Modeling Non-Functional Properties for Human-Machine Systems

Arunkumar Ramaswamy[1,2] **Bruno Monsuez** [1] **Adriana Tapus** [1]
[1]Department of Computer and System Engineering,
ENSTA-ParisTech, 828 Blvd Marechaux, Palaiseau, France
[2]VeDeCom Institute, 77 rue des Chantiers, 78000 Versailles, France
Email:{arun-kumar.ramaswamy; bruno.monsuez; adriana.tapus}@ensta-paristech.fr

Abstract

A Human-Machine system is a complex system consisting of many components and services that dynamically compose to achieve a specific goal. The functional and non-functional attributes of the components are considered to make 'who does, what, and when' decisions depending on the operational context. However, non-functional properties are not given sufficient importance compared to that of the functional requirements during the developmental stages. This paper highlights the importance of non-functional properties in human-machine systems and proposes a metamodel for modeling those properties. A case study on assistive lane keeping in automobiles is presented to demonstrate how the non-functional properties can be modeled. This is a part of the intermediate results of a research in progress for modeling decision architectures for autonomous systems.

1 Introduction

In earlier times, function allocation in human-machine systems was primarily a design decision. Consider the case of cruise control in today's cars, the steering control is allocated to the human driver while the automation is responsible for applying acceleration. This is an example for static function allocation. Some of the traditional strategies for function allocation are: (a) assigning each function to the most capable agent, (b) allocating to machine every function that can be automated, and (c) applying an appropriate allocation policy (Inagaki 2003). However, in adaptive human-machine systems the function allocation is a run-time problem. Consider the case of auto-pilot taking control of an aircraft in emergency situations. In such systems, there is some kind of control sharing and trading that happens. The first example of adaptive cruise control is a control sharing case, while the second one is a control trading problem. In the latter case the software takes the control of system with or without the consent of human.

In certain tasks, humans outperform automation while in some others, it is the opposite. However, in most of semi-autonomous systems involving humans, for example in cars, many functions that human performs are provided by the automation also. Although the software component and the human counterpart can perform the same functionality, it

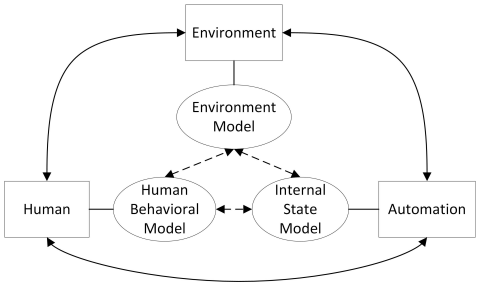


Figure 1: Models and their Interactions in a Human-Machine System

the quality expected from the entire system and the context that determines which one should be activated or deactivated in compliance with the regulation authority such as FAA for civil aviation. A typical human-machine system maintains three kinds of models: (a) human model, (b) environment model and (c) the internal state model of the system as shown in Figure 1. The functionality is achieved through the interaction of these three models. Most modeling languages provide support for the description of functional behavior, while the non-functional requirements are normally described using informal comments even though these properties play a vital role in determining the quality of the system. A decision making system estimates the quality provided by the human and that of the automation, and decides the authority for control. For example, attributes such as response time, accuracy and reliability may comprise the quality of the software component, while alertness, skills and expectation-intention correlation may decide the quality of the human.

Furthermore, selecting the services based on the quality have advantage not only between human and automation agents but also within the automation system itself. Since the functionalites are viewed as services it is possible to compose basic low level services in different fashion to provide different functionalities. For example, in cars, voice recognition service can be used in an entertainment system as well as in a navigation system. However, the bottleneck is more related to the business models existing in industries that develops such systems. The standard approach for automobile OEMs is to develop systems by assembling

components that have been completely or partly designed and developed by external vendors (Sangiovanni-Vincentelli and Di Natale 2007). Because of the increasing complexity of automobile systems with large number of distributed features, such an approach will also lead to various compositional issues commonly known as *feature interactions*. Therefore, Service Oriented Architectures (SoA) are gradually being adopted in these systems where various functionalities are provided as services and the assembled components are seen as service providers. This software engineering paradigm has many advantages in Human-Machine collaborative work. Advances in human behavior research helps to model various human actions. These actions can be seen as services provided by human, for example, steering, braking, applying acceleration by the driver can be viewed as services provided by the human driver. In some context, humans provide high quality services while in some others the machine counterpart does. For example, in the case of assisted parking, human steering control service is delegated to the machine still retaining the authority over acceleration with the driver.

