Affective Embedded Systems: a Requirement Engineering Approach

Millys F. A. Carvalhaes, Adson F. da Rocha, Marcus Fraga Vieira and Talles Marcelo G. de A. Barbosa

***Abstract*— This paper proposes an embedded system development process able to deal with affectivity requirements. This process includes W3C tools, such as EmotionML and SysML, in order to specify cognition and affectivity as system requirements. It combines three different integration models: Sparx's [1] and Wolf's System Development Processes [4] and Russell's Intelligent Agent Specification [5]. In order to validate our model, a pet robot case study was built. This application can be useful in many homecare applications, such as assistive robots for capturing attention of autistic children and robot therapy for elders. Moreover, it represents a futuristic embedded system design space for applying affective computing.**

***Index Terms* — Affectivity, Affective computing, System Development Process, EmotionML, SysML.**

I. Introduction

The contemporary robotics technology is broadening its applications from factory to more general-purpose applications in domestic and public use, e.g., partner to the elderly, rehabilitations, search and rescue, etc. If robotics technology is to be successful in such complex, unstructured, dynamic environments with high level of uncertainties, it will need to meet new levels of robustness, physical dexterity and cognitive capability [2]. In addition, these new applications demand different types and levels of affectivity as new generation of system´s requirements.

Emotions play an important role in human experience and it may influence decisions, perception, learning and communication. According to Schröeder [3], a great challenge in the human-computer interaction area refers to human adaptation instead of machine adaption in order to satisfy human necessities.

Cognitive Systems and Affective Computing areas should be integrated in order to produce new embedded systems applications. However, traditional requirement analysis inherits from Software Engineering Area does not accommodate these new applications of embedded systems. It does not cover all aspects needed for embedded system design, such as affective requirements. In addition, it is very difficult (time consuming) to define certain aspects of complex embedded systems, such as emotions and cognition. These new aspects increase the complexity of some usual attributes tests, such as correctness, completeness, verifiability, consistency, modifiability and traceability.

In this paper cognition and affectivity requirements have been combined from the software engineering perspectives to propose a practical system's development process. The proposed methodology is based on three different approaches: Sparx System's Development Process [1], Wolf's Development Process [4] and Russell's Intelligent Agents Specification Methodology [5]. As case study an emotive pet robot was build. It can be useful in many homecare applications. Assistive robots for jointing attention of autistic children and robot therapy for elders represent typical applications. Preliminary result reveals an efficient development framework that can be easily adapted to different types of embedded applications specially those required to include affectivity requirements.

II. MATERIALS AND METHODS

A. *SysML and EmotionML*

System Modeling Language (SysML) is a general purpose modeling language for system engineering, derived from a UML customization proposed by OMG consortium to minimize the gap between software and system engineering. SysML intends to unify different modeling languages currently used by system engineers [6]. For example, SysML has been applied for specifying electronic fuel injection systems in order to reduce design complexity because all those supported diagrams improve the designer comprehension about system properties [7]. However, traditional automobile designs do not consider emotional requirements. For example, certain data obtained from the car driver physiological condition (e.g. stress levels) could be used as input to system controllers improving system´s efficacy and adding new functionalities such as those related to car accident prevention.

EmotionML support emotion representation by standard Extensible Markup Language (XML). It provides tags to describe emotion, producing a fine-grained description and a well-structured specifications artifact. It avoids ad hoc

Millys Fabriele is with Biomedical and Biomechanics Laboratory, Federal University of Goiás, Goiânia, GO 74001-970 Brazil Adson F. da Rocha is with UnB-Gama Faculty, University of Brasília, Gama, Brazil

Marcus Fraga Vieira is with Biomedical and Biomechanics Laboratory, Federal University of Goiás, Goiânia, GO 74001-970 Brazil

Talles Marcelo G. de A. Barbosa is with the Department of Computing, Pontifical Catholic University of Goiás, Goiânia, Brazil

attempts to deal with emotions requirements, offering a W3C compatible solution. However, the most typical misunderstanding appear during the specification activities because system designers tend to ignore different levels of affectivity related to a single emotion classification [3].

