Reflective Pervasive Systems

(a submission to the special issue on Pervasive Adaptation)

NIKOLA SERBEDZIJA, Fraunhofer FIRST, Berlin
AND
STEPHEN FAIRCLOUGH, School of Natural Sciences & Psychology, Liverpool John Moores University

Pervasive adaptive systems are concerned with the construction of 'smart' technologies capable of adapting to the needs of the individual in real-time. In order to achieve this level of specificity, systems must be capable of monitoring the psychological status of the user and responding to these changes in real-time and across multiple systems if necessary. This paper describes a number of conceptual issues associated with this category of adaptive technology. The biocybernetic loop describes different approaches to monitoring the status of the user from physiological sensors to overt behaviour. These data are used to drive real-time system adaptation tailored to a specific user in a particular context. The rate at which the technology adapts to the individual user are described over three different phases of usage: awareness (short-term), adjustment (medium-term) and co-evolution. (long-term). An ontology is then proposed for the development of an adaptive software architecture that embodies this approach and may be extended to encompass several distinct loops working in parallel. The feasibility of the approach is assessed through implemented case studies their performance and functionality.

Categories and Subject Descriptors: I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—Multiagent systems; C.2.4 [**Computer-Communication New York**]: Distributed System—Distributed applications; H.1.2 [**Models and Principles**]: User/Machine Systems—Human factors

General Terms: Design, Human Factors, Languages

Additional Key Words and Phrases: Biocybernetic loop, Physiological computing, Ontology, Middleware

ACM File Format:
Serbedzija, N and Fairclough, S. 2010, Reflective Pervasive Systems

1. Introduction

Seamless and implicit human-computer interaction (HCI) is an important characteristic of smart technology [Norman 2007]. In order for technology to be 'smart', the system requires some implicit means of assessing the context of system usage that does not necessitate any form of explicit user intervention. The development of smart computer systems requires technology to interface with the behaviour of the user in order to deliver real-time adaptation that is perceived as both timely and intuitive from the perspective of the user.

"What you like is what you get" [Serbedzija 2009] is the ultimate principle of this new generation of reflective technology. In the context of this work, the term *reflective* is used to denote a novel approach for highly personalized pervasive adaptation. This approach makes diagnosis of the psychological state of the user the central driver for the adaptive strategy. To optimize the system awareness of user's inner state, the approach combines several temporal control loops that deploy different observation data and allow for reasoning and "appropriate" system behaviour. Thus, the system response reflects the user state observing also its own reaction and performing a constant self-improvement. Each instance of human-computer interaction (HCI) has unique characteristics that may be determined by the person, the system, or the environment. A reflective system is designed to adapt to the specific features of the HCI (e.g. characteristics of the person, the task or the environment). This shift from designing for the general to the specific attributes of the user has been called individuation [Hancock et al. 2005]. This process is complex and requires proactive intelligent adaptation on the part of the system in order to tailor the interaction to the individual. Technological systems can fulfill their potential for individualization by monitoring the user over a period of time in order to both learn about preferences and to predict intentions/adaptation strategies in real-time.

A variety of application domains for this reflective approach to system design ranges from entertainment, E-learning and ambient assisted living to real-time embedded systems such as adaptive automation. For example, advertisement in public places may achieve greater influence if display devices could discover the level of interest from the viewer and adapt content in order to sustain interest levels [Beyer et al. 2009]. Smart homes [Bishof 2007] or smart vehicles [Serbedzija 2008] could increase safety and comfort if system adaptations were tailored to the individual. Furthermore, it is possible for reflective technology to be connected and for the representation of the user that drives the adaptive process to yield a pervasive and distributed ‘web’ of hardware/software environment that support the activities of the individual.

The paper brings together multidisciplinary advances in areas of physiological computing [Fairclough, 2009] and software engineering for autonomous and adaptive systems. One goal of physiological computing is to create a symmetrical form of human-computer interaction [Hettinger 2003] - where the system adapts to the state or behavior of the user, who in turn reacts to system adaptation. This is a genuine dialogue between person and machine that is dynamic, sustained and long-lasting. The biocybernetic loop [Pope 1995] describes the translation of data from physiological activity of the user to software adaptation. Adaptive control is an important part of the control theory where feed forward or feed backword control loops are used. A well known concept can be seen in IBM autonomic computing initiative [Kephar and Chess, 2003], where the Monitor, Analyse, Plan, Execute (MAPE) loop is defined. The MAPE-loops (or similar variants) have been playing an important role in a number of adaptive systems implementations [Kramer and Magee, 2007; Huebscher and NcCann, 2008].

A number of frameworks have been developed that support contextual and pervasive computing. The context toolkit [Dey et al. 2001] features rapid creation of context-aware pervasive applications, whereas Aura architecture [Sousa et al. 2002] allows content to follow its user through transparent content migration. However, these approaches focus more on transparent distribution in a pervasive context suited well for ambient assistance, lacking the major dimension of this approach, namely physiological computing. More recent solution for programming pervasive systems concentrates on mobile applications. The TOTA approach [Marco and Zambonelli 2009] proposes a coordination model that is event based, aiming at providing agents that can facilitate both the contextual activities and the definition of complex distributed coordination patterns. The TOTA approach, though promising for mobile pervasive agent-based systems would require inclusion of agent support into reflective framework. In a tradeoff between complexity and effectiveness, reflective approach has included simpler ontology-based concept to deal with psychological diagnoses and pervasiveness.

There are numerous solutions for adaptive processing in literature. Many approaches deal with adaptive computing and reflection in terms of dynamic changes within the software system itself. Most often, adaptive systems [McKinley et al. 2004] focus on middleware - the layers of services separating applications from operating systems and network protocols – allowing software to adapt dynamically to its environment. Often, reflection mechanism is deployed to enable software to modify its structure and behavior in run-time according to the changes in its execution environment [McKinley 2004]. This approach builds upon latest results and deploys state-of-the-art concepts in middleware and component composition. However, the major focus here is not adaptation of the software itself, but rather awareness-rich adaptive behavior relative to the contextual changes obtained from the persons involved and the environment.

This paper describes the concept of reflective pervasive technology both theoretically and pragmatically. Various techniques to monitor the behaviour or psychological status of the user are explained in details. The concept of biocybernetic loop is further developed to illustrate system adaptation in real-life situations. Different time scales for distinct modes of user-system coupling is achieved allowing for a short, medium and long term adaptation. The approach is fully implemented using ontology for system modeling and service and component orientation for structuring the software. A generic and distributed reflective framework eases system composition and deployment which makes the reflective applications truly pervasive. The approach is illustrated with a number of case studies.

2. Methods for Monitoring the User

Monitoring technologies used to capture the user status may be divided into three distinct categories: (1) overt actions (e.g. location, looking, pointing), (2) overt expression (e.g. changes in behavior associated with psychological expression), and (3) covert expression (e.g. changes in physiology associated with psychological expression). These options cover both overt and covert indicators of user state. Overt behavior and actions may be captured at a low- and fine-grained level of detail; the location and movements of the user may be logged via GPS, user activities in terms of active engagement with software/hardware may be monitored (e.g. recording keystrokes and cursor movements) and cameras can provide details of user actions, such as eye movements and gross actions, such as standing or sitting. This behavior can also be captured at high fidelity such as measuring indices of productivity linked to performance, e.g., number of tasks completed in a set time.

