Affective Stack — A Model for Affective Computing Application Development

Nik Thompson*, Tanya Jane McGill
School of Engineering & Information Technology, Murdoch University, South St, Murdoch, Western Australia

* Corresponding author. Tel.: +61 8 93601285; email: n.thompson@murdoch.edu.au
Manuscript submitted May 12, 2015; accepted July 3, 2015.
doi: 10.17706/jsw.10.8.919-930

Abstract: Affective computing applications hold promise to revolutionize human-computer interaction by enabling more natural and intuitive interaction modalities. These may include the communication of vocal expressions, physiological signals or other non-verbal indicators of underlying affective state. Although the field has experienced substantial expansion in recent years, the tools and techniques utilized are not yet mature or well established. A notable issue is the one-off nature of currently implemented affective computing systems. There is as yet no notion of standardized program architecture and there is no straightforward way to extend the functionality of existing software to include affective components. This paper introduces a new model which describes the affective computing application in terms of a set of loosely coupled functional components. This model encourages a uniform and replicable approach to affective application development in which functional components can be improved independently and subsequently re-used. The model also incorporates existing third party software as a functional component highlighting the potential to build upon existing, well established, software packages. It is hoped that this model and discussion spurs further, focused development in this growing field.

Key words: Affective computing, human-computer interaction, software development architecture, user interfaces.

1. Introduction

Affective computing applications are applications that can detect and respond to users' emotions. This paper addresses some of the issues hindering widespread development of affective computing applications and presents a component based model to support affective computing application development. This model, termed the Affective Stack Model is discussed in terms of its development and initial evaluation. A further new concept is introduced: that of retrospectively adding affect support into an existing piece of software, and this is made possible by the methodology and approach described. The paper is structured in terms of problem discussion, followed by solution suggestion, solution development and evaluation and lastly some concluding remarks.

2. Background

The central goal of affective computing is to detect and appropriately respond to the underlying affective state of a user during a human-computer interaction. The incorporation of affective and non-verbal

communications holds the potential to greatly improve the quality, usability and experience of human-computer interactions by making the interaction more natural and intuitive. However, human-computer interaction is generally both explicit (via traditional input mechanisms such as the keyboard) and asymmetric. This means that whilst there are numerous ways in which the computer may provide rich and multi-modal information to the user, the user may not always possess equivalent means of communicating his or her status to the computer [1]. This disparity is most pronounced when we consider non-standard forms of communication, such as affective state, as there are no well-established or widely deployed means for communicating this information.

Therefore, for affective computing to be successful, this disparity in communication must be reduced by enabling implicit and bidirectional communication. The creation of additional input modalities beyond the traditional keyboard and mouse can support this goal [2]. For instance these modalities may be based on physiological signals, observable traits such as image recognition of facial expressions, audio analysis of vocal patterns or a variety of other senor based input means which can signify changes in the user's emotional state, and can interpreted by affective computing applications.

McMillan, Egglestone and Anderson [3] suggested that a different interaction paradigm is required for sensor based human-computer interaction (e.g. affective feedback from a new input modality) to the traditional and widely adopted electromechanical based human-computer interaction. Unfortunately, there has been little systematic exploration how this type of sensor based interaction is best utilized in applications such as affective computing [4]. Affective computing applications are often built in the same way as more traditional applications, with the affective functionality inserted into the program architecture in an ad-hoc manner wherever the developer may deem it to be appropriate. There are also no structured ways to extend the functionality of existing or third party software with affective components, and thus the vast range of established software packages may not be able to avail of the potential benefits of affective computing.

Consequently, it could be argued that the current trend for ad-hoc development in affective computing is hampering progress. It has also been noted that research in the area was disparate and uneven, and it seems that little progress has been made in this area [e.g. 4; 5]. One goal that has been identified in the affective computing literature may be termed device independence – that is, any successful solution to handle new user input modalities must be capable of abstracting over multiple sensing devices which may have different outputs, manufacturers and operating requirements [6]. Therefore there is a clear need for a reusable design approach or model which is sufficiently abstracted from implementation considerations in order to be applicable to a wide range of operating and sensing environments.

