Toward a Software Product Line for Affective-Driven Self-Adaptive Systems

Javier Gonzalez-Sanchez
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, AZ, USA
javiergs@asu.edu

Abstract—One expected characteristic in modern systems is self-adaptation, the capability of monitoring and reacting to changes into the environment. A particular case of self-adaptation is affective-driven self-adaptation. Affective-driven self-adaptation is about having consciousness of user’s affects (emotions) and drive self-adaptation reacting to changes in those affects. Most of the previous work around self-adaptive systems deals with performance, resources, and error recovery as variables that trigger a system reaction. Moreover, most effort around affect recognition has been put towards offline analysis of affect, and to date only few applications exist that are able to infer user’s affect in real-time and trigger self-adaptation mechanisms. In response to this deficit, this work proposes a software product line approach to jump-start the development of affect-driven self-adaptive systems by offering the definition of a domain-specific architecture, a set of components (organized as a framework), and guidelines to tailor those components. Study cases with systems for learning and gaming will confirm the capability of the software product line to provide desired functionalities and qualities.

Index Terms—Affective Computing, Self-Adaptation, Software Product Line, Framework, Patterns, Software Architecture

I. MOTIVATION

The computer's ability to recognize human affective states (emotions), given physiological signals, and to change its behavior driven by those affective states is gaining popularity; this is motivated by the realization that emotions are inextricably bound to human cognitive processes and express a lot about human necessities [1]. Emerging systems equipped with those abilities can be called “affect-driven self-adaptive” systems. Affective-driven self-adaptive systems aim to take advantage of sensing devices to perceive signals of user’s affective state changes, understand the meaning of those changes, learn from them, and react in consequence to improve system’s functionality and user experience. Learning environments, health care systems, and video games are just a few examples of systems that stand to benefit from affective-driven self-adaptive capabilities.

II. PROBLEM

Providing affective-driven self-adaptive capabilities to software systems faces three main challenges.

First, like most self-adaptive systems deployed today, affective-driven self-adaptive systems implement their self-adaptive capabilities in an application-specific and disperse "hardwired" way; which make them brittle, difficult to generalize, and costly and difficult to modify and maintain [2]. Additionally, self-adaptation models are reported mainly for performance and work balance (resources) issues. Self-adaptability based on human factors, such as affect measurement, seems to be a different kind of challenge.

Second, although Picard et al. have demonstrated the feasibility of developing physiological-signal-based emotion recognition systems [1], studies’ reports indicate that most work is related with offline affect analysis (creation of models) not with real-time awareness, i.e. software systems working with users in real-time, without human supervision, in which affect recognition trigger self-adaptation.

Third, since affective computing is a relatively young field, most efforts around it are focused on specific systems as one-of-a-kind endeavors; efforts around manufacturing in great scale affect-driven self-adaptive software are rare and moving craftsmanship of proof-of-concept systems to manufacturing is something that has not been successfully done.

The goal of this work is to approach these challenges by standardizing a feasible and cost-effective way to add affective-driven self-adaptive capabilities, either into new or existing systems, enabling systematic reuse and achieving the adequate software qualities.

III. RELATED WORK

Existing approaches limit their applicability to support the creation of one product instead of a common product platform (i.e. product families). Developing a product is useful to solve one problem, but massive production of affective-driven self-adaptive systems or even just affective-aware systems requires involving expertise centered on the assets and procedures associated with affect and self-adaptation domains. The following paragraph summarizes some milestone works.

Burleson et al. developed a platform for multimodal sensing and interpretation of affective information, which is able to provide a response in real-time through an expressive agent [3]. Showing that affect can be used as a control variable is the main contribution of this work. Burleson et al. platform aims to help end-users (researchers) to run investigations and specifically to test theories about affect responses in learning.
Therefore, the scope is not an intentional and concerted effort to create and apply multi-use software artifacts, throughout, to support developers to create affective-driven adaptation.

Garlan et al. recognize that adaptation mechanisms highly specific to one application and tightly bounded to the code are costly to build and difficult to modify. Therefore, they proposed a middleware for supporting adaptation [4]. The middleware leverages a framework, called Rainbow, and guidelines to work with it. Their goal is to be able to add self-adaptation capabilities to a wide variety of systems adding external control mechanisms and using a reusable infrastructure. However, their study cases address self-adaptation driven by performance and work balance (resources) issues; dealing with affect measurements is not in their scope.

Clay et al. proposed an architecture pattern, named emotion recognition branch, as a guide for the engineering of affective systems with three piped components for capturing, analyzing and interpreting data [5]. Clay et al. work is related to affective systems in general, i.e. what the system does with the information about the affective state of the user is not supported.