The measurement of overt expression has been heavily researched in the domain of affective computing [Picard 1997]. A number of systems have been developed to capture the expression of emotion as manifested with respect to facial or vocal expression [Russell at al. 2003] and body posture [Ahn et al. 2007]. These categories of expression involve subtle changes that are not always under conscious control. In this case, expressive behavior that is overt and observable is monitored and categorised as indicative of key emotional states [Bartlett et al. 2003]. There is a degree of overlap between this category of expressive behavior and actions described in the previous paragraph. The user may be identified as sitting in front of the computer according to one level of analysis, however the posture of the user may indicate engagement or disengagement with task activity [Ahn et al. 2007]

Covert expression is achieved when the user communicates with a computer system via physiological changes. These signals represent internal channels of communication between various components of human central nervous systems. There is a long literature in the physiological computing tradition [Fairclough 2009] inspired by work on affective computing [Picard 1997], specifically the use of psychophysiology to discern different emotional states and particularly those negative states such as frustration

[Kapoor et al. 2007] that both designer and user wish to minimise or avoid. A parallel strand of human factors research [Pope et al. 1995; Prinzel et al. 2003] has focused on the detection of mental engagement using electroencephalographic (EEG) measures of brain activity. The context for this research is the development of safe and efficient cockpit automation; see [Scerbo et al. 2003] for summary of this work. The same approach was adopted to monitor the mental workload of an operator in order to avoid peaks (i.e. overload) that may jeopardize safe performance [Wilson and Russell, 2007]. In these examples, psychophysiology is used to capture levels of cognitive processing rather than emotional states. Psychophysiology may also be used to quantify those motivational states underlying the experience of entertainment technology [Mandryk and Atkins 2007]. This application promotes the concept of adaptive computer games where software responds to the state of the player in order to challenge or help the individual as appropriate [Gilleade et al. 2005].

The use of overt or covert measures to capture user state are not intended to be mutually exclusive. The detection of user states is achieved via multimodal data collection [Pantic and Rothkrantz, 2003] where specific states are characterized by ‘fusing’ information from a variety of sensors, e.g. gestures, facial expression, pressure on the mouse, as well as psychophysiology. Recent work on the detection of user frustration [Kapoor et al. 2007] demonstrated the utility of this approach by combining multiple measures to predict subjective feelings of frustration. These authors measured skin conductance in combination with posture analysis, detection of head gestures (head shakes and nods), facial expression (smiling) and haptic monitoring. These measures were used to predict self-reported episodes of frustration, which was accurately detected in 79% of all cases (chance level = 58%); the strongest predictors of frustration were non-verbal expressions (fidgets, head velocity, postural changes) as well as skin conductance level. In this case, measures of user state originating from different sources are combined to provide a valid and reliable representation.

To summarize, the status of the user may be represented by overt actions/location, expressive behavior and covert psychophysiology. This representation of user status may be captured and stored in real-time and used in order to direct system adaptations, i.e. to tailor the response of the system to the needs and preferences of the individual. The reflective approach combines all these categories of measurements.

3. The Biocybernetic Loop

The dynamics of the reflective computing system are based upon the biocybernetic loop as described originally by [Pope et al. 1995]. This loop describes how psychophysiological data regarding the status of the user is captured, analyzed and converted to a computer control input in real-time. The function of the loop is to monitor changes in user state in order to initiate an appropriate adaptive response. The biocybernetic loop is designed according to a specific rationale, which serves a number of specific meta-goals. For instance, the biocybernetic loop may be designed to:

- Promote and sustain a state of positive engagement with the software/task
- Minimize health or safety risks inherent within the HCI

The capability of the biocybernetic loop to sustain engagement has been demonstrated within the context of the computer game [Rani et al. 2005]. The goal of research into biocybernetic control of adaptive automation is to avoid the use of automation during hazardous states of awareness, e.g. fatigue and boredom, when safety may be jeopardized [Prinzel 2002]. The same protective logic underpins the use of psychophysiology to detect negative states of frustration [Kapoor et al., 2007], which may have implications for the health and wellbeing of the user in the long-term. From a design perspective, the formulation of meta-goals for the biocybernetic loop is an important development. These goals are often implicit in the design of a technology and rarely defined with a high degree of precision. By contrast, the biocybernetic loop requires specific data in order to adapt to the individual in real-time; therefore, these goals are translated into specific directives about when to offer help or switch off system automation.

The biocybernetic loop is equipped with an array of adaptive interventions to promote specific meta-goals [Gilleade et al., 2005], e.g. to provide help, to give emotional support, to manipulate task difficulty. The triggering of these interventions is controlled by the loop in order to 'manage' the psychological state of the user. Correspondingly, the way in which person responds to each adaptation is how the user 'manages' the biocybernetic loop. This improvisatory crux achieves human-computer collaboration by having person and machine respond dynamically and reactively to responses from each other. It may be useful for the loop to monitor how users respond to each intervention in order to individualize and refine this dialogue. This generative and iterative model of HCI requires an elaborate repertoire of adaptive responses to cover the full range of possible outcomes over a period of sustained use of human-computer dialogue. The latter point is particularly important for 'future-proofing' the reflective computing system as user and machine are locked into a co-evolutionary spiral of mutual adaptation.

The biocybernetic loop represents a basic component of reflective technology where raw data from user behaviour and physiology are translated into computer control. The biocybernetic loop also incorporates the goals of system adaptation, e.g. to engage the user, to improve the safety of performance.

4. Temporal Dynamics of System Adaptation

The interaction between user and system via the biocybernetic loop, as defined here, may be differentiated in terms of timescales of system adaptation. The biocybenetic loop must respond initially to rapid changes in user state that fluctuate over minutes or hours. In the longer term, the process of adapting system performance to the traits of the individual, e.g. personality, preferences, proficiency, may take place over a timeframe of several days or even weeks. The system may also be designed to incorporate changes that take place over a longer timescale of months or years. The user is not a stable system and the representation of the user must evolve in line with the process of maturation and aging process. The development of the biocybernetic loop from a generic system to one that is tailored to a specific individual user may be described with respect to three phases of interaction:

- awareness/improvisation (seconds/minutes)
- adaptation/reciprocal coupling (hours/days)
- co-evolution (months/years)

The initial period of system *awareness* provides the technology with its first opportunity to monitor and assess the state of the user in situ. In reflective systems awareness is achieved through sensor devices that capture, quantify and operationalize events in the environment (including the user). At a fundamental level, awareness describes the 'field of view' of the system, i.e. the range of events that the reflective system is capable of monitoring and responding to. When the loop triggers an adaptive response during this period of *awareness/improvisation*, there is significant potential for error because the system is working from a generic template.