B. Rationality Specification Model

According to Russell and Norvig [5], rationality means take the best action from the input data available at this time. Rationality is related to an entity, called intelligent agent. A Rational Agent is the one that does everything right, perceiving the environment through sensors and acting though actuators. Conceptually, for each agent function there is at least one performance measure. Besides, all agent features description must be arranged in a tuple organization, called PEAS (Performance, Environment, Actuators and Sensors). PEAS introduce a development process classified as iterative and incremental. To start with PEAS is necessary to define the agent functions. Associated with each agent function there is a PEAS description. For example, an environment description related to a single pet robot agent function can be described as partially observable, stochastic, episodic, dynamic and continuous.

The main PEAS advantage is its inherent simplicity to describe environment´s properties. However, the main disadvantage is it's time-consuming because PEAS requires a description set for each system´s agent function, otherwise the system's definition results in a poor quality description and become inadequate to express rationality.

C. Hybrid Embedded System Development Process

A development process is a predefined set of activities. It helps designers to produce a system reducing costs and time, but increasing quality.

According to Wayne Wolf [4], an embedded system development process is very important for three reasons, it keep a scorecard on a design to ensure that was done and everything that needs to be done, allow to develop a computer-aided design tools, and finally makes more easier the communication between a design team members. By defining the process, team members can easily understand what they need to do, what they should receive for other team member and what they need to deliver when they complete the assigned step.

This paper proposes a hybrid development process, based on Sparx's [1] and Wolf's models [4]. Certain activities templates were inherited from Sparx's process, such as *System requirements, Structure, Behavior, Simulation, Implement Hardware and Implementation*. However, certain modifications were made to include applications necessities, such as PEAS description. From Wolf's process *Specification, Components and System Integration* were incorporated. In addition, new activities were proposed in order to better accommodate affective requirements. These requirements should be described according to EmotionML syntax during the system requirements definition.

D. SysML Tools an alternative approach

Since system modeling tools have a high cost, the acquisition of these tools to several projects may become unaffordable. Thus, considering reducing costs, in this paper were used UML tools combined with stereotypes which allow UML metamodels extend SysML models to different purposes.

A stereotype is an extensibility mechanism of UML that allows designers to extend the UML vocabulary in order to create or represent new elements, derived from other existing models, e.g. Fig. 1 shows the requirement diagram defined by SysML Specification. While specific system modeling tools provide support to this new kind of diagram, UML tools do not offer a specific diagram to represent requirements. However, by the means of stereotypes designer might represent requirement modifying existing models, such as, Class Diagram presented in Fig. 2. The class diagram is defined by stereotype <<Class>>; the same diagram could conceptually represent a functional requirement diagram using a stereotype <<functionalRequirement>> or a non-functional requirement diagram using a stereotype <<nonFunctionalRequirement>>.

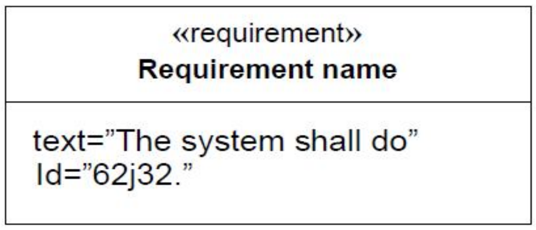


Fig. 1 - Requirement model defined by SysML specification

Therefore, free or low cost UML tools could be used to system modeling by means of stereotypes. However, UML tools not support source code generation based on SysML specification like provide specific SysML tools. Thus, in this context all source code derived from diagrams should be created manually.

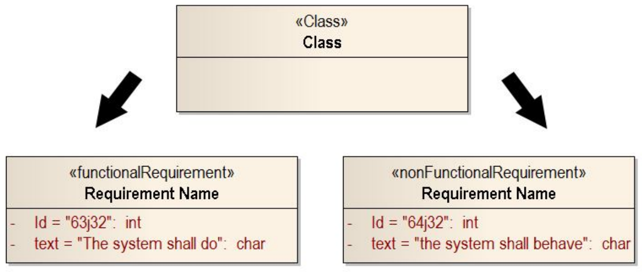


Fig. 2 - Customization of UML Class Diagram to represent requirements by means of stereotypes

III. THE AFFECTIVE DEVELOPMENT PROCESS

The requirement definitions are the first activity presented in Fig. 3. **System Requirements** are informal descriptions of the system. These requirements represent customer's necessities and what they are expecting from the proposed system.

The next activity is creates a more detailed definition of requirements. It is called **Specification** and it is based on Volere techniques [8]. Specification activity refines requirements into a formal specification including enough information to begin the system architecture design.