Affective computing applications have potential uses in practically any situation where a human-computer interaction is taking place. Technology that can recognize and even express affect can provide insights into human-computer (and in some cases human-human) interactions. This may allow the system to be improved by being able to respond in a more natural and realistic way. Measuring the stress or difficulty caused by a system may also make it possible for developers to evaluate various configurations and to pinpoint potential usability problems in new systems. These technologies have been successfully implemented in very diverse environments. These include robotic personas [7], wearable computers [8], learning companions [9], [10] and games [11] .

Wearable computers provide a rich and diverse ground for evaluating and implementing affective technologies. The close contact with the user enables easy communication of subtle non-verbal cues that may be valuable indicators of affective state. In some cases, the affect detection capabilities may even be used to improve the user's own abilities to perceive emotions in others, and thus improve human-human communication. For example "expression glasses", developed at MIT provide the wearer with feedback

about the emotional expressions of others [12], a technology that may improve the quality of life for those with autism or other disorders that impair human-human communication [13]. Existing devices that are in close contact with the user may also have potential to be used as affect sensing devices, for example a mouse may sense the user's stress levels [14], or a car steering wheel may sense when the user is falling asleep or identify lapses in attention [15].

The diverse nature of development in affective computing highlights several points. Firstly, that such technology can be successful and adaptable to a wide range of situations with positive outcomes. However, given the disparate and "one-off" nature of implementations, it is potentially difficult to transfer findings from one particular domain to a new application. Secondly given that this is a new and emerging field, there is little evidence of "shared best practice" aside from the existing understanding of indicators of affective state [16].

3. Problem: The Ad-Hoc Nature of Affective Applications

Affective computing applications often diverge from one another on a seemingly fundamental level. As discussed in the previous section, the areas in which affective computing applications have been developed are very diverse with little tying them together. In terms of implementation, there is also a wide array of hardware and software in use. Design and implementation decisions may be based on ease of use, familiarity with a particular hardware or software environment or simply cost. In other instances, if the goal is to explore novel approaches to affective input processing then entirely new tools may be developed during the course of the research. Software in particular is often quite unique so at this level too there is naturally a huge amount of variation due to the differing preferences, skills and goals of the developers.

Within the application, there is also room for a lot of variation in the way the interpretation and classification of affective states is done. The interplay between complex emotions is not fully understood and many, often divergent, models of affect exist [17], [18]. Some software may utilize the concept of a few basic component emotions [19], others may attempt to directly map non-verbal cues to specific named emotions [20], or adopt a dimensional view of emotions and attempt to plot the most likely emotional state in terms of its components of arousal and valence [21].

It becomes apparent that the one commonality amongst affective computing applications is the extent to which developments are unique and tied to a particular implementation. Hamming [22] stated that "a central problem in all of computer science is how we are able to get to the situation where we build on top of the work of others rather than redoing much of it in a trivially different way" (p. 10) – an observation that is valid to this day. As affective applications are highly specialized and complex, there has to date been no discussion regarding the concept of reusing existing affective applications in new problem domains and situations. Furthermore, the potential for adding affect support as an additional layer above existing software has not been investigated thoroughly to date.

Aist, *et al*. [23] demonstrated the utility of adding emotional support (provided by a human) to an existing tutoring system, and noted that this approach may be useful for future developments in intelligent tutoring systems. Certainly, the ability to augment existing software with affect sensing capabilities could for the most part turn the entire operating system and all its application software and tools into an affective computing application. This would be a breakthrough for those who envision affective computing as being a part of the entire computing experience rather than the domain of a few isolated applications.

The next section addresses these issues by introducing a modular design architecture for affective applications. By allowing different components to be developed in isolation from one another, approaches with apparently irreconcilable differences may co-exist with the minimum of redevelopment. The nature of the model also facilitates integration with third party software, thus opening up possibilities for a unified

affective interface running on top of existing software applications.

4. Solution Suggestion

The practice of modular programming may be applied to the solution by splitting the affective system up into logical units that may be developed independently of one another. A modular system or application, differs from a monolithic system in which every unit may interface with any other, as it is composed of smaller, separated chunks of code or hardware that are isolated. By decoupling components that are likely to change, modules may be changed independently with reduced impact on the rest of the application [24]. These modules may be developed by separate teams, with their own development methodology and requirements.

This approach renders it possible for developers to concentrate on the actual logic of their application unit while building upon the strengths of complementary modules written and developed by others. Advantages of modular approaches commonly cited include shortening the time needed to develop new applications and enabling those with particular expertise to develop and enhance functionality in their own area of knowledge whilst others work on different modules, often simultaneously [25].

