1

Model-Driven Development of Intelligent Mobile Robot Using Systems Modeling Language (SysML)

Mohd Azizi Abdul Rahman[1], Katsuhiro Mayama[2], Takahiro Takasu[2], Akira Yasuda[2] and Makoto Mizukawa[2]
[1]*University Technology of Malaysia*[1]
[2]*Human-Robot-Interaction Lab, Shibaura Institute of Technology*
[1]*Malaysia*
[2]*Japan*

1. Introduction

In Japan alone the number of disabled and elderly people are growing rapidly and they usually require personal transportation vehicles such as cars and wheelchairs. However, people currently have limited capabilities or difficulties to operate these vehicles in such complex environments like shopping malls or tourist-attraction centers.

In recent years, a great number of studies of mobile robot applications have been proposed. However, only few of these studies have realized the model-based design approach in their entire development process. NEDO's Intelligent Robot Technology (RT) Software project (Okano et al.,2009) started to promote the robot technology as the basic knowledge and technology to solve various problems in daily life. In this context, an intelligent mobile robot has been developed for providing mobility to the elderly and physically unfortunate people. Besides that, the National Institute of Advanced Industrial Science and technology, also known as AIST has developed the RT-Middleware (RTM) in order to achieve efficiency in developing robot software components (Ando et al., 2005).

The main idea of this research is to develop robot software modules by looking from the system engineering analysis point of views. This is because most robotic systems are complex embedded systems. System engineering approaches focus on designing and constructing the complete system, and also on providing model reuse capabilities. Moreover, these approaches can enhance communications among the development teams, specification and design qualities and reuse of system specification and design artifacts. Modules' reusability is our main concern in this paper.

Past development efforts of robot software using Model Driven Architecture (OMG MDA, 2003) Model Driven Architecture® (OMG MDA, 2003) approach seem insufficient to support the demand of current industrial-to-domestic robot transitions. Developing intelligent robots in large scales is very demanding for experiment purposes. Thus, almost all robot systems have some common functions. However, much usable design information went to waste because of a serious lack of sharing and reusability. This motivates us to explore the

use of the MDA for the design and development of robot software modules in order to achieve module reusability.
This paper outlines the need for the MDA approach in developing reusable robot software modules that are expressed as models. Additionally, this approach is employed to encourage the use of model-driven engineering methodology (MDE, 2008), as it is less attractive to robotic software developers (Bruyninckx, 2008). More detail about MDA is presented in section 2.
The key issue is robot software module reusability and this paper concerns about the design process of robot software modules by using a model-based approach. This is shown by how existing RT-Components (RTCs) (Ando et al., 2008) can be mapped to the SysML models (see Figure 1).

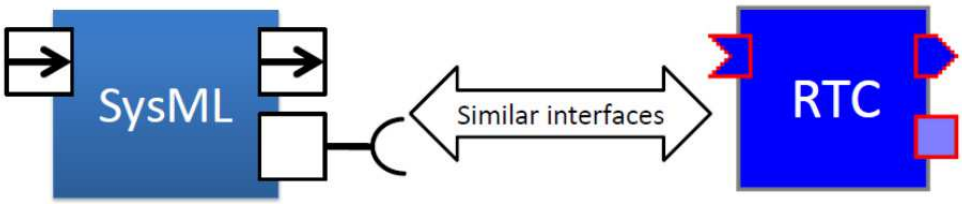


Fig. 1. SysML module to RTC block mapping concept

The scope of this study is to develop reusable intelligent RT modules that can be used to provide safe and convenient transportation service for disabled people in order to promote a barrier-free society. A mobile robot platform is currently developed for real implementation purpose at Shibaura Institute of Technology, Japan. In this study, we only concentrate on the initial phase of designing the software modules and the a mapping strategy of the SysML module into some existing RTCs (i.e. samples taken from other robotics project).

2. Model driven architecture, its need, and systems modeling language

In order to achieve the purpose of this study, we adopt an MDA approach to using models in robot software development. It prescribes certain kinds of models to be used, how those models are prepared, and the association of the relationships of the different kinds of models.

2.1 MDA approach

MDA (OMG MDA, 2003) is a software design approach for the development of software systems that describes the content of models by developing a language-neutral and separate from the implementation design software. MDA is a method known in model-driven engineering and the purpose of MDA is to change design information from implemented software into independent software by describing the coded content of a specific language-independent model. This allows the design information to be developed independently, making it possible to be changed without the use of implementing software. In MDA, the abstract model of information is called platform independent models(PIMs) and the model that corresponds to a specific programming language is called platform specific models (PSMs).

PIM describes higher abstraction level of a system design than a program does but does not describe anything regarding to the platform. Its content falls under the behavior and the internal structure of the system. The application information for the platform added into the PIM is called PSMs. Inside MDA, the PIM is converted into a coded programming language (PSM) using a tool. With the abstract representation model, the design information itself will become reusable and the production of the program that was reflected immediately from design information due to the semi-auto code generation is also improving.
The use of MDA allows the construction of design information that is no longer depending on the specified development techniques and technologies. With the design depending on which platform is being clearly specified, not just abstract design information, specific system information from a voluntary platform can also be reusable. Figure 2 shows MDA basic process and its usage in developing complex systems.