The **Architecture Definitions** (third activity) explore functional modularization. It defines how functionalities should be implemented. In this development process it is divided in two main activities. The first activity (called **Structure**) is responsible for defining all system components while **Behavior** activity describes the system's functionalities and how they relate to each other.

An embedded application usually has a huge variety of attributes, sometimes expressed as mathematical equations. All these attributes must be specified at **Constraint Blocks** activity.

To verify produced prototypal a simulation activity was proposed. **Simulate** is useful for system testing and it must be performed assisted by Software Verification Tools.

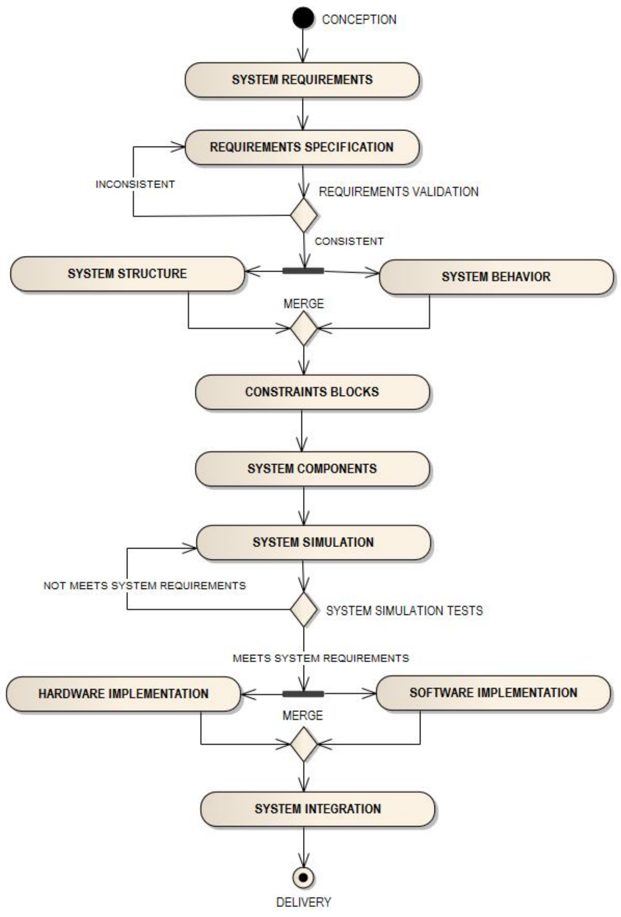


Fig. 3 - An embedded system development process.

Components activity (see Fig. 3) defines system parts. Moreover, it can be used to ensure that the components are in conformance with system architecture.

The next step is related to **hardware and the software implementation**. Finally, **System Integration** (the last activity) is responsible for producing a complete system version. It includes several tests to assure uncovered bugs will be fixed.

IV. INSTANCE OF THE AFFECTIVE DEVELOPMENT PROCESS TO DIFFERENT APPLICATIONS

The advantage of the Affective Development Process is its applicability to several applications in various domains. Since designers can extend development process to support specific project features by the means of stereotypes, it allows the process to fit in several applications, such as, Homecare and Cognitive Rehabilitation using virtual reality in a Biomedical Engineering approach. Moreover, using free or low cost UML tools reduces project cost.

A. Cognitive Rehabilitation using virtual reality in a Biomedical Engineering approach

The cognitive rehabilitation therapy is considered a central point for treatment programs developed for individuals with encephalic trauma [14]. For example, a stroke complication is the loss of basic cognitive functions such as memory, attention and executive functions, which control other abilities and behaviors [13].

Therapists use different techniques to improve the functions recovery during rehabilitation. However, the optimal learning occurs when people are motivated, practice various tasks related to a specific activity, and is provided a "feedback" in order to allow the central nervous system time of storing the sensory information learned [15] [16] [17].

Based on this scenario, one of the possible uses of the affective development process is provide support to designers incorporate affective requirements into application. Thus, the application could store emotional information’s about patient and use them to encourage and create a feedback according to patient motivational state.

V. A HOMECARE CASE STUDY: THE MOLLY PETBOT

In order to validate top-down proposed development process a pet robot was built. It is called Molly. In this paper the Molly purpose is to imitate a pet, such as a little dog demonstrating happiness, pleasure, disgust, indifference and anger during a normal situation of being hungry. For example, making the comparison between a pet and a pet bot, hunger can be seen as synonymous with battery discharge. In other words, emotional states of this pet bot should be affected according to its different battery levels. Moreover, Molly is able to avoid or to overcome certain obstacles. Molly can be also very useful in telemedicine applications especially those related to human monitoring. For example, Poh [9] has developed a simple, low-cost method for measuring multiple physiological parameters using a basic webcam. By applying independent component analysis on the colour channels in video recordings, it can extract the blood volume pulse from the facial regions, heart rate (HR), respiratory rate, and HR variability.