There are many potential advantages to a modular approach to affective applications. As components are not necessarily implementation specific, it may facilitate porting the developments to novel hardware platforms or environments. Different sensors or new input methods may be also explored and later incorporated into the affective system with reduced development and implementation time.

5. Development of Solution: The Affective Stack

The proposed solution suggests that the functional components of an affective system should be abstracted into a generalizable and re-usable model which will allow developers to build upon their successes iteratively and incrementally – this has been termed the Affective Stack Model. The use of this model as a blueprint for affective computing systems will facilitate modularization of solutions and allow separate groups to work on different functional components within their own area of expertise. This is crucial for cross disciplinary endeavours such as affective computing as this may draw from a number of specialist fields such as computer science, psychology, or physiology to name a few. This is a generalizable model, utilizing principles that are broadly applicable to the wider area of affective computing and will provide a valuable framework for the future development and implementation of affective computing applications.

The proposed Affective Stack Model is shown in Fig. 1, which illustrates the functional components and data flow between components. From a high level perspective an affective computing system can be seen to consist of several logical sub-units. These are logical units and at this level of detail no distinction is made between the hardware or software functionality implemented within them.

The Affective Stack represents a design architecture in which affective interaction components may be integrated into any application software. The Rule Set encodes the affective inputs and their associated responses for a given application context whereas the actual measurement and associated processing of affective cues is carried out within the Affective Platform component, which consists of both hardware and signal processing features. Every affective input of interest, or combination of these, is mapped to one or more rules describing the action that the host software may take at this point. This modularization enables the flexibility to develop adaptive rule sets, whilst also ensuring that implementation specific information about the user or task is not required within the Affective Platform itself.

The Event Mapper forms the bridge between the modules (or programs) which are dealing with affective information, and those modules which are directly interacting with the user (typically the user interface

classes). The provision of a suitable Event Mapper renders it possible to build upon potentially closed source commercial applications and thus give them some form of affect support capability. All of the components in the Affective Stack Model are discussed in more detail in the following sections.

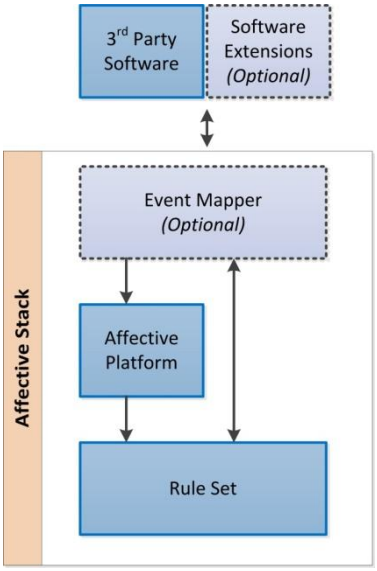

Fig. 1. Affective stack model.

5.1. Third Party Software and Software Extensions

The Third Party Software component may consist of any software, website or even operating system application that calls for affective functionality either as an initial development objective or later addition. The interaction between this component and other Affective Stack components is bidirectional, that is, there must be some way to detect the user's actions and application context and also some way to respond to this appropriately. This bridging between third party software and other affective components is carried out separately by the Event Mapper component. Depending on the type of application, there are several ways that the Event Mapper component may detect and monitor the user's actions; these are discussed in more detail in the following section.

In terms of the responses, the third party software may already possess the functionality required to interact with the user in response to the affective inputs. For example, most software already has a built in online help which may be the suitable response to trigger if the user is experiencing difficulty. However, in some cases the third party software may not already possess this functionality and may require additional components to facilitate the affective communication with the user, these optional components are termed Software Extensions.

5.2. Event Mapper

The optional Event Mapper component is included in the model as an intermediary between third party (and closed source) software and the Affective Platform component being used. The Event Mapper will provide the "hooks" by which the affective components may be attached to some existing software. Therefore, this is the only component that is potentially unique to the third party software being used. The target platform and environment will dictate the form that the Event Mapper will take or determine indeed if it is a required component.

For a completely closed source application, the event mapping may have to be performed at the interface level. At this level, an Event Mapper may take the form of a screen scraper or screen reader. A screen reader

is an application which attempts to determine what is being displayed on the screen and then presents this in an alternate form. A common use of this is a text to speech engine used for blind users. There are a number of such applications commercially available and they are often bundled with leading operating systems such as the SAPI system for Windows [26] or VoiceOver for Apple [27].

