Model Driven Developed Machine Vision System for Service Robotics

S.M. Grigorescu*, *Student Member*, *IEEE*, O. Prenzel†, A. Gräser*, *Member*, *IEEE*
*University of Bremen/Institute of Automation, Bremen, Germany
†Rheinmetall Defence Electronics GmbH, Bremen, Germany

***Abstract*—In this paper the modeling of the machine vision architecture ROVIS for service robotics is presented. The focus of the paper is not on the image processing algorithms used for robot environment understanding, but on the *Model Driven Development* (MDD) approach used for the design and implementation of the vision system. Starting from the basic requirements of ROVIS, the development process goes through three typical design phases: requirements analysis, system functional analysis and architectural design. The flow of information from the user of the rehabilitation robot to the vision system and back is modeled within the so-called *shared-control framework*. The proposed machine vision architecture is used in implementing the visual perceptual capabilities of the rehabilitation robot FRIEND.**

***Index Terms*—Model driven development, robot vision, service robotics.**

I. Introduction

In service and rehabilitation robotic systems, which have to operate in cluttered scenes under variable illumination conditions, a proper design of the robot's vision system is crucial [1]. Depending on the visual information precision, autonomous robotic actions can be performed. A robot controlled based on visual sensing is FRIEND (*Functional Robot arm with frIENdly interface for Disabled people*) [2], a semi-autonomous rehabilitation robot in its third generation designed to support disabled and elderly people in their daily life activities (Fig. 2). FRIEND has been developed at the Institute of Automation of University Bremen since 1997, within different projects [2]. Within the last research project AMaRob (*Autonomous Manipulator control for rehabilitation Robots*) the goal has been achieved of supporting disabled people with impairments of their upper limbs in Activities of Daily Living (ADL) (Fig. 1(a)) and professional life (Fig. 1(b)). Disabled patients gained independence from nursing staff for at least 1,5 uninterrupted hours. In different robot working scenarios a large number of action sequences like "pour and serve a drink", "take, prepare and serve a frozen meal", "fetch and handle a book" are necessary to fulfil the robotic system user's demands. On the system functional level, complex tasks are composed of a series of sensors and actuators operations.

The visual perceptual capabilities of the FRIEND robot are represented by the ROVIS (*Robust Machine Vision for Service Robotics*) machine vision architecture, described in [3]. The main features of ROVIS are the closed-loop image processing methods used for object recognition and 3D reconstruction [4], [5]. Here, closed-loop image processing refers to the automatic adjustment of image processing parameters with respect to external influences, e.g. variable illumination conditions, or scene uncertainty. In this type of feedback control system, the control signal is a parameter of image processing operation and the controlled variable a measure of image processing quality [5]. Hence, such a system should not be confused with traditional visual feedback control for active vision [6] or visual servoing [7].

(a) (b)

Fig. 1. Typical service robotic scenes from the FRIEND environment. (a) Activities of daily living (e.g. taking a meal out of a refrigerator). (b) Working at a library desk.

In this paper, the implementation of the ROVIS architecture using a *Model Driven Development* (MDD) approach is presented. Details regarding proposed image processing methods used in ROVIS can be found in [3], [4], [8]. Software engineering of image processing has been long treated in literature [9]. In [10], [11] basic C++ object oriented programming for general purpose image processing and analysis is suggested. The architectural design of robot vision system has been considered in many papers. In [12] a resource adaptive image processing framework for autonomous service robots is introduced, whereas in [13], [14], [15] real time and multimodal robot vision systems are considered. Hardware aspects concerning image processing modules for robots are treated in [16]. Vision architectures for manipulative tasks have been considered in quite a high number of publications. To cite only a few of them, in [17], [18] robot vision architectures aiming at achieving precise object grasping and manipulation are presented. In contrast to the referred papers, where robot vision systems are described mainly using block diagrams, the literature lacks robotic image processing frameworks described using standardized languages which enables rapid and transparent conversion into executable machine code. The MDD paradigm makes

Fig. 2. The rehabilitation robotic system FRIEND operating in complex environment.

use of the *Unified Modeling Language* (UML) [19], [20] to graphically represent software systems, as also hardware components. With the help of UML, in [21], [22] an agent based architecture for controlling a mobile robot is introduced.

The contribution of this paper is on the MDD development of the ROVIS system [3]. In Section II the concept of ROVIS is presented, followed in Section III by the modeling of the architecture. Section IV deals with hardware aspect regarding the implementation of the vision system. Finally, conclusions are given in Section V.