Non-Functional Properties (NFPs) and Quality of Service (QoS) models can be applied in various software development approaches also. Modern complex software systems are realized by mainly three paradigms - Component Based Software Engineering (CBSE), Model Driven Software Development (MDSE), and Service Oriented Architecture (SOA). In CBSE, the system is a combination of interacting components, which have predefined interfaces and perform a specific functionality (Brugali and Scandurra 2009). Models are first class entities in MDSE and the approach starts with defining abstract models and platform specific details are progressively added to get the final implementation (France and Rumpe 2007). SoA view systems as a composition of various services to perform a functionality. In all these approaches, the final system can be viewed as a Systems of Systems (SoS) with components, models, or services as heterogeneous independently operable systems. In such complex systems, the emergent behaviors caused by the interaction of different components supplied by different vendors cannot be overseen without a formal model based approach (Ramaswamy, Monsuez, and Tapus 2013). The problem gets further aggravated when these components are used in safety critical systems such as autonomous driving in automotive domain. In addition with the introduction of drive-by-wire in automobiles, the compositional problem is becoming more pervasive. Hence the concept of QoS is a global issue not only to service oriented systems.

The problem of dynamic adaptation is to maintain the QoS of the system at a certain level. In order to do that in a system involving humans, the QoS of human models as well as for automation models needs to be specified in a common framework. The challenge is that various attributes that determine a NFP is heterogeneous in nature while considering human and machine models. Our paper tries to address this problem by providing a common metamodel that can be used to specify the QoS of human, machine, and their interactions.

The paper is organized as follows: Section 2 provides related works. A reference architecture for a human-machine system is described in section 3 in order to illustrate the concept of service composition. The concepts of NFP and QoS are explained in section 4 and a metamodel is proposed in section 5. Section 6 demonstrates the application of NFP models with the help of a case study on assistive lane keeping in automobiles. Section 7 concludes the paper with future research directions.

2 Related Works

Although there is little literature that deals directly with modeling NFPs in human-machine systems, this section will point out works in metrics used in human-machine systems and application of NFPs in general software engineering.

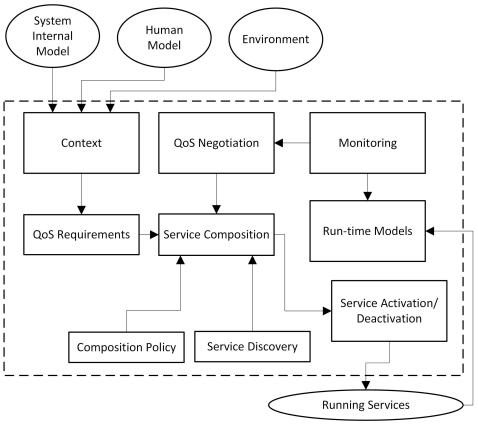
Metrics for standardization for Human-Robot Interactions (HRI) have been proposed by Olsen and Goodrich, 2003, Burke et al. 2004, Steinfeld et al. 2006, and Saleh and Karray 2010. The proceedings of the workshop on 'the Metrics for Human-Robot Interaction' proposed several guidelines for the analysis of human-robot experiments and forwarded a handbook of metrics that would be acceptable to the HRI community (Burghart and Steinfeld 2008).

