AI Dimensions in Software Development for Human-Robot Interaction 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

In this paper, we highlight the usage of AI in software development process for Robotic systems, in general and HRI systems, in particular. The software as well as the software development methodology and associated tools are knowledge-based systems. The key challenge is to represent domain knowledge that enables the process and model evolution to built complex software intensive HRI systems.

1 Introduction

Robots are becoming more social with the help of underlying software that employs cognitive processes behind user mental models. Currently, the software development process for designing and implementing Human-Robot Interaction (HRI) systems are ad-hoc in nature and are mainly concentrated on delivering 'proof of concepts' in order to substantiate researcher's idea. The software thus developed are least reusable and the researchers have to re-develop from scratch in order to built on existing work. Not only the software (product), but also the software development methodology (process) that facilitates the development of HRI systems can be considered as knowledge-based systems. In this paper, we highlight the application of AI in our software development framework - Self Adaptive Framework for Robotic Systems (SafeRobots).

2 SafeRobots Framework

SafeRobots is model-driven software development toolchain that supports systematic software development for robotic and HRI systems. It is based on three orthogonal software engineering paradigms - knowledge-based engineering, model-driven software engineering, and component-based software development. The software development process in SafeRobots resides in four conceptual spaces - knowledge space, problem space, solution space, and operational space, as illustrated in Figure 1. The application of AI techniques in SafeRobots framework can be broadly categorized into two parts: Search-based software engineering and Semantic knowledge supported model evolution.

Search-Based Software Engineering

Search-Based Software Engineering (SBSE) reformulates software engineering as a search problem (Harman 2012). Our research work (Ramaswamy, Monsuez, and Tapus 2014b) on problems faced during software development in robotics indicates that the existence of large solution space available for system designers leads to poor software quality. Multiple algorithms are available for implementing a functionality in robotics and early decisions assuming future operating conditions tend to degrade the software quality. We have proposed a modeling language called Solution Space Modeling Language (SSML) that models the solution space of a given problem. Once the solution space for a given problem is modeled, computational search techniques can be applied to evaluate the system objectives and to converge to a reduced set of solutions. The system can be a robot or human-robot team with a goal specified in the problem space. For example, the system's objective to maximize its Quality of Service (QoS) can be defined as:

$$Maximize \sum_{i=1}^{k} QoS(m_i) \tag{1}$$

where m_i is an abstract term that represents an algorithm (algorithmic sequence), model (human or automation), or a control flow. Detailed discussions on QoS and NFP, and its modeling benefits for HMI systems can be found in our previous research paper (Ramaswamy, Monsuez, and Tapus 2014a). We have demonstrated this by modeling NFP of human driver, automation and their interaction in an assistive lane keeping system for vehicles. The QoS of the m_i can be defined as:

$$QoS(m_i) = \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2}$$

where, c_{ij} are application-specific coefficients and x_{ij} are model parameters.

Offline Solution Search In offline search of solution space, the system designer computes the optimum solution using the application context and design time information. For example, in the case of a human-machine system, the design decisions include the task allocation to human or machine counterparts. Consider the case of cruise control in

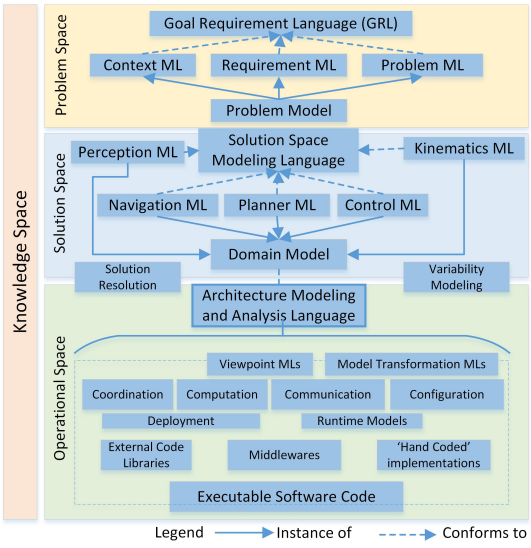

Figure 1: Overview of conceptual spaces, modeling languages, and models involved in SafeRobots Framework

today's cars, the steering control is allocated to the human driver while the automation is responsible for applying acceleration. In contrast, for an assisted parking application, the steering is controlled by the machine while the acceleration is provided by the human driver. This is an example for static function allocation considering the QoS of composed agents ($\sum m_i$).