The *awareness/improvisation* phase of adaptation is characterized by trial-and-error learning on the part of the system. In order for this to occur, the biocybernetic loop must exhibit a sufficient level of awareness in seconds/minutes to assess the impact of system adaptation on the state of the user. For example, if an e-learning system automatically activates a help window in response to frustration, it stands to reason that effective assistance should bring about a fairly rapid reduction of frustration. If frustration is sustained or even increases after the provision of help, then the loop must assume that the information was inappropriate or useless from the user's perspective. Once this reflexive assessment has been completed, the loop may trigger a second category of assistance, which hopefully should be more useful. This kind of second-order monitoring and awareness (i.e. where the system monitors the user response to its own behaviour) is an important mechanism to allow the loop to learn about the preferences and adaptive strategies that are favored for a particular user during a specific type of task activity.

When the loop has been exposed to the user for a period of several hours or days, it is anticipated that a certain degree of personalization has been achieved via second-order monitoring described in the previous paragraph. At this stage, the reflective system is building a database regarding user preferences as well as the reliability of those preferences. In other words, the loop is learning to distinguish short-term fluctuations of the user state (that occur in seconds or minutes) from longer-term variations that are characteristic of individual traits. With repeated exposure to stereotyped patterns of target states (e.g. frustration, boredom, discomfort), the system is learning how to counteract those undesirable states in a way that is optimized for that person. Therefore, this second phase of adaptation describes a series of adjustments in the longer term that are described as *adaptation/reciprocal coupling*. This phrase is intended to capture the complementary bond between reflective system and user who are both learning about the other.

The reflective system builds a database via exposure to user behavior. With time and usage, this database extends both depth and breadth of knowledge regarding user behavior, expectations and preferences. This second phase of personalization involves the construction of a mathematical model describing the state of the user and how changes in user state should be translated into timely and appropriate system adaptation. It is anticipated that any database will be very stable as it incorporates many hours of system use over the preceding months and even years. What prevents the database becoming fixed and rigid is the changing status of the user; as a person uses the system to perform a task over a long period of time, they will become more skilled and in doing so, may possibly require different categories of system adaptation to meet their needs. The biocybernetic loop within the reflective system must remain alert to the possibility that user preferences will change over long periods of time due to skill acquisition or simply boredom or even the process of aging. At this point, the biocybernetic loop begins a second cycle of *co-evolution* which shares some characteristics with the initial stage of improvisation, but will often involve a process of fine-tuning existing responses to a slowly evolving process of individual change.

The extension of reflective technology across multiple systems is an orthogonal concept to the temporal dynamics of adaptation. Pervasive reflective adaptation introduces a new dimension where the adaptive responses from different devices must be coupled and coordinated in line with changes in user state and known preferences. This concept of pervasive reflective systems opens up a new dimension of of multiple control loops running simultaneously, sharing information about the user state or preferences between different systems. Pervasive technology highlights issues of synchronization (between different adaptive loops), conflicts (between different adaptive responses) and privacy (i.e. the capability of devices to share information about the user) for adaptation across distributed systems. In addition, pervasive technology brings the potential to enrich information about the current state of user by pooling information across systems.

The performance of reflective technology should be improved by sustained monitoring of the user state. As the system develops a database of user characteristics, adaptive interventions should appear to be intuitive and appropriate from the perspective of the user. In order to develop this facility, the system must develop a capability for second-order monitoring, i.e. capturing the impact of its own responses on the state of the user. This kind of second-order monitoring would be augmented by the requirement for a pervasive reflective system to monitor the activities of other adaptive loops, which may be trigger by the proximity of the user.

5. Reflective Approach

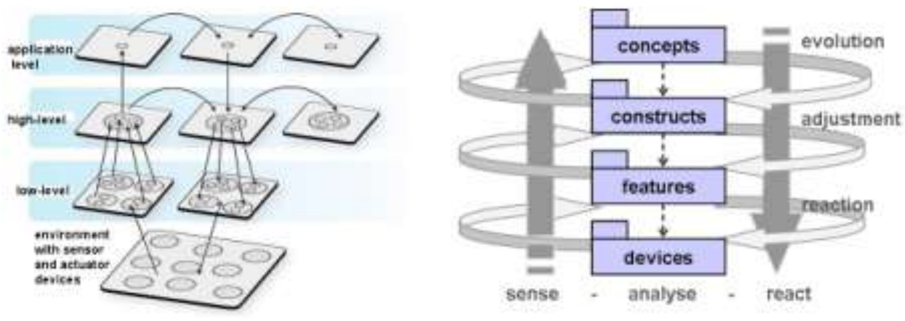
A reflective approach is defined as technology that has the potential to diagnose the user state, understand the context of human-computer interaction and to respond or adapt in a manner that is both timely and intuitive. Reflective technology is designed to maximise usability and the quality of the user experience via an implicit mode of human-computer interaction, i.e. where user state and behaviour is monitored on a passive basis. This type of technology can only be accomplished if the control system has the sensory means to understand covert and psychological aspects of human behaviour as well as explicit episodes of behaviour within the context of system usage. However, reflective systems also monitor the impact of their own real-time adaptation on the user state. For instance, if a system senses frustration and offers help information to the user, the software could monitor and respond to user response to that help information. This 'second-order' monitoring allows the system to monitor the efficacy of its own adaptive performance. The term "reflective" denotes a system's capability to respond appropriately (i.e. to *reflect*) to a complex situation. It should be clearly distinguished from the programming language vocabulary, where "reflection" [Demers and Malefant 95] denotes the capability of software to perform internal inspection. In order to function as reflective technology, the system must contain the following components:

- sensor apparatus to detect the behaviour of the user
- a dynamic representation of the behaviour/status of the user (awareness of the user)
- actuators or adaptable interfaces for system adaptation
- an adaptive controller containing the rules of the system to coordinate changes in user representation and system response

The reflective approach imitates the adaptation process as it appears in nature and is thus highly multi-disciplinary, requiring the know-how from psychology, social sciences, bio-engineering and computer science.

5.1 Reflective Ontology

From the previous examples, it can be seen that a reflective system is a multi-dimensional problem space with numerous orthogonal dimensions: awareness/improvisation as a immediate action caused by a biocybernetic loop, dynamic adjustment/reciprocal coupling represents adaptation over shorter period of time, coevolution is achieved via system memory gained over a longer period of time and pervasive collaboration. To cope with such complexity, a thorough modeling approach is required that encapsulates the essential problems described earlier and leads to a generic solution. The reflective ontology has been specified [Kock et al. 2009] for this purpose - to classify the problem space, construct major entities and describe the relations between structures. For the ontology description the UML - Unified Modeling Language is used [Kogut 2002] as a widely accepted method for high-level software modeling.



a) Conceptual view *b) Ontology view*

Figure 1. Reflective Systems Modeling

Figure 1(a) sketches this conceptual view. The basic idea is to organize the sense-analyze-react cycle in a hierarchical fashion. At the bottom, an environment exists that is populated with sensor and actuator devices. Above, low-level software components perform the first and the last steps in each sense-analyze-react cycle (features are extracted from sensors or are used to control actuators). The high-level components deal with abstract constructs describing the user context (e.g. psychological, cognitive or physical states), or high-level system goals. At the top application level, the diagnosed user constructs are related to system goals used to control the system. The reflective ontology in Figure 1(b) classifies the application domain into: (1) concepts, (2) elements, (3) features and (4) devices. Concepts are high level entities that include application scenario description, system's goals and chronology. Constructs encompass major entities like user states (emotional, cognitive, physical), and corresponding (re-) active effects (actuator states). Features are lower-level measurements used to derive constructs, e.g. actual measures that are used to construct the user state. Finally, at the lowest level devices represent end-appliances connected to the system. Each of these entities is further decomposed until most of system modules are fully described. Considering the temporal diversity of system adaptations, one can distinguish between immediate reactions (improvisation/awareness), reciprocal coupling (adaptation), and evolutionary changes (coevolution).