Mylopoulos, Chung, and Nixon proposes two complementary approaches for using Non-Functional information - process oriented and product oriented. Process oriented approach used NFP to guide the development of software systems. While in the product oriented approach, the non-functional characteristics of the final product are explicitly stated, making it possible to examine if products fall within the constraints of non-functionality. There are three UML profiles that are standardized by the Object Management Group (OMG) that deal with modeling QoS for software components - QoSFT, SPT, and MARTE profiles. In Dobson, Lock, and Sommerville a QoS ontology for service centric systems has been proposed.

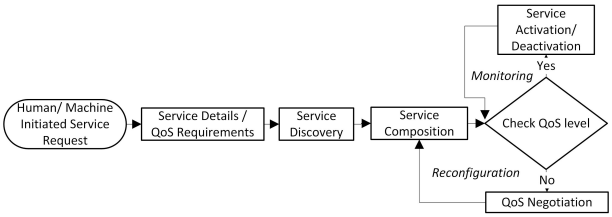
3 Reference Architecture

Figure 2a shows an abstract architecture of a service oriented system. The reference architecture is used here only as a way to explain the concepts hiding all the technology and platform specific details. The architecture is comprised of a number of low level services that provide support for service discovery, QoS negotiation, supervisors, etc.

Figure 2b shows a typical sequence of activities in the system. The request for service can be generated either from human or machine. The service description with all the required parameters and required levels of QoS is computed with respect to the context. For example, emergency braking service will be at a high level while a music streaming service will be at a lower level. Appropriate services found from a repository are composed to check for conflict, redundancy etc. and QoS is computed for the composed service. The composed service is verfied with QoS specification and the required service is activated. A supervisor continuously monitors the quality level of the running services and reconfiguration is done when anomalies are detected.



(a) Reference Architecture



(b) Activity Flow

Figure 2: Service Oriented Architecture

4 Non-Functional Properties and Quality of Service

Non-functional properties define how a functionality operates, for example, performance, availability, effectiveness, etc. QoS is the aptitude of a service for providing a quality level to the different demands of the clients (MARTE 2011). There is no general consensus in the community about the concepts of NFP and QoS. Non-Functional Requirements are not implemented in the same way as functional ones. NFPs are seen as *by products* when a functionality is implemented. In software engineering terms, usability, integrity, efficiency, correctness, reliability, maintainability, testability, flexibility, reusablity, portability, interoperability, performance, etc. constitute NFPs (Chung and do Prado Leite 2009). At the same time, what determines QoS is highly domain specific. For example, throughput and bandwidth determines QoS for a network; performance, fault-tolerance, availability and security for an embedded system; personality, empathy, engagement, and adaptation for social robots (Tapus, Mataric, and Scassellati 2007); resource utilization, run-time adaptation for service robots (Inglés-Romero et al. 2013).

In Steinfeld et al. the authors identified some common metrics for tasks in navigation, perception, management, and manipulation for Human-Robot Interaction. For example, effectiveness of a navigation task can be measured by:

- Percentage of navigation tasks successfully completed
- Coverage of area
- Deviation from planned route

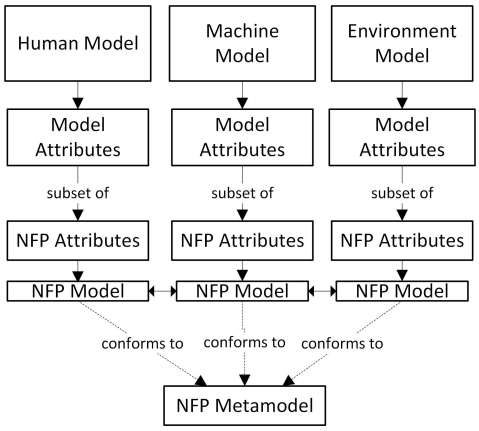


Figure 3: Model Relationships

- Obstacles that were successfully avoided
- Obstacles that were not avoided, but could be overcome

Efficiency can be measured by time to complete the task, operator time for task, average time for obstacle extraction, etc. Using the empirical data, the following three hypotheses are proposed:
Hypothesis 1: There can be one or more quality attributes that directly or indirectly influence the NFP of the system.
Hypothesis 2: The attributes that determine the NFP differs with the context, viewpoint, and current state of the system.
Hypothesis 3: Even when two systems has same attributes that determine the NFP, the quality levels of NFP does not necessarily be the same.

