Towards a Pattern Language for Affective Systems

Javier Gonzalez-Sanchez, Maria Elena Chavez-Echeagaray,
Robert Atkinson, Winslow Burleson
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University Tempe, Arizona, US
University Drive and Mill Avenue, Tempe, AZ 85287
+1 (480) 965-9253
{javiergs, helenchavez, robert.atkinson, winslow.burleson}@asu.edu

Abstract. There is growing interest in how to leverage information about users' emotions as a mean of personalizing the response of computer systems. This is particularly useful for computer-aided learning, health, and entertainment systems. Such systems are still designed and developed from scratch and the experience from their implementation is not documented, resulting in forcing the development teams to 're-invent the wheel'. Therefore, there are few architectures, frameworks, libraries, or software tools that allow developers to easily integrate emotion recognition into their software projects. This paper presents an approach of recording the design experience in the form of patterns for emotional-aware systems and aims to develop a pattern language for those systems.

Keywords: Patterns, Pattern Languages, Emotional-Aware Systems, Affective Computing

1 Introduction

Affective Systems (AS) have the ability to accurately recognize, understand, and respond to human emotions (Gonzalez-Sanchez et al. 2011b). The design and implementation of such systems is not an easy task, since they are complex systems that: (a) incorporate several sensing devices (hardware); (b) need to apply diverse machine learning algorithms to deal with the vast amount of data generated by those sensing devices; and (c) collaborate with existing code as a subsystem.

There are several examples of research conducted on creating AS to support learning (Arroyo et al. 2009, Woolf et al. 2007, D'Mello et al. 2007), patient monitoring in health care (Chao and Zhiyong 2008), and videogames (Gilleade et al. 2005). However, the majority of this research does not focus on the creation of reusable software, software frameworks, or the best methodological practices for those purposes. Instead, these approaches are focused on creating a proof-of-concept system to collect data and validate technology approaches.

Therefore, systematic disciplined approaches must be devised in order to leverage the complexity and assortment of AS and achieve overall product quality within specific time and budget limits aiming to design and implement AS based on reusable design experience gained over several years of try-and-error attempts. One such approach is the use of patterns.

Patterns describe a problem which occurs over and over again and then describe the core of the solution to that problem, in such a way that one can use this solution a million times over (Alexander et al. 1977). Patterns are not conceived but rather discovered or mined after numerous implementations of the same solution for a given problem, usually by different people. Patterns can be grouped in a pattern language, which is a collection of related patterns that collaborate inside the boundaries of an application domain (Lyardet et al. 1998) and can guide the designer through step-by-step design guidelines. Several repositories of patterns exist for various disciplines and offer design-expertise reuse to the corresponding communities. For example: the object-oriented software community has documented the design patterns initiated by (Gamma et al. 1995); the hypermedia community has established a repository of patterns in (Hypermedia 2012); the HCI community has also launched a repository of patterns documented in (HCI Patterns 2012); and the learning community has started a similar endeavor lead by (Iba 2011).

This proposal aims to move research a step towards that direction by proposing an initial set of design patterns for AS. The patterns in this paper are meant to work synergistically and become part of a pattern language. Researchers in affective computing field have solved AS challenges repeatedly and have implemented solutions developing design patterns implicitly. As part of our work, we harvested domain-specific patterns in the attempt to document problem's solutions present on AS. Designers of AS, especially inexperienced designers, could take advantage of those patterns (i.e. previous design expertise) and save time and resources assuring software quality.

The structure of this paper is as follows: Section 2 provides background about AS classification and functionalities aimed to become patterns; the template for documenting patterns is also described here. Section 3 provides the catalog of the found patterns and enumerates their unique characteristics. Finally, Section 4 presents conclusions and ideas for future work.

2 Background

This section provides a background about AS classification (considering common implementation characteristics) and functionalities. Also present the template to be used for pattern documentation.

2.1. Classification and Functionalities

The first step for the harvesting process was to classify AS in categories according with their common implementation characteristics; two dimensions were considered: static or dynamic (system ability to react in consequence to affect) and autonomous or dependent (from other systems). As shown in the following table (Table 1) these dimensions became three categories: Loggers, Adaptive Systems, and Companions.