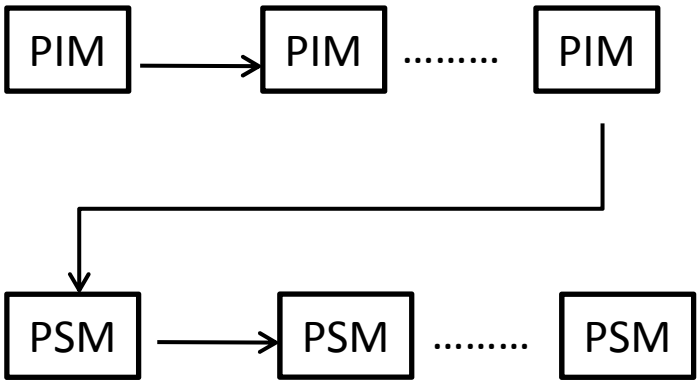


Fig. 2. MDA basic process of complex systems (OMG MDA, 2003)

2.2 The need of MDA in robotic software development

MDA is considered very useful in robotic software development. The OMGRobotics-Domain Task Force (OMG Robotics DTF, 2005) that aims to promote and integrate the OMG robot standards, has laid out one example of why MDA which was designed for robotic systems should be introduced and extended throughout the robotic field.
The MDA is considered very effective in handling functional decomposition problems in the intelligence module. Functional decomposition problems are problems that occur during the division of a robotic system due to different development teams having different ways of splitting functionality into components, making it difficult to use the independent developed modules mutually.
In order to solve this problem, it is important to standardize the division unit that will suit to robot functions. If intelligence modules are functionally decomposed with a mutual cooperation, not only that different module can be easily applied together, but also the exchangeability and continuity of the modules can be achieved. This can be compared to the benefit of implementing service-oriented architecture (SOA) that highly depends on the partition in services that is not a trivial problem. As we are concerned about reusability issues, about reusability issue, both MDA and SOA provide a concrete solution for that, but how we split functionality into components make an MDA approach is more preferable.

In this research, the focus is the output of the module and if the output is the same as the input, the implementation method used is considered correct. Therefore the major thing that has to be considered during this stage is how individual modules are divided. This is no other than the PIM in the MDA. By promoting an intelligence module with the same functional decomposition as well as combining different intelligence module with the same purpose will make it possible to realize a variety of application systems. If we apply MDA to the robotic planning stage and construct the PIM with sufficiently repeated discussion on the functional division, we will be able to develop a much proper robotic module development.

2.3 SysML

SysML is a general-purpose modeling language that is only associated with descriptive semantics. It was developed to support the specification, analysis, design, verification and validation of a wide range of complex systems (OMG SysML, 2010). The <<block>> is the basic unit of structure in SysML and can be used to represent the systems that may include hardware, software, information, processes, personnel, and facilities. The objective of SysML is to unify the diverse modeling languages currently used by system engineers. SysML reuses a subset of Unified Modeling Language (OMG UML, 2010) and provides additional extensions needed to address systems engineering aspects not covered by UML2. It includes nine diagrams (see Figure 3) that can be used to specify system requirements, behavior, structure and parametric relationships. Requirements diagram and parametric diagrams are the new diagram types proposed by SysML.

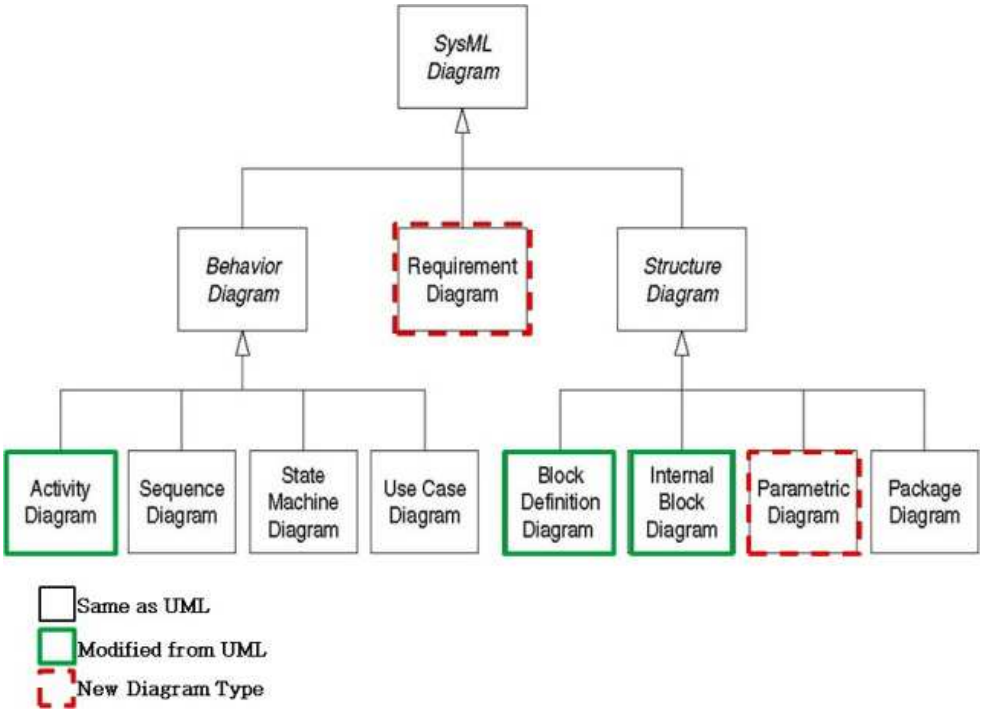


Fig. 3. SysML diagram taxonomy and its comparison with UML (OMG SysML, 2010)

SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements. The requirement diagram can be used to represent the relationships that exist between requirements and to visualize them. It provides a bridge between traditional requirements management tools and other SysML models. It can be used to depict the requirements in graphical, tabular, or tree structure format and highlight their relationships between requirements and other model elements that satisfy or verify them.

3. Modeling approach

3.1 Overview

In this paper, the progress of our model-based approach is presented by adopting the international standard modeling language SysML throughout our design process. All diagrams are made of SysML constructs.We study two problems for improving module reusability such as intelligent module division and system development based on the intelligent modules.

3.2 Purpose of this study

This paper describes our effort to adopt model-based approach to make robot models independent from any specific hardware and software platform by deriving reusable and versatile robot model. Thus, other developers will be able to refer to our designated models and customize it to meet their robot specifications. For making effortless transition from models to real system, we employ existing RT-Components developed by NEDO's project to reduce our burden on the development, as well as extending the reusability of RT modules. We have the following purposes to:

- make the model for each process to develop reusable models

In this research, we adopt model-based approach to each design process from intelligent mobile robot basic functional analysis to final system analysis, depending on the implementation. This approach enables us to choose appropriate design decision expressed as models. For example, if the requirement is same, developers will reuse our requirement diagram. Otherwise, only design workflow should be reused.

- design PIM level and PSM level separately

Modeling process consists of platform independent model design and platform specified model design as previously described in Section 2.

- improve PSM reusability

We make use the RT-Middleware in our development as a software platform to improve software module and system reusability. Mappings between abstract blocks and RTC blocks were achieved.

3.3 Modeling process

Modeling activity conducted in this study is shown in Figure 4. The red box means "movement intelligence basic design" while blue box means "movement intelligence system design", and green box means "internal software system design".The paper-like diagrams are shown as the output of each analysis regarding to SysML diagrams. For brevity, we disregard some explanations about the design steps throughout the paper. However, the reader is advised to contact the authors for the full specifications. The following contents explain each of the steps:

1. Context analysis; this analysis reveals the supposed environment and is used for deriving requirement analysis and functional analysis. As a result, a context diagram is obtained in this analysis.
2. Requirement analysis; the functional and non-functional requirements based on the supposed environment and limitations are organized. In the process we make the requirements as well as making the connection between related requirements; and the solution for each requirement is shown.
3. Use case analysis; the required functions that based on the functional requirement are described. The use case diagram is used in this phase.
4. Hardware structure analysis enables us to organize abstracted hardware connection by using Block Definition Diagrams.
5. Software functional analysis; shows software including its relations. Each software block is abstracted from the necessary functions.
6. Software structure analysis defines the interface of each module using Internal Block Diagrams (IBDs).
7. Software specification analysis shows how its internal behavior is defined.
8. Behavioral analysis; Software system behaviors are designed using Sequence Diagram (SD) and State Machine (STM).
9. RTCs selection; RTC blocks are chosen to be mapped from the designed software modules.
10. Implementation of the system; Final step to test and evaluate some parts of the designed modules implemented on a real robot platform (not covered in this paper).