5 NFP Metamodel

A Model is an abstraction of reality and hence, it is only a specific viewpoint of the reality. The concept of having multiple models for modeling different viewpoints are termed as *separation of concerns* in Model Driven Engineering (MDE) terminology. A model capturing the Non-functional characteristics of a system does not encapsulate the entire functional model of the system. It will capture a subset of the model attributes as illustrated in Figure 3. In this case, the NFP models conforms to a common abstract NFP metamodel.

A metamodel can also be called as a modeling language that is at a higher abstraction level than the model itself. Currently, there are two ways for modeling a modeling language. One approach is by using an extensible general purpose modeling language like Unified Modeling Language (UML) and the other by developing a Domain Specific Language (DSL). This section proposes a metamodel using the profiling mechanism of UML based on the hypotheses mentioned in the previous section.

In our NFP metamodel terminology, the attributes that determines NFPs are called *QoS attributes*. Each NFP has at least one *QoS profile* associated with it. A QoS profile consists of a set of QoS attributes and a *QoS policy* that defines how the attributes affect the quality of NFP. The policy can

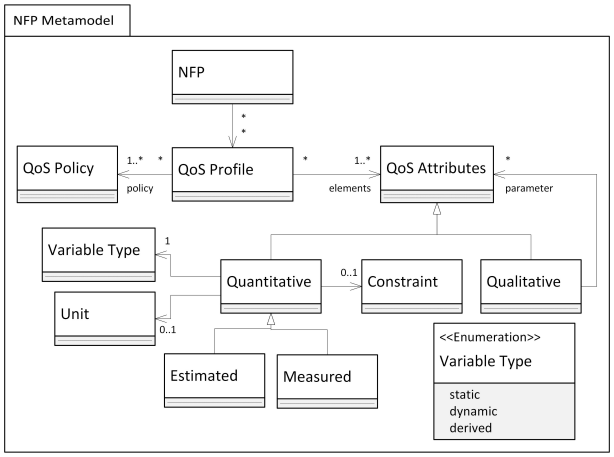


Figure 4: NFP Metamodel

be defined based on logic systems such as a first order logic, temporal logic, predicate logic, etc. (Rosa, Cunha, and Justo 2002).

Figure 4 shows our metamodel diagram of NFP using UML notation. NFP and QoS Profile are the first class entities in this metamodel. NFP can have multiple QoS profiles associated with it. QoS Profile consists of QoS Attributes and QoS Policy. Each QoS Profile must have at least one QoS Policy associated with it. QoS attributes can be Quantitative or Qualitative. Quantitative attributes can be directly measured or can be estimated. Quantitative attributes can have metric units to specific along with it, for example: seconds for responsiveness, bits per second for throughput, etc. The quantitative attributes can be static, dynamic, or derived. Static attributes will not change during the course of system operation, for example, time required by a human driver to press the brake pedal of the vehicle. Dynamic attributes can change during system operation, for example, task load on human driver during driving. Qualitative attributes refer to discrete characteristic that may not be measured directly but provide a high level abstraction that are meaningful in a domain, for example, for indicating an individual personality traits such as introversion, agreeableness, etc. (Fong, Nourbakhsh, and Dautenhahn 2003). Sometimes a qualitative attribute needs some quantitative measures also, for instance, energetic person with a certain activity level (Tapus, Cristian, and Matarić 2008).

6 Case Study: Assistive Lane Keeping

Assistive lane keeping is a perfect example for dynamic function allocation in Human-Machine system. In this case study, we specifcally consider system initiated invasive lane keeping feature in vehicles. The automation part is the lateral control of the vehicle to keep the vehicle in the same lane. The system takes control by applying the required torque on the steering wheel in case of lane departure situations. The case study is to demonstrate how NFPs of Human and Machine can be formally modeled using the proposed NFP metamodel. The Non-Functional property, efficiency is modeled as an example. We define QoS for efficiency as the response time in which the human will react to a critical event (here lane departure).