5.2 Modeling with reflective ontology entities

The devices module contains entries for sensors, actuators, or other devices, e.g. hard disks containing configuration settings. Features may be measurements like pulse or temperature stemming from sensors, actuating variables like tone or volume belonging to actuators, or facts like age or gender stemming from a hard disk. More complex examples for features are the mean heart rate or the mean temperature.

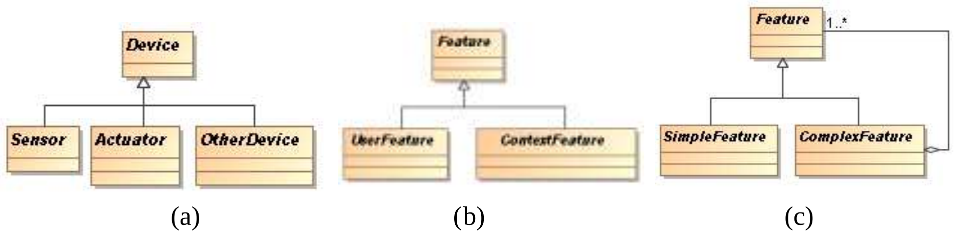


Figure 2. Modeling Devices and Features

Figure 2 illustrates both the structure and content of the device and feature entities: a) Sensor examples are camera, thermometer or blood pressure device, Actuator examples are led light or radio, a hard disk (used to store personalized data) and OtherDevice may be a communication interface (used for interaction with other reflective systems); b) a Feature can be a UserFeature or a ContextFeature, and c) the feature as either SimpleFeature or ComplexFeature Constructs model present user state: emotional (e.g. frustration), cognitive (e.g. high mental effort), or physical (e.g. comfort and movement). System goals are application oriented entities that drive the actuators (in accordance to the user state).

5.3 Reflective software architecture

Developing software to control biocybernetic loop involves tasks like real-time sensor/actuator control, user and scenario profile analyses, affective computing, self-organization and adaptation. To accomplish these requirements, a service- and component-oriented [Schroeder et al. 2008] middleware architecture, based on reflective ontology, has been designed that insures a dynamic and reactive behavior featuring different biocybernetic loops. According to the reflective ontology, the reflective software is grouped into three layers:

- Tangible layer - a low-level layer that controls sensor and actuator devices. It offers its atomic services (sensor measurements/actuator controls) to the rest of the system.
- Reflective layer - a central layer that combines atomic services of the lower layer with user profile and scenario description. This allows for more complex services and components that evaluate user emotional/cognitive/physical states and application situation and trigger system (re-)action, according to the application goals.
- Application layer - a high level layer that defines application scenario and system goals. By combining low and high level services and components from other layers, application layer runs and controls the whole system.

The overall design goal is to have a generic software architecture that mirrors the patterns of immediate, short and long term adaptation (exercising different biocybernetic loops) and is capable of dynamic configuration and efficient adaptation.

The **tangible layer** is the lowest layer of the system that features improvisation (immediate adaptation) deploying the concept of awareness. It interfaces/connects to the sensor and actuator devices and offers functionalities to the rest of the system. In the case of collecting the multiple psychophysiological features – it provides the reflective layer with concrete values that can be used in deducing higher level user states (e.g. increased blood pressure + increased activity from corrugator muscle = anger). It also executes commands (coming from the higher levels) needed to control actuator devices. The tangible layer is capable of autonomous processing and can execute urgent (safety) actions reflecting the overall system strategy and performing improvisation as a form of immediate adaptation. Sensors observe the environment and produce raw data that need to be classified and analyzed in order to offer some meaningful features about observed phenomena. Actuator devices are any kind of appliances that may do a useful work, influence users or the surrounding or fulfill a certain task. The tangible layer is dynamically configured according to the application needs. The tangible layer effectively implements the first phase of biocybernetic loop – the improvisation and can bypass the upper layer in order to perform safety actions. It also controls the communication devices that can exchange different reflective data structures, allowing for pervasive collaboration with other reflective systems.

The **reflective** layer is event driven, starting its loops with the features obtained from the tangible layer, combining them (according to the semantic rules defined within reflective ontology) to diagnose user states and suggesting action according to the system goals. The actions are further given to the tangible layer in form of outgoing services that triggers actuator control commands. The system reaction caused by reflective layer is slower than the tangible layer reaction. Several cycles are needed to collect a complete image of the situation and to consider impact that previous system reaction had (allowing fine-tuning of system functioning). Reflective layer deploy logic for both positive and negative feedback of the biocybernetic loop. Its functioning depends on the higher-level meta-goals described in the application scenario. It contains a rule-based engine that can interpret data dependence defined in the ontology as a correlation between the features and constructs. This, in a combination with short term system memory, effectively deploys the knowledge and expertise needed for diagnosing user states and psychological constructs implementing the second phase of the biocybernetic loop (reciprocal coupling). In another words, this layer provide the concept of dynamic adjustments as a medium term adaptation. To support reflective pervasive behaviour, the reflective layer can also perform exchange of higher level user information (e.g. user preferences, habits or “typical“ reaction in certain situation). In that way different reflective system can collaborate using each-other functionality.

The role of the **application** layer is to put the system components together, describe the application scenario and define system goals. At this level of software, a concrete correlation between sensing and actuating devices is given. The application layer corresponds to the highest ontology level of “concepts”. It implements the control strategy of the whole system by defining system goals and the strategy by which these goals should be achieved. It also takes into account the recorded user behaviour patterns. For example, if the user mood is to be improved by music and lightening, emotional user states are directly correlated with music player control and lighting control [Schroeder et al. 2008]. For the long term adaptation purposes the application layer keeps track of individual user characteristics which provides means of coevolution between system and user. This system goal cannot be achieved immediately. The process of coevolution takes time to transform a generic, anonymous system into a personal assistant that knows habits and emotional/cognitive characteristics of the persons, thus allowing for more effective and faster immediate and short term adaptive behaviour. As a reflective system (at the application level) is “aware” of individual user profiles, this information can be communicated between other reflective systems, personalizing new systems as the user approach them.

