# SOFTWARE FRAMEWORK FOR ASSISTIVE MOBILE ROBOT SYSTEM

Motoki Takagi, Yoshiyuki Takahashi, Takashi Komeda, Hiroyuki Koyama, Shin-ichiro Yamamoto

Shibaura Institute of Technology, Saitama-city, Japan

m705431@sic.shibaura-it.ac.jp

**Abstract: The Assistive MObile Robot System (AMOS), which helps a daily life of physically handicapped persons and interact with changing and unknown environments, must be controlled robustly. In order to achieve robust control, many sensors including vision sensors must be equipped. So the amount of sensors and functionality are increased and system becomes more complicated. In order to develop the complex and large system, the software framework for AMOS has been developed. It allows to be used in a distributed environment and easy development.**

## Introduction

In aging society with declining birth rate, the population of the elderly people with some kind of functional disorders will be increased. Also, the labor population of helper will be decreased. So the physical and mental burdens on the helpers will be increased. AMOS has been developed to help and reduce these burdens. The aim of AMOS is to pick up and transport daily use objects and placing them in designated indoor location semi autonomously.

AMOS is required to be operated in the same space of humans, also possibly pets as well and sensitive and expensive objects found in the home of private people, hospital or rehabilitation facility. It is important that the robot can avoid any undesired contact and collision with all of these, the robot must know the surounding information and controlled robustly. Therfore many sensors including vision sensors must be equipped. But the amount of sensors and functionality are increasing and system becomes more complicated.

In order to develop these complex and large system, the software framework for AMOS has been developed. It allows to be used in a distributed environment and easy development.

Figure 1: AMOS in Action

## Software Framework for AMOS

The system like AMOS, which interacts with changing and unknown environments and humans, must be controlled robustly and tasks to achive the complex activities must be controlled and coordinated redundantly in the system with restricted resources. Therefore, we have been developing the software framework which concept is based on the service principle [2] and shared responsibility approach [1]. CORBA Trading services [4] are used as a back bone of the framework. It allows easy component distribution and development. Component can be developed both under Windows and Linux environments.

In case the service requests the special resources like view, View Contract Managers and View Controller are developped which enable View Contract Manage approach [1]. If the properties of the pre-selected service indicates that the service requires specific resources, like view, the requesting component (View Requesting Service: VR), which is independent of the resources and has no knowledge about dependencies, sets up and delegates the final decision, which services (View Providing Service: VP), to be started on which robots and cameras, to the View Contract Manager. This decesion includes enable redundant processing, suppress conflicting combinations and keep costs low in order to allow for starting new services without stopping active ones. Costs and quality of the services, calculated on View Controllers, are used for the decision which is based on fuzzy rules.

The View Controller checks feasibility of the request on the given robot, i.e. if it can be reached considering the robot's dexterous workspace. The View Contract Manager requests a bid from individual View Controllers at what local costs and quality the service can be processed and which local (resource) conflicts with currently processed services will arise. From the bids of the View Controllers, the View Contract Manager selects the globally (sub-) optimal subset of services and their assignments to robots.

The view providing components are able to request the space of the interest (SOI) to be observed, which is called view request, to the View Controller. The view request is independent of the robot on which it is performed.

A main feature of View Contract Manage approach is that each View Controller's bid include several alternative options, which consists of their local quality,


