Addressing Multi-Domain Integration Challenge in Robotics using Model-Based Approach

Arunkumar Ramaswamy[1,2] and 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, Versailles, France
{arun-kumar.ramaswamy,bruno.monsuez,adriana.tapus}@ensta-paristech.fr

Abstract. Software development for robotic systems require knowledge from several domains. This paper highlights the application of domain modeling in SafeRobots Framework for designing robotic systems. We discuss how heterogeneous domain models can be modeled and integrated for systematic software development, and how it can be employed in intelligent model evolution.

Keywords: robotics, model-driven software engineering, knowledge-based systems

1 Introduction

System engineering robots poses an interesting challenge of integrating heterogeneous problem domains while designing the system. This heterogeneity can be seen in terms of conceptual domains (e.g., perception, control systems, mapping), models of computation (e.g., synchronous, asynchronous, discrete, continuous time), application domains (e.g., service robots, industrial robots, driver-less cars), etc. In the last decade, developing software for such complex systems was addressed by 'divide and rule' strategy by adopting component-based software engineering techniques [1]. This has resulted in a large number of middlewares (e.g., ROS [2]), code libraries (e.g., PCL [3]), and component frameworks developed by different research laboratories and universities. To a large extent, these frameworks have helped in rapid prototyping of individual functionalities, but system integration and analysis still remains an issue. The main reason is that these frameworks handle the system development at the level of software code. The system integration problem can be managed efficiently by increasing the level of abstraction.

In this paper, we highlight the application of domain modeling for robotic system design in our 'SafeRobots' framework [4]. Specifically, we discuss the following two questions with respect to SafeRobots Framework:

1. How domain knowledge is modeled in SafeRobots Framework?
2. How the domain models helps in model evolution during software development?

2 Related Works

By learning from the shortcomings of code-based approaches, the software engineering community in robotics is gradually moving towards Model-Driven Software Development (MDSD) approach. The Smartsoft framework is based on a model driven toolchain that support formal modeling of component skeleton that act as a wrapper around the user code [5]. The European project on Best Practices in Robotics (BRICS) provides guidelines and a framework to develop robotic components [6]. They are based on the separation of concerns between different aspects of Computation, Communication, Coordination, Configuration, and Composition. Currently it is in the developmental stage and only limited concepts have been integrated in the toolchain. RobotML, developed in the framework of the French research project 'PROTEUS' is a DSL for designing, simulating, and deploying robotic applications [7]. V^{3}CMM component meta-model consists of three complementary views: structural, coordination, and algorithmic views. However, it has not addressed any robotic domain specific aspects [8].

3 SafeRobots: A model-driven toolchain for software development in Robotics

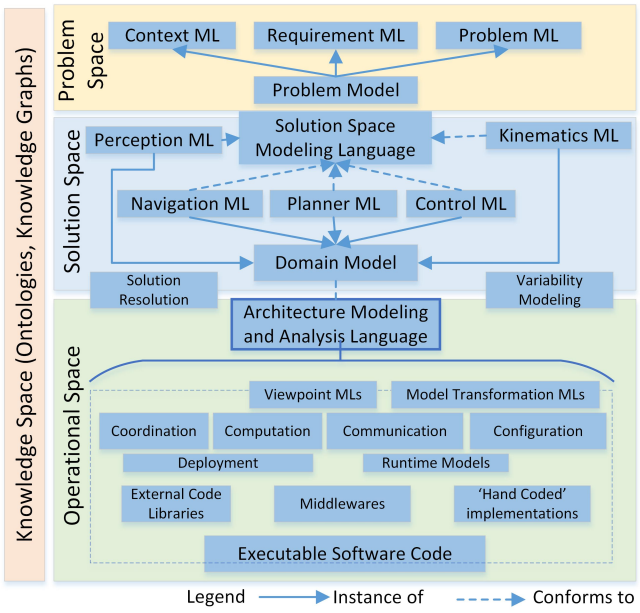


Fig. 1: Ecosystem of models in SafeRobots toolchain

Self Adaptive Framework for Robotic Systems (SafeRobots) is a model-driven toolchain that is currently under development in our lab at ENSTA-ParisTech. SafeRobots is based on three software engineering paradigms: Knowledge-based engineering, Model-driven engineering, and Component-based software engineering. A comprehensive discussion on SafeRobots can be found in [4].

In SafeRobots framework, the entire software development process is conceptually divided into three spaces: problem space, solution space, and operational space. Each space is supported by a knowledge space which acts as a bridge between the three spaces. The complete ecosystem is illustrated in Figure 1. In problem space, the problem, the requirements, and the contexts are modeled using appropriate Modeling Languages (ML). The solution space model captures the knowledge relating to domain concepts, computational algorithms, execution sequence, and their non-functional properties in a formal way. The Architecture Modeling and Analysis Language (AMAL) and its open semantic framework enables concrete architecture modeling in the operational space.