Table 1. AS classification

Dimensions	*Autonomous*	*Dependent*
Static	Logger	Logger
Dynamic	Adaptive System	Companion

a) **Logger.** These are AS implemented to gather data to posterior analysis. They collect affective signals and data about user's interaction with the environment and store them in files or databases.
b) **Adaptive System.** These are AS able to change their behavior in real-time, aiming to show empathy to the user. For example, modifying an element of the environment to increase the engagement in a game or changing the difficulty level of a task to reduce frustration in a learning activity.
c) **Companion.** These are an extension of **adaptive systems**; companions live inside other systems and their work is to complement them as an independent extension. For example, affective companions are used in intelligent tutor systems to provide affective support to the learner. The tutor and the companion interact but they are independent of each other. Tutor can work with or without the companion, activate or deactivate it, or even call different companions as needed.

We tried to discover common functionalities among some documented AS of each category (logger, adaptive system, companion). If these functionalities were indeed found in at least three or four AS, then these functionalities were considered widely adopted and applicable and were therefore regarded as AS patterns. The methodology used in this paper for 'pattern mining' is governed by such a philosophy.

The common functionalities detected in our exploration are summarized as follow:

a) **Sensing.** Measuring signals from a hardware device (sensors). The collected measurements are a binary stream of raw data. For example: skin conductivity.
b) **Perception**. Parsing a binary stream of raw data to obtain a measure of an affective state. For example: skin conductivity measurements are parsed to obtain arousal levels.

c) **Emotional Intelligence**. When systems are conformed of several sensors and therefore measure several signals, it makes necessary to take advantage of that data to infer an affective state. This is a functionality characteristic of multimodal systems. For example: skin conductivity, pupil dilatation, and face gestures can be used to infer the engagement or frustration of a user.

d) **Synapsis**. Communicating the affective state with other systems or subsystems of the current system. For Loggers this means put the information in a storage mean, for adaptive systems and companions this implies a communication effort.

e) **Introspection**. Gathering information about the task the user is doing and the status of the task: UI events, system failure, etc. This is useful to adaptive systems and companions to be aware of the context related with the current affective state.

f) **Rapport.** Executing a behavior accordingly with the detected affect state while the user is doing a specific task.

g) **Behavior Coding Repository**: Defining rules and policies to be applied for a specific affective state while doing a specific task. These rules and policies define the behavior of the system.

The relationships between these functionalities are shown in Fig 1.

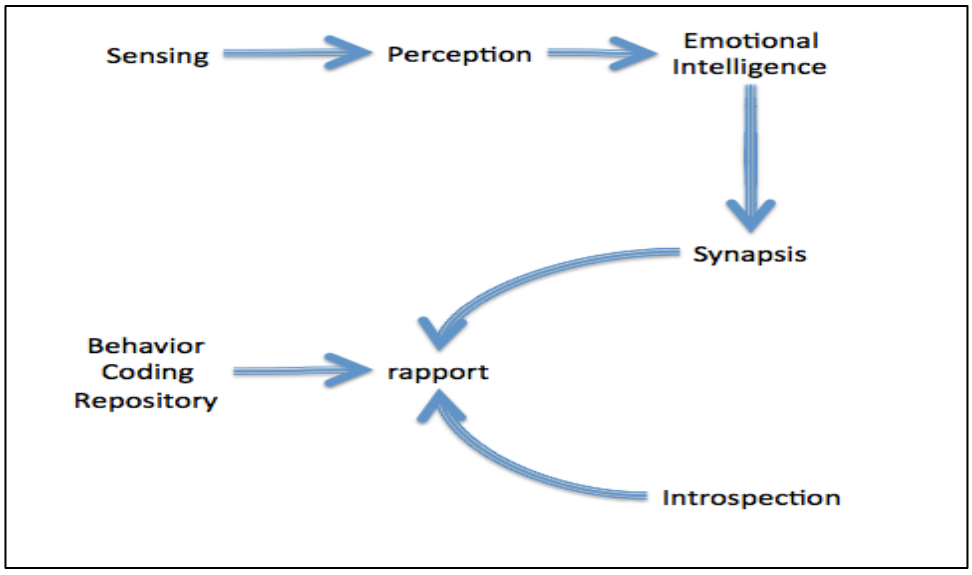

Figure 1. Common functionalities. Sensing, Perception, and Emotional Intelligence are common to all AS. The others functionalities are more common in adaptive systems (including companions) but not exclusive of them.