Power consumption is another important issue. It can affect Molly´s emotional states because low battery charge levels should affect its current mission. In this paper affective requirements will be focused on this scenario. Molly Robot should operate according to two affective states: *(i)* follow its owner and *(ii)* low battery charge level. The first affective state defines all possible emotions expressed when Molly is following its target. The next describes Molly's behavior when its battery charge is almost empty. Both situations influence Molly behavior according to the current state of charge. In both cases a triggered emotions should be generated in response to events, such as an obstacle.

A. System Requirements Definition

Requirements should be grouped into three categories. **Functional requirements** define system´s functionalities while **non-functional** requirements describe inherent characteristics of the system, such as performance. The last category is the **Affective Requirements**. They describe all expected system emotional states, such as anger and frustration. In the proposed process described in this paper all requirements should be defined using SysML stereotypes based on UML. SysML provides a standard and well-structured description avoiding misunderstanding and ambiguity.

After the activity of requirements analysis, requirements will be organized into hierarchy packages. It is important to avoid requirement mixtures. Otherwise, it could become impossible to build requirement traceability matrixes.

Finally, the traceability matrix should be built according to traditional Requirement Engineering techniques [10]. It is important to verify the requirements correctness. Besides, a traceability matrix allows implement activities to control and manage the project implementation, for example, to analyze risks and its impacts.

Non-functional requirements define characteristics and limitations of the application. They are composed by features and requirements. For example, Performance Measure [5] - a cognition parameter - should be expressed as feature. Requirements may be associated with an annotation to describe its characteristics, such as requirements details, implementation details, etc.

Functional requirements describe all functionalities performed by robot. They are composed by performance measures and environment description defined according to Russell´s PEAS Model [5]. Moreover, they may be associated with an annotation to describe its characteristics.

The requirements diagram presented in Fig. 4 shows a system functional requirement "Avoid obstacle" associated to its performance measure "not hit any obstacle". This description can be useful to evaluate the success of its implementation while environment package defines all environment properties.

Environment properties showed in Fig. 4 should be more detailed using annotations. An Environment description

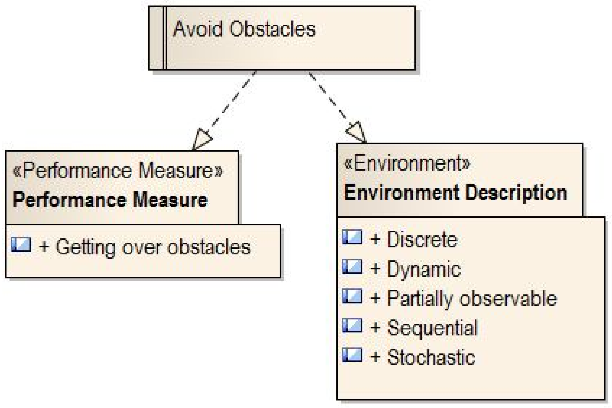


Fig. 4 - Functional requirement avoid obstacles description.

should have information enough to help designers about the environment properties related to specific agent actions. Fig. 5 shows environment description of *Avoid obstacles*, a sort of functional requirement.

Affective requirements are described by means emotional states using State Machine Models. They represent all possible

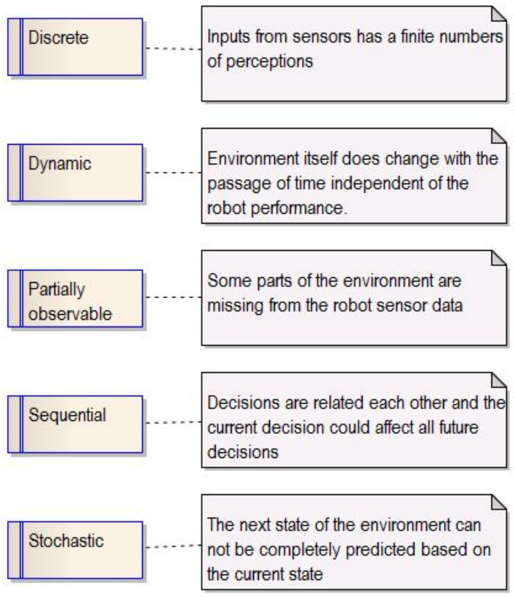

Discrete
Inputs from sensors has a finite numbers of perceptions
Dynamic
Environment itself does change with the passage of time independent of the robot performance.
Partially observable
Some parts of the environment are missing from the robot sensor data
Sequential
Decisions are related each other and the current decision could affect all future decisions
Stochastic
The next state of the environment can not be completely predicted based on the current state