The screen reader does not actually "view" what is being displayed on the screen – instead it acquires this information programmatically through the operating system. Operating system developers include Accessibility APIs for the purpose of providing an alternate representation of what is being displayed on screen. One such example is Microsoft UI Automation (UIA) which is compatible with all commonly used Windows environments including Windows 95, XP, Vista and Windows 7. Through the UIA an application can query the operating system and receive updates when activity occurs on the screen. For example, the application may be notified when the user has clicked on a button, moved a window or accessed a menu item. This information can be accessed by the Event Mapper component and used in the affective application.

Some types of interface already make their screen contents available through system calls and will not even require the UIA API to be used. An example of this is an HTML or XML based application, such as a web interface. Objects in such an interface are represented within the Document Object Model (DOM), and these may be accessed programmatically. Thus, tasks such as retrieving the state of a button or the contents of a text box are trivial in this instance.

Design that uses an intermediary Event Mapper is not a new concept, but is commonly employed in software development of loosely coupled modules. As noted above, every implementation does not necessarily require the development of an Event Mapper component. Indeed if software is being built with affective components as stated functional requirements, then developers may make provisions to allow the program status to be queried, thus removing the need for an Event Mapper. However, good design practice would necessitate its inclusion as a component in the system architecture to clarify the logical separation between these functional components.

5.3. Affective Platform

The Affective Platform contains the hardware and software required to acquire the affective data and to convert this into a usable format. The hardware comprises the physical interface between the user and the computer, including any sensors and associated components needed to acquire a signal. The software comprises of the routines used to transform this acquired signal into a meaningful result in the context of an affective state. The resulting information is then communicated to any affective application components which require it.

Examples of hardware platforms that may meet the functional requirements of the Affective Platform component described in this model include the commercial platforms developed by ProComp and BioPac [28] and the open affective platform developed by Thompson, Koziniec and McGill [29]. This component may be used to potentially detect any indicators of affect and is not limited to physical measurements such as physiological signals. Reusability and support for iterative development are considered to be crucial indicators of the success of this model and thus developers should be able to use all of the tools and methods at their disposal without having to re-invent other's efforts.

5.4. Rule Set

The Rule Set encodes the affective inputs and forms the decision making layer which will ultimately influence the behaviour of the software. In contrast to the Event Mapper which is tightly coupled with the (third party) application software, the Rule Set is completely detached from this. The Rule Set is affective implementation specific, in that the knowledge about how to classify and respond to affective states, as well

as the specific action to be taken, is encoded in this component. Therefore the Rule Set may vary depending on the types of application behaviours required. Actions may be to control an on-screen agent, to modify interface components, to trigger an alarm, or any action that is appropriate to the situation. For example, a learning application Rule Set may direct the software to respond to some type of input from the Affective Platform with desired instructional feedback.

Fig. 2 illustrates the extent to which model components are influenced by either third party application software requirements or the implementation of the underlying model of affect. As noted previously, there are numerous models describing the structure of affect, and this naturally influences the software rule set significantly.

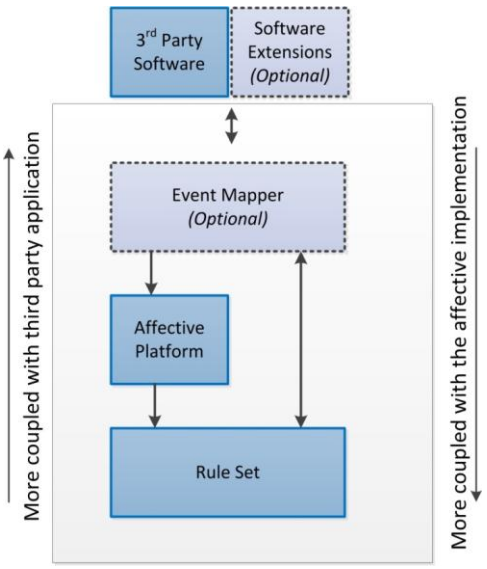

Fig. 2. Dependencies of affective stack components.