2.2. Sample of Systems

The set of AS used to harvest patterns is shown in Table 2, which includes the classification of the AS and the detected functionalities.

Table 2. AS reviewed to harvest patterns. Detected functionalities: (1) sensing, (2) perception, (3) emotional intelligence, (4) synapsis, (5) introspection, (6) rapport engine, (7) behavior coding repository.

AS	Category	1	2	3	4	5	6	7
Wayang Outpost by University of Massachusetts Amherst. (Cooper et al. 2009)	Logger	X	X	X	X	X		
Multi-sensor Affect Recognition System by MIT Media Lab (Kapoor and Picard 2005)	Logger	X	X	X	X			
A Platform for Affective Agent Research by MIT Media Lab (Burleson et al. 2004)	Adaptive System	X	X	X	X	X	X	X
Smart Sensor Integration by University of Augsburg (Wagner, André, and Jung 2009)	Adaptive System	X	X	X	X			
Affective Intelligent Tutor System: Emilie-1 and Emile-2 by IEEE (Nkambou 2006)	Companion	X	X	X	X	X	X	X
SEMAINE by German Research Center for Artificial Intelligence (Schroder 2010)	Adaptive System	X	X	X	X			
The Emotion Branch: A Unified software architecture by Université Bordeaux (Clay, Couture, and Nigay 2009)	Logger	X	X	X	X			
Multimodal Affect Recognition based on Decision Fusion Technique by University of Sydney (Hussain and Calvo 2009)	Logger	X	X	X	X			
Fusion Framework for Adaptive Multimodal Affect Recognition of an audience by VTT Technical Research Centre of Finland (Vildjiounaite et al. 2009)	Logger	X	X	X	X			
Replicants by University of Amsterdam (Sebe, Cohen, and Huang 2005)	Logger	X	X	X	X			
MAUI by University of Central Florida (Lisetti and Nasoz 2002)	Companion	X	X	X	X		X	X

The analysis done on these systems allows us to describe how their functionalities were implemented, find commonalities among the systems, harvest patterns, and document those patterns in a suitable format.

2.3. Pattern Description Template

Almost all of the approaches that have proposed patterns in a domain have also suggested a novel way of describing and cataloging them. Gamma et al., suggest in (Gamma et al. 1995) that it is more difficult to describe patterns than to actually find them. That is not our case for AS. We used a pattern description template taken from (Avgeriou et al. 2003) with the following fields or attributes:

a) **Name**. A unique name to distinguish the pattern and uniquely refer to it.
b) **Problem**. A brief description of the design problem at hand.
c) **Motivation**. An explanation of the origins of the problem, probably with an example for better communicating it. It may also contain the context of the particular problem if it is necessary in order to make it more understandable.
d) **Solution**. A description of the solution proposed by this pattern that addresses the problem and motivation stated earlier.
e) **Forces.** A list of the issues, pulled from the problem, which are addressed by the solution.
f) **Known uses**. Examples of the pattern in current AS. This is an important attribute of a pattern since it is claimed that a proposed pattern gets accepted by the corresponding pattern community, only if there has been two or three examples of its use by someone other than the one who is suggesting the pattern (Buschmann et al. 1996).
g) **Related Patterns**. Other patterns associated to this one in some way.

This template does not delve into implementation details, but rather expresses a generic solution.

3. Catalog of Patterns

This section shows the application of the template proposed in the previous section, for the harvested AS patterns. Notice that the template in the previous section contains seven items per each pattern and the subsections below contain only six, due to the fact that the first item on the template is the name of the pattern that is the title of each subsection. The relationships between the described patterns are depicted in Fig 1.