Fig. 5 - Complete PEAS environment description

emotional states expressed by application. Each emotional state should be detailed according to EmotionML specification. However, for each EmotionML element is possible define a custom vocabulary to represent their characteristics. Each vocabulary is created using a set of words in order to represent appropriated emotion types, such as "Arousal", "Dominance" and "Pleasure". In certain situations numeric attributes can be used to quantify emotion intensity. Fig. 6 shows "Arousal: 1.0". It represents the maximum intensity while "Arousal: 0.0" represents the minimum intensity.

An Emotional State is described using Action Tendencies, Dimensions, Appraisals and Categories (see Fig. 6). An Action Tendency represents actions or emotions with strong influence on the motivational state of the application. For example, the current emotional state, showed in Fig. 6, may influence Molly robot to perform three different actions: *(i)* tendency to "Charge battery", *(ii)* moderate tendency to "Find shelter" or *(iii)* very low tendency "Continue the activities".

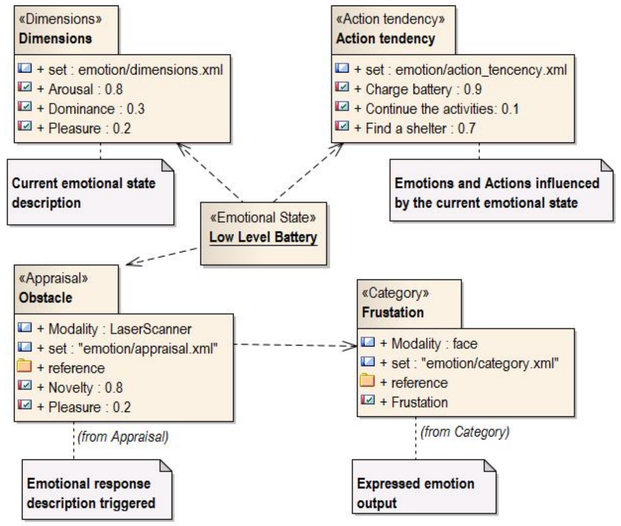


Fig. 6 - Emotion Specification related to “Low level battery” state.

Dimensions represent one or more emotions used to quantify the current emotional state. In Fig. 6 an emotional state is quantified using three dimensions: a high “Arousal”, a low “Dominance” and a very low “Pleasure”.

Appraisal is used to characterize emotional responses due to specific events. A single Appraisal example is showed in Fig. to describe the pet bot behavior when pet bot perceives an obstacle. The attribute "Modality" defines a sensor category, such as laser scanners, distance sensor, etc. The attribute "URI" defines the data path into storage manager.

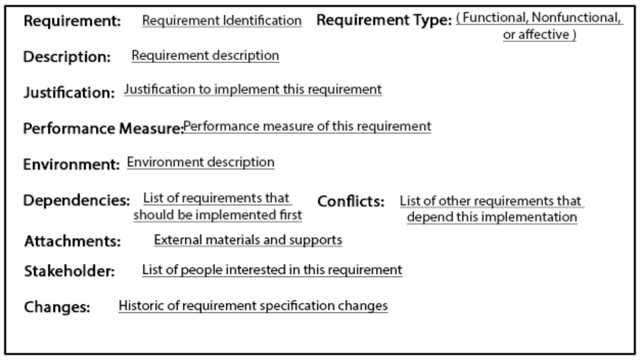
Category is used to describe one or more emotions triggered by an event defined in Appraisal. The attribute modality defines the type of output used to express this emotion, e.g. an image.

B. *Specification*

The proposed process presented in this paper recommends Volere Requirements Specification Template [8] to produce a high quality specification document. Fig. 7 shows a document instance customized according to the current case study.