Human Driver Behavior Model

Modeling the human driver in ACT-R architecture (Trafton et al. 2013) is considered in this section. The ACT-R model was selected because of the availability of large number of tunable parameters in models representing various aspects of human driver such as memory, perception, control, monitoring, and decision skills. A brief description of the model with only relevant properties are described here (see Salvucci 2006 for more detailed descriptions). In this model there are control, monitoring, and decision making components. The control component is responsible for low level perception and, lateral and longitude control. Lateral control is based on two features - near point and far point. The near point represents the vehicle's current lane position. It is measured at 10m from vehicle's center. The far point can be: (a) the vanishing point of a straight road, (b) the tangent point for the curve, and (c) the lead vehicle. The control law for steering angle can be expressed as (Salvucci and Gray 2004) :

$$\Delta\varphi = k_{far}\Delta\theta_{far} + k_{near}\Delta\theta_{near} + k_I\theta_{near}\Delta t \quad (1)$$

where φ is the steering angle, θ_{far} and θ_{near} are visual angles of near and far points, respectively, $\Delta\varphi$, $\Delta\theta_{far}$,$\Delta\theta_{near}$, and Δt are the difference of the respective parameters with respect to the last cycle. In short, the control law imposes three constraints on the steering angle: a steady far point($\Delta\theta_{far} = 0$), a steady near point ($\Delta\theta_{near} = 0$), and a near point at the center of the lane ($\theta_{near} = 0$). Similarly, the longitude control law can be expressed as:

$$\Delta\psi = k_{car}\Delta thw_{car} + k_{follow}(thw_{car} - thw_{follow})\Delta t \quad (2)$$

where ψ is the acceleration value, thw_{car} is the time headway to the lead vehicle, Δthw_{car} is the difference of thw_{car} with respect to the previous cycle, and thw_{follow} is the time headway for following a lead vehicle. The monitoring model is defined by a random probability measure ($p_{monitor}$) that checks left or right lane, and forward and backward with equal likelihood. The decision model is defined by the safe distance (d_{safe}) and the lead vehicle distance that is considered as safe by the driver. The memory is modeled as total number of references that can be stored in declarative knowledge, decay time for the memory, and memory creation time. The cognitive architecture is designed in such a way to include limitation and constraints on models that mimic the human system (refer Salvucci 2006 for more details). The NFP model for efficiency of the human driver is shown in listing 2 in textual form. The model structure conforms to the proposed NFP metamodel. The template used for textual representation is shown in listing 1.

```
import (external NFP and QoS_Profile Models}
NFP: (NFP name);
QoS_Profile: (Qos ID):(QoS Name);
```

```
        Qos_Attributes:
                (comma separated list of
                    attributes
                in the following format
                ID:Name:Variable Type:Unit);
Qos_Policy: policy_function();
```

Listing 1: Textual Model Representation

```
import control_skill,monitor_skill,
    decision_skill,memory_skill
NFP: efficiency_human;
QoS_Profile: rt:response_time;
        Qos_Attributes:
                cs:control_skill:derived:,
                ms:monitor_skill:derived:,
                ds:decision_skill:derived:,
                mem:memory_skill:derived:;
Qos_Policy: rt.cs>thcs & rt.ms>thms & rt.ds>
    thds;
-----------------------------------------
QoS_Profile: control_skill;
        Qos_Attributes:
                pt:prep_time:static:ms,
                et:exec_time:static:ms,
                kflt:kfar_lateral:dynamic:,
                knearlt:knear_lateral:
                    dynamic:,
                kilt:ki_lateral:dynamic:,
                kfarln:kfar_speed:dynamic:,
                knln:knear_speed:dynamic:,
                kiln:ki_speed:dynamic:,
Qos_Policy: contrl_policy();
-----------------------------------------
QoS_Profile: monitor_skill;
        Qos_Attributes:
                pm:prob_monitor:dynamic:,
Qos_Policy: monitor_policy();
-----------------------------------------
QoS_Profile: decision_skill;
        Qos_Attributes:
                sd:safe_distance:dynamic:m,
Qos_Policy: decision_policy();
-----------------------------------------
QoS_Profile: memory_skill;
        Qos_Attributes:
                mem:memory:dynamic:,
                ct:creation_time:static:sec,
                dr:decay_rate:static:,
Qos_Policy: memory_policy();
```

