Accompanying Observation Modes and Software Architecture for Autonomous Robot Software

Zhe Liu, Xinjun Mao, and Shuo Yang
College of Computer,
National University of Defense Technology
Changsha, Hunan, P. R. China
Email: {liuzhe, xjmao, yangshuo}@nudt.edu.cn

***Abstract*—To support robust task execution in open environment, autonomous robots (AR) should include reactive capabilities to cope with the dynamics and uncertainties from real-world environment. The uncertainties pose great challenges for robots being sensitive to the environmental changes and flexible to adjust self-behaviors. To this end, this paper aims to improve the sensing and acting capabilities of autonomous robots by novel behavioral theories, observation modes and software architectures. Specifically, this paper has three main contribution: (1) presents an accompanying model that specifies a novel accompanying pattern for interacting robot behaviors; (2) proposes four types of accompanying observation modes that coordinate multiple robot sensing behaviors; (3) proposes a concrete multi-agent software architecture that implements aforementioned accompanying model and accompanying observation modes. To demonstrate the applicability and validity of our accompanying modes and MAS-based software architecture, this paper conducts a case study to implement a domestic service example, which requires the robot to run in a highly dynamic environment and can adapt its behaviors to unexpected situations.**
*

Keywords-Autonomous robot software; accompanying model; accompanying observation mode; multi-agent system

I. Introduction

Nowadays, more robots are increasingly applied in open environments (e.g., family, hospital, battlefield) and expected to autonomously behave without human beings intervention to achieve assigned tasks [1], [2]. Typically, we call them as autonomous robots (AR), which is a complex software-driven system. The software (Autonomous robot software, ARS) expects to (1) manage and control physical devices (e.g., arm, motor, leg) of robots; (2) make decisions on robots behaviors and drive robots to act; (3) perform computations on robot sensing data.

In presence of dynamics and uncertainties, the autonomous robot software is challenged with the following issues: (1) no explicit behavior theories that specify interacting processes between robot sensing and acting behaviors; (2) limited observation modes that effectively coordinate various robotic sensing behaviors; (3) lack of suitable software architectures that suitable to implement the novel behavior theories.

DOI reference number: 10.18293/SEKE2018-088.
*This research is supported by research grants from Natural Science Foundation of China under Grant No. 61532004.

In the field of the behavior of autonomous robots, there are many researches focus on behavior-based model for robot [3]–[5]. And in recent year, [6] presented a behavior-based hierarchical architecture and defined fourteen robot behaviors to assist telepresence control a humanoid robot. [7] reproduced the kind of individual recognition and attention that a human can provide in robot based on observed behavior. However, the relationship between robotic behaviors and behavior theories are not hot research directions. Sony company had simply classified and described robotic behaviors in their behavior module [8], but we think their behavior classification cannot be used for AR's behaviors because their method didn't meet the characteristics of AR. And for software architectures of ARS, there are several researches has combined ARS with MAS (multi-agent system). VOMAS (Virtual Operator Multi-Agent System) [9] is developed to support spontaneous generation of a task without the need to re-plan. COROS [10] is a multi-agent software framework for cooperative robots in which each of the agents represents a robot machine or a monitoring/control workstation.

This paper aims to improve the sensing and acting capabilities of autonomous robots by novel behavioral theories, observation modes and software architectures. Specifically, this paper has three main contribution: (1) presents an accompanying model that specifies a novel accompanying pattern for interacting robot behaviors. (2) proposes four types of accompanying observation modes that coordinate multiple robot sensing behaviors. (3) proposes a concrete multi-agent software architecture that implements aforementioned behavior models and observation modes.

The rest of this paper is organized as follows: Section 2 analyzes the features of autonomous robot behaviors through a motivating example and presents an accompanying model. Section 3 proposes four types of accompanying observation modes. Section 4 introduces a MAS-based architecture. Section 5 illustrates a case study and conclusion is made in Section 6.