C. *Architecture*

There are two different activities for specifying the system architecture: *(i) System structure* should be represented by block diagrams, which show the components modularization according to its functionalities; *(ii) System behavior description* performed in two activities: **use case diagrams definitions** and **state machine diagrams definitions**. Both are inherited from traditional UML. The use case diagrams show an overview of system's functionalities and its relationship while state machine diagrams represent the system behavior descriptions.



Requirement: Requirement Identification **Requirement Type:** (Functional, Nonfunctional, or affective)

Description: Requirement description

Justification: Justification to implement this requirement

Performance Measure: Performance measure of this requirement

Environment: Environment description

Dependencies: List of requirements that should be implemented first **Conflicts:** List of other requirements that depend this implementation

Attachments: External materials and supports

Stakeholder: List of people interested in this requirement

Changes: Historic of requirement specification changes

Fig. 7 - Volere Requirements Specification Template.

D. *Constraints Blocks Definition*

The constraints blocks definitions are important activities to systems modeling. They define and standardize all constants and attributes needed to system applications, such as mathematical expressions and global variables. Fig. 8 shows a constraint blocks definition instance. Each block specifies the input and output data types. Furthermore, they define the constants useful in system´s operations.

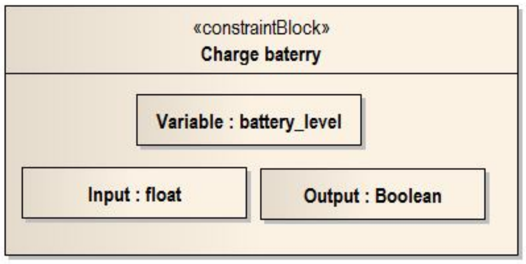


Fig. 8 - Constraints Blocks Definition

E. *Simulation*

Simulation is an essential part of embedded systems development. It helps designers to build a system in order to meet customer needs, avoiding additional rework while allowing designers to identify and fix project bugs before development. In this case study Proteus ISIS Professional rel. 7.7 v.2 was evaluated. It supports different hardware technologies and allows deploying binaries software artifacts in order to produce a complete embedded system simulation template.

F. *Hardware and Software implementation*

This activity is responsible for transforming specifications into software artifacts and electronic circuits. According to Pressman [11], it should combine several techniques to ensure the system quality, such as unit testing, programming in peers and successive refinements. This activity should be customized according to organizational processes and adopted technologies. The case study proposed in this paper has focused on reusing hardware components by means of commercial off-the-shelf components. It allows rapid prototyping, costs reduction and to parallelize software and hardware development. Fig. 9 show a hardware and software prototype capable to express affectivity.

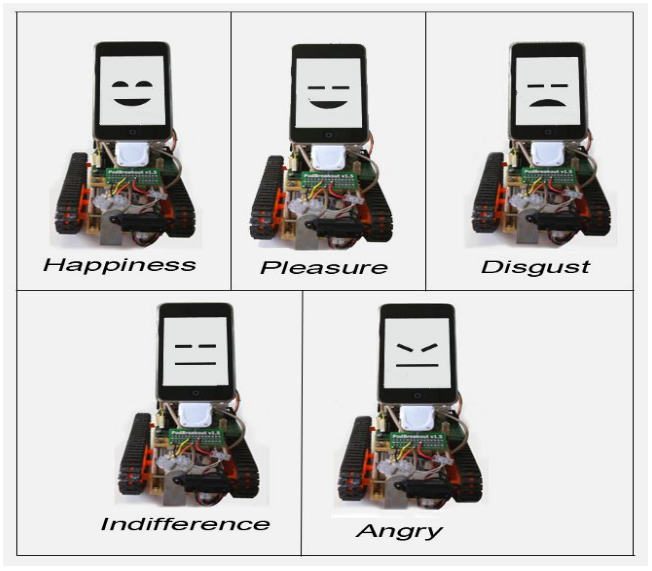


Fig. 9 - The Molly Petbot.

G. *System Integration*

System Integration is related to combine system's modules to produce a complete system. For example, at system integration several tests are performed to find bugs, not only related to software but those including all system. It is fundamental to define test plan. System integration often is a hard activity because it usually uncovers problems. Thus, it is often necessary successive refinements to meet all conformances.

VI. Conclusion

In this paper an affective development process was proposed. It has been developed to support designers to include affectivity requirements combining EmotionML and SysML technologies. So far, the scientific community has been working to capture, recognize and represent patterns related to affectivity [9] [13]. However, there is a lack of research in development processes to model and represent these affectivity requirements. Thus, the proposed development process provides a link between Affective Computing and Software Engineering.