Listing 2: Efficiency Model for Human Driver

Steering Automation Model

There are a number of control algorithms in the literature for automated steering control for highway driving, lane keeping, and lane changing situations (Guldner, Tan, and Patwardhan 1997). The non-functional parameters of the execution model in the final platform consist of both algorithm as well as hardware platform specific details. Listing 3 shows an example NFP model for efficiency for automatic steering control using vision sensors.

```
import platform_capability,sensor_capability
    ,algorithm_parameters
NFP: efficiency_machine;
QoS_Profile: rt:response_time;
        Qos_Attributes:
                et:execution_time:static:ms,

Qos_Policy: response_time_policy();
-----------------------------------------
QoS_Profile: platform_capability;
        Qos_Attributes:
                pt:process_load:dynamic:
                    number,
                ra:resource_availability:
                    derived:,
                sp:scheduling_policy:
                    qualitative,
Qos_Policy: decision_policy();
-----------------------------------------
import camera_capability
QoS_Profile: sensor_capability;
        Qos_Attributes:
                fc:front_camera,
                fr:rear_camera,
Qos_Policy: camera_policy();
-----------------------------------------
QoS_Profile: camera_capability;
        Qos_Attributes:
                res:resolution:static:
                    pixelsperframe,
                fr:framerate:static:fps,
                intf:interface:static:kbps,
Qos_Policy: camera_policy();
-----------------------------------------
QoS_Profile: algorithm_parameters;
        Qos_Attributes:
                vel:vehicle_velocity:dynamic
                    :kmps,
                fang:
                    front_wheel_steering_angle
                    :dynamic:rad,
                sa:slip_angle:dynamic:rad,
Qos_Policy: algorithm_policy();
```

Listing 3: Efficiency Model for Automated steering control

Interaction Model

Listing 4 shows the NFP model for interaction between the Human Driver and the Steering Automation system. The policy $interaction\_policy()$ defines the strategy for maximizing the efficiency of the entire system.

```
import efficiency_human, efficiency_machine
NFP: efficiency_interaction;
QoS_Profile: rt:response_time;
        Qos_Attributes:
                eh:efficiency_human:dynamic:
                    derived:,
                em:efficiency_machine:
                    dynamic:derived:,
Qos_Policy: interaction_policy();
```

Listing 4: Efficiency Model for Interaction between Human Driver and Automation

7 Conclusion

The importance of Non-Functional properties in human-machine systems were discussed. Modeling those properties are necessary in architectures where functionality alone cannot be used for making both design time and run-time decisions. Our NFP metamodel provides a generic base for specifying the non-functional aspects of both human and machine models. The challenge of dealing with heterogeneous attributes defining the human and automation models are addressed by categorizing the attributes into profiles hierarchically and then using policies to compare at a higher abstraction level. In this direction, the next logical step is to formally specify the interaction policies and analyzing what logic systems, such as a first order logic, predicate logic, etc. are suitable for providing formal proofs.

8 Acknowledgment

This research is funded by VeDeCoM Institute, a French automotive cluster on mobility research.

References

Brugali, D., and Scandurra, P. 2009. Component-Based Robotic Engineering (Part I) [Tutorial]. *Robotics & Automation Magazine, IEEE* 16(4):84–96.

Burghart, C. R., and Steinfeld, A. 2008. Proceedings of Metrics for Human-Robot Interaction, a Workshop at ACM/IEEE HRI. Technical Report Technical Report 471, School of Computer Science, University of Hertfordshire, Hatfield, UK.