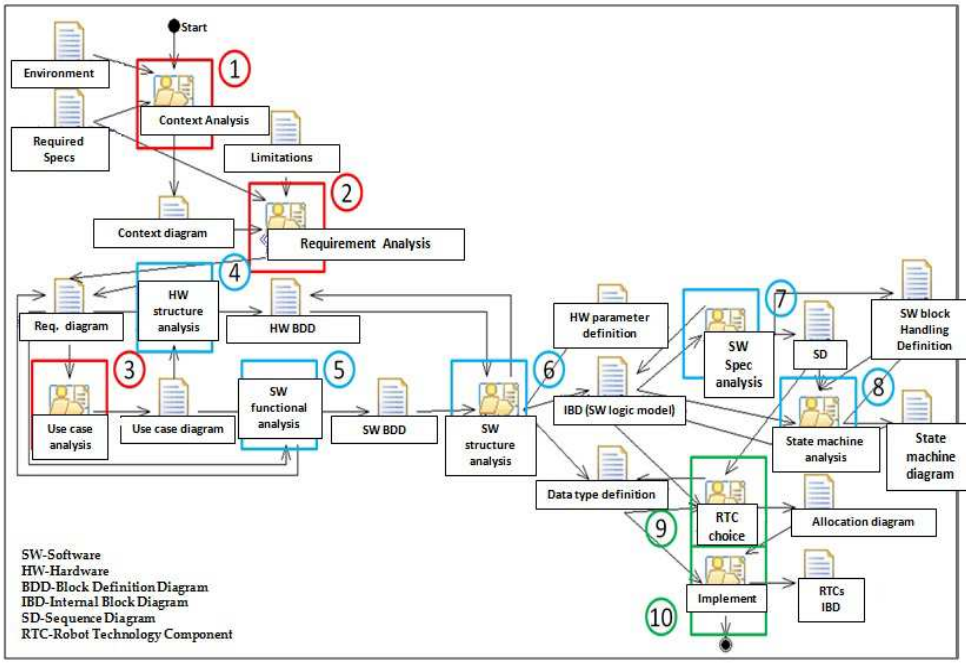


Fig. 4. Overall design process and modelling workflows

3.3.1 Context analysis

The PIM level modeling activity starts by abstracting information about the robot system in environments. Our mobile robot operates in an outdoor surrounding with pedestrians, bicycles, cars and various road conditions such as on the grass, pavement, slopes or ramps, and so on. The robot must be able to avoid the obstacles and make a correct path in various situations. Figure 5 shows an example of the context diagram that includes expecting objects in the surroundings and the disturbance elements in our mobile robot operating environments. Disturbance elements (e.g. light and building) are considered as the possible distraction sources that may affect the robot's sensing ability. The passenger is identified to be a user who interacts with our robot system. This categorization of object and environment is for deriving functional robot requirements.

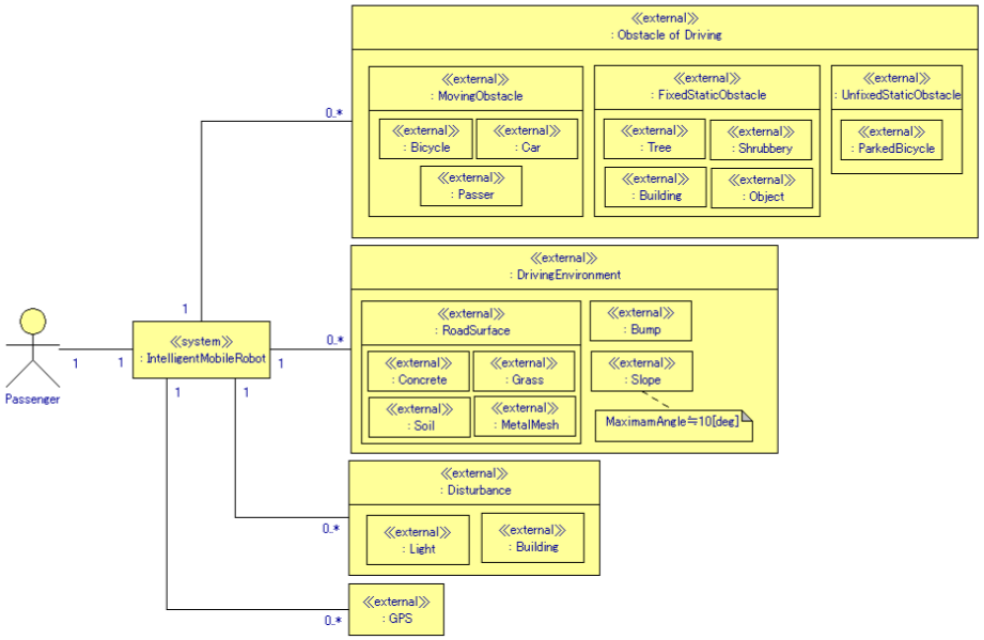


Fig. 5. Context diagram for operating environment analysis

3.3.2 Functional requirement analysis

A requirement diagram is a new diagram type in SysML that can be used to model the system requirements in more detail. Our mobile robot system requirements are described in the following SysML's requirement diagrams. They focus on the requirement mobility, the safety for operating in the environment and the extension of RT modules reusability.Figure 6 shows the basic requirements derived from the previous operating environment analysis. The top-level requirements are identified as "SafetyLocomotion" and "ExtendingRT-

ModuleReusability". The "SafetyLocomotion" is composed of other sub-requirements with respect to the locomotion strategies which are called "Controlled by Passenger", "AutonomousLocomotion", and "Enhanced Safety". Figure 7 illustrates the derived requirements and functions of robot locomotion for manual control mode (i.e. a joy-stick controller may be used as an input device to the system) and for autonomous locomotion mode. These requirements represent abstract levels of functional requirements to the concrete description with the specific devices or software modules. The autonomous locomotion is further derived into several specific components. The '<<deriveReqt>>' stereotype is associated for specifying a desired destination, planning a path, tracing trajectory, self-localization, and obstacle avoidance functionalities. In this paper, we omit the discussion on the "EnhancedSafety" requirement as a future work possibility by considering safety standardization imposed in Japan.

In Figure 7, the red rectangles represent the actual hardware of the robot system with the <<satisfy>> stereotypes associate to its upper layer blocks meaning that the chosen hardware should satisfy the designed requirements. As we can see, they are arranged somewhat hierarchically, with low-level derived requirements closer to the hardware. Constructing the system in a hierarchical manner like this helps us to maintain good design pattern. It also helps with traceability because we will have a clear understanding of the dependencies between requirements. These abstraction layers allow for easier model re-use and exchange.

