Architecture of Navigation System Using CBD Method and UPnP Middleware

Joong-Tae Park
Dept. of Mechatronics
Korea University
Seoul, Korea
geullu@korea.ac.kr

Jae-Bok Song
Dept. of Mechanical Eng.
Korea University
Seoul, Korea
jbsong@korea.ac.kr

Sangseok Yun
Center for Intelligent Robotics Frontier
21 Program at Korea Institute of
Science and Technology
Seoul, Korea
ssyun@kist.re.kr

Abstract − This paper proposes the new integrated navigation system of a mobile robot in indoor environments. The main contributions of this system are as follows. First, the component based development (CBD) method used in this paper can make the navigation system scalable and flexible. Second, modules and components of the system are designed by the Unified Modeling Language (UML). Class diagrams in UML enable developers to clearly identify components and modules. Third, the proposed approach uses the UPnP (Universal Plug and Play) which has many advantages over other middleware like CORBA. The proposed navigation system has successfully been applied to several robots. Experimental results show that the architecture of this paper is efficient and easy to use.

Keywords − Navigation system, Software architecture, Integration, UML, UPnP.

1. Introduction

The service robots, which have drawn much attention these days, have many functions such as navigation, HRI (Human-Robot Interaction), manipulation, and so on. Among these, navigation is one of the most fundamental and important functions that a mobile robot possesses.

The navigation system consists of several technologies such as localization, mapping, path planning, and so on. For dependable navigation, these technologies should be integrated based on the well-defined navigation architecture. The needs for the architecture are as follows.

1. The well-defined navigation architecture is required for coherent and systematic combination of several components and modules into an integrated system.
2. There are many relations between each component. For example, the localization component obtains sensor data from the sensor components.
3. Navigation performance becomes highly dependent on the architecture, as new components are added, or existing components are removed or changed.
4. The well-defined navigation architecture ensures reusability of components.

So far, there have been some related research activities on architecture such as Minerva [1], RoboX [2], and PSR [3][4]. The Minerva was placed in the Smithsonian Institution in the US as a museum tour-guide robot. Minerva worked well even in complex environments. The RoboX was a guide robot shown in the national exhibition of Switzerland and operated for 5 months. In RoboX, various functions such as face recognition and range sensor based human detection were included. The PSR (Jinny) demonstrated at the 2003 Korea Science Festival. Also, it was tested in the exhibit hall of Hyundai Heavy Industries.

Despite the importance of the integration strategy and architecture, the above-mentioned approaches did not fully satisfy the following requirements.

1. To communicate with distributed components, the architecture includes the middleware. Besides, the architecture should be independent of robot platforms.
2. The components should be defined well and classified clearly.
3. The hardware interface components for sensors and actuators such as range sensors and wheel actuators should be developed so that they are hardware independent. If the interface components are defined well, the components can be transplanted to other robots without major modification.
4. The behavior components should be defined clearly in order to execute the various tasks such as autonomous navigation, human following, SLAM, etc.
5. The UML [5] based approach is recommended for the reusability of components and the extendable architecture. The UML is one of the best object oriented modeling languages and it provides the standard notation for software analysis and design.

This paper proposed a new architecture of navigation system which satisfies these requirements. The proposed architecture has two major advantages over the previous ones. First, the navigation system requires a lot of software components and modules for high functionalities. The UPnP [6] middleware integrates them systematically and increases the interoperability and portability between each component. Second, the proposed architecture provides the UML based diagrams. The diagrams also help to develop and integrate the components and modules. The interfaces of the components and modules are identified using the diagrams. The system can change the components and modules easily, which makes the architecture applicable to many robots without major modification of the architecture. Besides, the integration process is intuitively completed by just following the procedure as shown in the UML diagrams.

The remainder of this paper is organized as follows. Section 2 illustrates the architecture design. Section 3 deals with the UPnP middleware of the navigation system. Experimental results are shown in section 4 and finally in section 5 conclusions are drawn.

