Model Driven Software Development for Human-Machine Interaction Systems

Arunkumar Ramaswamy
[1]Department of Computer and System Engineering, ENSTA-ParisTech, 828 Blvd Marechaux, Palaiseau, France
[2]VeDeCom Institute, 77 rue des Chantiers, Versailles, France
arun-kumar.ramaswamy@ensta-paristech.fr

Bruno Monsuez
[1]Department of Computer and System Engineering, ENSTA-ParisTech, 828 Blvd Marechaux, Palaiseau, France
bruno.monsuez@ensta-paristech.fr

Adriana Tapus
[1]Department of Computer and System Engineering, ENSTA-ParisTech, 828 Blvd Marechaux, Palaiseau, France
adriana.tapus@ensta-paristech.fr

ABSTRACT

In a typical Human-Machine Interaction (HMI) system, a task is performed by cooperation of the human and the automation component. The system adopts a cognitive architecture to model human psychology and makes optimum decisions on dynamic task allocation between human and the machine counterpart depending on the context. However, such architectures do not define how those systems are implemented in software. Various models involved in Model Driven Software Development (MDSD) approach in developing HMI systems is presented. This paper proposes a metamodel for modeling Non-Functional Properties (NFP) in HMI systems and provides a case study on assistive lane keeping in automobiles to demonstrate the approach.

Categories and Subject Descriptors

I.6 [**SIMULATION AND MODELING**]: Model Development—*Modeling methodologies*; H.1.2 [**User/Machine Systems**]: Human factors

Keywords

Model driven engineering; Non-functional properties

1. INTRODUCTION

Humans outperform computers in certain tasks, while in some others, it is the opposite. Consider the case of an assisted parking system in today's cars, the steering control is allocated to the computer while the driver is responsible for applying acceleration. However, in most of the semi-autonomous systems involving humans, for example in automobiles, 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 the quality expected from the entire system and the context that determines which one should be activated or deactivated. In order to do that in a system involving humans, the Quality of Service (QoS) and NFPs of human models as well as for automation models need 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. In addition, we also provide a general picture on the various models involved in MDSD of HMI systems.

HRI'14, March 3–6, 2014, Bielefeld, Germany.
ACM 978-1-4503-2658-2/14/03.
http://dx.doi.org/10.1145/2559636.2559824.

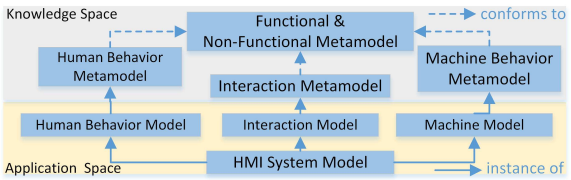

Figure 1: Models in Human-Machine Systems

2. MODEL DRIVEN SOFTWARE DEVELOPMENT

The common approach in Model Driven Software Development (MDSE) begins with defining meta-level models or Domain Specific Languages (DSL) that captures the knowledge of a specific domain. Abstract models are then designed based on the application requirements that conforms to metamodels or DSLs. The models undergo a series of Model to Model transformations (M2M) to reduce the abstraction levels by gradually including platform specific details and finally, Model to Text transformation (M2T) is applied to generate an executable code.

Figure 1 illustrates three models - human, machine, and interaction models that conform to the respective metamodels defined at the knowledge level. In order to bring homogeneity and facilitate reasoning based on functional and NFP, a common Functional and Non-Functional Metamodel

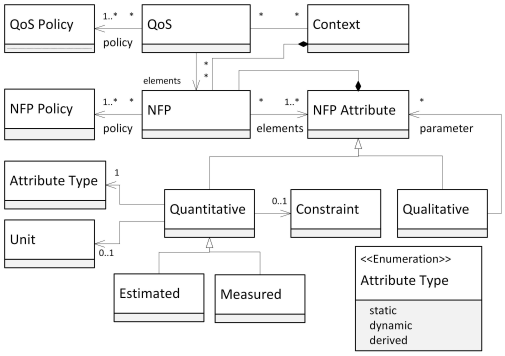

Figure 2: NFP Metamodel

(F&NFM) defined at a higher abstraction level is needed. In short, F&NFM provide syntactical and basic semantic constraints to define Functional and NFP of the models.