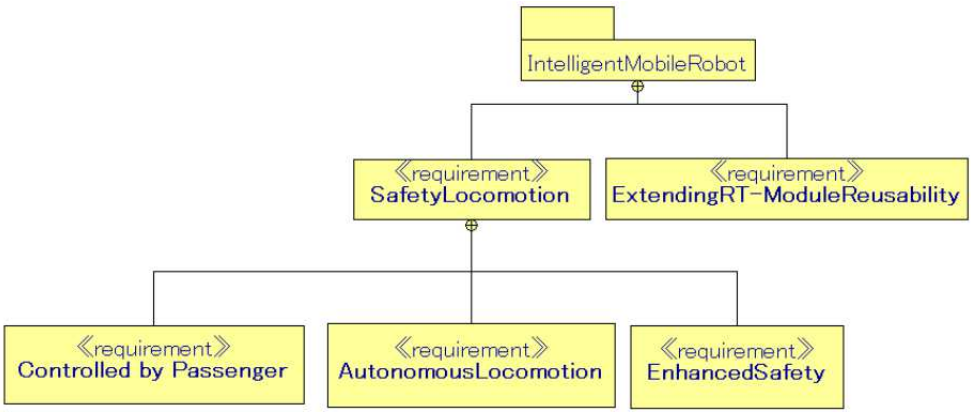


Fig. 6. Top-level requirements derived from operating environment analysis

For necessary functional analysis, we identified the following core functions for our mobile robot system to perform

- Specifying a destination point "InputGoal" function.
- Navigating to the specified destination by using path planning, path generating, trajectory generating, obstacles detecting, position localizing and error detecting modules.

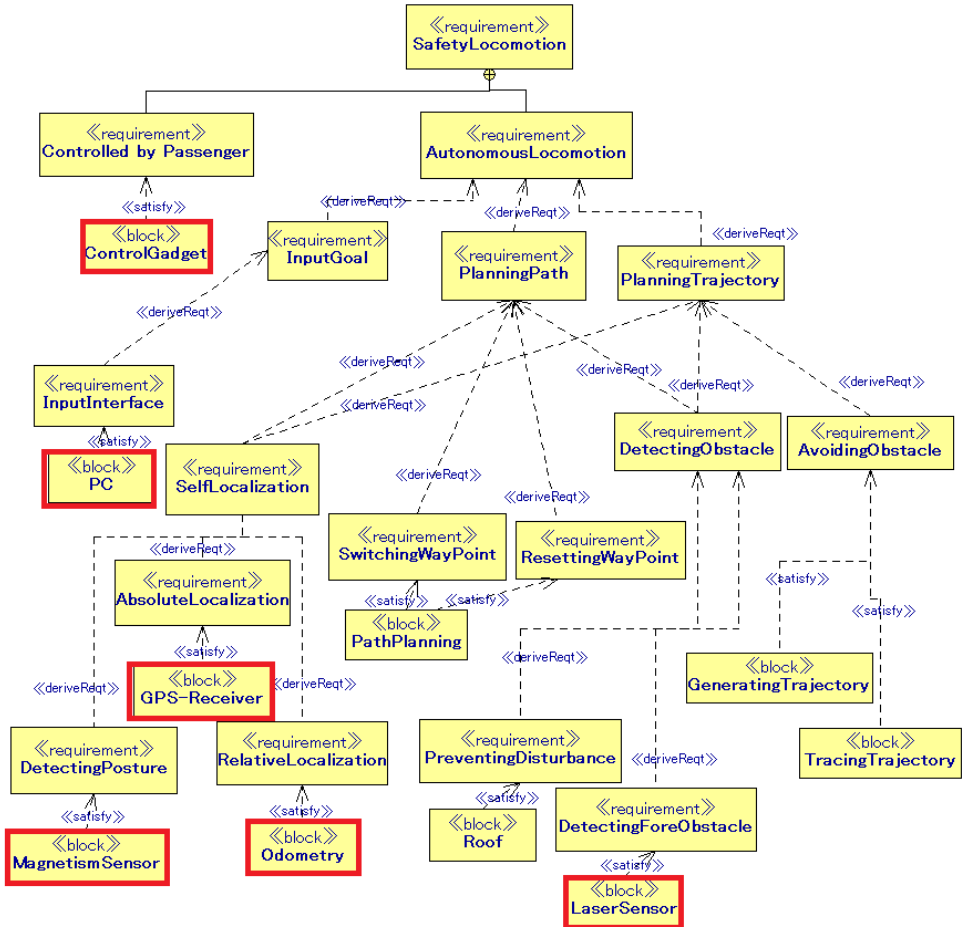


Fig. 7. Functional requirements and its derivation of "SafetyLocomotion" block.

3.3.3 Software and hardware composition

This section describes how we model the required software and hardware components, and decide the internal interfaces between these components. By implementing the common interfaces that were proposed by the working group of the NEDO project, our robot modules can easily adapt and apply the existing RTCs.