3.1 Sensing

Problem. External hardware devices, called sensors, measure signals of affective (emotional) changes. Those measurements are streams of binary data that are complex and diverse, them can range from brain-waves (EEG) signals and physiological reactions readings to face-based and gesture-based emotion recognition to posture and pressure sensing (Gonzalez-Sanchez et al. 2011).

Motivation. Minimize dependency on the hardware devices used to measure signals of affective (emotional) changes and provide genericity to access those signals.

Solution. The Sensing pattern defines a generic interface with a serial port device. The main intention here is to completely encapsulate the interface with the serial port hardware device. All components interfacing with the serial port will not be impacted by changes in the hardware device. Thus, data is gathered no matter the hardware interface (USB, Bluetooth, serial port, or any other communication approach).

The process to follow implies to:

1. Open a connection with the device establishing adequate parameters for speed, number of data-bits per character, parity, and number of stop bytes. For example: 9600 bps, 8 data bits, 1 stop byte, and no parity check.
2. Establish a sampling rate. For example skin conductivity is measured at 2 Hz, EEG is measured at 8 Hz.
3. Gather a group of bits from the connection and put them in a repository.

Forces. Isolating the hardware from the process. Sensing pattern keeps the hardware-dependent commands confined in a component of the system; therefore simplifies the software port to new hardware.

Known uses. The eleven AS described in Table 2 work in this way.

Related Patterns. `PERCEPTION`.

3.2. Perception

Problem. The organization, identification, and interpretation of sensed information in order to match it with a value that represents a level of a specific affect. This implies a process that transforms signal measurements from the environment (groups of bits) into encoded meaningful values (magnitudes).

Motivation. Measured signals by themselves are not useful, there is the need to process them and came with an understanding of that data, and parsing them into a standardized format. The goal is to provide a value that represents the magnitude of an affect state.

Solution. Define a family of algorithms (called perception mechanisms), encapsulate each perception mechanism, and make those perception mechanisms interchangeable. Perception pattern allows applying different algorithms independently from the components that use it.

The process to follow implies to:

1. Gather sensed information from a specified source.
2. Use this information as input for the perception algorithm. A perception algorithm varies from regression models to inference networks. Researchers report diverse approaches to implement perception using well-known machine learning algorithms.
3. Report the algorithm output.

Forces. Encapsulating perception mechanisms, isolating them from the rest of the system, and making them interchangeable. Therefore, simplifying the software modifiability.

Known uses. The eleven AS described in Table 2 use this pattern.

Related Patterns. `SENSING` and `EMOTIONAL INTELLIGENCE`.

3.3 Emotional Intelligence

Problem. Affect detection is commonly implemented as multimodal, i.e. using several sensing devices where each of them is associated with a specific perception algorithm. The use of multiple inputs modality aims to increase accuracy. This modality could provide magnitudes of the same affect or magnitudes of related affects. Multimodality deals with lots of information that needs to be organized, structured, and conjugated. For example: EEG sensors provide magnitudes for boredom and frustration, and face-based sensors report magnitudes for interest and sureness; using those magnitudes (boredom, frustration, sureness, and interest) makes possible for a system to make an intervention or not, and if decided, the kind of the intervention.

Motivation: Emotional Intelligence is a key component in multimodal systems that collect several signals and apply several perception mechanisms to increase the accuracy of the affect state detected. The goal is providing the system with the ability to join diverse perceptions in one affect state.

Solution: To use a collection of independent programs that fills cooperatively a common data repository. Each program is providing a perception value using its own resources, and all programs share their info to converge in a resulting affect state. Programs are independent of each other. They do not call each other, nor is there a predetermined sequence for their activation. A central control shell component evaluates the current reported values and infers the true about the affect state. This data-directed control regime is referred to as opportunistic problem solving, allows experimenting different algorithms, and allows experimentally derived heuristics to control processing. This is close to Blackboard pattern used to describe the situation where a group of human experts sit in front of a real blackboard and work together to solve a problem. But in this case only one

expert is solving the problem and the knowledge sources are providing the information to solve it.