3. NON-FUNCTIONAL METAMODEL

NFP define how a functionality operates, for example, performance, efficiency, effectiveness, etc. of a human operator. QoS is the aptitude of a service (human or machine) for providing a quality level to the different demands of the clients (or context). In order to set a general consensus, based on empirical observations, the following assumptions are made: NFP are determined by a set of non-functional attributes (e.g., efficiency can be measured by time to complete the task, operator time for task, average time for obstacle extraction, etc. [2]). QoS is a high level property for comparing a functionality in different contexts (e.g., QoS of a steering maneuver for a human driver is different for cruising and parking scenarios). Policies associated with NFP and QoS determine how these properties are estimated from its constituent attributes. Policies are functions that act on a set of attributes, and defines how these properties are estimated. In a nutshell, `NFP_Policy(NFP attributes)` defines NFP, `QoS_Policy(NFPs, Context)` defines QoS of a functionality.

4. NON-FUNCTIONAL MODEL OF ASSISTIVE LANE KEEPING SYSTEM

Assistive lane keeping is a perfect example for dynamic function allocation in Human-Machine system. 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 NFP, 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).

Modeling the human driver in ACT-R architecture [3] is selected for demonstrating the NFP modeling 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 detailed description of the model can be found in [1]. The efficiency model of the human driver, machine and their interactions based on the proposed NFP metamodel is shown in listing 1. The policy functions are defined based on logic systems, such as a first order logic, predicate logic, etc., by the system designer.

```
//HUMAN_MODEL:
import control_skill,monitor_skill,decision_skill,memory_skill
NFP: eh:efficiency_human; NFP_ATTRIBUTES: cs:control_skill:derived, ms:
       monitor_skill:derived, ds:decision_skill:derived, memory_skill:
       derived; NFP_POLICY: eh.cs>thr_cs & eh.ms>thr_ms & eh.ds>thr_ds;
-----------------------------------------
NFP: control_skill; NFP_ATTRIBUTES: prep_time:static:ms, exec_time:static:
       ms, kfar_lateral:dynamic, knear_lateral:dynamic, ki_lateral:dynamic,
       kfar_speed:dynamic, knear_speed:dynamic, ki_speed:dynamic; NFP_POLICY
       : contrl_policy();
-----------------------------------------
NFP: monitor_skill; NFP_ATTRIBUTES: prob_monitor:dynamic, NFP_POLICY:
       monitor_policy();
-----------------------------------------
NFP: decision_skill; NFP_ATTRIBUTES: safe_distance:dynamic:m, NFP_POLICY:
       decision_policy();
-----------------------------------------
NFP: memory_skill; NFP_ATTRIBUTES: memory:dynamic:, creation_time:static:
       sec, decay_rate:static:, NFP_POLICY: memory_policy();
//MACHINE_MODEL:
import platform_capability,sensor_capability,algorithm_parameters
NFP: efficiency_machine; NFP_ATTRIBUTES: response_time; NFP_ATTRIBUTES:
       execution_time:static:ms,       NFP_POLICY: response_time_policy();
-----------------------------------------
NFP: platform_capability; NFP_ATTRIBUTES: process_load:dynamic:number,
       resource_availability:derived, scheduling_policy:qualitative,
       NFP_POLICY: decision_policy();
-----------------------------------------
NFP: algorithm_parameters; NFP_ATTRIBUTES: vehicle_velocity:dynamic:kmps,
       front_wheel_steering_angle:dynamic:rad, slip_angle:dynamic:rad,
       NFP_POLICY: algorithm_policy();
//INTERACTION_MODEL:
import efficiency_human, efficiency_machine
NFP: efficiency_interaction; NFP_ATTRIBUTES: efficiency_human:dynamic:
       derived, efficiency_machine:dynamic:derived,
NFP_POLICY: interaction_policy();
```

Listing 1: Efficiency model of human driver and machine, and their interactions

5. CONCLUSION

MDSD is a proven approach in software development for automotive and avionics domain. In HMI domain, models are widely used for specifying human behaviors and their interaction with the environments. Our initial experience in adopting MDSD for implementing HMI systems shows reduction in developmental cycle and enhance reusability and portability of the models as compared to the code-based approaches. Modeling NFPs is necessary in architectures where functionality alone cannot be used for making design time and run-time decisions. Our NFP metamodel provides a generic base for specifying the NFP 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 analyze suitable formal methods for defining the policies.

6. REFERENCES

[1] D. D. Salvucci. Modeling driver behavior in a cognitive architecture. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 48(2):362–380, 2006.

[2] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 33–40. ACM, 2006.

[3] J. G. Trafton, L. M. Hiatt, A. M. Harrison, F. Tamborello, S. S. Khemlani, and A. C. Schultz. Act-r/e: An embodied cognitive architecture for human robot interaction. *Journal of Human-Robot Interaction*, 2:30–55, 01/2013 2013.