ISSN: 1727-1983 

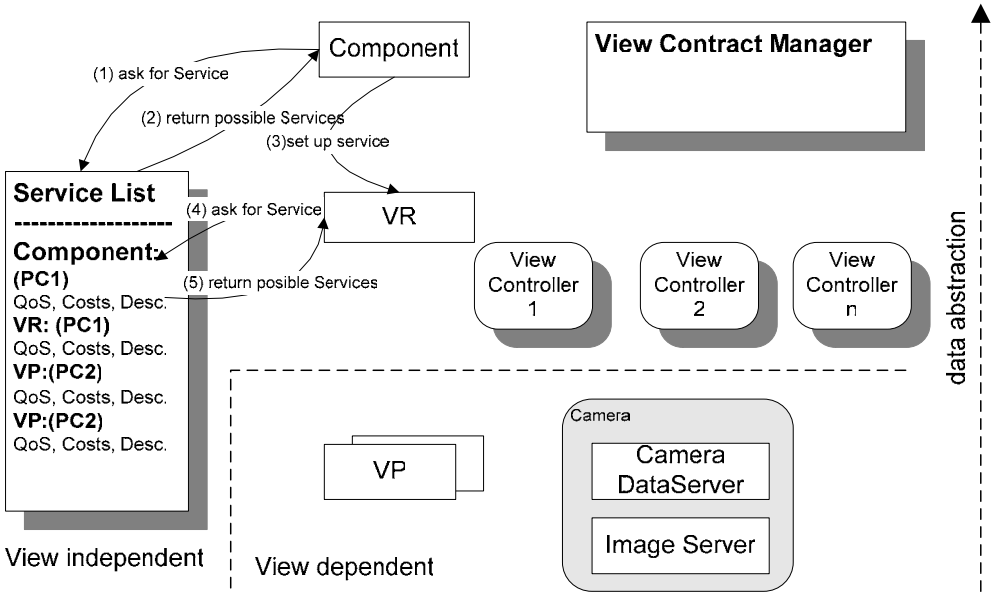


Figure 2: Setting up of the normal Service and pre-selection of VP

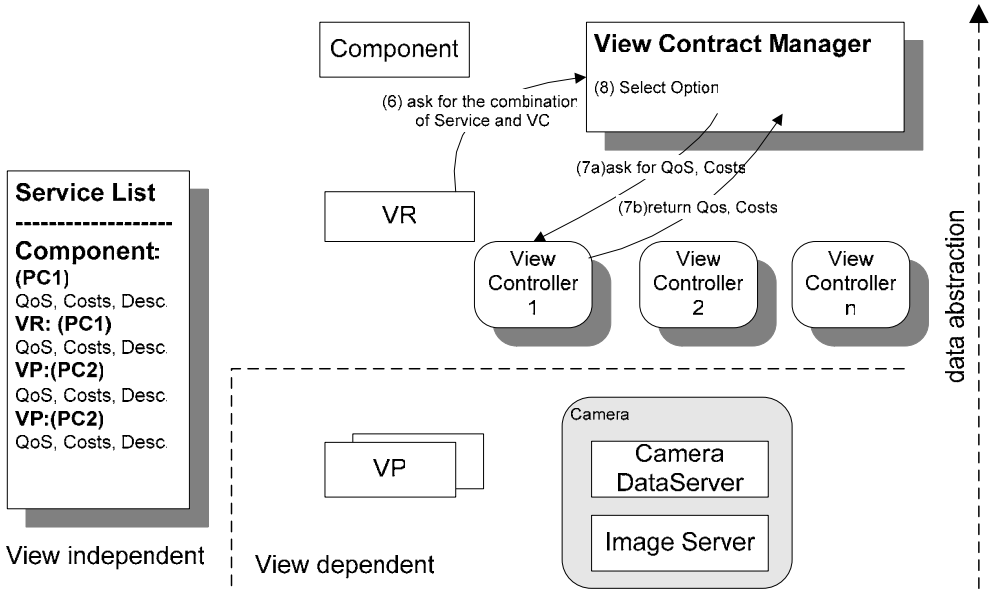


Figure 3: Delegate the decision of fitting VP and VC to VCM

cost and implications on the other services. Then View Contract Manager selects a globally sub-optimal service and location dynamically, balancing performance optimization by redundant processing with the avoidance of conflicts for the limited resources and the constraint of high reactivity alternative.

From Figure 3 to Figure 5 indicate the sequence of setting up normal services and view-dependent services. Service List, normal Component, View Requesting Service (VR), VP (View Providing Service), Camera Data Server, Image Server, View Controller and View Contract Manager are available in this example settings. At first all the components which can provide services offer the service and registered to the Service List. And then the component asks for the service to Service List and gets a selection of all fitting services. Then the component chooses one of pre-selected fitting service (VR) and set up it. And then VR asks for VP service to Service List and gets the candidate of the fitting services (Figure 2). Then VR ask for the fitting combination of VP and View Controller to View Contract Manager. Then View Contract Manager asks for the bit to the available View Controllers, and then finally View Contract Manager determines the best fitting service-robot combination based on the options (Figure 3). After the VP receive the response from View Contract Manager, it set up the VP service. Then VP set up the services of View Controller, Camera Data Server and Image Server based on the constraint of selected combination (Figure 4). Finally all the communications between services are established (Figure 5).