In addition, our designed models separate device specific components from components for the transporting functions. As a consequence, our models are truly independent from the specific devices especially for the wheeled platform, and easy to be reused in future. Figure 8 shows robot software composition diagram while Figure 9 shows its hardware profile's composition. The former diagram is made of SysML's internal block diagram (IBD) and the latter is made up of SysML block definition diagram (BDD). In Figure 8, we structure the software components as identified previously during the necessary functions analysis by connecting each software component through interface ports provided in SysML. To make

the system complete, hardware configuration is shown in Figure 9 by profiling each hardware implementation using various sensors for robot localization, obstacle avoidance, battery malfunction, emergency switches and so on. Figure 10 displays our mobile robot platform, the Four-X developed at Shibaura Institute of Technology, Japan. This type of mobile platform is equipped with swivel-steered drive system. The following list specifies some parts of its hardware components:

1. Hemisphere A100 Global Positioning System (GPS) receiver.
2. On board PC (Panasonic CF-19).
3. Emergency Switch Button.
4. On board PC for data logging.
5. Notification lamp.
6. Bumper switch.
7. Motor controller (EPOS 70/10).
8. On board power supply (intelligent battery).
9. Laser range finder (UTM-30LX).

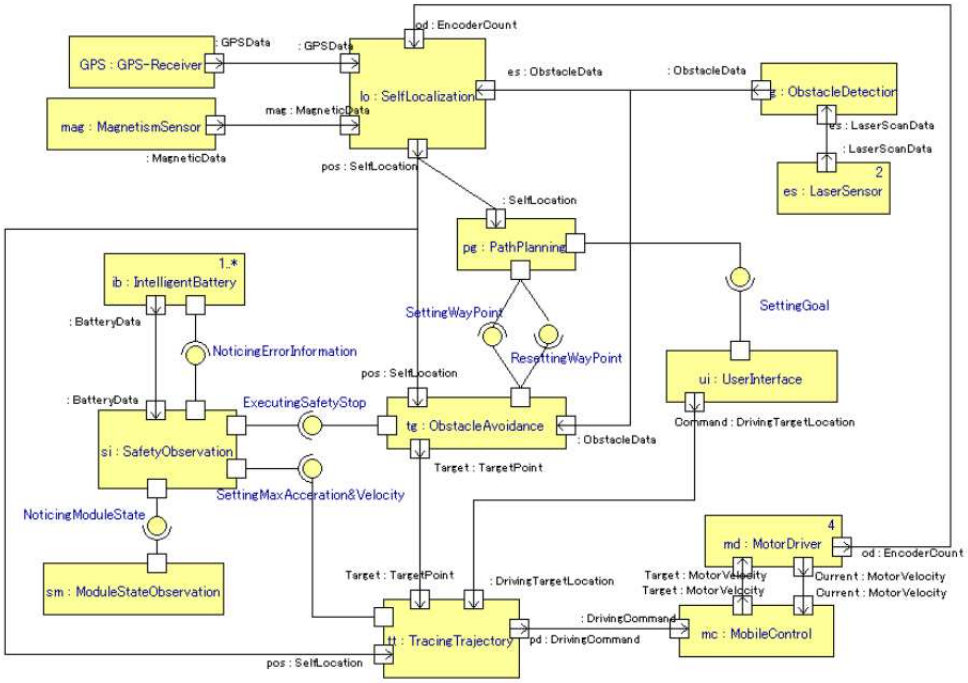


Fig. 8. Software modules composition and ports configuration modelled with the internal block diagram (IBD).