The suggested implementation of the Rule Set is a decision network in which the incoming data from the Affective Platform, as well as any other information that the application may provide, may be used to ultimately choose the course of action. Within the decision network, observations (such affective cues and interface events) may be used to calculate the probability that a particular hypothesis is true using Bayesian inference. Bayesian inference involves collecting evidence that will be either consistent or inconsistent with various outcomes; as the evidence accumulates, the degree of certainty about a particular hypothesis being true will be altered. This is expressed as a numerical estimate of the degree of confidence in a hypothesis before any evidence has been observed, and a numerical estimate of the degree of confidence in the hypothesis after a set of evidence has been observed. Therefore Bayesian inference may be used to discriminate between several hypotheses simply by accepting the one with the highest amount of support from observations.

For instance, the goal of a software interface may be to detect stressful conditions and respond accordingly. If repeated observations are made that are associated with stressful conditions (e.g. increased heart rate or breathing), then there is a higher degree of support that the user is experiencing stressful conditions as opposed to being calm. Other interface based events may also contribute to this overall inference. For instance, if the user has made repeated unsuccessful attempts at performing a software task, then this may also give further support to the hypothesis that the user is experiencing stress. All of these observations may be included and weigh into the outcome of the decision network. A number of affective applications have successfully used decision networks in the past, for example, to model indicators of affective state to guide the actions of an embodied agent [30]. It has also been suggested that such

embodied agents would be more engaging if the underlying decision network had access both to users' affective reactions, as well as other non-affective feedback [31].

Such a decision network may be implemented using a tool such as Netica [32], a package that is used for solving complex problems using Bayesian inference. The network would be constructed to contain nodes pertaining to the affective measures in use (i.e. the output from the Affective Platform), as well as other interface events to provide valuable context that may have had an influence on the emotional state of the user at that particular point in time.

An example of how a Rule Set may be constructed is shown in Fig. 3. In this basic example, the top row of nodes shows the inputs to the decision network. These include two biosignals (from the Affective Platform) and one other interface signal from the (third party) software. The combination of these values is used to infer a value for the arousal and valence nodes, which then subsequently determine the action that the software will take.

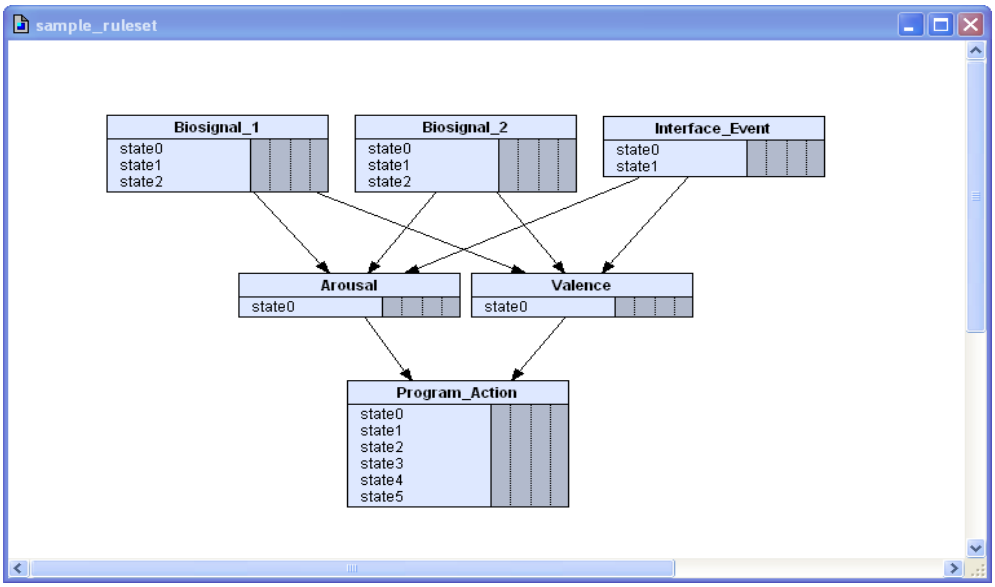

Fig. 3. Example decision network containing 3 input nodes.

This approach allows potentially any number of nodes to be included for different types or sources of data that need to be evaluated. One potential application of this flexibility may be to build a knowledge base of information regarding how users respond to various types of affective interventions. This may then serve as an internal feedback mechanism to tailor the program behaviour to use the most successful interaction strategy available at any given point. The internal feedback to implement an adaptive rule set is outside the scope of this research, although the structure of the model is such that this kind of future development is facilitated. It should be noted that the management and processing of rules within this component does contribute to the overall complexity of the developed system. Therefore, different applications may emphasize (or de-emphasize) the Rule Set component in keeping with the desired level and type of functionality of the affective computing application.