II. The ROVIS Concept

The objective of the rehabilitation robotic system FRIEND is to help disabled patients in their daily life activities. Thus, the robot operates in a human, unstructured environments, as depicted in the scene from Fig. 2. FRIEND is equipped with a 7 *Degrees of Freedom* (DoF) manipulator arm mounted on an electrical wheelchair and a series of sensors used by the robot to understand its environment, like a stereo camera system attached to a pan-tilt head. The purpose of ROVIS is to provide reliable visual information to the robot in order to plan the 7-DoF manipulator motion and to grasp the recognized objects of interest [23].

ROVIS is integrated within the Reactive Layer of the overall control system of the robot, presented in Fig. 3. Visual information is provided to the *Sequencer* which plans sequences of operations needed to perform specific tasks and to activate the planning of the manipulator arm. The Sequencer plays the role of a Discrete Event Controller that plans sequences of operations by means of predefined task knowledge. Through the functioning of the system, the computed data is shared between the modules with the help of the *World Model*. The World Model defines the information produced and consumed by the operations in the Reactive Layer. The *Human-Machine Interface* (HMI) operates at the user interaction level. The user commands are acquired with the help of different input methods such as speech recognition, chin control and Brain-Computer Interface and translated further into machine language for interpretation [24], [25].

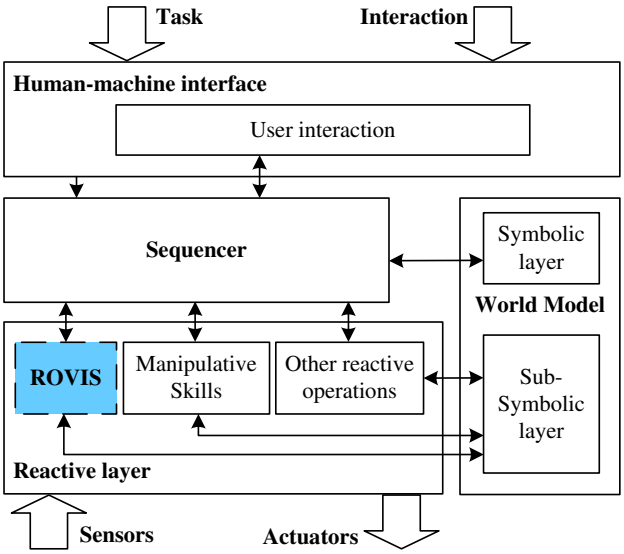

Fig. 3. Overall control architecture of the service robot FRIEND.

The software interconnections between the processing layers are implemented using the *Common Object Request Broker Architecture* (CORBA).

Fig. 4 shows a schematic overview of the ROVIS building blocks. Arrows connecting the blocks illustrate the flow of information through the ROVIS system as well as the connections of the ROVIS components with the external modules, the HMI and other reactive operations of the system FRIEND. As can be seen, there are two main ROVIS components: hardware and the object recognition and reconstruction chain. The ROVIS hardware consists of a Bumblebee® stereocamera system mounted on a pan-tilt head placed on a special rack behind the user, above his head, as illustrated in Fig. 2. Using a special input device such as a chin joystick, the user of FRIEND navigates the system in front of the container related to the particular scenario, for example a fridge. The stereo cameras view the scene in front of the robotic system including the manipulator and the tray mounted on the wheelchair in front of the user. In the ROVIS initialization phase the extrinsic camera