2. Architecture Design

2.1 Requirements Analysis

Analysis of the functional requirments of the system defines what the system provides to the user. Furthermore, it is the first step in software development. Some requirements needed for the development of the navigation architecture system were defined by this approach and these are described as follows.

- Navigation Behavior: It is the behavior which assists a human through navigation such as following a human. The navigation system consists of many navigation behaviors, each of which should be componentized.
- Behavior Component (BC): It is defined by the navigation behavior unit.
- Control Component (CC): It supervises the behavior component, communicates with other components through the standard XML message, and executes its own processes independently.
- Primitive Module (PM): It is the essential element for constitution of behavior components. For example, the ‘AutoMove’ component consists of various modules such as ‘Path Planner’, ‘Motion Control’ and so on.

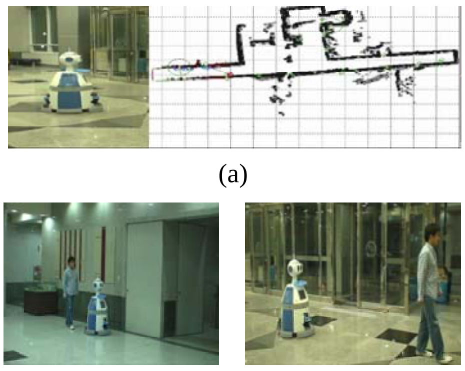

Fig. 1. Three types of navigation behaviors; (a) autonomous map building, (b) autonomous navigation, and (c) human following.

Some typical navigation behaviors which are necessary to the service robots can be explained as follows.

- Autonomous map building (AutoMapBuilding): For navigation of a service robot, mapping is very important because the map of the environment is required to estimate the robot pose. It can be built by exploration or SLAM algorithm. Figure 1(a) shows the process of autonomous map building.
- Autonomous navigation (AutoMove): It is the behavior for a robot to move to the goal without any collision, when a user inputs the goal pose. The process of autonomous navigation with the environmental map generated by the map building behavior is shown in Fig. 1(b).
- Human following (HumanFollowing): The human following behavior is one of the important techniques to assist a human. For example, the robot which coexists with a human should be able to robustly follow the specific person. Figure 2(c) shows that a robot follows a human through the human following behavior.

2.2 Component Based Architecture

This section describes the architecture of the navigation system. The navigation system used in this research works using both a range sensor and a vision sensor. All components are distributed on several SBCs (Single Board Computer).

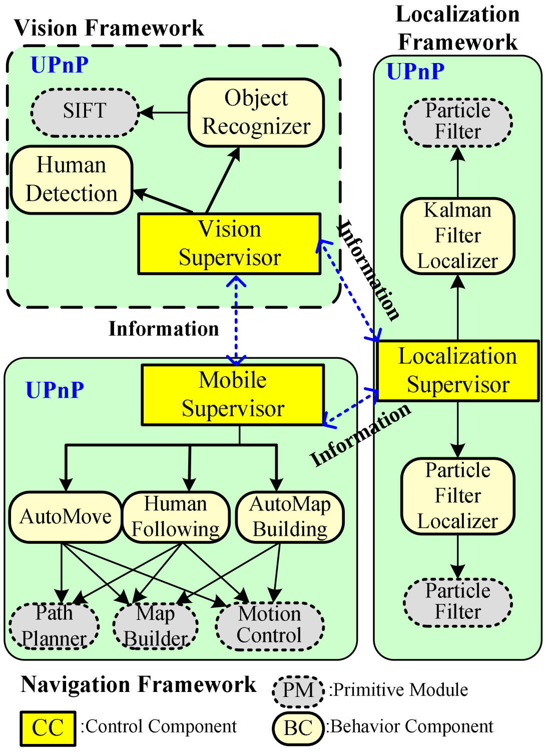

Fig. 2. Architecture of navigation system.

Figure 2 shows the architecture of the integrated navigation system. This architecture is classified into three parts; a vision framework, a navigation framework and a localization framework. Each framework consists of general components which can be divided into a behavior component and a control component which supervises behavior components.