ISSN: 1727-1983 

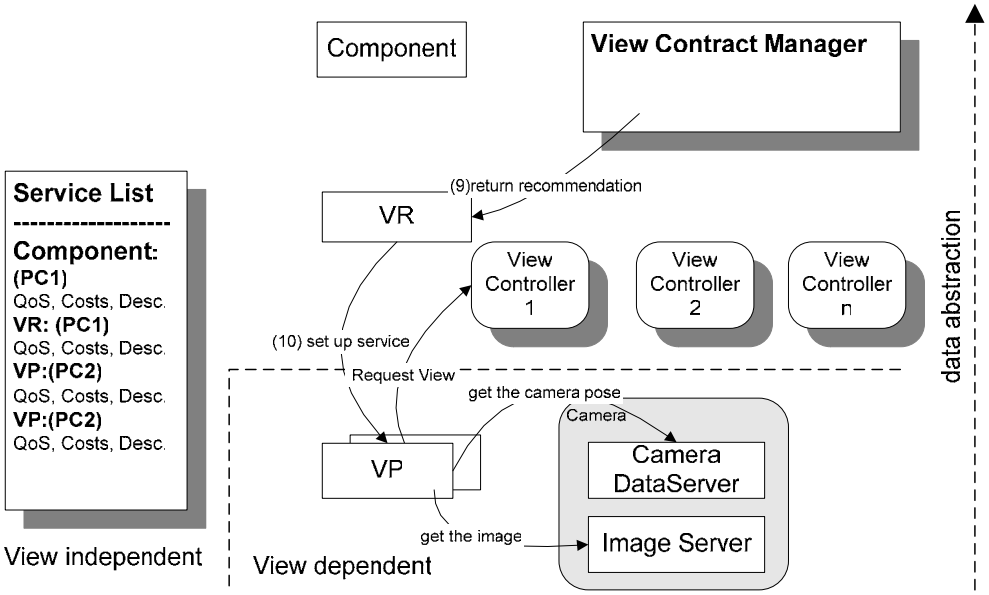


Figure 4: Setting up best fit VP and then services of CDS, IS and View Controller are set up.

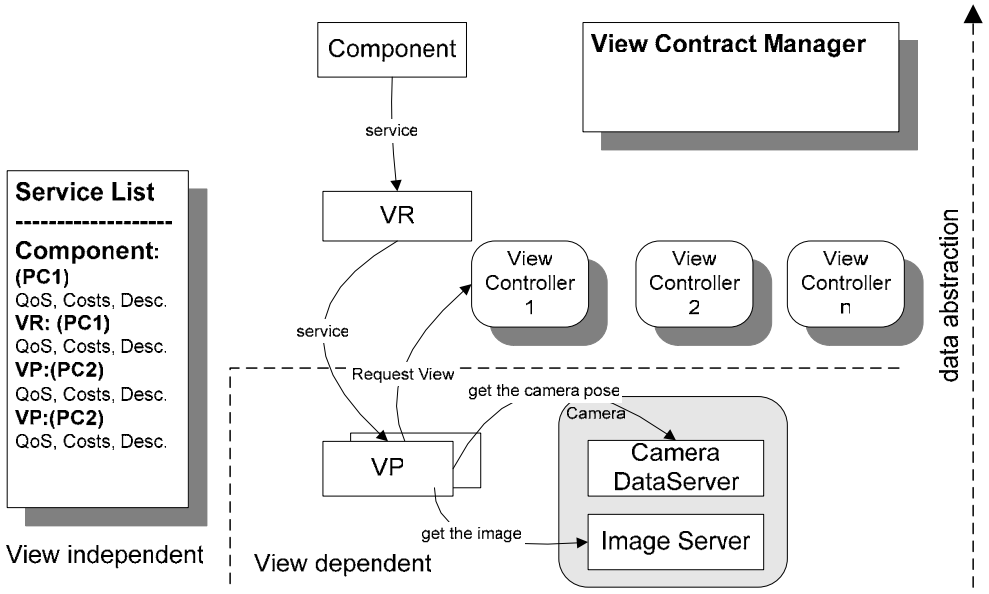


Figure 5: Communications between services are established.

**Software components for AMOS**

The software components for AMOS are developed. The scenario of the AMOS is defined as below. The operator controls the pan-tilt camera by touching touch panel to find the interest object in the room to transport. Then operator clicks the object in the touch panel. Afterwards, the Mobile Robot starts moving to the place where object is located and at same time the environment is extracted by the stereo camera, ceiling mounted cameras and range sensors. When AMOS reach to the place, then the manipulator is controlled to locate the robot hand in the specific pose to picks up the object and grasp it. Afterwards the object is transported to the specified place.