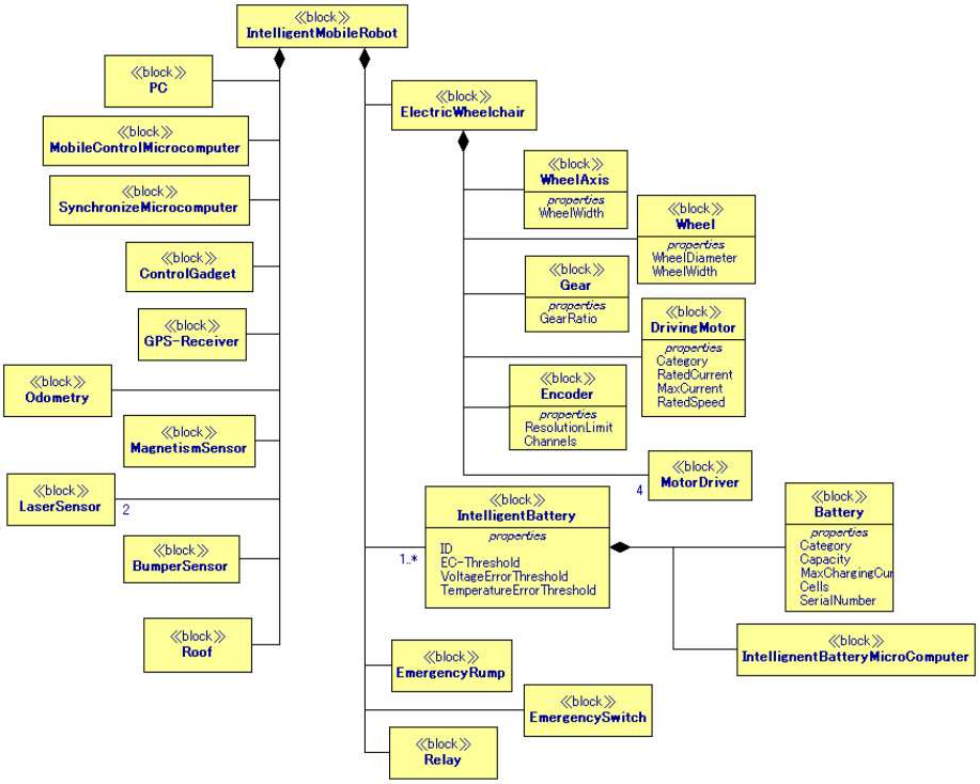


Fig. 9. Hardware modules composition using a block definition diagram (BDD).

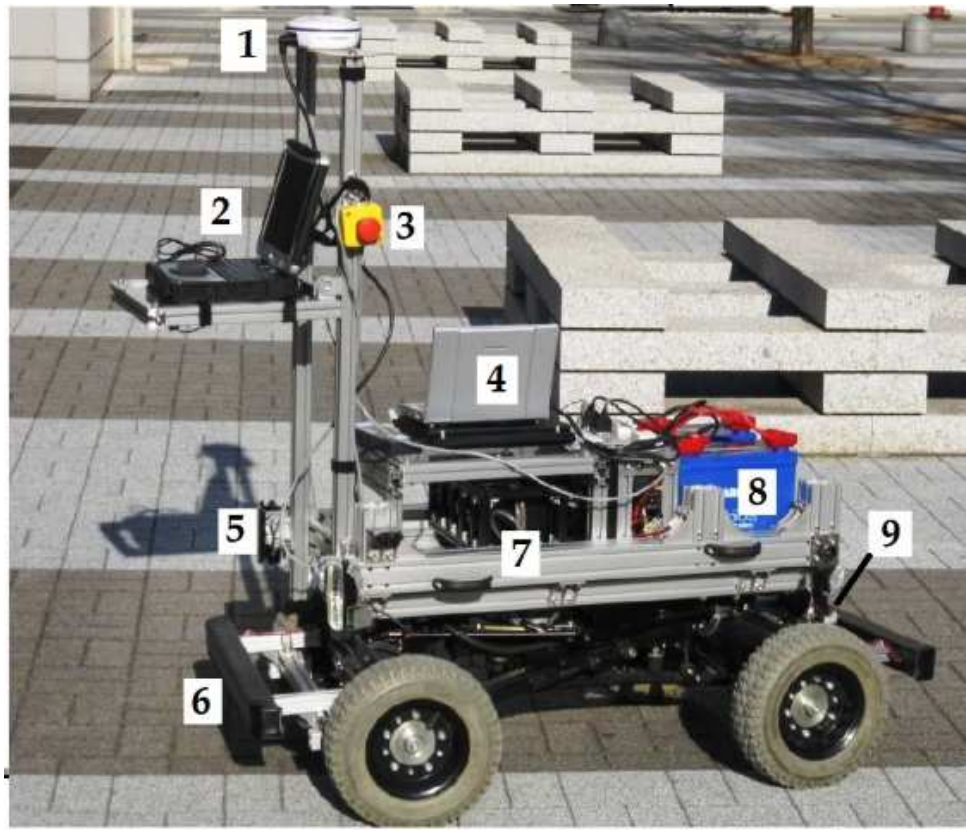


Fig. 10. The Four-X mobile robot platform

3.3.4 Mapping to the RT-components

The PSM modeling activity starts from here. We adopt a sample from Tohoku University's RT-Cs robot project to be reused in our software development. They have developed a versatile R-TC navigation system implemented on the Segway RMP 200 platform to complete the whole course in the Real World Robot Challenge 2009(Segway, 2009). Refer (Yuta et al., 2010) for information about the Tsukuba Challenge event. So, it is proven that their robot has a robust navigation system for using in outdoor environments. Outdoor navigation capability is one of the reasons why we adopt these RTCs beside its algorithm similarity. In addition, these RTCs also adopt a Global Positioning System (GPS) map, self-localization by using GPS and odometry, and obstacle avoidance by laser range finder (LRF). These similarities make our robot software development efforts slightly reduced because the concept of reusability is applied although with a different mobile robot platform.

In this phase, software modules are replaced with compatible RTCs blocks as we propose to select the suitable RTCs for the functionality of each SysML modules. As depicted in Figure 11, "SelfLocalization" module consists of four RTCs compatible modules,

which are “CorrectorGPS”, “CorrectorURG”, “CurrentVelocity”, and “PredictorOdometry”. By making a wide variety of software elements, we can have more choices of compatible RTCs.