Online Solution Search The offline search will not converge to a single solution in many applications. It is due to the fact that there may be multiple solutions that satisfy the given constraints or due to the partial context information available during design time. For example, in adaptive human-machine systems the function allocation is a run-time problem. For example, assistive lane keeping in automobiles 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. For explanatory purpose, in the Equation 1, there will be two model compositions - m_1 and m_2 representing `human_steering_control` and `automation_steering_control` respectively. In this example we use ACT-R human driver model proposed by (Salvucci, Boer, and Liu 2001). The QoS of m_1 and m_2 is defined as:

$$QoS(m_1) = c_{11}.\texttt{lane\_change\_score} + c_{12}.\texttt{production\_cycle\_time} \quad (3)$$

$$QoS(m_2) = c_{21}.\texttt{response\_time} + c_{22}.\texttt{confidence\_level} \quad (4)$$

where $\{c_{11}, c_{12}, c_{21}, c_{22}\}$ are application-specific coefficients that determine the interaction policies between human and automation models. The parameters (x_{ij}), such as `lane_change_score`, are ACT-R driver model parameters. The QoS composition represented by Equation 1 is subject to the following constraints:

$$\texttt{lane\_change\_score} \geq b1 \quad (5)$$

$$\texttt{production\_cycle\_time} \leq b2 \quad (6)$$

$$\texttt{response\_time} \leq b3 \quad (7)$$

where, $\{b1, b2, b3\}$ are context-specific parameter that are dynamically determined during run-time.

Semantic Knowledge Supported Model Evolution

The conceptual spaces in SafeRobots framework is hierarchically arranged in such a way that the lower layer uses the knowledge gained in the upper layer. The knowledge space consists of domain concepts represented using ontologies. The problem is modeling in the form of goals (i.e., hard goals and soft goals) and requirements modeled using Goal and Requirement Language (GRL). By applying the functional constraints provided by the problem space on the domain conceptual knowledge, the solution space for the given problem is modeled using SSML. The SSML can be extended by users to define domain-specific semantics, for example, ACT-R cognitive architecture semantic model. The NFP properties are specified along with their functionalities in the solution model. The strategy is to shift the decisions on NFP at a later stage when more information on operating context such as external environment, hardware, operating system, etc., are available. The operational model is filtered from solution model by applying non-functional constraints. The operational model contains concrete architectural model with variabilities that are resolved dynamically during runtime. The SafeRobots toolchain is being implemented using Eclipse Modeling Framework. The model evolves during the software development by incorporating the knowledge created in the form of models and helps to shift the critical decision when more information is available. There is a close relationship between machine learning and model evolution in software engineering complex HRI systems as both involves progress during optimization that can be viewed as a learning process.

3 Conclusion

In this paper, we discussed how AI techniques are applied in intelligent software tools and process to develop complex HMI systems. The main challenge is to adopt a systematic software development process so as to identify the parameters and optimization strategies to develop efficient and reusable software for HRI systems.

References

Harman, M. 2012. The role of artificial intelligence in software engineering. In *First International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, 1–6. IEEE.

Ramaswamy, A.; Monsuez, B.; and Tapus, A. 2014a. Modeling non-functional properties for human-machine systems. In *2014 AAAI Spring Symposium Series, Formal Verification and Modeling in Human-Machine Systems, Palo Alto, USA*.

Ramaswamy, A.; Monsuez, B.; and Tapus, A. 2014b. Solution Space Modeling for Robotic Systems. *Journal of Software Engineering for Robotics (JOSER)* 5(1):89–96.

Salvucci, D. D.; Boer, E. R.; and Liu, A. 2001. Toward an integrated model of driver behavior in cognitive architecture. *Transportation Research Record: Journal of the Transportation Research Board* 1779(1):9–16.