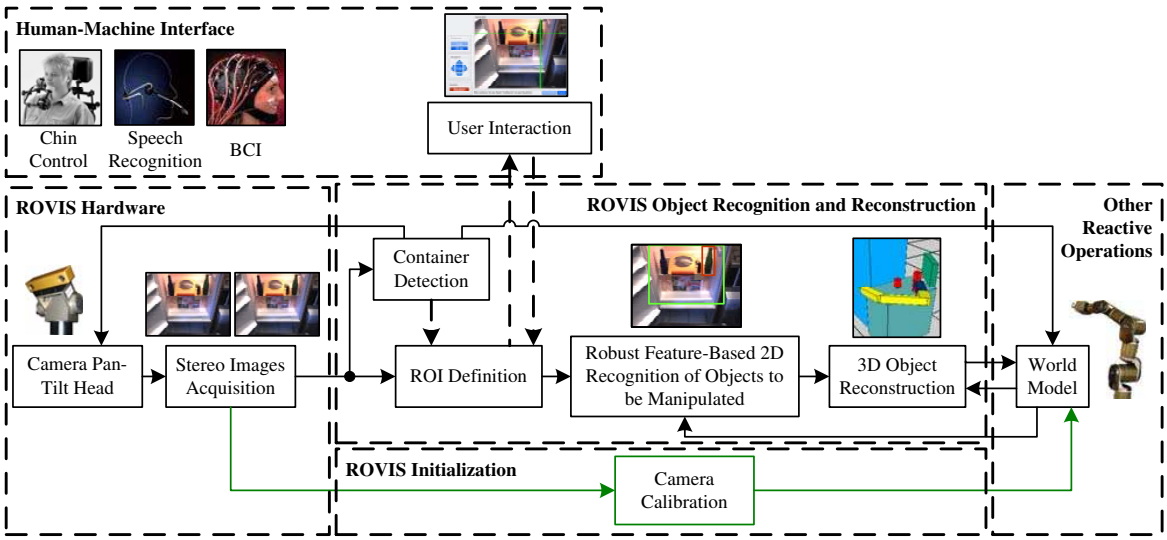

Fig. 4. Block diagram of ROVIS, the robust vision architecture of the service robotic system FRIEND.

parameters are calculated through camera calibration. The viewing angle of the sensors can be changed through the pan-tilt control so that the container can be detected in the image. This is illustrated in Fig. 4 by the feedback from Container Detection to the Camera Pan-Tilt Head block. The ROVIS object recognition and reconstruction chain consists of a sequence of image processing operations used for the extraction of features needed for both 2D recognition and 3D reconstruction of the objects present in the manipulator's environment. The main concept of ROVIS is to apply the image processing operations on an image *Region of Interest* (ROI) rather than on the whole image. This is motivated by the observation that people focus their visual attention on the region around an object when they grasp it as illustrated in Fig. 2. Information with respect to the 3D position of objects is needed for a "look-and-move" type of robot control. In order to achieve satisfactory precision of 3D stereo reconstruction the high image resolution of 1024x768px is used.

III. Modeling Strategy

A solution for the modeling strategy of ROVIS, starting from the block diagram from Fig. 4, is the *Unified Modeling Language* (UML) [19], [20]. UML wraps together several graphical language tools for modeling object-oriented problems. The birth of UML dates back to 1997 when it was standardized by the *Object Management Group* (OMG) consortium [26]. At the center of UML are nine types of modeling diagrams from which five are used for modeling the proposed machine vision system: *Use Case Diagrams*, *Class Diagrams*, *Sequence Diagrams*, *Statechart Diagrams* and *Activity Diagrams*. Over the past years UML has been extended with several other tools for coping with new problems suited for the UML standard. One such extension is the *System Modeling Language* (SysML), a tool used by systems engineers to specify and structure systems [19]. The adopted design process is divided into three phases:

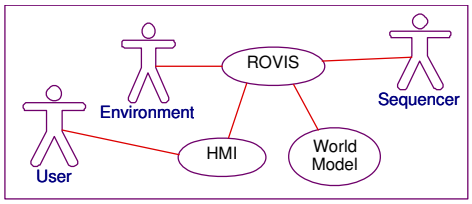

Fig. 5. ROVIS general use case diagram.

- *Requirements analysis*,
- *System functional analysis*,
- *Architectural design*.

These steps will be further detailed.

A. Requirements analysis

This phase represents the analysis of the requirements that define what the machine vision system should do, in other words its functional requirements. The graphical modeling interface for this step are use case diagrams, which describe specific operational aspects of the system. The system is represented here as a black box.

In Fig. 5 the interconnections of ROVIS with external systems can be seen. ROVIS is connected with two other use cases: the HMI and the system's World Model. The user of FRIEND is modeled in Fig. 5 as an actor. He interacts with ROVIS through the HMI. The *Environment*, or scene, in which FRIEND operates is modeled also as an actor connected to the vision architecture. The *Sequencer* is modeled as the requester of visual information. The predefined task knowledge with which the Sequencer plans sequences of operations is formally specified and verified a priori in a scenario-driven process [27]. It is flexibly applicable in different situations and environments due to the usage of object classes, as detailed below. The visual data processed by ROVIS is finally stored in the World Model.

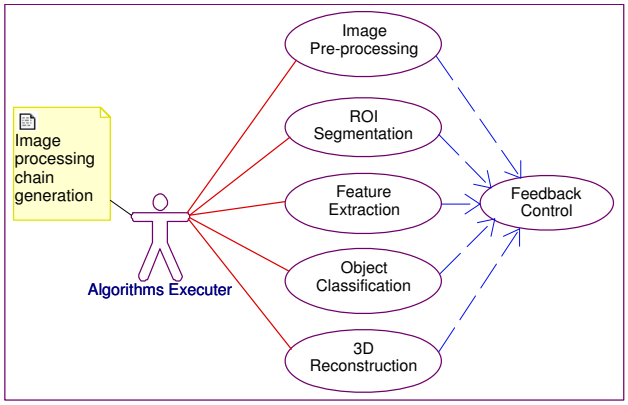

Fig. 6. ROVIS object recognition and reconstruction chain modeling.