The process to follow implies that:

1. A common knowledge base is iteratively updated by a diverse group of knowledge sources (the perception mechanisms).
2. Each knowledge source updates the knowledge base with its own inference of a user's affect.
3. A control shell, the expert, is responsible to infer a common affect state joining the reported values (selecting and rejecting values) from the diverse knowledge sources.
4. In a loop process, the knowledge sources and the control shell continue working together to solve the problem handling the solution as a sum of its parts.

Forces. Provide support for experimentation combining knowledge sources (perception mechanisms) and improve fault tolerance and robustness (failure in one perception mechanism has less impact on the whole system).

Known uses. The eleven AS described in Table 2 use this pattern.

Related Patterns. `PERCEPTION` and `SYNAPSIS`.

3.4 Synapsis

Problem. Infer affective states is only one step in the process, now the final goal is to communicate this affect state with other components that can use it to adjust or react and make a system aware of the user's affect state.

Motivation. Communicate or share affect states maintaining the sources decoupled from the destinations; thus, sources and destinations can vary independently.

Solution. Provide a communication infrastructure based in a message-queue paradigm for inter-process communication. Senders of messages publish the messages, without of knowledge of what if any, receiver there may be; messages are sent to the systems or components that are interested in receiving those messages.

The process to follow implies that:

1. Listening elements (receivers) exist and are willing to subscribe to specific messages.
2. Senders elements (sources) put messages in the message queue without of knowledge of what if any, receiver there may be for it.

3. A control unit associated with the queue filters the message and each receiver is notified only of those messages it was interested on.

Forces. Promote integration and scalability, concentrating in one point the access to the data required by several components. Became a facade for external systems.

Known uses. The AS 3, 4, and 5 listed in Table 2 use this pattern approach.

Related Patterns. `EMOTIONAL INTELLIGENCE` and `RAPPORT ENGINE`.

3.5 Introspection

Problem. Determine the context of user's affect state, i.e. what the user is doing and what is receiving in consequence. To do that, it is necessary to examine the computer's program inputs (events received) and outputs (status) at runtime including the values, properties, and functions of the system.

Motivation. Decide wisely the kind of intervention that is required to know the context on what the user is working, i.e. having information about the actions (events created or received) he is performing. For example: if it is reported that the user has a high level of frustration it makes sense to introspect about the task he is doing and the status of the computer (such as system failures).

Solution. Add additional responsibilities to some components into the system in order to monitor its execution and maintain a log of relevant events (input and outputs) at runtime.

The process to follow implies:

1. To wrap the original system into new component.
2. Calls to the original systems will be received by the new wrapper element.
3. Wrapper calls to the system functionality and also reports the execution of the action.
4. Wrapper provides data about the actions that the user is performing, the task, and the system status while user's affect state is been inferred.

Forces. Add to the components the functionality of maintaining and updating a log of their actions, so when one changes its state, a log is maintained and updated automatically recording inputs and outputs.

Know uses. The AS 1, 3, and 5 listed in Table 2 use this pattern.

Related patterns. `RAPPORT ENGINE`.

3.6. Rapport Engine

Problem. There are several and different ways (policies) in which a system could be empathetic. How to choose the best option? Execute the proper reaction (behavior) for a detected affect state while doing a specific task is key. It is required to provide some form of artificial intelligence, which consists primarily of a set of rules about behavior.

Motivation. Provide the mechanism necessary to select and execute the predefined behavior ad-hoc for the current situation in order to achieve some goal. Having systems that are able to be empathetic with the user offers a social and affective support that has been proven to have positive impact. To be empathetic it is necessary to show a behavior compatible with the current affective information and the context of the user.

Solution. Define an element on the system able to combine the information shared by the synapsis process and the introspection of user's context to select the proper reaction to be done. This element is able to define which would be the proper way to proceed accordingly and bases its decisions on rules and polices previously defined and validated.

The process to follow implies:

1. Having access to synapsis data.
2. Having access to introspection data.
3. Having access to a set of defined behaviors that the system is able to execute.
4. Defining rules as conditions and providing a mechanism for prioritizing those rules when more than one is triggered.
5. if a rule or condition matches the current state, of the world the condition is triggered and the associated behavior fired. The Rapport Engine often has to choose between mutually exclusive rules - since actions take time, only one action can be taken. Two steps are necessary: (a) matching rules against the database, (b) selecting which of the matched rules to apply and executing the selected actions.