II. Accompanying Behavior Theory

In this section we analyze the features of autonomous robot behaviors through a motivating example, and present the accompanying behavior model that specifies the novel behavior patterns.

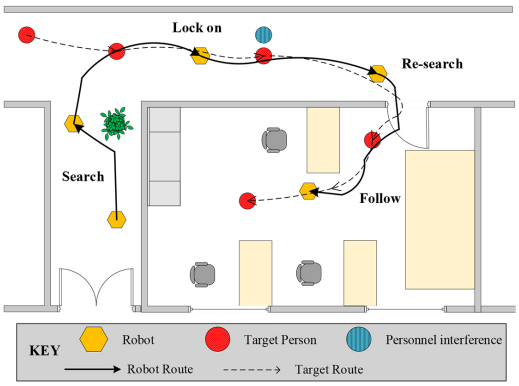


Figure 1. The motivating sample of autonomous robot

A. A motivating sample

Let us consider a domestic service robot example that motivates our research (see Figure 1). The service robot operates in an open home environment with several rooms and is designed to care for elderly inhabitants. Specifically, the robot expects to identify whether they have fallen off and provides the necessary information services, e.g., calling an ambulance or notifying the family. As the elderly person moves around, the service robot should follow the person within a safe distance to timely perceive his moving information, determine his safety status, and answer his service requests.

- *Scenario 1 (searching for the elderly person)*: the robot should search for the target by moving through the rooms autonomously and recognizing his body features. If the target is lost, the robot repeats the search for the target.
- *Scenario 2 (following the elderly person)*: when the target is found, the robot follows the target when he moves. The robot should follow the target closely to avoid losing the target.
- *Scenario 3 (lock on to the elderly person)*: when there is someone else next to the target, the robot should lock on to the target and avoid mistakes on recognizing the target person.

Through the example, we have found some common features of the autonomous robot behaviors: (1) the robot needs to plan a rational route and timely process multi-feedback from multiple robotic behaviors; (2) the robot should cope with the environment uncertainties by adjusting its behaviors to avoid obstacles or lock on to the right target. From the concrete example, we can claim that: (1) close connection and interaction between different behaviors of robots is important for acting robustly in open environments; (2) massive sensing information is necessary for robotic behavioral adjustment and adaptation.

B. Accompanying behaviors

For a complex task, robots usually need to carry out diverse behaviors to cooperatively reaching a common task goal. Some of the behaviors are responsible for observing the environment and sensing changes, whereas others drive physical actuator devices to achieve physical tasks. In this view, robot behaviors can be classified into two types: task behaviors that performed by actuators (e.g., arms, legs, and motors), and observation behavior typically performed by sensors or probes (e.g., sonar and cameras).

Although the two types of behaviors are relatively independent, the robot should enhance their synergy when achieving tasks, so that task behaviors can obtain on-demand feedback to adjust, optimize and self-manage the planned behaviors. Such a synergistic relationship between task behaviors and observation behaviors was defined as *accompanying behavior*.

C. Accompanying model for ARS

Furthermore, we found that accompanying relation doesn't only occur between observation and task behaviors, but can also occurs between observation behaviors. Multiple observation behaviors is the basis of accompanying behavior. For example, when robot follows the target, the robot should use the camera to lock on the target and use the radar to measure the distance between it and the target. There are several features in accompanying relations with observation behaviors:

- Interactivity: the observation and task behaviors of autonomous robots must interact with each other frequently and rapidly. The sensing information from observation behaviors allows the task behaviors informed of the status of task execution and make adaptation timely.
- Concurrency: when accompanying relation occurs, there are always more than one observation behaviors executing. Multiple observation behaviors can observe the environment in multi-dimension and improve the perception ability.
- Temporality: when multiple observation behaviors execute in parallel, the processing of massive observation information will show a kind of timing. Usually, these information will be processed sequentially. However, sometimes, one kind of information is more important for task execution and it should be given a high processing priority.
- Dependence: different accompanying behaviors reply on different types of information while interacting with each other. Some interactions depend on data-type messages, and others may depend on event-type messages.