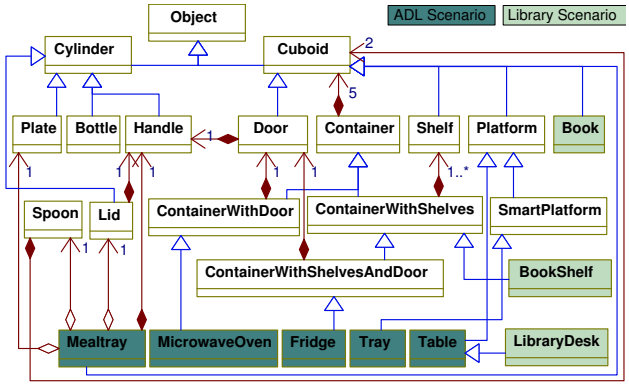

Fig. 7. Hierarchical ontology of objects that are involved in two rehabilitation robotic scenarios (ADL and library scenarios).

Another important requirement for ROVIS is the automatic construction of the object recognition and 3D reconstruction chain, also called image processing chain, which puts together sequences of image processing operations needed for object detection. The model of this process is depicted in Fig. 6. The five types of basic image processing operations, or primitives, are modeled as five use cases: *Image Pre-processing*, *ROI Segmentation*, *Feature Extraction*, *Object Classification* and *3D Reconstruction*. The ROI definition algorithms and the camera calibration methods are considered pre-processing operations. In order to achieve a high robustness of the vision system with respect to external influences the five types of image processing methods are connected to an extra use case which models feedback structures used within image processing operations. Depending on the type of objects that have to be recognized, appropriate object recognition operations are called. For example, for the case of uniform colored objects, region based recognition is to be used in object detection [28]. On the other hand, the detection of textured objects is performed via boundary based object recognition [28]. Such methods, based on the inclusion of feedback control at image processing level, are included in the use cases from Fig. 6. The dynamic image processing chain is automatically constructed by the *Algorithms Executer* which connects the proper vision methods needed by a specific scenario.

Keeping in mind the complexity of the FRIEND scenarios mentioned in introduction, the machine vision system has to deal with a variety of objects, including bottles, glasses, meal-trays, books, tables and containers (e.g. refrigerators, microwave ovens, etc.). From the image processing point of view, the objects to be recognized are classified into two categories: “container” objects, such as the fridge, microwave oven and book shelf, and objects to be manipulated such as bottles, glasses, meal-trays and books. The scenario-related task knowledge is provided by the Sequencer through *object classes*. Whenever the Sequencer activates a system operation, relevant information of object classes involved in the operation are made available in the system’s World Model. This information is specified via object class characteristics encoded in an extensible ontology. A simplified section of this ontology is depicted in Fig. 7, where the objects involved in the ADL scenario as well as in the library scenario are pointed out. In case of the fridge, which is a part of the ADL scenario, the characteristics *IsContainer*, *HasShelves* and *HasDoor* will be made available. For the tray with the meal (meal-tray) the knowledge about its components *Plate*, *Spoon* and *Lid* is supplied.

Besides providing object class characteristics during task execution, the task planner within the Sequencer also operates on the basis of object classes and plans context-related operations. Among others, the following class-based categories of ROVIS operations can be activated by the Sequencer:

- AcquireObjectBySCam: Determine the object’s location and size via the stereo camera (SCam) system. This operation is used to determine single objects or parts of objects (e.g. a handle) or containers where other objects are placed (e.g. fridge, table, gripper).
- AcquireGrippedObjectBySCam: Determines the gripping location and size of the object in the gripper via SCam.
- AcquireObjectInContainerBySCam: Determine location and size of an object in a container via SCam.
- AcquireObjectOnPlatformBySCam: Determine location and size of an object on a platform via SCam.
- SearchFreeLocationOnPlatformBySCam: Determine a free location via SCam which is suitable to place a certain object on.