Considering that system modeling tools have a high cost, the affectivity development process creates a customization of low cost UML tools using stereotypes. The customization provide support to designers extend UML vocabulary to create new models derived from SysML specification. Moreover, stereotypes allow designers to create new vocabularies to meet specific project needs, allowing the process to fit in several applications.

In the Case Study the proposed process was applied to a Homecare application. Preliminary results show at least two advantages when compared to ad hoc attempts. First, requirements diagrams provided by SysML allow engineers to specify affectivity using standard tools. Second, EmotionML utilizes a custom emotion jargon facilitating to reuse the same development process for different applications.

PEAS and SysML hybridization improves the quality of requirements specification because PEAS descriptions provides a complete agent environment description. It is useful to include and modify systems policy.

Future work will concentrate to deal with safety critical aspects. Failures or malfunction in critical systems may result in severe damage to system, death or serious injury to people.

Acknowledgment

This study was supported by Microchip's Academic Program (AP-0280) and FAPEG *(Fundação de Amparo à Pesquisa de Goiás)* at Pontifical Catholic University of Goiás.

References

[1] R. D., Mancarella. Embedded System Development using SysML, An illustration using Enterprise Architect. Sparx System Pty Ltd and ICOMIX, 2010. Available: http://sparxsystems.com/resources/booklets/ebook_list.htm
[2] D. Katié .Advances in Cognitive Robotics,"Achievements and Challenge". Proc. of 10th Symposium of Neural Network Applications in Electrical Engineering. NEUREL, Serbia, 2010.
[3] M. Schröder. Emotion Markup Language (EmotionML) 1.0, 2 W3C. Available: http://www.w3.org/.
[4] W. Wolf. Computers as Components, Principles of embedded computing System design. ELSEVIER, 2nd edition, 2008
[5] S. Russell, P. Norvig. Intelligent Agents" in Artificial Intelligence: A modern Approach. 2nd edition, 2004.
[6] System Modelling Language (SysML) Specification. Version 1.2, June 2010. Available: http:// www.omgsysml.org/.
[7] J. A. Silva. An Empirical Study of SysML in the Modeling of Embedded Systems. SMC, 2006.
[8] S. Robertson. Mastering the Requirements Process. Addison-Wesley, 2006.
[9] M. Poh, J. D. McDuff, R. Picard. Advancements in Noncontact, Multiparameter Physiological Measurements Using a Webcam. IEEE Transactions on Biomedical Egineering, vol 58, no. 1, 2011.
[10] I. Sommerville. Software Requirements in Software Engineering. 8th edition, 2006.
[11] R. S. Pressman, R. S. Software Engineering, A practitioner's Approach. 6th edition, 2005.
[12] Carvalhaes, M. F. A.; Ferreira, José Olímpio ; Silva Neto, Olegário Correa da ; Rocha, Adson Ferreira da ; Barbosa, T. M. G. A. . Including Affectivity Requirements in Embedded Systems. In: IEEE International Systemas Conference, 2013, Orlando-FL. Proceedings of 7th Annual IEEE International Systemas Conference. Piscataway, NJ: IEEE, 2013. v. 1. p. 229-234.
[13] COSTA, R. M. E. M.; CARVALHO, Luis Alfredo Vidal de. The acceptance of virtual reality devices for cognitive rehabilitation: a report of positive results with schizophrenia. Computer Methods and Programs in Biomedicine (Print), EUA, v. 73, n.3, p. 173-182, 2004.
[14] Sohlberg, M. M.; Matter, C. A.; Cognitive Rehabilitation – An Interrogative Neuropsychological Approach; The Guildford Press, New York , 2012
[15] Cirstea et al., Feedback and cognition in arm motor skill re-acquisition after stroke. Stroke, v.37, p.1237-1242, 2006.
[16] Cirstea, M.C., Levin, M.F. Improvement of arm movement patterns and endpoint control depends on type of feedback during practice in stroke survivors. Neurorehabilitation and Neural Repair, v.21, p.398-411, 2007
[17] Winstein et al., Methods for a multisite randomized trial to investigate the effect of constraint-induced movement therapy in improving upper extremity function among adults recovering from a cerebrovascular stroke. Neurorehabilitation and Neural Repair, v.17, p. 137-152, 2003.