Based on the accompanying behavior and the analysis above, we present an accompanying model for autonomous robot software (see Figure 2). In the model, we divide these behaviors into system-level behaviors that perform common robotic capabilities (such as planning behavior, scheduling behavior, etc), and application-level behaviors that relate to specific tasks (task behavior and observation behavior).

As shown in Figure 2, multiple observation behaviors are the core pf the accompanying model: (1) when accompanying relation occurs between observation behaviors, it can let software receive more accurate environmental information. (2) when accompanying relation occurs between task behavior and observation behaviors, it can let software know task execution more accurately and rapidly. Therefore, accompanying model have achieved two purposes for ARS: (1) On the one hand,

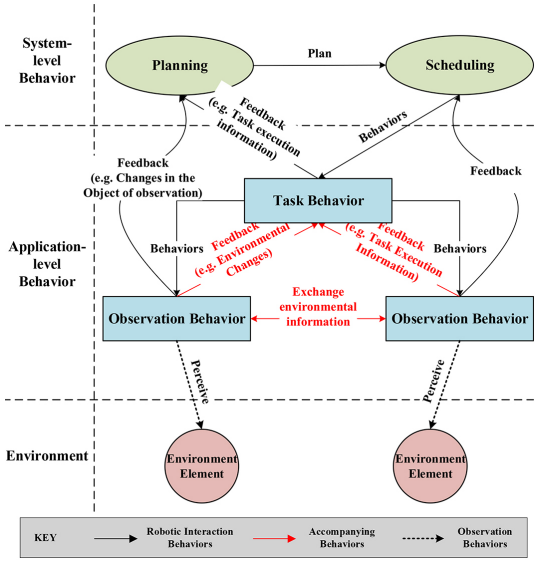


Figure 2. The accompanying model for autonomous robot software

multiple observation behaviors can enhance the sensitivity of the robot to the dynamic environment by more than one sensors. The accompanying behavior can help the robot make plan more accurately by multi-feedback. (2) On the other hand, accompanying model can assist autonomous robot software adjust its behaviors and tasks more flexibly with interaction between different behaviors.

III. Accompanying Observation Modes

This section identifies four types of accompanying modes for the autonomous robot software in runtime phase.

A. Classification of accompanying observation modes

As robots operate in open environments, different observation behaviors can form into diverse accompanying relations. In this section, we try to analyze the essential types of accompanying observation in following cases:

- Purpose-oriented: in aforementioned sections, there are two main purposes for accompanying relations based on multiple observation behaviors. One purpose is to improve the perception ability to the dynamic environment, while the other one is to facilitate flexibility and adaptivity for robot behaviors. From this view, we divide the accompanying observation modes into two major categories: (1) environment perception modes, (2) task assistant modes.
- Multi-dimension consideration: as illustrated in section 2, there are many characteristics for accompanying behaviors, and these characteristics of different dimensions can also result in diverse accompanying observation modes. For example, for interactivity, sometimes observation behavior needs to interact with other observation behaviors and sometimes neednt, considering this we design a cooperation mode for the former situation.

Based on aforementioned classification principles, we have identified four types of accompanying observation modes: *accident mode, observation behaviors cooperation mode, priority mode, non-priority mode*. And the characteristics of these modes in multiple dimensions as show in Table I. In Table I, **Interactivity** means the types of behavior which interact with each other in the mode; **Concurrency** means whether there are multiple observation behaviors occurs at the same time; **Temporality** means whether there are behaviors with high processing priority; **Dependence** means the type of message between behavioral accompanying in the mode.

TABLE I.
THE CHARACTERISTICS OF ACCOMPANYING OBSERVATION MODES IN MULTIPLE DIMENSIONS.

