Service-Oriented Robot Software Development Framework for Distributed Heterogeneous Platforms

Extended Abstract

Hyesun Hong
Seoul National University
Seoul, Korea
hshong@snu.ac.kr

Hanwoong Jung
Samsung Electronics
Seoul, Korea
jhw7884@gmail.com

KangKyu Park
Seoul National University
Seoul, Korea
gorgeous622@snu.ac.kr

Yeonggyo Yoon
SAP Labs Korea
Seoul, Korea
czerny92@gmail.com

Soonhoi Ha
Seoul National University
Seoul, Korea
sha@snu.ac.kr

ABSTRACT

A key technical challenge to make multiple robots work together is software development how to specify the mission at the user level and how to program each robot separately. In this paper, we propose a novel software development framework in which a mission is specified with a novel scripting language and the individual robot behavior with an extended dataflow model at the task level. How to relate these two specifications and how to generate the robot code automatically are also addressed in the proposed framework. The viability of the proposed methodology is validated with a preliminary experiment.

KEYWORDS

Software Development Framework; High-level Specification

ACM Reference Format:
Hyesun Hong, Hanwoong Jung, KangKyu Park, Yeonggyo Yoon, and Soonhoi Ha. 2018. Service-Oriented Robot Software Development Framework for Distributed Heterogeneous Platforms. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

In the near future, it will be common for a user to make diverse robots work together in various fields, including small mobile robots with limited energy and weak computation power [1][2][13]. A key technical challenge to realize this vision is how to specify the mission at the user level and how to program each robot considering the resource and/or energy constraints.

The traditional method to program a robot is to use the robot-specific programming environment provided by the robot manufacturer [3][7][8][9]. To increase the reusability of the software on various robot hardware platforms, several robotic software platforms have been developed recently for systematic software development. The most prominent robot software platform is Robot Operating System (ROS) [16], which is based on a component-based software

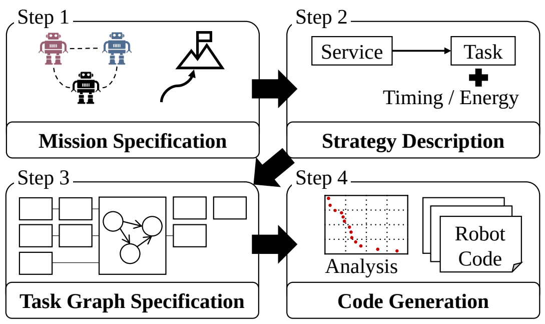

Figure 1: Overview of the proposed framework

```
MasterTeam: Robot r …
MasterTeam.Action.Move {
  move(Seoul National Univ)  … }
MasterTeam.A_MODE {
  set(Action, Move) …}
MasterTeam.main {
  case (A_MODE):
     catch(emergency): mode = RC_MODE
  case (RC_MODE): …. }
```

Figure 2: Mission scripting language example

design methodology. However, it has several weaknesses. Its resource requirement may be too high for miniature robots since it assumes a Unix-based operating system such as Linux. It is also not easy to control and coordinate multiple robots [6].

In this paper, we propose a novel software development methodology that separates mission specification and robot behavior programming. A new scripting language with dynamic mode changes and multitasking is devised for mission specifications. For robot behavior programming, on the other hand, we use an extended dataflow model for task-level behavior specification of each robot. The actual robot software is automatically generated from the extended dataflow model.

