OpenAffect API: a proposal for enabling an ecosystem of emotion awareness tools

Olivier Liechti
University of Applied Sciences and Arts Western Switzerland
Yverdon-les-Bains, Switzerland
olivier.liechti@heig-vd.ch

Rodney Reis
Avalia Systems
Yverdon-les-Bains, Switzerland
rodney@avalia.systems

I. Introduction

We propose an affective API designed with a focus on agile development. The OpenAffect API enables different types of applications to produce and consume affective measurements. At the workshop, we will present the details of the REST API and illustrate its use with a demonstration.

Our motivation to standardize the capture of affective state is twofold. Firstly, from an agile practitioner's perspective, we know how critical it is to maintain awareness about the complex interplay between cultural values, work practices, emotions, motivation, software quality and team performance. While one-to-one and group discussions help maintain this awareness, they do not scale and should be complemented by other, automated means. Secondly, from a scientific perspective, we see a growing interest in mining software repositories to study the same questions at scale. So far, studies had to rely either on *a posteriori* manual labelling or on machine learning to extract emotions from software artifacts. Future studies would benefit from normalized affective measurements captured in the context of daily activities.

II. Use cases

Agile professionals are engaged in all sorts of activities: they capture requirements, write code, fix bugs, have meetings, etc. Through these activities, they produce artefacts: source files, documents, etc. They also produce metadata to describe the activities and the artefacts: messages in git commits are metadata about code changes, comments on bug reports are metadata about the bug fixing activity. The activities occur in different temporal cycles, marked by ceremonies. People have emotions before, during and after performing the activities. They have emotions about the evolving state of artefacts. They have emotions when thinking about specific cycles, such as the *coming week*, the *last hour* or the *current sprint*. The following use cases illustrate how different tools could be used to capture affective state and benefit from a unified API. The list is obviously not exhaustive.

A. Report affective state when getting a bug report

Bob has been assigned an issue to fix. In the bug tracking system, he does not clearly understand the problem and does not have instructions for reproducing it. That makes him *angry* and *worried*. In the UI, it takes him a couple of seconds to label his comments with two emoticons. The bug tracking system sends a measure to the OpenAffect server.

B. Report affective state when submitting a pull request

Alice has completed the implementation of a feature. She feels *proud* about her work and *confident* that the code is solid and well tested. When she submits the pull request, she uses tags to report her feelings. The OpenAffect server is notified.

C. Report affective state when starting a code review

Carlos has received a request to review some code three days ago. He is doing too many things in parallel, is late and feels *stressed*. He will only have the time to do a quick review and feels a bit *guilty* about it. He uses tags to record his affective state. The code review tool sends data to the OpenAffect server.

D. Report affective state about temporal cycles

Sacha is using a *quantified self* mobile app. In the UI, he can regularly record how he feels about cycles such as the coming day, the past week, the past sprint, etc. If Sacha does not report affective state for some time, he is prompted for the same information via instant messaging, by a bot. The mobile app and the bot are OpenAffect clients.

E. Report affective state during a planning meeting

The team is planning the next iteration. Everybody is connected to a collaborative web app. Stories can be moved from the backlog to the center to be discussed. While people speak, they can push buttons to express how they feel about the story. Their feedback is sent to the OpenAffect server.

F. Trigger introspective discussions in the team

The data collected across tools is processed, aggregated and presented in an analytics dashboard. During retrospective meetings, the team is using the dashboard to discuss questions such as: *is there growing anxiety about a systemic quality? Is the team progressing in its mastery of a practice? Are there parts of the system that trigger negative emotions? Are there examples that could help us write better bug reports?*

978-1-5386-2793-8/17 $31.00 © 2017 IEEE
DOI 10.1109/SEmotion.2017..9

G. Manually label a comment in the issue tracker

Dr. Jane is doing a study to investigate whether there is a correlation between the emotions expressed in bug reports and the time needed to start the bug fixing activity. The team did not use any emotion-aware tooling, so the artefacts are not labelled. Dr. Jane is reading all comments and labelling them herself. To do that, she uses a special tool that then sends the data to the OpenAffect server.

H. Create a tool to automatically label comments

Dr. Kim is doing a similar study but is using machine learning to automatically extract emotions from bug reports. In the tool she has implemented, she has used the OpenAffect client library to send the output results to the server.