Hussain and Calvo proposed a framework for multimodal affect recognition targeted for learning systems but flexible to other applications [6]. Hussain and Calvo framework, like Burleson et al. platform, aims to support end-users (researchers) to run investigations and specifically to test theories about affect responses. There is not an intentional and concerted effort to create and apply multi-use software artifacts, throughout, to support developers to create affective-driven adaptation.

Several others' efforts exist; most of them are similar to one of the previously described (therefore present similar limitations), such as: (a) multimodal affective user interface by [7]; (b) agent-based intelligent tutoring system using facial recognition system by [8]; (c) a framework for affective intelligent tutor system: Emilie-1 and Emile-2 by [9]; (d) SEMAINE framework by [10]; and (e) multimodal emotion recognition by [11].

IV. APPROACH

My hypothesis is that it is possible to support a broad creation and integration of affective-driven self-adaptation as core capability inside new and existing systems by (1) extending and adapting self-adaptive software fundamentals (such as the ones described in [4] for performance-driven and resource-driven self-adaptation) to support the creation of systems that use affect as a control variable, and (2) implementing a software engineering manufacturing approach, such as software product lines. Software product lines (SPL) adopt a manufacturing vision encoding proved practices and capturing knowledge of how to produce applications that share common characteristics and make that knowledge available in the form of assets (architecture models, patterns, frameworks, and tools) within a package of integrated guidance to systematically apply them. Developing products using a suitable SPL aims to improve productivity (reducing cost and time), quality, and evolution capability.

V. CONTRIBUTIONS

There are three assets to consider outlining a SPL approach:

1) An architectural model that defines how systems must be built and how that architectural model can be applied to different class of systems; the architectural model advances common problems in affective-driven self-adaptive software design and generalizes them to different kinds of systems holding a promise for cost-effectiveness and dynamicity (real-time operation).

2) A framework, as a collection of tailorable parts that allows developers to target the infrastructure to specific classes of systems. It is a reusable infrastructure that systematically defines interfaces and components. The framework will enable systems’ developers (software engineers) to add affective-driven self-adaptive capabilities to their systems. The goal of the framework development will not be to invent any new fusion or classification methods, but rather to facilitate researchers using the existing ones.

3) Guidelines to combine proprietary and framework components while creating new systems. The underlying principle here is to separate the system from adaptation and adaptation from affect recognition following a divide-and-conquer method to confront the problem. Guidelines are documented to specify objectives, properties of interest, conditions for change, and strategies for adaptation.

The assets (architecture, components and guidelines) to be implemented in this proposal have their origins in the phases documented in [2] for engineering self-adaptation (monitoring, modeling, and controlling) as well as in the categories proposed in [12] for dynamically adaptive systems (monitoring, decision-making, and reconfiguration infrastructure). Having these works as a starting point we (my team and I) harvested common functionalities among some documented affective systems. The common functionalities detected in our exploration are summarized as follows: (a) sensing - measuring signals from hardware devices; (b) perception - parsing a binary stream of raw data to obtain a measure of an affective state; (c) emotional intelligence - when systems are conformed of several sensors (therefore, measure several signals) it is necessary to fusion that data to infer an affective state; (d) synopsis - communicating the affective state with other systems or subsystems of the current system; (e) introspection - gathering information about the task that the user is doing and the status of the task (such as UI events and system failure) to make system aware of the context related with the current affective state; (f) rapport - executing a behavior accordingly with the detected affective state while the user is doing a specific task; and (g) behavior 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).

In my dissertation proposal, I present a research agenda to create or adapt an architectural model for affective-driven self-adaptive systems, implement domain-specific components (organized as a framework), and explore the use of design patterns as templates that document the required guidelines.
(besides the use of object-oriented common documentation techniques).

VI. PROGRESS

Preliminary work demonstrates the feasibility of the overall approach while partially addressing the challenges related with affect recognition, usage of patterns to document guidelines related with affect, and designing of a framework that provides coarse-grained components for affect recognition (a first step on the road to affective-driven self-adaptation). The following paragraphs present a list of publications and developments done that represent advances of the work proposed here.

Working with colleagues, I have tackled the issue of detection of affective states. Detecting affective states requires that the computer senses information that is complex and diverse, it can range from brain-waves signals and biofeedback readings to face-based and gesture emotion recognition to posture and pressure sensing. Obtaining, processing, and understanding that information requires the use of several sensing devices, algorithms, and tools. I have already researched and present work about multimodal emotion recognition showing my understanding of methodologies and tools in [13].