As an example of the operation scheme using 'AutoMove' behavior, when a robot receives an order to move to the goal, the navigation system activates the 'Mobile Supervisor', 'Vision Supervisor' and 'Localizer Supervisor' components. The next steps are as follows:

- Step 1: The Control components load the behavior components such as 'AutoMove', 'Particle FilterLocalizer', and 'ObjectRecognizer.'
- Step 2: The components load the specific modules (PathPlanner, MapBuilder, and so on.).
- Step 3: 'Localizer Supervisor' estimates the current robot pose using the 'ParticleFilterLocalizer' component. In this case, 'Vision Supervisor' obtains the visual information from the 'Object Recognizer' component, and transmits it to 'LocalizationSupervisor'.
- Step 4: The 'MapBuilder' module builds the map.
- Step 5: The 'PathPlanner' module generates a path to the goal.
- Step 6: 'AutoMove' commands the translational and rotational velocities to the 'MotionControl' module to move.
- Repeat from Step 3 to 6 until the robot reaches the goal.

2.3 Component Design

The components are composed of several primitive modules. These modules must have reusability and scalability for high performance of components. Therefore, the modules are designed by the strategy pattern [7].

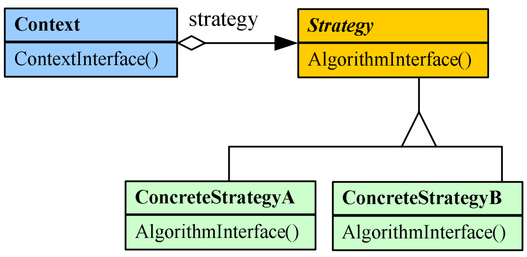

Fig. 3. Structure of strategy pattern

The strategy pattern defines a group of algorithms, encapsulates each one, and makes them interchangeable. This pattern allows the algorithms to vary independently from clients that use them. Fig. 3 shows the structure of the strategy pattern.

Figure 4 shows the whole structure of the 'AutoMove' component which used the strategy pattern. The 'AutoMove' component consists of various sub-modules which also consist of various algorithms. In this case, the strategy pattern is very useful. For example, when the 'AutoMove' component wants to change the 'LeftWallFolling', an algorithm of 'Motion Control' for 'DynamicWindow', another algorithm of 'MotionControl', The 'AutoMove' just change the 'pMC' object as shown in Fig. 5. For this reason, the strategy pattern increases the effectiveness of the components.

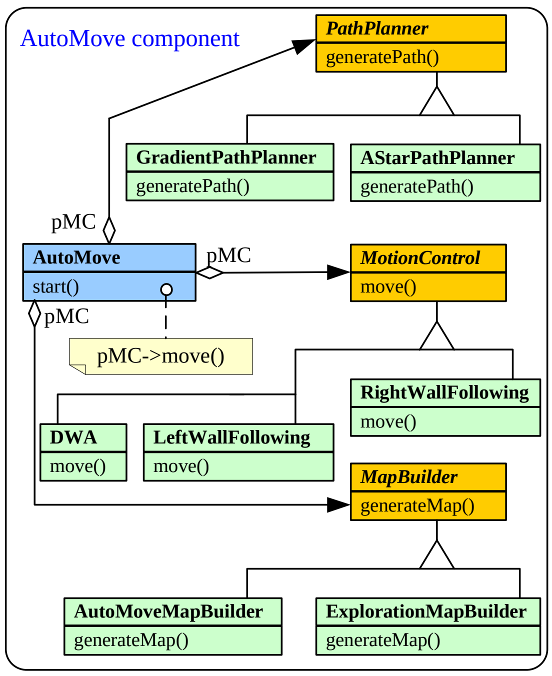

Fig. 4. Class diagram of AutoMove component.

```
MotionControl* pMC;
 Switch( localization condition)
 {
    case normal:
         pMC = new DynamicWindow;

    case abnormal:
         pMC = new LeftWallFollowing;
 }
pMC-->move();
```

Fig. 5. Sample code of strategy pattern.

2.4 Implementation Using UML Notation

In this paper, the class diagram, one of the UML notations, is used to develop the component. The class diagram is designed to implement various types of software modules. It represents association and hierarchy between components by following the UML standard.