In order to develop the system easily, object-oriented design is introduced and each activities are designed as a component. Components are designed as simple as possible, so the each component provides only simple functionality. Like extracting object, move the mobile, picking up object are designed as single components. If the components only provide simple functionalities and designed simply, the reusability is expected to be increased. If the complex tasks are required, then these tasks are carried out by combining many simple components and services. AMOS's components are designed as hierarchical structure (Figure 7). The components located upper side in the hierarchy deal more abstract data. Rough layouts of the components and hard ware structure of the AMOS is given Figure 6.


ISSN: 1727-1983 

Human Machine Interface sends motion command to the Pan-Tilt Camera based on the operator's command, then the operator click the object to be picked up in the graphical user interface. Then the large SOI is generated and set as a rough destination of the mobile robot. When the mobile robot is moving, the range data from the sensors are used to build up the grid map and self location of the mobile robot is estimated. Afterwards, the requested object is extracted and location of the object is measured by the stereo camera mounted on the mobile robot. And also the type of object is recognised to decide the location of gripper to grasp the object. Then the manipulator is controlled to locate the gripper to pick up the object. Detail description of the components is available in below.

**Human Machine Interface**
It provides user interface to the operator. The operator can controls pan-tilt camera and can click on the touch panel to choose the object to be picked up.

**Activity Controller**
This component controls the sequence of whole activities which AMOS provides.

**Carrying Object Activity Controller**
This component controls the activity of the picking up object and mobile robot's activity.

**Picking up Object Activity Controller**
This component controls the sequence of picking up objects including the activity of image processing.

**Grasping Activity Controller**
This component is responsible for the manipulator's trajectory and gripper's motion.

**MobileController**
This component control the mobile robot's position, when motion commands are sent to this component.

**SlamSolver**
This component builds up the grid map and estimates self location of the mobile robot.

**PassPlanner**
This component estimates the pass from starting point to the distination based on the grid map build by the SlamSolver.

**Obstacle Detector**
This component extracts the obstacles in the environment.

**Object Extractor**
This component extracts the objects which is pickupable.

**Object Tracker**
This components tracks the object which is choosen by the operator.

**Hand Trajectory Generator**
This component generates the trajectory of the Robot Hand which is mounted on the manipulator.

**Moving Object Detector**
This component detects something moving in the environment.

**Object Recogniser**
This component recognises the object.

**Hand Motion Controller**
This components manages the motion command to the the robot hand.

**Range Data Server**
This component provides distance data taken from Ultrasonic Sensors or P.S.D. sensors.

**Image Server**
The component provides the image data to the components via shared memory or over network.

**Camera Data Server**
This component provides the camera parameters and poses to the components.

## Conclusion

We have developed the software framework for AMOS, which is based on service principle and shared responsibility principle. CORBA Trading services are used as a back bone. It allows easy component distribution and development. The components, which is under this framework, can be developed both under Windows and Linux environments. This framework allows to be used in a distributed environment and provides reusability. And also the components and services to realize the complex tasks of AMOS are designed.

The components designed in this paper will be implemented and evaluated in the experimnts in the next step.

## Reference

[1] TAKAGI M., EBERST C., UMGEHER G., (2003): 'Control of Redundant Attentive and Investigative Behaviors in an Active Cognitive Vision System', 3rd International Conference on Computer Vision Systems - ICVS 2003 Workshop on Computer Vision System Control Architectures

[2] PONWEISER W., UMGEHER G., VINZE M., (2003): 'A Reusable Dynamic Framework for Cognitive Vision System', 3rd International Conference on Computer Vision Systems - ICVS 2003, Workshop on Computer Vision System Control Architectures

[3] PRÖLL K., 'Contract Management for Transfer Jobs in Multiagent Robotic Systems'

[4] Object Management Group (OMG), CORBA services: Common Object Service Specification; March 1995

[5] STEFAN A. BLUM, 'A CORBA-based System Architecture for the Exploration of Indoor Environments with an Autonomous Robot'