Additionally, I led into the development of an agent-based software architecture for a multimodal emotion recognition framework. This work offered a first step to provide structural reference (architecture) and components (framework) to develop affective-driven self-adaptive systems. This previous work addressed: the modeling of an agent-driven component-based architecture for multimodal emotion recognition, called ABE, and the use of ABE to implement a multimodal emotion recognition framework to support affect recognition [14]. So far, I have successfully prototyped the framework and demonstrated its functionality within a first iteration of implementation. Also, I addressed in [15] this shortcoming in models and developer’s guidelines by proposing the use of software design patterns for modeling a multimodal emotion recognition framework. The design of the framework offers to: (a) integrate existing sensing devices and SDK platforms, (b) include diverse inference algorithms, and (c) help to correlate measurements from diverse sources. We (my team and I) described our experience using this model and its impact on facets, such as creating a common language among stakeholders, supporting an incremental development, and adjusting to a highly shifting development team, as well as the qualities achieved and trade-offs made. To fulfill my dissertation, I will need to complete the design and perform subsequent iterations of implementation to include self-adaptation assets and moving from affect recognition to affective-driven self-adaptation.

Finally, a preliminary set of assets was tested with undergraduate student developing an adaptable 3D video game, which uses user’s affective states as input to alter and adjust the gaming environment; brain waves were used as input to infer meditation, excitement, and engagement into the game; lighting and colors were altered according with the inferred user’s affective state [16].

VII. EVALUATION

Each of the challenges and, therefore, the results need to be validated and evaluated in a different way, involving formal analysis of software assets, empirical evaluation, and user study cases. For study cases with users, two kinds of users are considered for different evaluation: developers and system users. I plan to perform evaluation as follows:

a) Framework test bed. Cheng et al. [2] used for their self-adaptive framework evaluation a dedicated test bed with study cases of systems with different concerns, and they qualitatively demonstrated how self-adaptive frameworks could be generalized across different styles of systems and concerns. The main goal of their experiments was to test whether the framework provides all desired functionalities in a fairly convenient way. The experiments needed to confirm that the framework easily adapts to input data available; allows to flexibly combine feature-level, class-level, and decision-level fusion methods; allows to reason data along timeline; and to select the appropriate classifier in real time. I propose to follow their approach for qualitative analysis.

b) Framework and patterns in action. This aims to evaluate that developers understand the architecture, and are able to use the framework components following the guidelines. These users could be undergraduate students involved in capstone projects, or undergraduate research assistants participating in the creation of systems. This will include gathering and interpreting data from developers’ experience and run case studies that apply the SPL on learning and gamming systems to demonstrate generality and composability. I plan to assess the cost-effectiveness based on measurement of effort and informal user studies. As part of my dissertation, I will evaluate how well such framework supports modular application of patterns as guidelines.

c) User case studies. An affective-driven self-adaptive system is a unique and site-specific mixture of program, computer, hardware, software, and computational architecture. The evaluation of programs and hardware is important, as well as the architecture structure and models, but also the HCI component. This proposal will be used as a basis to built affective-driven systems; testing the fundamentals from the software point of view requires, as described before, formal methodologies or empirical studies with developers. But that does not evaluate users’ (other than developers) experience with what the SPL provides. Thus, I propose a complementary evaluation of that as follows: having understood that running user studies cannot be done with the SPL assets but can be done with products created using the SPL approach. The results of the evaluation of the product will need to be analyzed carefully, since the user’s experience per se will be in part due to the internal functionality of the product, and in part due to the qualities and operation of the assets provided by the SPL approach. The assets of this proposal are essential to the realization of appropriately emotionally adaptive experiences; therefore, I believe that to test the accuracy of my proposal it is also important to evaluate users operating those systems. Moreover, as claimed by Mark Weiser “the best computer is a quiet, invisible servant”, it is important to the goal of this thesis.
to demonstrate that SPL assets have the qualities to be those servants inside applications. Without matter, this evaluation can be ambiguous because it is related with user experience with the application itself, with the SPL assets, and with the accurately application of process and guidelines by the developers involved. Several factors are involved but it is fair to run case studies around it.

d) Formal evaluation of the architecture. It has not been concluded if the architectural model will be created or adapted. If it is adapted, the evaluation of the architectural model could be part of what was previously described. If a new architectural model will be required, then it could be necessary a formal evaluation of the architecture to make sure it is the one that will do the job. A new architecture must be evaluated because so much is riding on it. Architecture evaluation provides a relatively low-cost risk mitigation capability. The evaluation of the architecture could be conducted using Architecture Tradeoff Analysis Method (ATAM) [17]. ATAM’s purpose is to help choosing a suitable architecture for a software system by discovering trade-offs and sensitivity points.

ACKNOWLEDGE

This research was supported by the Office of Naval Research under Grant N00014-10-1-0143 awarded to Dr. Robert Atkinson and by the National Science Foundation, Award 0705554, IIS/HCC Affective Learning Companions: Modeling and supporting emotion during teaching Dr. Beverly Woolf and Dr. Winslow Burleson.

REFERENCES