There are two practical reasons for using class diagrams, even though the class diagram does not have theoretical meaning. First, the proposed architecture contains more than 100 classes. Therefore, the researchers should know association between classes for maintenance. Second, it is necessary that several researchers can modify or add their components without affecting other components. To sum, the class diagram helps the researchers can understand the whole software architecture.

3. UPnP Middleware

The Universal Plug and Play (UPnP) is the standard technology which uses the standard protocols and web technologies such as TCP/IP, UDP, HTTP, XML, and so on [8]. It enables the devices such as PCs, peripherals, intelligent appliances, and wireless devices to know each other automatically, when the devices are plugged into a network. For example, if you have a laptop and a printer connected to the network and need to print out a document, you can command such as 'print' to the laptop, and then the laptop sends a signal request if there is any printer on the network. The printer would identify itself and send its location in the form of a URL (Universal Resource Locator).

The main components of UPnP architecture are 'Devices', 'Control Points', and 'Services.' The 'Device' is an entity which provides services and a 'Control Point' is a service requester. A Service is a unit of functionality implemented by a device [9]. These concepts are applied to architecture of navigation system as shown in Fig. 6.

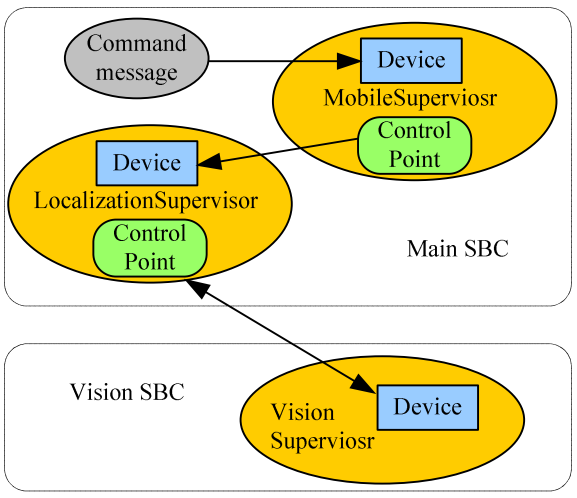

Fig. 5. Concept of UPnP middleware of navigation architecture.

In this architecture, the networking of UPnP is divided into five steps. Table 1 shows their simple descriptions.

Table 1 Steps of UPnP networking.

Steps	Description
Device Discovery	Control point finds interesting devices
Device Description	Control point learns about device capabilities
Device Control	Control point invokes actions on devices
Event Notification	Control point listens to state changes of device
Presentation	Control point controls devices and/or views device status using HTML UI

- Device Discovery step: When a device is added to the network, the UPnP discovery protocol (SSDP) allows that device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the SSDP allows that control point to search for devices of interest on the network.

- Description step: After the device discovery step, a control point should know the services which are executed by each device. Therefore, for the control point to know more about the device and its capabilities, or to interact with the device, the control point must retrieve the device's description from the URL provided by the device in the discovery message.

- Device Control step: After a control point has retrieved a description of the device, the control point can send actions to a device's service. To control a device a control point sends a suitable message using XML or SOAP. In response to the message, the service returns specific action values of fault codes.

- Event Notification step: A UPnP description for a service includes a list of actions the service responds to and a list of variables that model the state of the service at run time. The service publishes updates using event messages when these variables change. A control point may subscribe to receive this information.

- Presentation step: A control point controls device or views device status using the HTML user interface.

As an example of the UPnP networking scheme using a 'LocalizationSupervisor' and a 'MobileSupervisor', when the 'LocalizationSupervisor' is connected to a network, it advertises its service. For this result, the control point of the 'MobileSupervisor' finds the device. After that, these two components can communicate with each other through the control point and device, although each component is distributed on different SBCs. The above steps are done automatically. Thus, UPnP enables a zero-configuration network. From this result, the components which are wrapped by UPnP can join the network and also leave from the network easily.