ISSN: 1727-1983 

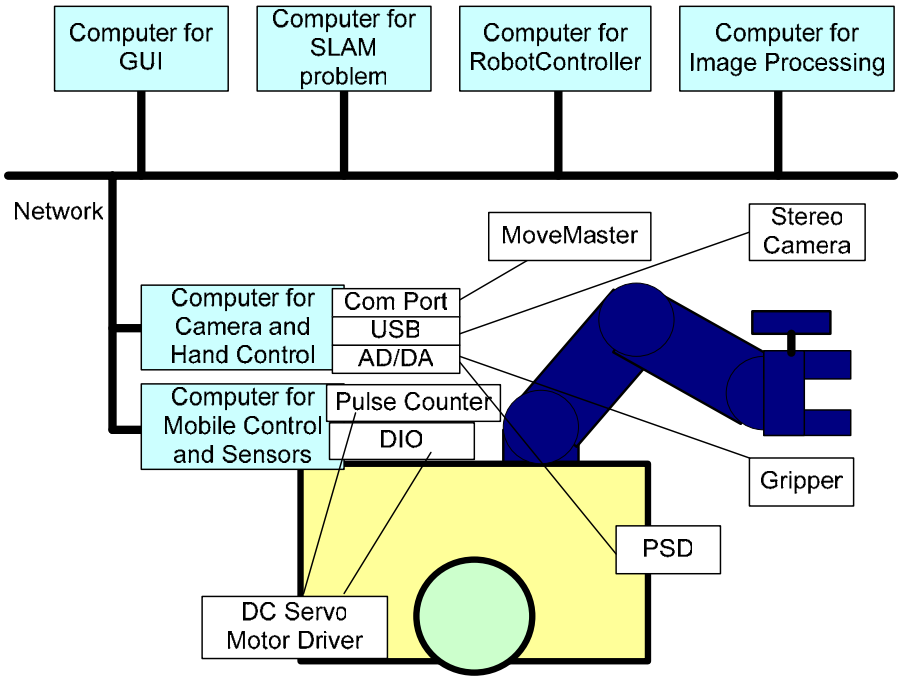


Figure 6: The system structure of the AMOS

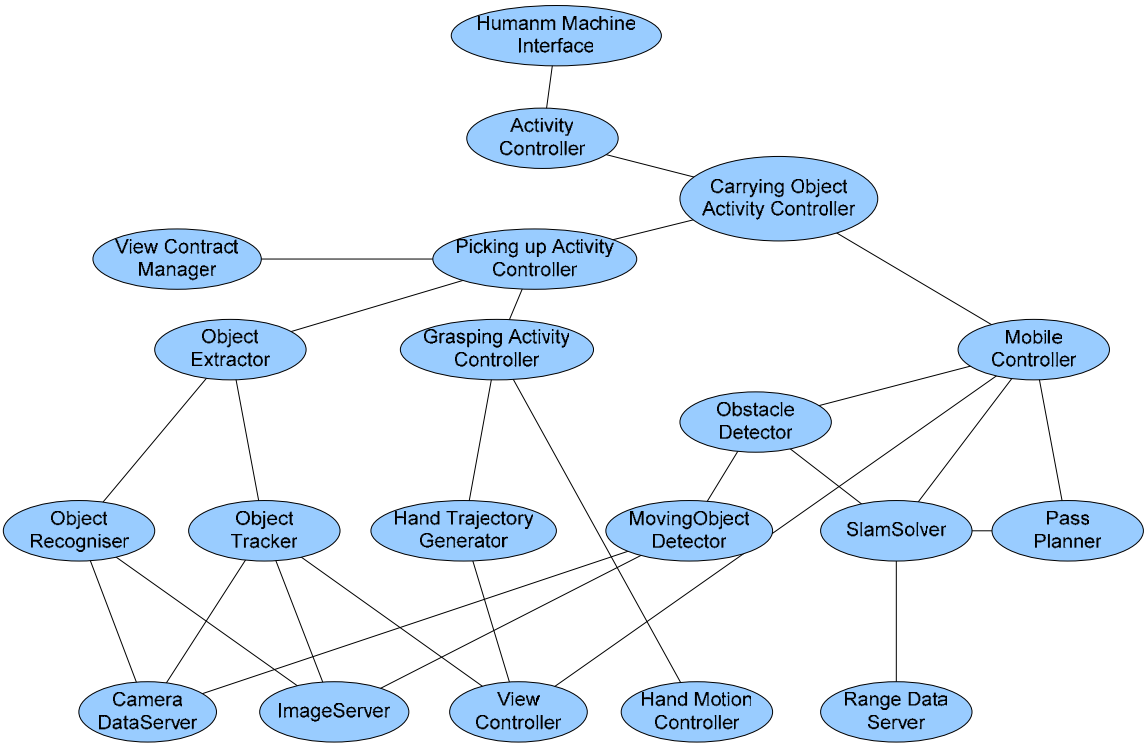


Figure 7: The hierarchy of the software components of AMOS


ISSN: 1727-1983 