6. Evaluation

As one of the aims of the model is to address the ad-hoc nature of current development, for the model to be considered to be successful, it must be broadly applicable to current and future operating environments. The success of this model may thus be judged by the ability of the concepts to be generalized to new situations and needs. Allanson [33] noted that a measure of abstraction is desirable so that the application

is not linked closely to a particular manufacturer's hardware. In this research, this concept is extended to the uncoupling of the model from all operating constraints: these include the hardware, the application software which utilizes the affective input, and also the underlying user model which maps users' non-verbal signals to affective states and ultimately directs the course of program action.

Throughout the development of the Affective Stack Model, all design decisions were subjected to ongoing evaluations in an iterative process of refinement and testing. To clarify the extent of this evaluation, the concept of independence has been introduced. In this paper, this term is defined as the uncoupling of the logical system architecture from its operating environment. In this situation, a greater degree of independence indicates a more adaptable, flexible and robust application that is suitable for deployment into a wider range of conditions than just a controlled study. The following sections will discuss the application software, user model and hardware independence of the Affective Stack Model in more detail.

6.1. Application Software Independence

Application software independence refers to the way in which the affective components interact with the application software. A generalizable and reusable affective solution will not be inseparably bound to the application software, or its specific problem domain.

The Affective Stack Model places any domain specific knowledge and information in a separate Rule Set component. Thus the Affective Platform (containing the hardware and signal processing features) is uncoupled from the target application software.

The one constraint, however, is that the array of affective input signals being used must map throughout the model components. That is, the outputs of the Affective Platform must be encoded (as inputs) in the Rule Set for the system to function. Thus, even though redevelopment and later modifications are simplified by the model, the selection of affective signals to monitor must be given careful consideration, as the utility of the entire developed system is potentially influenced by these design choices.

6.2. User Model Independence

There are numerous, sometimes conflicting models considering the role and structure of affect in human interactions [e.g. 17; 34; 35]. Indeed, it may well be possible that different approaches are better suited for evaluating certain kinds of activities, in certain operating environments. As the underlying user model of an affective computing application essentially encodes how the application will interpret and respond to affective state on a fundamental level, this is has far reaching implications. Decisions regarding how the underlying user model is interpreted would generally be made early in the development and must remain concrete due to the number of internal dependencies that this aspect of functionality would have.

In the Affective Stack Model, however, this aspect of functionality is abstracted into its own component: the Rule Set. The Rule Set is a component which, like any other module of a program, may be refined and modified independently, meaning that the developer need not be restricted in this way. Future affective computing applications may even employ adaptive solutions, in which the Rule Set component itself is dynamically learning and adjusting. This would provide a finer level of specificity and customized interaction than would be available if the program "logic" was to be fixed at development time.

6.3. Hardware Independence

Hardware independence considers the physical interface between the user and the computer (i.e. sensor design) and the hardware used to capture this raw signal (e.g. the implementation of an analogue to digital converter). Achieving hardware independence is a significant undertaking, as most implementations described in the literature to date appear to be tied, often inseparably, to the choice of hardware made by the researchers. This is an issue, as sensor design is an area with a vast amount of flexibility and therefore

potentially one that will undergo much development and improvement over time. The use of open Affective Platform component, such as that described by Thompson, Koziniec and McGill [29], provides an example of how hardware independence can be achieved. This platform utilizes two sensors to obtain physiological indicators of affective state and describes this state in terms of two dimensions. These values can then be quantified and used as an input into other components such as the Rule Set. However, affective systems built using this component would not be restricted to the use of these sensors or platform, the influence of sensor design on the rest of the application would be entirely constrained within the Affective Platform. That is, even the most fundamental changes to sensor design and implementation would necessitate a change only in the affective platform and no other component. For example, future sensor interfaces might be fully concealed within existing input devices such as mice or keyboards. Devices such as the HandWave Bluetooth device [28] may provide wireless physiological monitoring and transmit this to the affective platform. There is even potential for the next stage of interfaces to incorporate non-contact sensing capabilities [36].