Dimension	Environment Perception Modes		Task Assistant Modes	
	AM	OM	PM	NPM
Interactivity	OB, TBs and SBs	OB and OB	OBs and TB	OBs and TB
Concurrency	No	Yes	Yes	Yes
Temporality	No	No	Yes	No
Dependence	Event-type	Data-type	Data-type	Data-type

[1] AM=Accident Mode, OM=Observation behaviors cooperation Mode, PM=Priority Mode, NPM=Non-Priority Mode
[2] OB=Observation Behavior, TB=Task Behavior, SB=System-level Behavior

B. Four types of accompanying observation modes

1) Accident mode: This mode is designed for some emergencies in runtime for software operation safety, which is a kind of event-type message dependence mode between one observation behavior, task behaviors and system-level behaviors. This mode can improve robotic emergency response to sudden changes from the environment. Autonomous robot software usually operates in a complex environment where many unexpected statuses may occur. These statuses sometimes have side effects on robotic task execution, and the software may not open all sensors to these uncertain statuses all the time.

For possible accidents from the environment, the software expects to plan corresponding observation behavior to detect them. When accident occurs, the observation behavior will send the event-type message to system-level behaviors. Then, the software will let the robot block current execution task and switch to pre-designed corresponding task to deal with the accident. As soon as the accident is removed, the software will let robot resume the task.

As described in the motivating example, when the depth sensor finds that there is an obstacle in front of the robot, it will send event-type message “Obstacle” to planning behavior, and the software will let robot stop to avoid the obstacle until there are no obstacle.

2) Observation behaviors cooperation mode: This mode is designed for the interaction between observation behaviors in runtime. The active interaction can help observation behaviors adjust themselves rapidly and on this basis robot can receive environmental information with more accuracy and efficiency. This mode is a kind of data-type message dependence mode.

Under observation behaviors cooperation mode, when the software needs multi-observation behaviors to observe a specific target or environment, observation behaviors can interact with each other to adjust their observation strategies.

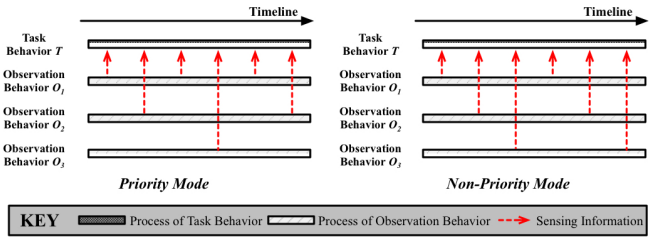


Figure 3. The processing sequence diagrams of Priority mode and Non-priority mode

3) Priority mode: This mode is designed for observation behaviors with different priorities. This mode is a kind of data-type message dependence mode between one task behavior and multi-observation behaviors. Generally speaking, different priorities mean different processing orders. In some tasks, the software should give some sensing information a higher processing priority than others. In other words, the software will process these sensing information more frequently and rapidly.

Under priority mode, for specific tasks, the processing priority of observation behaviors will be predefined in task behaviors or assigned by planning behavior. After collecting the sensing information, task behaviors will process these information according to their processing priorities.

4) Non-priority mode: This mode is designed for behaviors with the same processing priority and their information will be fused by task behaviors. This is a kind of data-type message dependence mode between one task behavior and multi-observation behaviors.

Under non-priority mode, observation behaviors have the same processing priority. The task behavior just need to receive the sensing information sequentially and process them as predefined according to tasks.

Figure 3 shows processing sequence of the same series of behaviors under priority mode and non-priority mode. In Figure 3, under priority mode, observation O_1 has a higher processing priority than O_2 and O_3. And under non-priority mode, all observation behaviors have the same processing priority and are processed sequentially.