These ROVIS operations are used during execution of a task, but also within the *initial monitoring* process, which is performed in the Sequencer after task activation by the user. Initial monitoring is the procedure which organizes the supply of scenario-related object characteristics in the World Model according to the object anchoring principle [29]. This sets the basis for distinguishing between the handling of indefinite objects of a certain object class (e.g. *a bottle*) within the ROVIS operations or the handling of a definite instantiation of a specific object class (e.g. *the small green bottle*). The difference between indefinite and definite objects is a runtime decision during anchoring, where a connection is established between symbolic object characteristics and concrete sub-symbolic

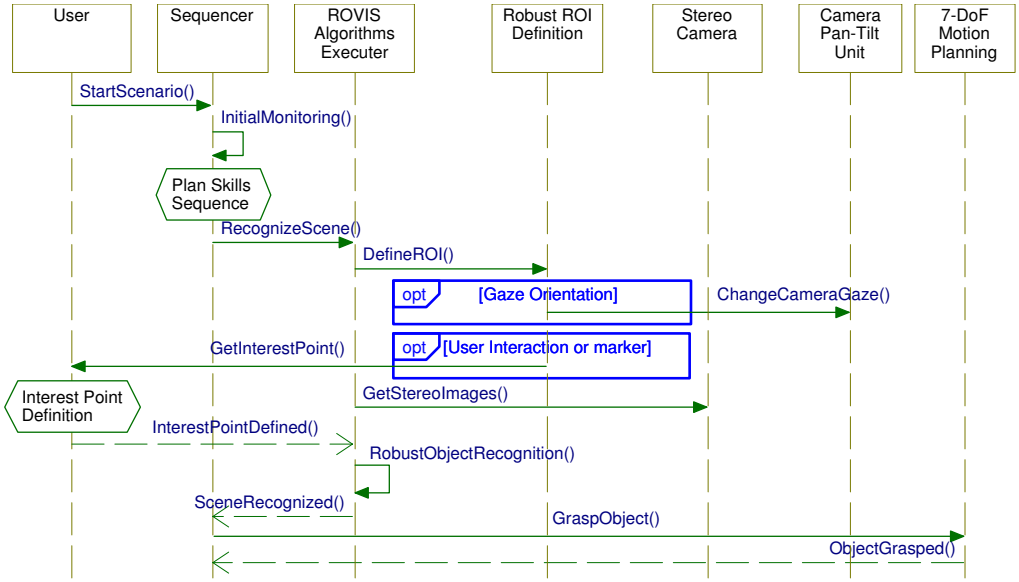

Fig. 8. Sequence diagram of the ROVIS operations involved in scene understanding.

data, like with the values *small* for the size and *green* and color of the bottle in this example.

Consequently, the pre-structured task knowledge, object classes and their characteristics allow to build universal operations in ROVIS as well as dynamic image processing chain construction according to a given scenario and context.

B. System functional analysis

Sequence diagrams are used to graphically represent the functional flow of information and the behavior of the system. Fig. 8 represents a sequence diagram of typical behavior of ROVIS. The process is scenario-driven and started by the user of the robot. After the user command is given, the control is taken by the Sequencer which plans the necessary sequence of actions needed to fulfill the requested scenario. For each scenario, an initial monitoring with object anchoring is executed first, followed by task execution. Both includes vision operations from the above listed categories. These operations, named *skills*, are called by the Sequencer. During skill execution, the image processing chain is constructed in order to recognize the objects in the scene for subsequent manipulator path planning and object grasping. As said before, the first step of the object recognition and 3D reconstruction chain from Fig. 4, is the definition of the image ROI. This is modeled in Fig. 8 by three complementary cases:

- *User interaction*,
- *Camera gaze orientation*,
- *Marker recognition* in an intelligent environment.

For the case of user interaction, the control of object recognition is given to the user of FRIEND who defines an interest point on the input left camera image through the HMI, as seen in Fig. 4. For the second and third case, *camera gaze orientation* and *marker recognition*, the control is further given to the 2-DoF pan-tilt head unit for changing the field of view of the stereo camera. The field of view change is sequenced by the calculation of the image ROI. After the image ROI is defined, the object recognition methods are applied and the 3D positions of the objects of interest are calculated and saved in the World Model. Finally the control of the process is given back to the Sequencer component which further calls the path planning and grasping algorithms for controlling the 7-DoF manipulator arm.

C. Architectural design

The third phase in the development of ROVIS is the *architectural design*. This phase represents the actual implementation of the physical and software elements.