7. Conclusion

The use of the Affective Stack Model proposed in this paper demarcates the scope of development of affective computing applications and opens up possibilities for development and enhancements to be made in one specific functional component without corresponding changes or updates necessary in adjacent parts of the design. The use of this model will yield new opportunities for development within the field of affective applications. The new concept of adding an affective layer on top of existing third party applications is also of significance as it brings affective application development within the reach of all software developers and end users rather than a select subset that possess specific knowledge and resources.

In a relatively new field such as affective computing, theories regarding emotion, tools, methods and software are constantly evolving and improving as our knowledge grows. The abstraction of different functions of the affective system into logical components means that these developments and lessons learned may be quickly integrated into the current software environment. A further benefit being that the complexity of a system may be developed iteratively whilst retaining many of the same functional components. In all cases every effort has been made to create a design which is both functional, and supportive of future developments, including substantial modifications which would ordinarily have called for a major redevelopment of the application software.

In this paper, the need for structured and uniform development architecture for affective platforms was considered. To address the issues that accompany the current ad-hoc nature of development, a model was introduced which includes a modular and component based design architecture for future developments. Each of the components was also explained, with a discussion of the tools and techniques suggested to build them and example implementations. The theoretical model was also evaluated in terms of its independence, which serves as an indicator of the potential for the concepts and methods to be applied to new environments and problem domains.

References

[1] Hettinger, L. J., Branco, P., Encarnacao, L. M., & Bonato, P. (2003). Neuroadaptive technologies: Applying neuroergonomics to the design of advanced interfaces. *Theoretical Issues in Ergonomics Science,* 220-237.

[2] Serbedzija, N. B., & Fairclough, S. H. (2009). Biocybernetic loop: From awareness to evolution. *Proceedings of the IEEE Congress on Evolutionary Computation*, Trondheim, Norway.

[3] McMillan, G. R., Egglestone, R. G., & Anderson, T. R. (1995). Nonconventional controls. In G. Salvendy (Ed.), *Handbook of Human Factors and Ergonomics*. New York: John Wiley and Sons.

[4] Fairclough, S. H. (2009). Fundamentals of physiological computing. *Interacting with Computers, 21(1)*, 133-145.

[5] Allanson, J., & Fairclough, S. H. (2004). A research agenda for physiological computing. *Interacting with Computers, 16(5)*, 857-878.

[6] Mader, S., Peter, C., Göcke, R., Schultz, R., Voskamp, J., & Urban, B. (2004). A freely configurable, multi-modal sensor system for affective computing. *Affective Dialogue Systems* (pp. 313-318).

[7] Breazeal, C. (2003). Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies, 59(1-2)*, 119-155.

[8] Picard, R. W. (1997). Affective wearables. *Personal Technologies, 1(4)*, 231-240.

[9] Sarrafzadeh, A., Alexander, S., Dadgostar, F., Fan, C., & Bigdeli, A. (2008). ''How do you know that I don't understand?'' A look at the future of intelligent tutoring systems. *Computers in Human Behaviour, 24(4)*, 1342-1363.

[10] D'Mello, S., Lehman, B., & Graesser, A. (2011). A motivationally supportive affect-sensitive AutoTutor. In R. A. Calvo & S. K. D'Mello (Eds.), *New Perspectives on Affect and Learning Technologies* (pp. 113-126).

[11] Gilleade, K., Dix, A., & Allanson, J. (2005). Affective videogames and modes of affective gaming: Assist me, challenge me, emote me. *Proceedings of the Conference* on *Digital Games Research Association* Vancouver, BC, Canada.

[12] Scheirer, J., Fernandez, R., & Picard, R. W. (1999). Expression glasses a wearable device for facial expression recognition. *Computers in Human Interaction* (pp. 262-264). Pittsburgh, USA.

[13] Kaliouby, R. E. (2006). Affective computing and autism. *Annals of the New York Academy of Sciences, 1093(1)*, 228-248.

[14] Kirsch, D. (1997). The sentic mouse: Developing a tool for measuring emotional valence. Retrieved November 5, 2012, from http://affect.media.mit.edu/projectpages/archived/projects/sentic_mouse.html

[15] Gusikhin, O., Filev, D., & Rychtyckyj, N. (2008). Intelligent vehicle systems: Applications and new trends in control automation and robotics. In J. A. Cetto, J.-L. Ferrier, J. M. Costa dias Pereira & J. Filipe (Eds.), *Lecture Notes in Electrical Engineering*. Berlin: Springer Heidelberg.