Burke, J. L.; Murphy, R. R.; Riddle, D. R.; and Fincannon, T. 2004. Task performance metrics in human-robot interaction: Taking a systems approach. Technical report, DTIC Document.

Chung, L., and do Prado Leite, J. C. S. 2009. On Non-Functional Requirements in Software Engineering. In *Conceptual modeling: Foundations and applications*. Springer. 363–379.

Dobson, G.; Lock, R.; and Sommerville, I. 2005. QoSOnt: a QoS ontology for service-centric systems. In *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on*, 80–87. IEEE.

Fong, T.; Nourbakhsh, I.; and Dautenhahn, K. 2003. A survey of socially interactive robots. *Robotics and autonomous systems* 42(3):143–166.

France, R., and Rumpe, B. 2007. Model-driven Development of Complex Software: A Research Roadmap. In *2007 Future of Software Engineering*, 37–54. IEEE Computer Society.

Guldner, J.; Tan, H.-S.; and Patwardhan, S. 1997. *On Fundamental Issues of Vehicle Steering Control for Highway Automation*. California PATH Program, Institute of Transportation Studies, University of California, Berkeley.

Inagaki, T. 2003. Adaptive automation: Sharing and trading of control. *Handbook of cognitive task design* 8:147–169.

Inglés-Romero, J. F.; Lotz, A.; Chicote, C. V.; and Schlegel, C. 2013. Dealing with Run-Time Variability in Service Robotics: Towards a DSL for Non-Functional Properties. *arXiv preprint arXiv:1303.4296*.

MARTE. 2011. *UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems*. Object Management Group, version 1.1 edition.

Mylopoulos, J.; Chung, L.; and Nixon, B. 1992. Representing and Using Non-Functional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering* 18(6):483–497.

Olsen, D. R., and Goodrich, M. A. 2003. Metrics for Evaluating Human-Robot Interactions. In *Proceedings of PERMIS*, volume 2003, 4.

QoSFT. 2008. *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification*. Object Management Group, version 1.1 edition.

Ramaswamy, A.; Monsuez, B.; and Tapus, A. 2013. Formal Models for Cognitive Systems. In *International Conference on Advanced Robotics (ICAR)*. IEEE.

Rosa, N. S.; Cunha, P. R.; and Justo, G. R. 2002. Process(NFL): a Language for Describing Non-Functional Properties. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 3676–3685. IEEE.

Saleh, J. A., and Karray, F. 2010. Towards generalized performance metrics for human-robot interaction. In *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*, 1–6. IEEE.

Salvucci, D. D., and Gray, R. 2004. A two-point visual control model of steering. *Perception-London* 33(10).

Salvucci, D. D. 2006. Modeling Driver Behavior in a Cognitive Architecture. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 48(2):362–380.

Sangiovanni-Vincentelli, A., and Di Natale, M. 2007. Embedded System Design for Automotive Applications. *Computer* 40(10):42–51.

SPT. 2005. *UML Profile for for Schedulability,Performance, and Time Specification*. Object Management Group, version 1.1 edition.

Steinfeld, A.; Fong, T.; Kaber, D.; Lewis, M.; Scholtz, J.; Schultz, A.; and Goodrich, M. 2006. Common Metrics for Human-Robot Interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 33–40. ACM.

Tapus, A.; Cristian; and Matarić, M. J. 2008. User-robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy. *Intelligent Service Robotics* 1(2):169–183.

Tapus, A.; Mataric, M. J.; and Scassellati, B. 2007. Socially assistive robotics. *IEEE Robotics and Automation Magazine* 14(1):35.

Trafton, J. G.; Hiatt, L. M.; Harrison, A. M.; Tamborello, F.; Khemlani, S. S.; and Schultz, A. C. 2013. ACT-R/E: An Embodied Cognitive Architecture for Human Robot Interaction. *Journal of Human-Robot Interaction* 2:30–55.