4. Experiment Results

4.1 Robot Platforms

In this research, the proposed architecture of a navigation system is applied to the two types of robot platforms. The overviews of the robot platforms help to understand the proposed architecture. The two types of robot platforms shown in Fig.6 are developed by Center for Intelligent Robotics in Korea. These were demonstrated not only at the 2005 Asia-Pacific Economic Cooperation (APEC) in Korea, but also in many exhibitions from 2005 to 2006. Furthermore, the "T-Rot" shown in Fig. 6(a) has been exhibiting at the demo room which is a ubiquitous computing environment at KIST (Korea Institute of Science and Technology). The "Infotainment robot" shown in Fig. 6(b) has two single board computers (SBC). In terms of software environment, Linux Red hat 9.0 and RTAI (Real-Time Application Interface) are used as OS. The serveral sensors (e.g., laser sensors, infrared sensors, carmeras, etc.) and acutuators are equipped. The T-Rot has also serval sensors and actuators and possesses four SBCs.

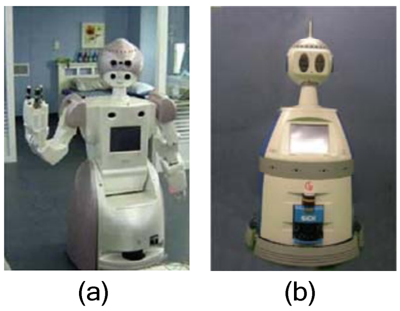

Fig. 6. Robot flatforms; (a) T-Rot and (b) Infotainment robot.

4.1 Verification of UPnP Middleware

This experiment was performed to verify the UPnP middleware using the ‘AutoMove’ behavior. To accomplish the ‘AutoMove’ behavior, three types of control components should be communicated with each other on the distributed environment, as shown in Fig.5. The navigation experiment was conducted successfully in the interaction room which is the ubiquitous computing environment at KIST. During navigation, the robot could estimate its pose correctly in real-time and avoided the obstacles. The robot can navigate successfully because the distributed components can communicate with each other within 10ms. It means the UPnP middleware helps to communicate easily between the distributed components on SBCs, and it does not affect the navigation performance. As the communication time between ‘Localization Supervisor’ and ‘MobileSupervisor’ increases, the robot cannot estimate its pose correctly.

Fig. 7. Easy interaction room.

4.1 Verification of Scalability and Effectiveness of Navigation Architecture

In this experience, the new navigation behavior is added to the architecture. Figure 8 shows the new architecture which includes a new behavior component. The role of a new behavior is ‘GlobalLocalization.’ When a robot performs the ‘AutoMove’ behavior, the robot should know its initial pose firstly. Therefore, the robot must have an ability which can estimate its pose in the unknown environment.

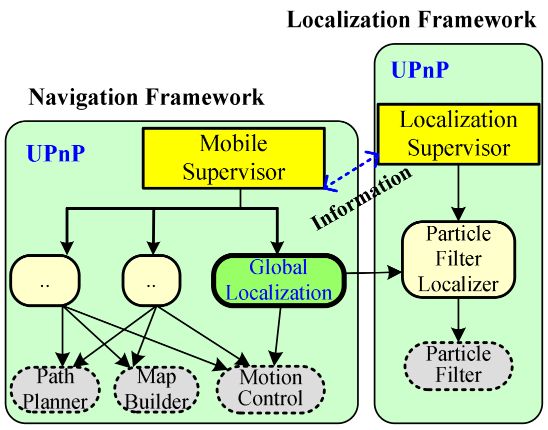

Fig. 8. Architecture which includes the new component.

```
NavigationBehavior* pNB;
 Switch( XML message)
 {
     case AutoMove:
             pNB= new AutoMove;
     case GlobalLocalization:
                   pNB= new GlobalLocalization;
 }
pNB->start();
```

Fig. 9. Sample code.

The new component can be added to the architecture with relative ease because the interface of components is clearly defined using the strategy pattern previously mentioned. Figure 9 shows the sample code which represents the new component is added to the existing architecture.