1) ROVIS software architecture: The overall software structure of ROVIS and the interconnection with the external systems can be seen in Fig. 9. The central part of the framework is the Core module which contains the Algorithms Executer. The ROVIS *Skill Server* acts as an interface from the machine vision system to the external world. Through it the Sequencer calls the image processing operations made available by the skill server interface. This server is only one of the skill servers used by the Sequencer in task planning. From the computer vision point of view, the necessary hardware is represented by the stereo camera and the pan-tilt head unit materialized here by the two hardware server interfaces: ICameraHardwareServer and IPTHHardwareServer. The connection to the World Model is also made from the skill server.

2) Execution of vision algorithms: In Fig. 10 the components involved in the execution of a typical skill are presented. In the center of the diagram is the ROVIS skill server interface which binds all the components together. For a better understanding of the process, a skill example, which provides as result the 3D position of an object, is considered: AcquireObjectBySCam. The input arguments of this skill are the task-related object names which are used to extract object-related information as provided

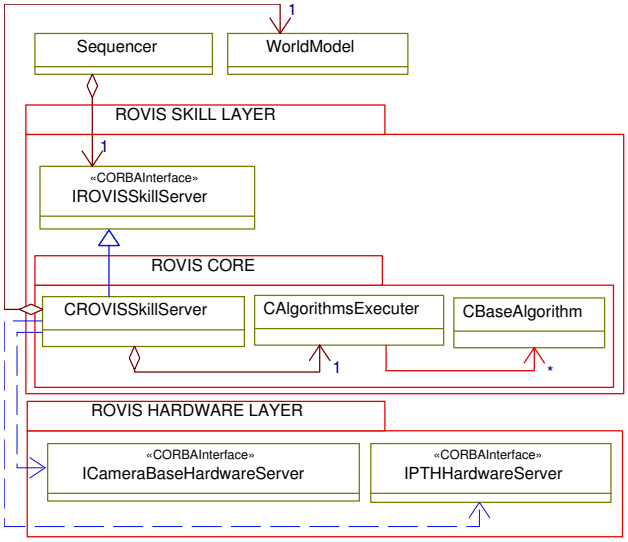

Fig. 9. ROVIS software architecture.

during initial monitoring or previous executions. Based on this information the dynamic generation of the image processing chain by the Algorithms Executer takes place.

This process is started by the Sequencer via skill call. First, the necessary hardware is actuated. For the case of the vision system, the stereo camera field of view is changed and the stereo images pair acquired. Within the skill, the Algorithms Executer combines vision algorithms with the help of their basic properties which reside in the algorithms base class. These are the properties Name, Description, Category, InputType and OutputType. The above properties are used when selecting appropriate algorithms with respect to the given object class, as well as its a priori known characteristics. In case of recognition of definite objects, a priori known characteristics are concrete data sets (e.g. color, shape descriptors, etc.) that are used to parameterize the vision algorithms.

The ROVIS methods are controlled via three functions called by the algorithms executer: run, stop and pause. These three functions modify the Status attribute of the algorithm. After the object of interest has been detected, its 3D position is saved in the World Model.

The overall structure of a skill, like AcquireObjectBySCam, can be seen in the flowchart from Fig. 11. The start and end information messages of the skill are written in a log file for later debugging. Also, at the beginning, an extra process is started to check the incoming commands to the skill (e.g. stop or pause). This process is terminated at the end of the structure. The skill can be executed in two ways:

- *Normal execution*, which includes the actuation of the vision hardware and the construction of the image processing chain,
- *Simulative execution*, used by the Sequencer to test task planning capabilities.

Just before the end of the skill, the encountered exceptions are properly handled.

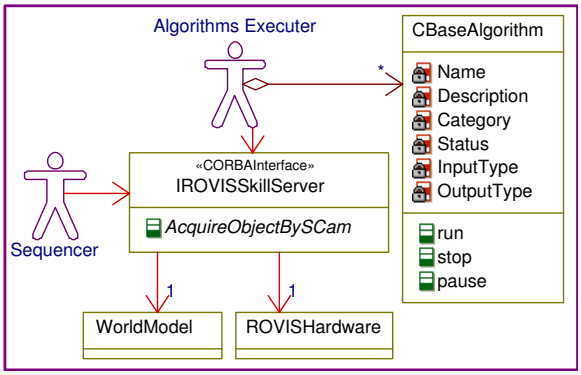

Fig. 10. Architectural design of a skill in ROVIS.

IV. Hardware Implementation Aspects

The computing system, on which ROVIS has been implemented, is represented by a standard PC computer with 8GB of RAM and two Intel XEON® QuadCores microprocessors, each working at a speed of 2.33GHz. The high computing power has been chosen so in order for the system to cope with the large amount of information data that has to be processed, especially from the machine vision system and motion planning algorithms of the manipulator. The computer is mounted at the backside of the wheelchair, behind the user, as can be seen in Fig. 2.