2 PROPOSED METHODOLOGY

The overall flow of the proposed software development methodology is shown in Fig.1 which can be understood as the refinement process among four levels of abstraction in software development.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

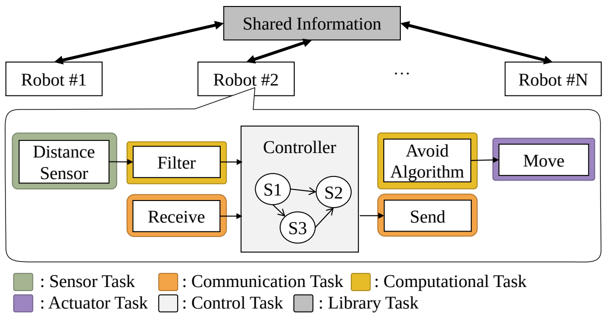

Figure 3: A task graph specification example for distributed robot platforms

The first step is *mission specification* at the highest level of abstraction with a scripting language. It can express team configuration, service-oriented programming, dynamic mode change of operation, and multi-tasking. To our best knowledge, no existent script language has such expression capability [4][12][15][18]. Fig.2 shows a snippet of mission specification written in our proposed script language. The function each robot can perform is abstracted with a *service* in our framework, and a user requests a service by its name. We also allow a user to define a *composite service* that executes the primitive services sequentially. Multitasking is not easy to express in popular script languages like python or lua. To support multitasking, we adopt the notion of *plan* from [5]; multitasking of a robot is represented by mapping one composite service per plan. A robot may have multiple operating modes depending on the environments and user requests. The generated event in the composite service can trigger a mode transition. In Fig.2, the mode conversion can be found on lines 7-9.

The second step is *strategy description* that defines the second level of abstraction. It provides more information on how to perform services. A service written in the mission can have different algorithms depending on various conditions and requirements. For service refinement, we describe the conditions and requirements for selecting the appropriate algorithm for each service. Non-functional requirements can be added at this step as well.

The next step is task graph specification that depicts the internal behavior of each robot to perform the mission. Unlike mission specification, we assume that the internal definition of a task is developed by a professional programmer. It is abstracted as a service function that a robot can perform. Recently compute-intensive services such as vision and machine learning are getting popular in robots. A compute-intensive service can be specified by a task subgraph that can be mapped to multiple processors for parallel processing.

To analyze the system behavior at compile-time, we apply a formal task graph model, extended from synchronous dataflow (SDF) [11] model for task graph specification. The SDF model that defines formal semantics for inter-task communication and task execution condition allows us to make the task scheduling decision at compile-time and estimate the performance and resource requirements. Our extended model uses finite state machine (FSM) to represent dynamic behavior [10] and a special type of task, called library task, to manage shared resources [14] among multiple robots.

Fig.3 illustrates a task graph for an autonomous driving scenario for a single robot. Distance sensor task is triggered periodically, and the triggered value is transferred to the control task. The control task is a special type of task, which plays the role of supervisor of the internal operation of a robot. Besides, Fig.3 shows how multiple robots share information. A library task supports shared resource management and server-client interaction, which is good for expressing the cooperation of heterogeneous robots.

The final step is to automatically generate the target code that runs on each processor from our extended dataflow model. This feature increases software design productivity by minimizing the possibility of human error in manual programming.

		ROS	Our Framework
The number of lines in the robot SW code	User-specified	1403	262
	Automatic generated	-	1152
	Total	1403	1414
Memory requirement (Unit: MB)	Application	73.51	45.37
	ROS framework	190.35	-
	Total	263.86	45.37

Figure 4: The number of lines in the robot software code and memory usage comparisons between ROS and our proposed framework for an autonomous driving scenario of a robot

3 PRELIMINARY EXPERIMENTS

To prove the viability of the proposed methodology, a preliminary experiment is conducted with a simple autonomous driving scenario whose task graph is represented in Fig.3, running on a V-Rep simulator [17] environment.

We compare our framework and ROS in two ways: the number of lines in actual robot codes and memory requirement. As shown in Fig.4, about 80% of the total code is automatically generated from the model-based task graph specification. The reduced number of manually written codes can serve as an indicator of how the productivity of software development is improved. Also, our experiment reveals that the memory requirement of ROS-based software is 5.82 times higher than the software designed by the proposed framework as seen in Fig.4. This is because additional processes such as *rosmaster* and *rosout* are executed to manage additional nodes and messages when ROS is running.