To support such multi-layer architecture, the reflective framework has been developed using the software components paradigm. Software components [Szypersk 1998] are units of software that make their communication capabilities explicit by means of ports (software types that describe the set of messages that can be received or send). The component based approach is dynamic, re-configurable and re-usable concept that allows for development of highly generic software.

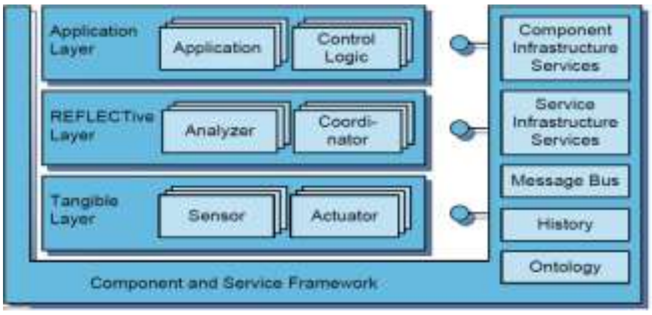


Figure 3. Reflective framework

Figure 3 illustrates the architecture of the reflective systems with its layers as well as generic component and service framework used to support the reflective ontology, a message bus and service and component infrastructures. The reflective framework is a distributed system that hides the physical location of the distributed parts and allow for transparent use of system components. The three hierarchical levels and a history module allows for the effective implementation of adaptation concepts at all three time scales, and communication infrastructure provide means for connecting to other systems fulfilling the needs for achieving pervasiveness. The framework with its supporting modules represent the running environment for each reflective system. With its transparent support for reflective ontology and all three phases of a byicybernetic loop processing, the framework builds a powerfull tool for development of concrete reflective systems.

5.4 Reflective Applications

Reflect framework offers a range of supportive tools for reflective application development. It also offers run-time environment for the low-level system functions leaving the system designer more time to concentrate on application goals and high-level concepts.

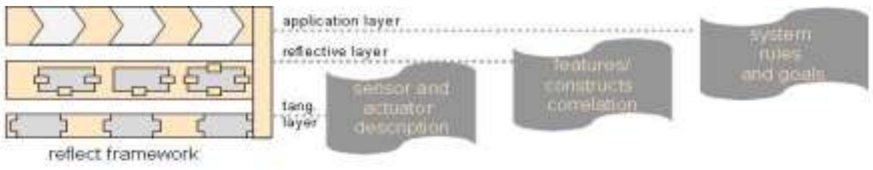


Figure 4. The deployment of Reflective Framework

Figure 4 illustrates the support that the developers gain through the use of reflect framework. There are three deployment phases: (1) selection the sensor/actuator devices, development of their reflect interface and the XML description of their features according to the reflect ontology (supported by the tangible layer) – this phase implements improvisation i.e. the short-term biocybernetic loop; (2) specification and the XML description of the psychological constructs that should be diagnosed with the provided sensor devices, as well as higher level actuator control action (that should affect the user state) – this phase is supported by the reflective layer components and corresponds to the short-term adaptation; (3) definition and the description of the rules and system goals that control the whole system and resolve possible conflicts in case of different control strategies – this phase corresponds to the long-term adaptation within a biocybernetic loop and is supported by the application layer components.

Due to the pervasive character of the reflective framework, different application can collaborate within each other or be amalgamated into a single operational entity. In that case features and constructs from one system can be combined with those from another system, thus enriching and complementing each other. The example of reflective pervasive technology is a complex case where several loops, which may be based on different sensors or data and may be designed with various goals or agendas, are active simultaneously. The goal of the reflective system is to coordinate adaptive loops that run in parallel by amalgamating each at the tangible, reflective and application layer. Therefore, distinct loops may share data or develop complex, two-dimensional representations of the user [Fairclough 2009] where physiology and behaviour are combined at the tangible layer. Complex representations of the user would be expressed at the reflective layer; more importantly, representations of a pervasive and adaptive system would also be generated at this level of hierarchy. This is very important as the components of a pervasive system and the adaptive capability available to the user would fluctuate over time, i.e. the pervasive system surrounding the user may change as he/she travels from home to the office. The application layer represents an ‘executive’ function in the sense that the goals of all active loops would be coordinated at this point and conflicts between loops with antagonistic agendas can be resolved.

6. Reflective Case Studies

The development and deployment of reflective applications is explained through three separate case studies which are at the end joined to a single system, illustrating a pervasive character of the reflective approach.

6.1 The Mood Player (Emotional Loop)

The mood player is a music player which selects music to match the mood of the user or selects music that may improve her/his emotional state [Zwaag and Westerink, 2010]. Several steps are needed for realizing the music player that operates within a closed-loop system. First, the input of the system must be defined so that it covers the current mood, measured unobtrusively with physiological sensors (e.g. heart rate, facial electromyography). The target emotional state selected by the user e.g., to feel more positive, or to feel more energetic, represents a second input to the system. The next step in the closed loop system involves pre-processing data input to obtain the significant features from the raw physiological signals. These features function as input to drive the selection of music. Third, music is selected based on the predicted influence the song will have on the diagnosed emotional state. These steps complete the biocybernetic loop for the mood player.

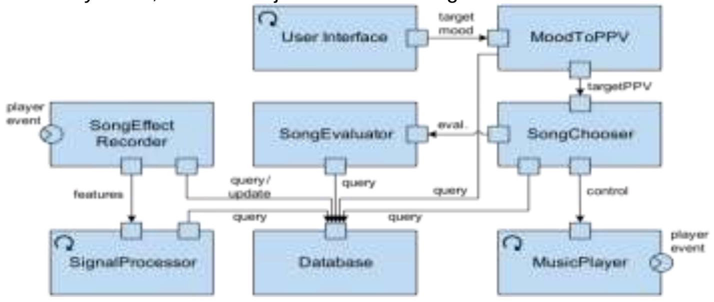


Figure 5. The mood player component architecture

The specific reflective component architecture [Schroeder et al., 2008] used for the application of the reflect framework applied on the mood player case study is presented in figure 5. The tangible layer controls the devices and the corresponding processing algorithms for the physiological sensors and music player. The physiological sensors sends the information to the SignalProcessor, which performs the pre-processing of the signals and passes information to the tangible layer. There is also a database component with chronological information for each user. The MusicPlayer component plays the music for the user. At the reflective layer there is the “SongEvaluator” component that fuses the sensor data with previous measurements and system goals and sets the “appropriate” music. This component receives the request to which emotional state the player should adapt to. It further instructs the SongChooser component which title to play. The Application Layer contains MoodtoPPV component that takes the target mode (system goal) into account when deciding which music to play for diagnosed emotional state. This case study features both medium term and long term adaptation. Mood player requires a number of biocybernetic cycles (over a couple of minutes) to detect the emotional state and the effect that the music has on it. Also in a longer term, as system uses chronological personal data, the system improves its performance evolving through confluence with the specific user.