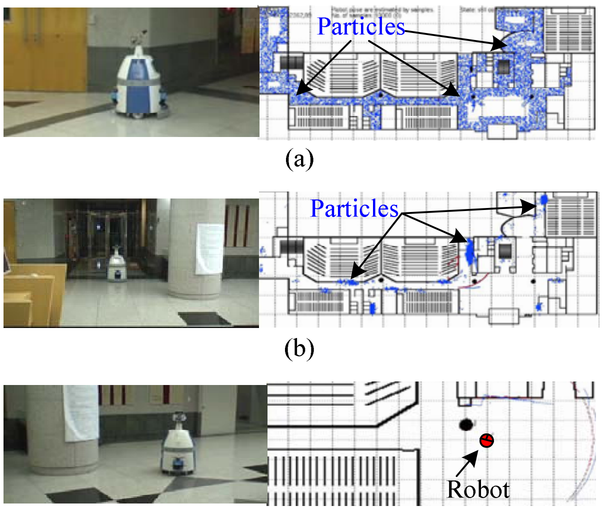

Fig. 10. Process of GlobalLocalization behavior.

Figure 10 shows the process of ‘GlobalLocalization’. The robot wanders in the building and draws particles on the map as shown in Fig. 10(a). The robot then estimates the pose probability of drawn particles until it finds its pose as shown in Fig. 10(b). When the robot finds its actual pose, the behavior is terminated as shown in Fig. 10(c).

5. Conclusions

In this paper, the architecture of the navigation system was proposed. The proposed architecture was applied to various robot platforms in a variety of task domains such as autonomous map building, autonomous navigation, and so on. The performance of the architecture was validated by a series of experiments. From this research, the following conclusions have been drawn.

1. The researchers can identify the components and modules through the UML properties such as the requirements analysis. Furthermore, the UML notations help to understand the whole architecture and decrease the development time. Thus, the UML method is suitable for architecture development.
2. In distributed environments, the robot architecture must have the middleware. In this research, the UPnP middleware was applied to the architecture successfully. Therefore, the UPnP is suitable for the robot middleware.
3. The CBD method increases the reusability of components and extensibility of the architecture.

Acknowledgements

This research was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

References

[1] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, and D. Fox, “MINERVA: A Second-Generation Museum Tour-Guide Robot,” *Proc. of the IEEE Conference on Robotics and Automation*, pp. 1999-2005, 1999.

[2] R. Siegwart, K.O. Arras, S. Bouabadallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, R. Philippsen, R. Piguet, G. Ramel, G. Terrien, and N. Tomatis, “Robox at Expo.02: A Large-scale Installation of Personal Robots,” *Robotics and Autonomous Systems*, Vol.42, No.3-4, pp.203-222, 2003.

[3] Gunhee Kim, Woojin Chung, Munsang Kim, and Chongwon Lee, “Tripodal Schematic Design of the Control Architecture for the Service Robot PSR,” *Proc. of the IEEE Conference on Robotics and Automation*, pp.2792-2797, 2003.

[4] Woojin Chung, Gunhee Kim, Munsang Kim, and Chongwon Lee, "Integrated Navigation System for Indoor Service Robots in Large-scale Environments ", *Proc. of the International Conference on Robotics and Automation*, pp.5099-5104, 2004.

[5] G. Booch, J. Rumbaugh, I. Jacobsen, *Unified Modeling Language User Guide*, Addison Wesley, Longman, 1997.

[6] UPnP, http://www.upnp.org.

[7] Erich Gamma, “Design Patterns: Elements of Reusable Object-Oriented Software”, ADDISON_WESLEY, 1995.

[8] Sang Chul Ahn, Jin Hak Kim, Kiwoong Lim, Heedong Ko, Young-Moo Kwon, and Hyoung-Gon Kim, “UPnP Approach for Robot Middleware,” *Proc. of the International Conference on Robotics and Automation*, 2005.

[9] Sang Chul Ahn, Jung-Woo Lee, Kiwoong Lim, Heedong Ko, Young-Moo Kwon, and Hyoung-Gon Kim, “UPnP Robot Middleware and its Application to Robot,” *Proc. of the International Conference on advanced Robotics*, pp. 2007.