4 Domain Knowledge Modeling

In SafeRobots, the domain knowledge is modeled in two different phases: Problem Independent Knowledge Modeling and Problem Specific Knowledge Modeling. These two phases correspond to the knowledge space and solution space in Figure 1.

4.1 Problem Independent Knowledge Modeling

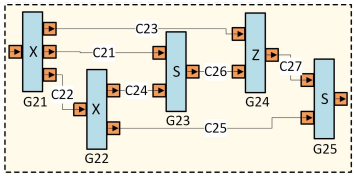
The domain knowledge modeling in this phase is independent of the problem specification or application constraints. The domain concepts are formally modeled using ontologies, Domain-Specific Languages (DSLs), Knowledge graphs, etc. The models at this level captures the robotic domain specific concepts, meta-data about the computational algorithms and standard interfaces, their structural dependencies, etc. The domain knowledge complements the various application specific development process by providing a knowledge base for abstract concepts such as image, point clouds, links, joints, platform, etc.

4.2 Problem Specific Knowledge Modeling

Problem-specific knowledge modeling or solution space modeling is performed with the help of functional requirements from the problem model as *constraints* applied to the domain model. In other words, a solution space model captures multiple solutions for the given problem by considering only the functional requirements and given domain knowledge base modeled in problem independent knowledge modeling phase. The strategy is to postpone the decisions on non-functional requirements at a later stage, since such properties can be estimated only when platform-specific decisions are made. In our approach, the solution

space is formally modeled using our modeling language, 'Solution Space Modeling language (SSML)'. Figure 2 shows a solution space model in SSML for lidar based vehicle tracking application. The connectors represents computational algorithms and their non-functional property model and the gates represents basic operations for composing different functional computational processes. The given model shown in Figure 2b capture three different solutions for the pointcloud segmentation problem. The three solutions that are modeled satisfies the functional goal of the problem model, but has different non-functional properties. A detailed discussion on SSML and the vehicle tracking application scenario can be found in [9].

(a) Experimental setup of lidar mounted vehicle



C21: 2.5D Projection,C22: Ground Classifier, C23: No Processing, C24: 2D Projection, C25: Euclidean Clustering, C26:Image Segmentation, C27: Box Mapping

(b) Solution space model of pointcloud segmentation operation

Fig. 2: An example for problem specific knowledge modeling in SSML for a vehicle tracking application

5 Knowledge Supported Model Evolution

The conceptual spaces from the SafeRobots framework shown in Figure 1 are 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. The non-functional properties (NFP) modeled using our NFP modeling language [10] are specified along with their functionalities in the solution model. The operational model is filtered from solution model by applying non-functional constraints provided by the problem model. The operational model contains concrete architectural model with variabilities that are resolved dynamically during run-time. Hence, the knowledge provided by the domain models are used by the tool to guide various phases of software development.

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.

6 Conclusion

In this paper, we discussed how modeling techniques are employed in SafeRobots framework to model domain knowledge and how they can be applied in intelligent software tools and process, to develop complex robotic systems. The main challenge is to adopt the domain model at the appropriate granularity to assist the system designer in systematic software development process to develop efficient and reusable software for robotic systems.

References

1. D. Brugali and P. Scandurra, "Component-based robotic engineering (part i)[tutorial]," *Robotics & Automation Magazine, IEEE*, vol. 16, no. 4, pp. 84–96, 2009.
2. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.
3. R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1–4.
4. A. Ramaswamy, B. Monsuez, and A. Tapus, "Saferobots: A model-driven approach for designing robotic software architectures," in *International Conference on Collaboration Technologies and Systems (CTS), Minneapolis, USA*. IEEE, May 2014.
5. C. Schlegel and R. Worz, "The software framework smartsoft for implementing sensorimotor systems," in *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1999, pp. 1610–1616.
6. R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Haegele, A. Pott, P. Breedveld *et al.*, "Brics-best practice in robotics," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK).* VDE, 2010, pp. 1–8.
7. S. Dhouib, S. Kchir, S. Stinckwich, T. Ziadi, and M. Ziane, "Robotml, a domain-specific language to design, simulate and deploy robotic applications," in *Simulation, Modeling, and Programming for Autonomous Robots.* Springer, 2012, pp. 149–160.
8. D. Alonso, C. Vicente-Chicote, F. Ortiz, J. Pastor, and B. Alvarez, "V3cmm: A 3-view component meta-model for model-driven robotic software development," *Journal of Software Engineering for Robotics*, vol. 1, no. 1, pp. 3–17, 2010.
9. A. Ramaswamy, B. Monsuez, A. Tapus *et al.*, "Solution space modeling for robotic systems," *Journal for Software Engineering Robotics (JOSER)*, vol. 5, no. 1, pp. 89–96, 2014.
10. A. Ramaswamy, B. Monsuez, and A. Tapus, "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*, March 2014.