IV. Software Architecture and Implementation Approach of Accompanying Observation Modes

A. MAS-based software architecture

The Multi-Agent System (MAS) provides an effective solution to address the development issues of high-level autonomous robot software for both modeling and implementation aspects. The autonomous robot software is a complex system that consists of a number of diverse and interacting components. All of these components work together to achieve the robots design objectives. Therefore, the autonomous robot software can be decomposed and organized as multiple autonomous agent entities. Furthermore, MAS maintains strong concurrency of plan execution, flexible interactions among diverse agent behaviors, and strong robustness of system functionalities. These features of MAS can greatly benefit

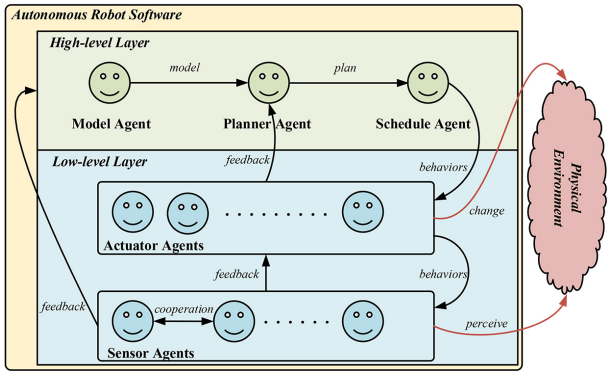


Figure 4. MAS-based software architecture for ARS

the implementation of accompanying model and observation modes, which require both concurrency and interaction mechanisms from implementation software architecture.

In this paper, we propose a multi-agent software architecture to implement the autonomous robot software and provide support for accompanying modes (Figure 4). In the architecture, each agent plays different role, takes distinct behaviours, and cooperates with each other to achieve the common task goals. Integrating with our previous works for the dual-loop control model [11], the multi-agent software architecture provides support for accompanying model and observation modes through agent communication mechanisms. However, in this MAS-based architecture, we don't consider the non-deterministic characteristic of MAS such as learning [12], self-adaptation and self-organization [13]. The description of the role that each agent plays in this multi-agent model is as follows:

- The *modeler agent* establishes the world model on the sensor inputs from sensor agents and offers the specifications of planning domain and problem to the planner agent for task planning.
- The *planner agent* performs planning jobs, including activities of establishing world models and planning over a specific problem domain.
- The *schedule agent* acts as a mediator that dispatches the generated plans to a specific actuator agent capable of the corresponding action.
- An *actuator agent* is implemented as the abstraction over the robot physical actuator and maintains a simple reactive structure, which will dispatche sensing tasks to sensor agents.
- A *sensor agent* controls an independent sensor device of the robot and is implemented to perform a stimulus-response behaviour aimed towards external state changes.

B. Implementation of accompanying observation modes

As shown in Figure 4, in our multi-agent software architecture, sensor agents (observation behaviors) can feedback sensing information to actuator agents (task behaviors) and high-level layer agents (system-level behaviors). Besides, sensor agents can interact with each other. Furthermore, multiple agents can be executed concurrently. These designs for the

architecture guarantee the implementation of accompanying modes.

In the implementation of this MAS-based architecture, we developed a multi-agent software framework *AutoRobot* [16] for developing autonomous robot software applications. In *AutoRobot*, MAS-based software architecture can be implemented under JADE [14] and robot controllers can be implemented under ROS [15]. For accompanying implementation, JADE has provided three communication behaviors for agents: One-shot, Cyclic and ThreeStep. Besides, we also design two interaction mechanisms for agents' communication: topics-based mechanism and services-based mechanism. These above communication behaviors and mechanisms guarantee the implementation of the accompanying modes.

V. Case Study

In this section, we implement the motivating example to validate the effectiveness and applicability of accompanying model and observation mode.

The hardware platform we adopted in the case study is the Turtlebot2 mobile robot, including a mobile base and a Kinect module that contains an RGB camera and a depth camera. The software infrastructure consists of the *AutoRobot* server and the ROS server.