6.2 The Cognitive Monitor (Effort Loop)

The main goal of cognitive monitor is to recognise instances of mental overload during a task and to perform action to alleviate this situation. For example, if a user is overwhelmed performing a multi-tasking control, the reflective system may automate sub-tasks in order to reduce mental workload. In the vehicular domain, the reflective assistant responds to dynamic changes in roadway environment that increase the mental workload of the driver, e.g. unfamiliar road, fog, high traffic density. The system responds to periods of high driver mental workload via a range of assistive strategies, some of which are covert (e.g. switch all incoming telephone calls to voicemail to prevent further distraction) or overt (e.g. activate driver aid devices to provide a warning if the vehicle drifts out of lane or follows at lead vehicle at insufficient distance). The form of support offered is on level of short and medium term adaptation as the system responds immediately, when sensors discover irregularly high physiological measures (indicative of increased workload) or after a short period of time needed to diagnose the situation (taking into account driver’s physiology and driving characteristics). The cognitive monitor also uses the vehicular CAN bus to collect data about speed, acceleration and steering wheel corrections. The warnings are restricted to the computer console installed in the cockpit.

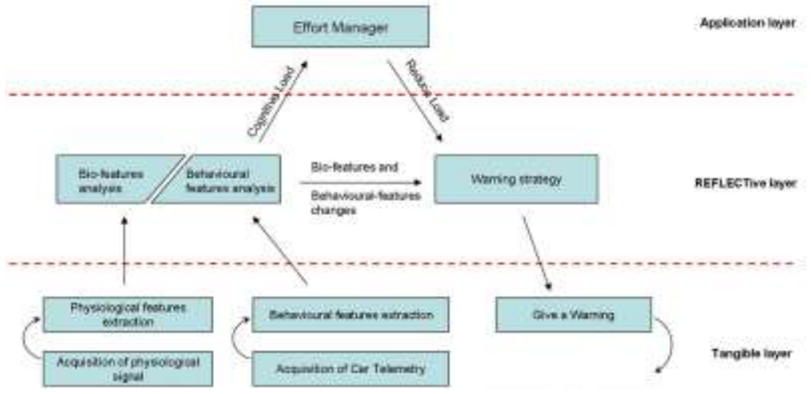

Effort Manager
Application layer
Cognitive Load
Reduce Load
Bio-features analysis
Behavioural features analysis
Bio-features and Behavioural-features changes
Warning strategy
REFLECTive layer
Physiological features extraction
Acquisition of physiological signal
Behavioural features extraction
Acquisition of Car Telemetry
Give a Warning
Tangible layer

Figure 6. Cognitive effort system set-up

Figure 6 presents the cognitive monitor system setup implemented on a small tablet PC with a touch-screen placed on dash-board between the driver and a co-driver. The system has been tested several times in the car, driving under various weather and road conditions. It features short and medium term adaptation as for an extreme physiological features – system must react immediately, where as for the minor changes in physiology, a system need more cycle and contextual information (about driving) to diagnose the state (medium-term adaptation).

6.3 The Adaptive Seat (Comfort Loop)

The aim of the seat adaptation is to monitor the postural behaviour of a driver and to use these data to detect episodes of driver discomfort. In these cases, the seat should re-shape to counteract the discomfort. The adaptive seat contains a number of pressure sensors that creates the pressure map indicating the sitting behaviour (comfort). These raw data are processed to calculate the Centre Of Pressure (COP), a parameter which is broadly used to study human sitting posture [Cho 2005]. Once the discomfort is detected, the air pumps are activated to inflate/deflate the air cushions within the adaptive seat.

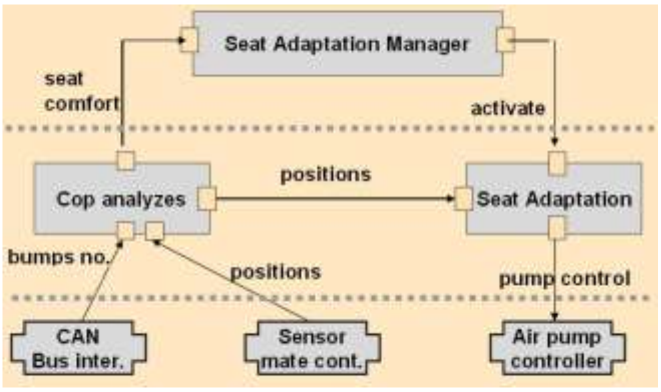


Figure 7. Simplified seat adaptation system component architecture

Figure 7 illustrates component architecture of the seat adaptation system. To make the system function more precisely, sensor information must be combined with the driving information. Driving parameters are extracted from the CAN bus (e.g. the current discomfort may be caused by sudden short-term acceleration or a bumpy road, which should not affect the seat adaptation strategy). The reflective layer deploys a rule based reasoning to perform the comfort diagnose and to trigger further system actions. A rule-based reasoning is a sub-system whose knowledge base is represented as a set of rules and facts. It consists of IF-THEN rules, a collection of facts and some inference engine [Browne 2009; Drool 2010].

Table 1. Example of rules used in seat adaptation example

IF // -- Physical Analysis Number of bumps = HIGH THEN Physical state = DISCOMFORT	IF // -- Physical Analysis Number of bumps = LOW AND COP speed = HIGH THEN Physical state = DISCOMFORT
IF // -- Seat Adaptation Manager Emotional state =NEUTRAL AND Cognitive load = NOT HIGH THEN ACTION_MODE	IF // -- Seat Adaptation ACTION_MODE AND Car setting = COMFORT AND Number of bumps = HIGH AND Cushions = OFF THEN Cushions = ON
IF // -- Seat Adaptation ACTION_MODE AND Car setting = COMFORT AND Number of bumps = LOW AND COP speed = HIGH AND Cushions = ON AND Time frame = 20 AND MOD (time, time frame) = 0 THEN Cushions = OFF	IF // -- Seat Adaptation ACTION_MODE AND Car setting = COMFORT AND Number of bumps = LOW AND COP speed = HIGH AND Cushions = OFF AND Time frame = 20 AND MOD (time, time frame) = 0 THEN Cushions = ON

Table 1 illustrates the rules used to program the seat adaptation strategy. The code that implements the rule engine is data driven, easy to modify and fits well to the component- and ontology-based architecture of the reflect framework. It also allows for a pervasive diagnoses taking care about possible conflicts within several biocybernetic loops or fusing new sensor features that may come from some other device (CAN bus) or other reflect systems (mood player).

The seat adaptation features medium and long term adaptation as it requires a number of iteration to diagnose the comfort of the driver. Also, keeping chronology for the specific driver improves system behaviour on a longer term.

6.4 Reflective Vehicle

The reflect applications outlined in the previous sections are described as separate systems reacting upon users' emotional cognitive and physical experiences, also called emotional, effort and comfort loops. The reflective vehicle system [Reflect, 2011] is a combination of all three modules configured into a single reflective application. The resulting system requires simple modifications at the reflective and application level, offering a “co-driver” style of a support [Serbedzija, 2008]. The system helps driver throughout the ride, observing her/his emotional, cognitive and physical condition and actively assisting in the process of driving.