The Engine chooses a behavior as follows:

1. Only one behavior can be active and in control at any time.
2. Each behavior has a fixed priority.
3. Each behavior has associated a condition that can determine if the behavior should be executed or not.
4. The active behavior has higher priority than any other behavior that should take control.

Forces. Encapsulate the decision-making capacity of the system**.**

Known uses. The AS 3, 5, 6, and 11 described in Table 2 use this pattern. Even thought the AS's strategies vary from the simple – IF conditions - to the complex – production systems or machine learning models, whichever strategy is implemented, the method is indeed crucial for the efficiency and correctness of companions and adaptive systems.

Related Patterns. SYNAPSIS, INTROSPECTION, and BEHAVIOR CODING REPOSITORY.

3.7. Behavior Coding Repository

Problem. Create a repository, emulating a working memory, which maintains data about rules and variables (state or knowledge) that defines the reactive behaviors for the system.

Motivation. Rules or polices, called behaviors, need to be defended as rationale of the empathetic. Those rules or policies assure that the way in which the system is reacting is the best according with the situation. Researchers define those rules with experience and common sense.

Solution. An element is required to hold information in the mind to do reasoning and comprehension and to make this information available for further processing.

The process to follow implies to:

1. Define and record behaviors into the repository including conditions, priority, and content.
2. Provide a mechanism for behavior localization.

Forces. Strive to create the simplest, most powerful solution possible; even if it takes slightly more time provide reusable approach. Allow adding and removing behaviors without even looking at the rest of the code.

Known uses. The AS 3, 5, 6, and 11 described in Table 2 use a rudimentary form of this pattern in which behaviors are if-else conditions, priority is driven by the structure of the conditions itself. Forces in those solutions are hard to say simple or reusable but researchers (outside of software community) could agree that if-else conditions are simple and reusable for them. Our pattern proposal pleads for a more software engineering way to implement this (such as using decision systems).

Related Patterns. RAPPORT ENGINE.

4. Conclusions and Future Work

This paper has attempted to initiate the establishment of a pattern language for AS, expanding the application domain of design patterns in areas such as Affective Computing and particularly in the development of AS. We believe that such a pattern language can provide many advantages for designers of AS, such as reduce time and cost of designing and developing AS, increase the software qualities on the AS especially in the usability of the system, and increase pedagogical quality of AS especially in learning effectiveness. Furthermore, an experimental AS is already being constructed following the patterns proposed in this paper. The aim is to illustrate the actual implementation of this pattern language by showing the implementation details and offering a complete description of the patterns' template.

Acknowledgments

We are grateful to Filipe Correia for his support during the writing process of this paper. This research was supported by Office of Naval Research under Grant N00014-10-1-0143 awarded to Dr. Robert Atkinson.

References

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S. 1977. *A Pattern Language*, New York: Oxford University Press.

Arroyo, I., Cooper, D. G., Burleson, W., Woolf, B. P., Muldner, K., and Christopherson, R. 2009 Emotion Sensors Go to School. In V. Dimitrova, R. Mizoguchi, B. du Boulay & A. Grasser (Eds.), Artificial Intelligence in Education. Building Learning Systems that Care: from Knowledge Representation to Affective Modelling (Vol. Frontiers in Artificial Intelligence and Applications 200), IOS Press, 17—24

Avgeriou, Paris, A. Papasalouros, S. Retalis, and M. Skordalakis. 2003. "Towards a Pattern Language for Learning Management Systems." Educational Technology & Society 6 (2): 11–24.

Burleson, Winslow, Rosalind W. Picard, K. Perlin, and J. Lippincott. 2004. "A Platform for Affective Agent Research." Workshop on Empathetic Agents, International Conference on Autonomous Agents and Multiagent Systems, Columbia University, New York, NY. http://www-3.unipv.it/webpsyco/bacheca/materiale/pessametodi0607EmotionModel5.pdf.

Buschmann F., Meunier, R., Rohnert, H., Sommertland P., and Stal, M. 1996. Pattern-Oriented Software Architecture, Volume 1: A System of Patterns, Chichester, UK: John Wiley & Sons.