The main sensor component of the robotic platform is the global vision module, represented by a Bumblebee® 2 Stereo Camera system used for environment understanding and 3D reconstruction. The camera is equipped with two 1/3“ Sony® progressive scan CCD ICX204 sensors that provide two synchronized 1024x768px RGB color images at a maximum framerate of 20 *Frames Per Second* (FPS) and a 4.65μm square pixels. The imaging sensors have a focal length of 6mm with 43° *Horizontal Field Of View* (HFOV) and a distance of 120mm between the two lenses. Also, the Bumblebee® 2 camera is pre-calibrated against distortions and misalignment. The conversion from the analog image signal to digital images is done through a 12-bit ADC (*Analog to Digital Converter*). Serial communication between the camera and the computing system is implemented using a 6-pin IEEE-1394a FireWire interface for camera control and data transmission. Various parameters of the stereo camera (e.g. exposure, white balance, shutter speed, etc.) can be set either to automatic or manual adjustment.

An important actuator used by ROVIS is the gaze orientation module of the stereo camera. This module is composed of a Schunk® 2-DoF *Power Cube* servo-electric pan-tilt head unit. The covered field of view of the pan-tilt is 1180° and 180° in the pan and tilt directions, respectively. For positioning and velocity control it uses two incremental encoders. The resolution used for the encoders of the motors has a value of 4 [Arcsec/Inc] for the pan and 5 [Arcsec/Inc] for the tilt angle, thus making it precise enough for the vision system. Another representative characteristic of the unit is its angular velocity, which has a maximum value of 248 [°/sec] for pan and 356 [°/sec] for the tilt. The communication

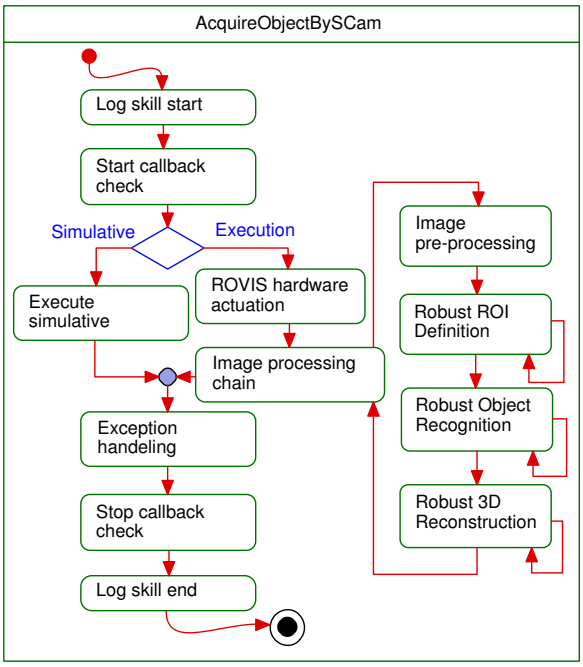

Fig. 11. The basic ROVIS skill structure.

between the positioning module and the main computing device is performed via a CAN bus interface.

V. Conclusions and Outlook

In this paper the model driven development of the ROVIS machine vision architecture for service robotics has been presented. The design and implementation of the architecture and the organization of the data flow was performed using the UML modeling language. Also, the concept of scenario-driven information was used in the automatic generation of the object recognition and 3D reconstruction chain required for scene understanding in the service robotic system FRIEND. Due to the flexibility of its design, the proposed vision system can be easily transfered to different robotic systems.

The efficiency of FRIEND and ROVIS has been demonstrated in real world application scenarios. Within the research project AMaRob, the functionalities of the proposed systems have been at display at different fairs and conferences, as, for example, ICORR 2007, CeBit 2008 and RehaCARE 2009.

References

[1] D. Kragic and H. I. Christensen, "Advances in robot vision," *Robotics and Autonomous Systems*, vol. 52, pp. 1–3, 2005.

[2] C. Martens, O. Prenzel, and A. Graeser, "The rehabilitation robots FRIEND-I and II: Daily life independency through semi-autonomous task-execution," *Rehabilitation Robotics*, 2007.

[3] S. M. Grigorescu, D. Ristic-Durrant, and A. Graeser, "ROVIS: Robust machine vision for service robotic system FRIEND," in *Proc. of the 2009 Int. Conf. on Intelligent RObots and Systems*, (St. Louis, USA), Oct. 2009.