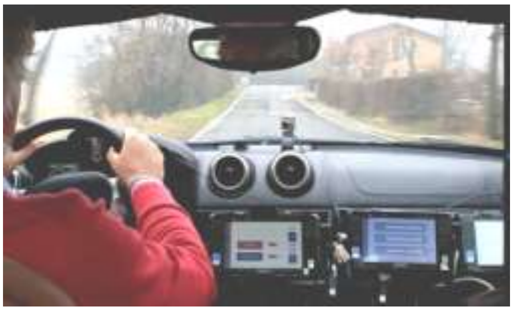


Figure 8. Reflective vehicle

Figure 8 shows the cockpit of the reflective vehicle. The sensors are: CAN bus, cameras, physiological measurement devices, equipped with reflective interface; the actuators are: adaptive seat and the 3 board computers (playing the role of the system monitor, mobile phone and media player). The functionality of each of the system components (three loops) remain the same, the performance and resource usage is not three times higher, but remains slightly higher than average of any of the single components.

Based on the information contained within the system, jet another loop (subsystem) has been added: a driving loop. It combines the sensor input from emotional, cognitive and comfort loops as well as data from the vehicular CAN bus, and monitor the driver's behavior.

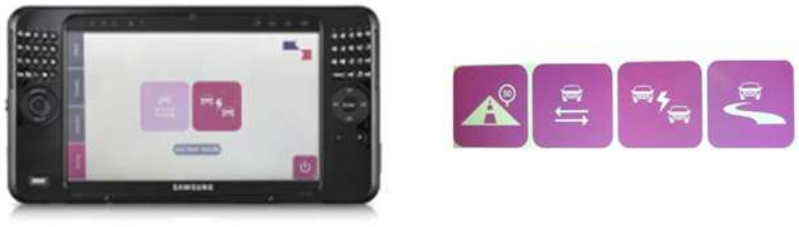


Figure 9 – Driving loop monitoring

Figure 9 illustrates the screen used for monitoring the drivers' behavior. It provides the warnings in case of four situations: sudden speeding (exceeding safety speed), aggressive driving (approaching road holding limit), acceleration feedback (abrupt control of the vehicle) and swerving feedback (erratic swerving). This addition effectively illustrates a pervasive nature of the reflective approach by its capability of re-using various sensory inputs from different loops. A combination of several reflective sub-systems enriches and refines the quality of each diagnosis without major computing overhead.

6.5 Implementation and Performance Details

The Reflect framework is implemented in the Java programming language (Java 1.6) using the Eclipse RCP Tools and Equinox OSGi environment [OSGI, 2005]. To optimize the distribution and pervasive character, a custom distribution is developed making the placement of different parts of the reflective system fully transparent and independent of the hardware configuration and the number of computers it runs at. There is also an efficient web server that allows monitoring on the smart phone devices (connected via *ad hoc* network)[Reflect 2011].

The vehicular installation runs on a network consisting of one notebook and three ultra tablet PCs (placed at the dashboard), as shown on Figure 8. A notebook is needed to run a compute-intensive imaging software; the three PCs share an equal distribution of the reflect framework and three adaptive subsystems (cognitive monitor, mood player and adaptive seat). The first touch screen is used for system control and monitoring of the application layer. Other two screens monitor the reflective layer (high level user states and tangible layer (sensor input). Actually, none of the screens is necessary for the system operation; they are used to monitor a seamless man machine interaction conducted by multiple control loops involving sensors and actuators.

Table 2: REFLECT system figures

	Mood Player	***Adaptive Seat***	***Cognitive Monitor***	***All together***	***Comment***
Size (source Java code)	8,6K lines	1,3K lines	2,6K lines	40,8	Reflect Framework: 28,4K lines (present in each) and tools for monitoring: 19,4K lines
Run time memory use	30 MB	28MB	30MB	35MB	Most of memory is used by OSGI and external libraries (common to all)
External devices	Nexus 10	Pressure map & accelerator	Polarion Belt	CAN Bus	Physiological measurements
CPU load at one ultra PC	60%	67%	30%	-	***Min. Requirements***: Windows XP, 128 MB RAM, 60MB hard disk, TCP/IP
Average CPU load for all UPCs				60%	
Development Environment	Java 1.6, Eclipse RCP Tool, Equinox OSGi, MySQL, Reflect custom developers tool (for interfacing external devices and rule engine)				

Table 2 illustrates some of the performance figures and implementation details of the vehicular prototype. It can be seen that even with low-performance ultra PCs, the system runs effectively and is flexible for further extensions. Up to the authors' knowledge there are no similar systems that could be used for performance or functional comparison.

7. Conclusion

The paper presents a novel approach in building smart technology responsive in real-time and across different devices. Reflective approach is defined and implemented through multi-level biocybernetic loops. At different time scales, biocybernetic loops feature different kinds of adaptation, from short term awareness/improvisation via medium term dynamic adjustments (reciprocal coupling) to a long-term adaptation (coevolution).

The reflective architecture is described as an effective implementation of the physiological computing concept. It is based on high-level modeling described with reflective ontology. The ontology developed is based on the existing approaches, but includes wider range artifacts needed for effective implantation of autonomous, adaptive and awareness-rich behavior, sensitive to various human experiences (psychological, social, behavioral). The final generic framework allows for high-level programming and deployment of a number of reflective applications featuring multiple levels of adaptation and pervasive behavior. Component based implementation deploys state-of-the art software technology yielding fast prototyping, numerous development tool support and efficient implementation of dynamic, reconfigurable, event driven and pervasive adaptive systems. The pragmatic orientation of the reflective approach is shown by three case studies and their joint deployment in the vehicular prototype.

A spectrum of challenges remains to be further researched. From the software/hardware point of view, more work is needed in the area of fine wireless sensor devices that should enrich psycho-physiological measurements and improve data analyses (artefects exclusion, efficient diagnosing). Novel methods are required for improving and optimizing adaptive control, learning and personalization. In the domain of human sciences further know-haws is required from physiology and medicine, especially in the domain of more precise diagnosing and from social sciences to supplement the system awareness with social and ethical perception. Pervasive character of reflective systems requires further investigation, especially in their shared use of different control loops. Finally more insight should be given into domain of further applications as well as on possible impacts, both positive and negative, that new technology has on us.

Acknowledgement

Most of the work presented here has been done under the REFLECT project [Reflect 2011] (project number FP7-215893) funded by the European Commission within the 7th Framework Programme, pervasive adaptation initiative. The authors acknowledge numerous constructive discussions with other members of the Reflect project team who contributed in the development of the concepts described in this paper.

References

Ahn, H., Teeters, A., Wang, A., Breazeal, C. and Picard, R. W. 2007. Stoop to conquer: posture and affect interact to influence computer users' persistence. Paper presented at the *Second International Conference on Affective Computing and Intelligent Interaction*, Lisbon, Portugal.

Bartlett, M., Littlewort, G., Fasel, I., and Morvellan, J. R. 2003. Real time face detection and facial expression recognition: development and application to human-computer interaction. Paper presented at the *Computer Vision and Pattern Recognition for Human-Computer Interaction*, Vancouver, Canada.

