emotion* as the outcome of one or more specific events and a *mood* as a longer lasting aggregation of the emotions for a specific user. For simplicity, Koko's model of mood takes in the output of the user's affect models and produces a *mood vector*, which includes an entry for each emotional state that Koko is modeling for that user. Each entry represents the aggregate intensity of the emotion from all affect models associated with that user. Consequently, if Koko is modeling more than one application for a given user, then the user's mood is a cross-application measurement of the user's emotional state.

4. METHODOLOGY

Koko's multiagent environment enables the development of applications which are both affective and social. To aide in the development of these applications we introduce Koko-ASM. Koko-ASM is a methodology for building affective, social (multiagent) applications that leverage Koko.

Koko-ASM focuses on the expressive communication between agents. Expressives originate from speech-act theory where they are defined as communicative actions that enable a speaker to express their attitudes and emotions towards a proposition [4] .

The methodology guides the developer through the process of identifying the expressives in the application. The identified expressives are then used to determine the key information needed to integrate the application with Koko. For example, one step of the methodology uses the identified expressives to determine which emotional states (e.g. hope, fear) should be modeled within Koko. The final result of the methodology is an affective, social application, which leverages Koko in order to maintain the affect model for each agent and communicate the expressives among agents.

5. CONCLUSIONS

Architectural improvements can have scientific significance. For example, separating knowledge bases from problem solving not only improved developer productivity, but also led to greater clarity and improvements in the concepts of knowledge representation and problem solving. Koko takes similar steps with respect to affect, especially by supporting expressive communication among agents.

Further, Koko-ASM provides a means by which developers can create applications which are both social and affective. We have conducted a case study to demonstrate the usefulness of the methodology. The subject of our 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.

The purpose of booST is to promote positive physical behavior in young adults by enhancing a social network with affective capabilities and interactive activities. To promote positive physical behavior, booST supports interactive physical activities among members of a user's social circle. The activities use the GPS to determine the user's progress toward achieving the activities goal.

Where booST departs from traditional social applications is in the use of *emotional status*. 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 (sad) to 10 (happy). Koko ensures that a user's *emotional status* is made available to both the user's agent, as well as the agents representing members of the user's social circle.

6. 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] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1970.

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