I. Conduct a study within an open source ecosystem

Dr. Karl is doing a similar study and is collecting data across an entire open source ecosystem. Because many teams in the ecosystem has used OpenAffect compatible development tools, a lot of high-quality data is available.

III. Design Questions

A. Why not use EmotionML?

This proposal has similar goals to EmotionML [1], a W3C recommendation published by the Multimodal Interaction WG. To our knowledge, EmotionML has not been widely adopted. The main reason to propose an alternative is to drive adoption by replacing the XML format with a more modern, API-driven approach. Another difference is that EmotionML has a very broad scope and leaves a lot of choices to be made by each application. We want to focus on a specific application domain and make more opinionated design decisions. This will facilitate interoperability and information sharing across tools.

B. Categories or dimensions?

Emotion researchers have developed many frameworks to describe emotions, which either use discrete categories (e.g. joy, fear) or multiple dimensions (e.g. arousal and valence). EmotionML supports both types. For the use cases that we have in mind, we believe that using discrete emotions is a better choice.

C. Which vocabulary?

A related question is whether we should i) select a single vocabulary, ii) support a list of pluggable vocabularies (similarly to EmotionML) or iii) let end-users extend a basis vocabulary with additional emotions. We propose to use one of the vocabularies supported by EmotionML, which is a list of 22 emotions proposed by Ortony et al. [2]: *admiration, anger, disappointment, distress, fear, fears-confirmed, gloating, gratification, gratitude, happy-for, hate, hope, joy, love, pity, pride, relief, remorse, reproach, resentment, satisfaction, shame*.

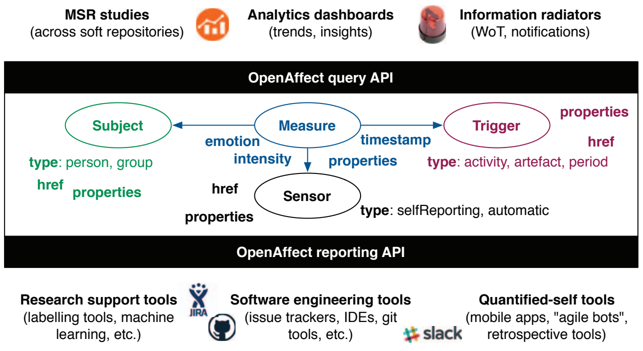

Fig. 1. The OpenAffect architecture

D. How to deal with privacy and encourage sharing?

An important area, where we have have open questions, is the provision of privacy mechanisms. The capture of affective state within a team raises questions. The disclosure of this state in public data sets raises further questions.

IV. Architecture, Data Model and API

The architecture that would enable an ecosystem of emotion awareness tools is shown in Figure 1. At the center, the server exposes APIs for reporting and querying affective measurements. The architecture supports both SaaS and private deployments. At the bottom, all sorts of tools make the measurements and send JSON payloads to the server. At the top, other tools send queries to retrieve measurements and aggregated statistics. The availability of OpenAffect client libraries in major programming languages make the development of awareness tools easy. The REST API exposes the following resources:

- A *subject* is an entity feeling an emotion. In most cases, it is a person, but it can also be a group. For instance, this is useful to capture the shared sentiment expressed a team during a retrospective meeting.
- A *trigger* is something that causes the emotion. It can be an artifact, such as a git commit or a bug report. It can be an activity, such as the participation to a meeting. It can also be a period, such as day or an iteration.
- A *sensor* is a tool which captures the affective state of a *subject* and generates a *measure*. Some tools allow subjects to self-report their emotions, others may detect affective state automatically.
- A *measure* is the record that a subject felt a certain emotion, possibly with a certain intensity.

Many of the SE tools expose *triggers* and *subjects* via a REST API. The *Subject* and *Trigger* resources managed in the OpenAffect server are proxies linked to these resources.

References

[1] F. Burkhardt, C. Becker-Asano, E. Begoli, R. Cowie, G. Fobe, P. Gebhard, A. Kazemzadeh, I. Steiner, and T. Llewellyn, "Application of emotionml," in *Proceedings of the 5th International Workshop on Emotion, Sentiment, Social Signals and Linked Open Data (ES3LOD)*, vol. 80, 2014.

[2] A. Ortony, G. L. Clore, and A. Collins, *The cognitive structure of emotions*. Cambridge university press, 1990.