Beyer, G., Kroiss, C. and Schroeder, A. 2009. Person Aware Advertising Displays: Emotional, Cognitive, Physical Adaptation Capabilities for Contact Exploitation. *1st Workshop on Pervasive Advertising at Pervasive*, Nara, Japan. 2009.

Bischoff, U., Sundramoorthy, V. and Kortuem, G. 2007. Programming the smart home, *Proc. Intelligent Environments, IE 07. 3rd IET International Conference*, pp:544 – 551

Browne, P. 2009. *JBoss Drools Business Rules,* Packt Publishing, Birmingham, April 2009

CAN Bus, 2010. CAN Bus, Vehicular Controller Area Network, available at: http://de.wikipedia.org/wiki/Controller_Area_Network

Cho, W.H. and Choi, H. 2005. Center of pressure COP during the Postural Balance Control of High-Heeled Woman, *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, Shanghai, China, pp. 2761-2764.

Dey, A.K., Abowd, G.D. and Salber, D. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2):97–166.

Demers, F-N. and Malenfant, J. 1995. Reflection in Logic, Functional and Object-oriented Programming. *Proceedings of the IJCAI '95 Workshop on Reflection and Metalevel Architectures and their Applications in AI*, pp. 29-38..

Drools, 2010. Drools Documentation Library, available at http://www.jboss.org/drools/

Fairclough, S. H. 2009. Fundamentals of Physiological Computing. *Interacting With Computers*, 21, 133-145.

Gilleade, K. M., Dix, A. and Allanson, J. 2005. Affective videogames and modes of affective gaming: assist me, challenge me, emote me. *Proceedings of DiGRA*.

Hancock, P. A., Pepe, A. A. and Murphy, L. L. 2005. Hedonomics: the power of positive and pleasurable ergonomics. *Ergonomics In Design*, 131, 8-14.

Hettinger, L.J., Branco, P., Encarnaco, L.M. and Bonato, P. 2003. Neuroadaptive technologies: applying neuroergonomics to the design of advanced interfaces. *Theoretical Issues in Ergonomic Science*, 4, 220-237.

Huebscher, M.C. and McCann, J.A. 2008 A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys*, 40(3):1–28, 2008.

Kapoor, A., Burleson,,W. and Picard, R.W. 2007. Automatic prediction of frustration. *International Journal of Human-Computer Studies*, 65: p. 724-736.

Kephart, J.O and Chess, D.M. 2003. The vision of autonomic computing. *Computer*, 36(1):41–50.

Kock, G, Ribaric, M. and Serbedzija, N 2009. Modelling User-Centric Pervasive Adaptive Systems - the REFLECT Ontology. In: *Intelligent Systems for Knowledge Management*, Vol. 252. Nguyen, Ngoc Thanh; and Szczerbicki, Edward Eds. Series: "Studies in Computational Intelligence", Springer, ISBN: 978-3-642-04169-3.

Kogut, P. et al. 2002. UML for Ontology Development, The Knowledge Engineering Review, 2002, 171, p. 61-64.

Kramer, J. and Magee, J. 2007. Self-managed systems: an architectural challenge. *Future of Software Engineering* (FOSE '07), IEEE Computer Society Press, 259–268.

Mandryk, R. L. and Atkins, M. S. 2007. A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. International *Journal of Human-Computer Studies*, 65, 329-347.

Marco M., M and Zambonelli, F. 2009. Programming pervasive and mobile computing applications: The TOTA approach, *ACM Transactions on Software Engineering and Methodology*, 18(4).

McKinley, P.K., Sadjadi, S.M., Kasten, E.P. and Cheng, B.H.C., 2004. Composing adaptive software, *Computer*, 37(7), 56 - 64.

Norman, D.A. 2007. *The Design of Future Things*, New York: Basic Books.

OSGI 2005. About OSGI Service Platform, OSGI White paper, OSGI Alliance, available at: http://www.osgi.org/wiki/uploads/Links/ OSGiTechnicalWhitePaper.pdf

Pantic, M., and Rothkrantz, L. J. M. 2003. Towards an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 919, 1370-1390.

Picard, R. W. 1997. *Affective Computing*. Cambridge, Mass.: MIT Press.

Pope, A. T., Bogart, E. H., & Bartolome, D. S. 1995. Biocybernetic system evaluates indices of operator engagement in automated task. Biological Psychology, 40, 187-195.

Prinzel, L. J. 2002. Research on Hazardous States of Awareness and Physiological Factors in Aerospace Operations No. NASA/TM-2002-211444. Hampton, Virginia: NASA.

Prinzel, L. J., Parasuraman, R., Freeman, F. G., Scerbo, M. W., Mikulka, P. J., and Pope, A. T. 2003. Three experiments examining the use of electroencephalogram, event-related potentials, and heart-rate variability for real-time human-centred adaptive automation design No. NASA/TP-2003-212442: NASA.

Rani, P., Sarkar, N. and Liu, C. 2005. Maintaining optimal challenge in computer games through real-time physiological feedback. *11th Human-Computer Interaction International*, Las Vegas, USA.

Reflect 2011. REFLECT project - Responsive Flexible Collaborating Ambient, available at, http://reflect.pst.ifi.lmu.de/

Russell, J. A., Bachorowski, J. and Fernandez-Dols, J. 2003. Facial and vocal expressions of emotion. *Annual Review of Psychology*, 54, 329-349.

Scerbo, M. W., Freeman, F. G., and Mikulka, P. J. 2003. A brain-based system for adaptive automation. Theoretical Issues in Ergonomic Science, 41-2, 200-219.

Schroeder, A., Zwaag, M. and Hammer, M. 2008. A Middleware Architecture for Human-Centred Pervasive Adaptive Applications, *1st PerAda Workshop at SASO 2008*, Venice, Italy, Oct. 21th 2008.

Serbedzija, N. et al. 2008. Vehicle as a Co-Driver, *Proceedings First Annual International Symposium on Vehicular Computing Systems - ISVCS 2008*, Dublin, Ireland.

Serbedzija, N. and Fairclough, S 2009. Biocybernetic Loop: from Awareness to Evolution, *IEEE Congress on Evolutionary Computation IEEE CEC 2009*, Trondheim, Norway.

Sousa, J.P. and Garlan, D. 2002. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. *17th World Computer Congress - TC2 Stream / 3rd Conf. Software Architecture*, Deventer, Netherlands, pp. 29–43.

Szyperski, C. 1998. *Component Software: Beyond Object-Oriented Programming*, ACM Press and Addison-Wesley.

Wilson, G. F., and Russell, C. A. 2007. Performance enhancement in an uninhabited air vehicle task using psychophysiologically determined adaptive aiding. Human Factors, 496, 1005-1018.

Yannakakis, G. N., Hallam, J., and Hautop Lund, H. 2007. Entertainment capture through heart rate activity on physical interactive playgrounds. *User Modeling and User-Adapted Interaction*, 18, 207-243.

Zwaag van der, M.D and Westerink, J. 2010 Musical induction and persistence of mood. Proceedings of the 11th International Conference of Music Perception and Cognition, Seattle, Washington, pp. 57-60.