4 CONCLUSIONS

In this paper, we propose a novel service-oriented robot software development framework for distributed heterogeneous platforms. The proposed framework includes the high-level mission specification with an easy-to-learn scripting language and the model-based task graph specification for algorithm-level behavior specification of each robot. The proposed methodology is verified with an autonomous driving scenario experiment.

ACKNOWLEDGMENTS

This research was supported by a grant to Bio-Mimetic Robot Research Center Funded by Defense Acquisition Program Administration, and by Agency for Defense Development (UD160027ID). Also, it was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2016R1A2B3012662).

REFERENCES

[1] Paul Birkmeyer, Kevin Peterson, and Ronald S Fearing. 2009. DASH: A dynamic 16g hexapedal robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), 2009*. IEEE, 2683–2689.

[2] Yufeng Chen, Hongqiang Wang, E Farrell Helbling, Noah T Jafferis, Raphael Zufferey, Aaron Ong, Kevin Ma, Nicholas Gravish, Pakpong Chirarattananon, Mirko Kovac, et al. 2017. A biologically inspired, flapping-wing, hybrid aerial-aquatic microrobot. *Science Robotics* 2, 11 (2017), eaao5619.

[3] Pololu Corporation. 2017. Pololu. (2017). Retrieved April 05, 2018 from https://www.pololu.com/docs/0J62

[4] Xiaocong Fan, John Yen, Michael Miller, Thomas R Ioerger, and Richard Volz. 2006. MALLET-a multi-agent logic language for encoding teamwork. *IEEE Transactions on Knowledge and Data Engineering* 18, 1 (2006), 123–138.

[5] Enrique Fernández Perdomo, Jorge Cabrera Gámez, Antonio Carlos Domínguez Brito, and Daniel Hernández Sosa. 2010. Mission specification in underwater robotics. (2010).

[6] Lee Garber. 2013. Robot OS: A new day for robot design. *Computer* 46, 12 (2013), 16–20.

[7] HumanRobotics. 2017. Pob Robotics-Pobtools, Ri2bee. (2017). Retrieved April 4, 2018 from http://pobtools.software.informer.com

[8] Parallax Inc. 2016. Parallax-PropBasic. (2016). Retrieved April 05, 2018 from http://developer.parallax.com/propelleride/

[9] Parallax Inc. 2017. Parallax-Spin. (2017). Retrieved April 05, 2018 from http://learn.parallax.com/tutorials/projects/christmas-scribbler/how-mixing-gui-and-spin-code-works

[10] Hanwoong Jung, Chanhee Lee, Shin-Haeng Kang, Sungchan Kim, Hyunok Oh, and Soonhoi Ha. 2014. Dynamic behavior specification and dynamic mapping for real-time embedded systems: HOPES approach. *ACM Transactions on Embedded Computing Systems (TECS)* 13, 4s (2014), 135.

[11] Edward A Lee and David G Messerschmitt. 1987. Synchronous data flow. *Proc. IEEE* 75, 9 (1987), 1235–1245.

[12] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 16.

[13] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. 2009. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, Vol. 1. IPCB: Instituto Politécnico de Castelo Branco, 59–65.

[14] Hae-woo Park, Hanwoong Jung, Hyunok Oh, and Soonhoi Ha. 2011. Library support in an actor-based parallel programming platform. *IEEE Transactions on Industrial Informatics* 7, 2 (2011), 340–353.

[15] Carlo Pinciroli, Adam Lee-Brown, and Giovanni Beltrame. 2015. Buzz: An extensible programming language for self-organizing heterogeneous robot swarms. *arXiv preprint arXiv:1507.05946* (2015).

[16] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. Kobe, 5.

[17] Eric Rohmer, Surya PN Singh, and Marc Freese. 2013. V-REP: A versatile and scalable robot simulation framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013*. IEEE, 1321–1326.

[18] Daniel Augusto Gama de Castro Silva et al. 2013. Cooperative multi-robot missions: development of a platform and a specification language. (2013).