A. Implementation of the motivating example

The case study for the aforementioned example is implemented using a multi-agent system prototype from the *AutoRobot* framework, and the concrete architecture for the case study is illustrated in Figure 5.

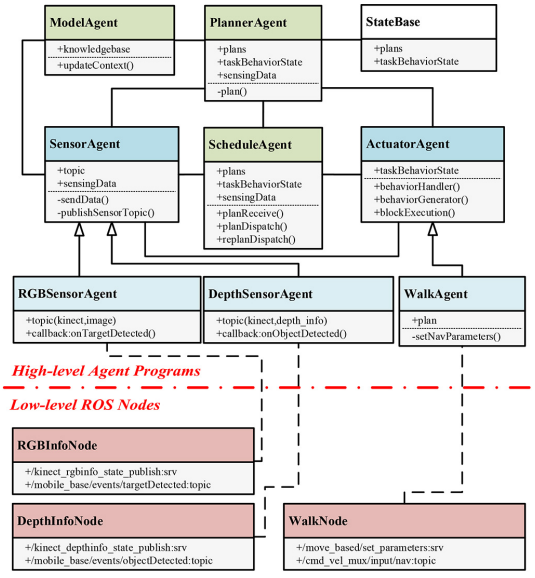
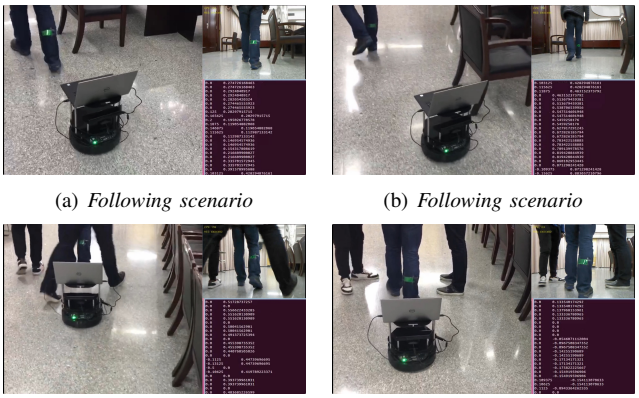


Figure 5. The multi-agent implementation architecture of the case study

For this specific example and the Turtlebot2 capability, we designed the following agent roles: 1) a WalkAgent to drive the mobile base to move around, 2) an RGBSensorAgent and DepthSensorAgent to obtain the RGB and depth images from the RGB and depth cameras.

Moreover, we implement the low-level robot controllers into ROS nodes that offer the fundamental robot functionalities. Each of the ROS node programs, such as the RGBInfoNode, corresponds to a high-level agent entity used to fulfil the agent's capability. The ROS nodes communicate with the high-level agent programs through the Java nodes that implemented by RosBridge middleware.

Under the implementation architecture, we implemented the service scenarios of *searching*, *following* and *locking on* the elderly inhabitants. In each scenario, the agent roles involved in the scenario include all agents in Figure 5. Figure 6 shows the actual effect of the implementation of these scenarios in human's view and robot's view.



(a) *Following scenario* (b) *Following scenario*

(c) *Lock on scenario* (d) *Lock on scenario*

Figure 6. The implementation of scenarios in real world. In (a) and (b) robot follows the target and avoids a chair. In (c) and (d) robot locks on the correct target when other people is surrounding.

B. Implementation of accompanying model and accompanying observation mode

The implementation details of accompanying model is shown in Figure 7 which is the sequence diagram of agent collaboration in the *Following* and *Lock-on scenario*. The accompany behavior is occurs between WalkAgent, RGBSensorAgent, DepthSensorAgent.