[16] Cacioppo, J. T., & Tassinary, L. G. (1990). Inferring psychological significance from physiological signals. *American Psychologist, 45(1)*, 16-28.

[17] Ortony, A., Clore, G. L., & Collins, A. (1990). *The Cognitive Structure of Emotions*. New York: Cambridge University Press.

[18] Roseman, I. J., & Smith, C. A. (2001). Appraisal theory — Overview, assumptions, varieties, controversies. In K. R. Scherer, A. Schorr & T. Johnstone (Eds.), *Appraisal Processes in Emotion*. New York: Oxford University Press.

[19] Alexander, S., Sarrafzadeh, A., & Hill, S. (2006). Easy with eve: A functional affective tutoring system. In G. Rebolledo-Mendez & E. Martinez-Miron (Eds.), *Proceedings of Workshop on Motivational and Affective Issues in ITS. 8th International Conference on ITS* (pp. 38-45).

[20] Picard, R. W. (1997). *Affective computing*. Massachusetts, USA: MIT Press.

[21] Prendinger, H., Mori, J., & Ishizuka, M. (2005). Using human physiology to evaluate subtle expressivity of a virtual quizmaster in a mathematical game. *International Journal of Human-Computer Studies, 62*, 231-245.

[22] Hamming, R. W. (1969). One man's view of computer science. *Journal of the ACM, 16(1)*, 3-12.

[23] Aist, G., Kort, B., Reilly, R., Mostow, J., & Picard, R. (2002). Experimentally augmenting an intelligent

tutoring system with human-supplied capabilities: Adding human-provided emotional scaffolding to an automated reading tutor that listens. *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*.

[24] Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM, 15*(12), 1053-1058.

[25] Myers, G. J. (1975). *Reliable Software Through Composite Design*. New York: Petrocelli/Charter.

[26] Microsoft Corporation. (2010). Microsoft Speech API 5.3. Retrieved 2014, from http://msdn.microsoft.com/en-us/library/ms720151%28v=VS.85%29.aspx

[27] Apple Inc. (2010). VoiceOver. Retrieved 2011, from http://www.apple.com/accessibility/voiceover/

[28] Strauss, M., Reynolds, C., Hughes, S., Park, K., McDarby, G., & Picard, R. W. (2005). The handwave bluetooth skin conductance sensor. *Affective Computing and Intelligent Interaction*. Heidelberg, Berlin: Springer.

[29] Thompson, N., Koziniec, T., & McGill, T. (2012). An open affective computing platform. *Proceedings of the IEEE 3rd International Conference on Networked and Embedded Systems for Every Application* (pp. 1-10). Liverpool, UK.

[30] Ball, G., & Breeze, J. (2000). Emotion and personality in a conversational agent. In J. Cassel, J. Sullivan, S. Prevost & E. Churchill (Eds.), *Embodied Conversational Agents* (pp. 189-219). Cambridge, MA: The MIT Press.

[31] Conati, C. (2002). Probabilistic assessment of user's emotions in educational games. *Applied Artificial Intelligence, 16*(*7-8*), 555-575.

[32] Norsys. (2003). Netica. Retrieved 11 June, 2014, from http://www.norsys.com.

[33] Allanson, J. (2000). *Supporting the Development of Electrophysiologically Interactive Computer Systems*. Unpublished doctoral dissertation, Lancaster Univesity, Lancaster.

[34] Roseman, I. J. (1991). Appraisal determinants of discrete emotions. *Cognition and Emotion, 5(3)*, 161-200.

[35] Russell, J. A. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology, 39(6)*, 1161-1178.

[36] Poh, M. Z., McDuff, D. J., & Picard, R. W. (2010). Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics Express, 18(10)*, 10762-10774.

Nik Thompson is the academic chair of cyber forensics and information security at Murdoch University in Western Australia. He holds the MSc and PhD degrees and teaches in the area of computer security and data resource management. His research interests include affective computing, human-computer interaction and information security.

Tanya McGill is an associate professor in I.T. at Murdoch University. She has a PhD from Murdoch University. Her major research interests include e-learning, ICT education, and information security. Her work has appeared in various journals including Computers & Education, Decision Support Systems, Journal of Computer Assisted Learning, Behaviour and Information Technology and Journal of Organizational and End User Computing.