Chao, X. and Zhiyong, F. 2008. A Trusted Affective Model Approach to Proactive Health Monitoring System. In Proceedings of the 2008 International Seminar on Future BioMedical Information Engineering. FBIE '08, IEEE Computer Society. 429--432

Clay, A., N. Couture, and L. Nigay. 2009. "Engineering Affective Computing: a Unifying Software Architecture." In, 1–6.

Cooper, David G., Ivon Arroyo, Beverly P. Woolf, Kasia Muldner, Winslow Burleson, and Robert M. Christopherson. 2009. “Sensors Model Student Self Concept in the Classroom.” User Modeling, Adaptation, and Personalization: 30–41.

D'Mello, S., Picard, R. W., and Graesser, A. 2007. Toward an Affect-Sensitive AutoTutor. In IEEE Intelligent Systems, (Vol. 22 no. 4), 53--61

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Gilleade, K., Dix, A., and Allanson, J. 2005. Affective Videogames and Modes of Affective Gaming: Assist Me, Challenge Me, Emote Me. In Proceedings of Digital Games Research Association. DIGRA'05, 16—20

Gonzalez-Sanchez, J., Chavez-Echeagaray, M.E., Atkinson, R., and Burleson, W. 2011. Affective Computing Meets Design Patterns: A Pattern-Based Model of A Multimodal Emotion Recognition Framework, Proceedings of the 16th European Conference on Patterns Languages of Programs.

Gonzalez-Sanchez, Javier, Robert M. Christopherson, Maria E. Chavez-Echeagaray, D.C. Gibson, Robert Atkinson, and Winslow Burleson. 2011. “How to Do Multimodal Detection of Affective States?.” Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on: 654–655.

HCI design patterns, http://www.hcipatterns.org/

Hussain, M.S., and R.A. Calvo. 2009. “A Framework for Multimodal Affect Recognition.” Learning Systems Group, DECE, University of Sydney. http://sf08.hcsnet.edu.au/files2/full_papers/A%20framework%20for%20affect%20recognition_20090630-2046.doc.

Hypermedia Design Patterns Repository, http://www.designpattern.lu.unisi.ch/HypermediaHomePage.htm.

Iba, T., 2011. Pedagogical Patterns for Creative Learning, 18th Conference on Pattern Languages of Programs (PLoP).

Kapoor, Ashish, and Rosalind W. Picard. 2005. “Multimodal Affect Recognition in Learning Environments.” In, 677–682.

Lisetti, Christine L., and Fatma Nasoz. 2002. “MAUI: a Multimodal Affective User Interface.” In, 161–170. New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/641007.641038.

Lyardet, F., Rossi, G., and Schwabe, D. 1998. Using Design Patterns in Educational Multimedia Applications. EDMedia'98.

Nkambou, R. 2006. “A Framework for Affective Intelligent Tutoring Systems.” In, nil2–nil8.

Schroder, Marc. 2010. “The SEMAINE API: Towards a Standards-Based Framework for Building Emotion-Oriented Systems.” Advances in Human-Computer Interaction 2010 (January). doi:http://dx.doi.org/10.1155/2010/319406.

Sebe, Nicu, I. Cohen, and T.S. Huang. 2005. “Multimodal Emotion Recognition.” Handbook of Pattern Recognition and Computer Vision 4: 387–419. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.1129&rep=rep1&type=pdf.

Vildjiounaite, E., V. Kyllonen, O. Vuorinen, S.M. Makela, T. Keranen, M. Niiranen, J. Knuutinen, and J. Peltola. 2009. “Requirements and Software Framework for Adaptive Multimodal Affect Recognition.” In, 1–7.

Wagner, J., E. André, and F. Jung. 2009. “Smart Sensor Integration: a Framework for Multimodal Emotion Recognition in Real-Time.” In, 1–8.

Woolf, B., Burelson, W., and Arroyo, I. 2007. Emotional Intelligence for Computer Tutors. In Supplementary Proceedings of the 13th International Conference on Artificial Intelligence in Education. AIED ‘07, 6--15