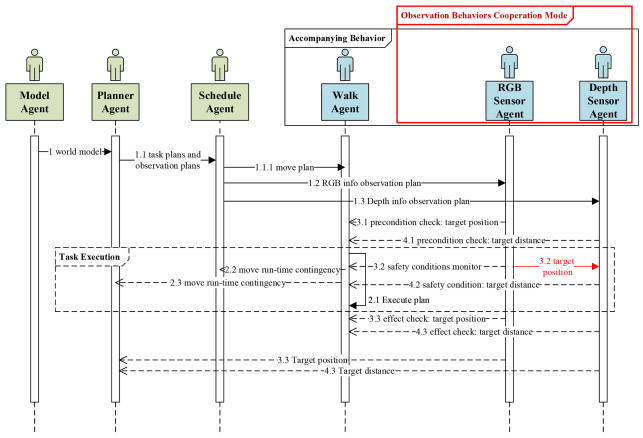


Figure 7. The sequence diagram of agent collaboration based on accompanying model in the *Following* and *Lock on scenario*

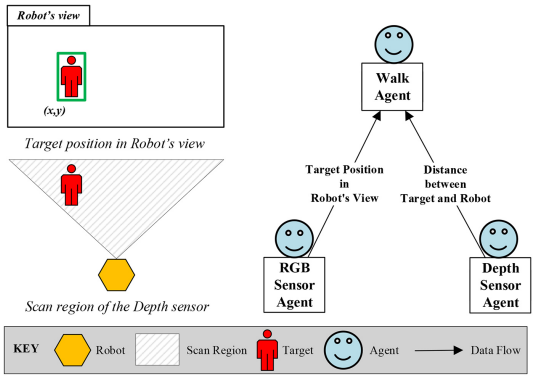


Figure 8. The scanning way of Depth sensor and the cooperation between agents in *Follow Scenario* in traditional way

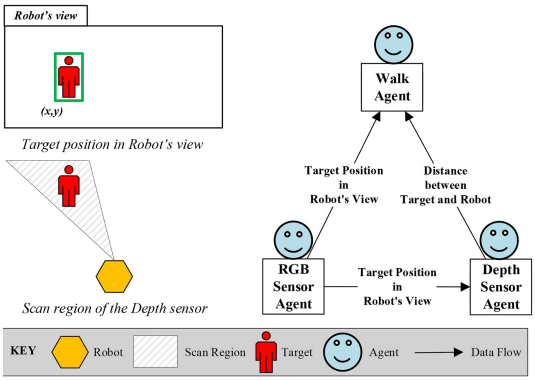


Figure 9. The scanning way of Depth sensor and the cooperation between agents in *Follow Scenario* with observation behaviors cooperation mode

The implementation details of one accompanying observation mode, observation behaviors cooperation mode, is shown in Figure 8 and Figure 9. In our case study, when robot is following the target, the software will open the observation behaviors cooperation mode between RGBSensorAgent and DepthSensorAgent. In *Following Scenario*, RGBSensorAgent is responsible to determine the position of the target in robot's view, while DepthSensorAgent is responsible to determine the distance between robot and the target. In a traditional way, RGBSensorAgent and DepthSensorAgent will feedback different sensing information to WalkAgent. Then WalkAgent will handle this information to calculate the location of the target and adjust the moving parameters. In real-world environments, the target may appear in any position in front of the robot and the DepthSensorAgent should scan a very large region to avoid losing of the target such as illustrated in Figure 8. The process of scanning and calculating the distance will cost lot of computing resource and spend more time than RGBSensorAgent, which will cause the robot losing the target.

When under observation behaviors cooperation mode, firstly, RGBSensorAgent will inform DepthSensorAgent where the target is to reduce the scan region as illustrated in Figure 9. And then RGBSensorAgent and DepthSensorAgent will send sensing information to WalkAgent. In this way, the processing time for the target location is greatly reduced and the sensitivity of the robot to the location of the target is greatly improved.