[4] S. M. Grigorescu, D. Ristic-Durrant, S. K. Vuppala, and A. Graeser, "Closed-loop control in image processing for improvement of object recognition," in *Proc. of the 17th IFAC World Congress*, (Seoul, Korea), July 2008.

[5] D. Ristic, *Feedback Structures in Image Processing*. PhD thesis, Bremen University, Institute of Automation, Bremen, Germany, Apr. 2007.

[6] D. J. Kriegman, G. D. Hager, and S. A. Morse, *The Confluence of Vision and Control*. Springer Verlag London, 1998.

[7] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, Oct. 1996.

[8] T. Heyer, S. M. Grigorescu, and A. Graeser, "Camera calibration for reliable object manipulation in care-providing system FRIEND," in *Proc. of the 2010 Int. Conf. ISR/ROBOTIK (to be published)*, (Munich, Germany), June 2010.

[9] D. W. Paulus and J. Hornegger, *Applied Pattern Recognition: A Practical Introduction to Image and Speech Processing in C++*. Braunschweig / Wiesbaden: Friedr. Vieweg und Sohn Verlagsgesellschaft mbH, 3 ed., 2001.

[10] D. W. Paulus, J. Hornegger, and H. Niemann, "Software engineering for image processing and analysis," 1999.

[11] D. W. Paulus and H. Niemann, "Object-oriented programming for image analysis," 1996.

[12] M. Beetz, T. Arbuckle, A. B. Cremers, and M. Mann, "Transparent, flexible, and resource-adaptive image processing for autonomous service robots," in *Proc. of the 13th European Conf. on Artificial Intelligence ECAI 98*, pp. 632–636, John Wiley and Sons Ltd., 1998.

[13] D. Westhoff and J. Zhang, "A unified robotic software architecture for service robotics and networks of smart sensors," 2007.

[14] S. Persa and P. Jonker, "Real time image processing architecture for robot vision," in *SPIE proceedings series*, (Bellingham WA, USA), Society of Photo-Optical Instrumentation Engineers, 2000.

[15] H. Utz, U. Kaufmann, G. Mayer, and G. K. Kraetzschmar, "Vip - a framework-based approach to robot vision," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 67–72, 2006.

[16] J. Sitte and P. Winzer, "Methodic design of robot vision systems," in *Proc. of the Int. Conf. on Mechatronics and Automation*, pp. 1758–1763, 2007.

[17] W. Ponweiser, M. Vincze, and M. Zillich, "A software framework to integrate vision and reasoning components for cognitive vision systems," *Robotics and Autonomous Systems*, vol. 52, pp. 101–114, July 2005.

[18] D. Kragic, Bjoerman, H. Christensen, and J.-O. Eklundh, "Vision for robotic object manipulation in domestic settings," *Robotics and Autonomous Systems*, vol. 52, pp. 85–100, 2005.

[19] H.-P. Hoffmann, "SysML-based systems engineering using a model-driven development approach," *White Paper, Telelogic*, 2008.

[20] R. Miller, "Practical UML: A hands-on introduction for developers," *White Paper, Borland Developer Network*, Apr. 2003.

[21] A. Chella, C. Massimo, I. Infantino, and R. Pirrone, "An agent based design process for cognitive architectures in robotics,"

[22] I. Infantino, C. Massimo, and A. Chella, "An agent based multi-level architecture for robotics vision systems,"

[23] D. Ojdanic and A. Graeser, "Improving the trajectory quality of a 7 DoF manipulator," in *Proc. of the Robotik Conf.*, (Munich, Germany), 2008.

[24] O. Prenzel, C. Martens, M. Cyriacks, C. Wang, and A. Graeser, "System controlled user interaction within the service robotic control architecture MASSiVE," *Robotica, Special Issue*, vol. 25, Mar. 2007.

[25] T. Lueth, D. Ojdanic, O. Friman, O. Prenzel, and A. Graeser, "Low level control in a semi-autonomous rehabilitation robotic system via a brain-computer interface," in *Proc. of the IEEE 10th Int. Conf. on Rehabilitation Robotics ICORR 2007*, (Noordwijk, Netherlands), June 2007.

[26] "http://www.omg.org/, Unified Modeling Language specification, version 1.5," *OMG document formal*, 2003.

[27] O. Prenzel, *Process Model for the Development of Semi-Autonomous Service Robots*. PhD thesis, Bremen University, Institute of Automation, Bremen, Germany, 2009.

[28] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. New Jersey: Prentice-Hall, 2007.

[29] O. Prenzel, "Semi-autonomous object anchoring for service-robots," in *Methods and Applications in Automation*, pp. 57–68, Shaker Verlag, GmbH, 2005.