VI. Conclusion

This paper presents an accompanying model and four types of accompanying observation modes for autonomous robot software to solve the challenge about being sensitive to the environmental changes and flexible to adjust self-behaviors. Our contributions are threefold. First, we investigate the features of behaviors in autonomous robot software and present an accompanying model based on accompanying relations. Second, we identify a number of important accompanying observation modes. Third, we propose a MAS-based software architecture to examine and specify the accompanying observation modes.

References

[1] Mao, X. J. (2016). Autonomous Robot Software Technologies, *Communications of the CCF*, 12(7): 60-65.

[2] Ziafati, P. (2013). Programming autonomous robots using agent programming languages. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (pp. 1463-1464). International Foundation for Autonomous Agents and Multiagent Systems.

[3] Ringert, J. O., Rumpe, B., and Wortmann, A. (2014). A requirements modeling language for the component behavior of cyber physical robotics systems. *arXiv preprint arXiv:1409.0394*.

[4] Topalidou-Kyniazopoulou, A., Spanoudakis, N. I., and Lagoudakis, M. G. (2013). A CASE tool for robot behavior development. In *RoboCup 2012: Robot Soccer World Cup XVI* (pp. 225-236). Springer, Berlin, Heidelberg.

[5] Liu, C., and Tomizuka, M. (2017). Designing the Robot Behavior for Safe HumanRobot Interactions. In *Trends in Control and Decision-Making for HumanRobot Collaboration Systems* (pp. 241-270). Springer, Cham.

[6] Zhao, J., Li, W., Mao, X., Hu, H., Niu, L., and Chen, G. (2017). Behavior-Based SSVEP Hierarchical Architecture for Telepresence Control of Humanoid Robot to Achieve Full-Body Movement. *IEEE Transactions on Cognitive and Developmental Systems, 9*(2), 197-209.

[7] Glas, D. F., Wada, K., Shiomi, M., Kanda, T., Ishiguro, H., and Hagita, N. (2017). Personal Greetings: Personalizing Robot Utterances Based on Novelty of Observed Behavior. *International Journal of Social Robotics, 9*(2), 181-198.

[8] Hoshino, Y., Takagi, T., Di Profio, U., and Fujita, M. (2004). Behavior description and control using behavior module for personal robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (Vol. 4, pp. 4165-4171). IEEE.

[9] Hsu, H. C. H., and Liu, A. (2007). A flexible architecture for navigation control of a mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 37*(3), 310-318.

[10] Koubâa, A., Sriti, M. F., Bennaceur, H., Ammar, A., Javed, Y., Alajlan, M., ... and Shakshuki, E. (2015). COROS: a multi-agent software architecture for cooperative and autonomous service robots. In *Cooperative Robots and Sensor Networks 2015* (pp. 3-30). Springer International Publishing.

[11] Liu, Z., Mao, X., and Yang, S. (2017). A Dual-Loop Control Model and Software Framework for Autonomous Robot Software. In *Asia-Pacific Software Engineering Conference (APSEC), 2017 24th* (pp. 229-238). IEEE.

[12] Briot, J. P., de Nascimento, N. M., and de Lucena, C. J. P. (2016). A multi-agent architecture for quantified fruits: Design and experience. In *28th International Conference on Software Engineering & Knowledge Engineering (SEKE'2016)* (pp. 369-374). SEKE/Knowledge Systems Institute, PA, USA.

[13] do Nascimento, N. M., and de Lucena, C. J. P. (2017). FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things. *Information Sciences, 378*, 161-176.

[14] Bellifemine, F., Caire, G., Poggi, A., and Rimassa, G. (2008). JADE: A software framework for developing multi-agent applications. Lessons learned. *Information and Software Technology, 50*(1-2), 10-21.

[15] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... and Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

[16] Yang, S., Mao, X., Yang, S., and Liu, Z. (2017). Towards a hybrid software architecture and multi-agent approach for autonomous robot software. *International Journal of Advanced Robotic Systems, 14*(4), 1729881417716088.