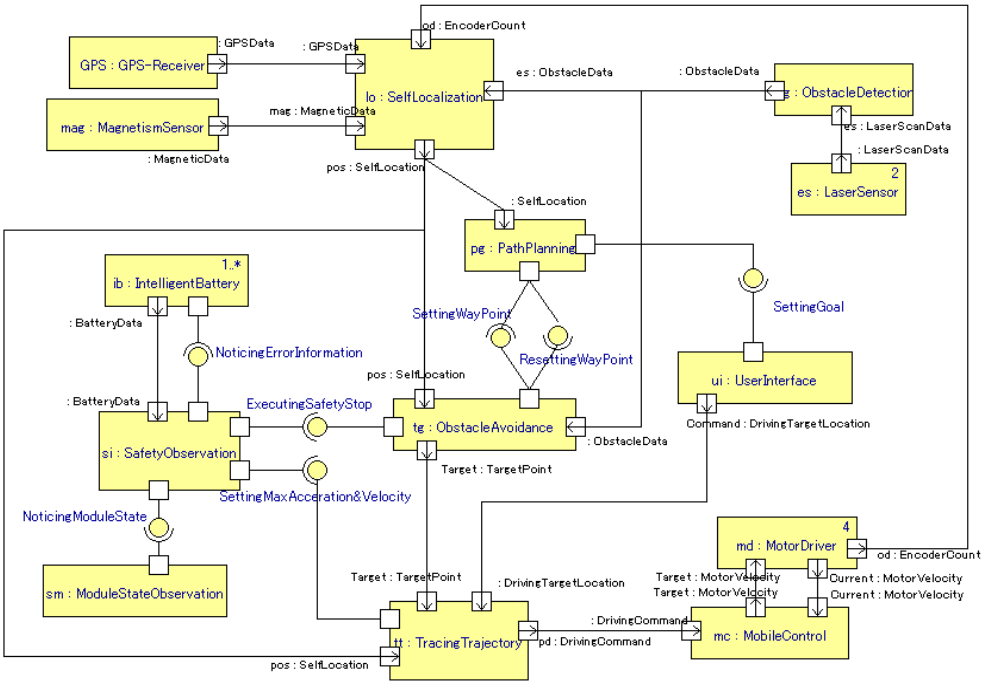


Fig. 11. Software modules (IBD) mapping technique

Thus, we set the suitable RTCs for functionality of software elements. To make a better understanding about this mapping technique, we provide a clearer mapping diagram by using <<allocation>> stereotypes (OMG SysML, 2010). Allocations are a set of notations (i.e. the new generic allocation dependencies) that allow different aspect of a SysML model to be related to each other. This is one of the SysML constructs to extend the UML capabilities. It is worth to notice that our SysML and RT-Middleware software platforms are based on C++ language.

As shown in Figure 12, the red box represents software element blocks (i.e. previously clarified in the analysis), and orange box determines real RTC system blocks. Therefore, these blocks have clear functionalities when each software block is allocated to other RTC system blocks with different specification or parameters. As a result, we model the full

specification of software modules that have been replaced with RTCs modules and its connection. An example is given in Figure 13. This mapping technique has reduced our development effort as well as promoting module reusability.

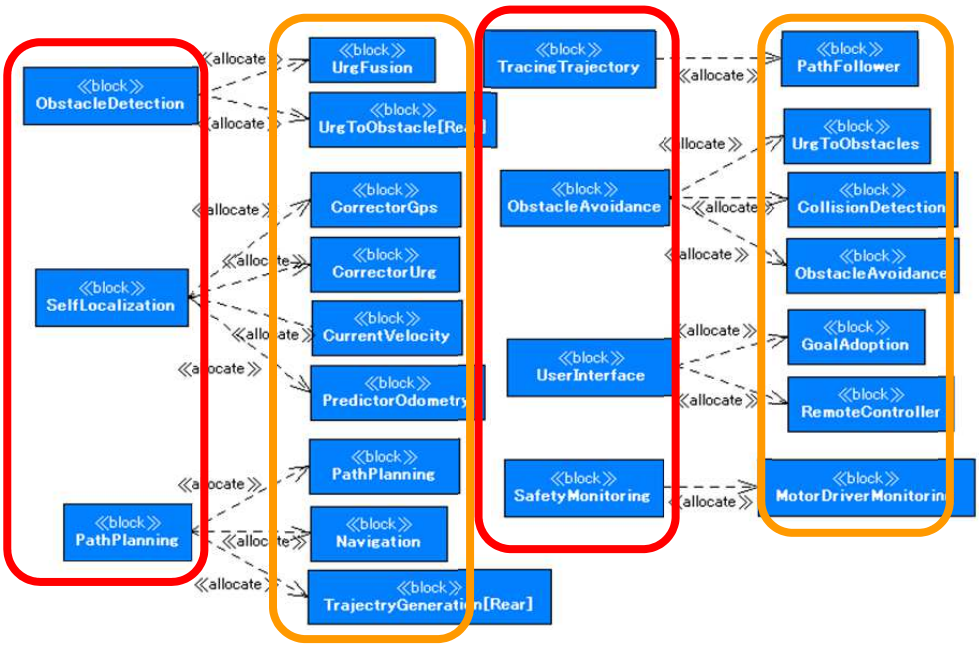


Fig. 12. Allocation diagram for mapping software component to compatible RTCs blocks.

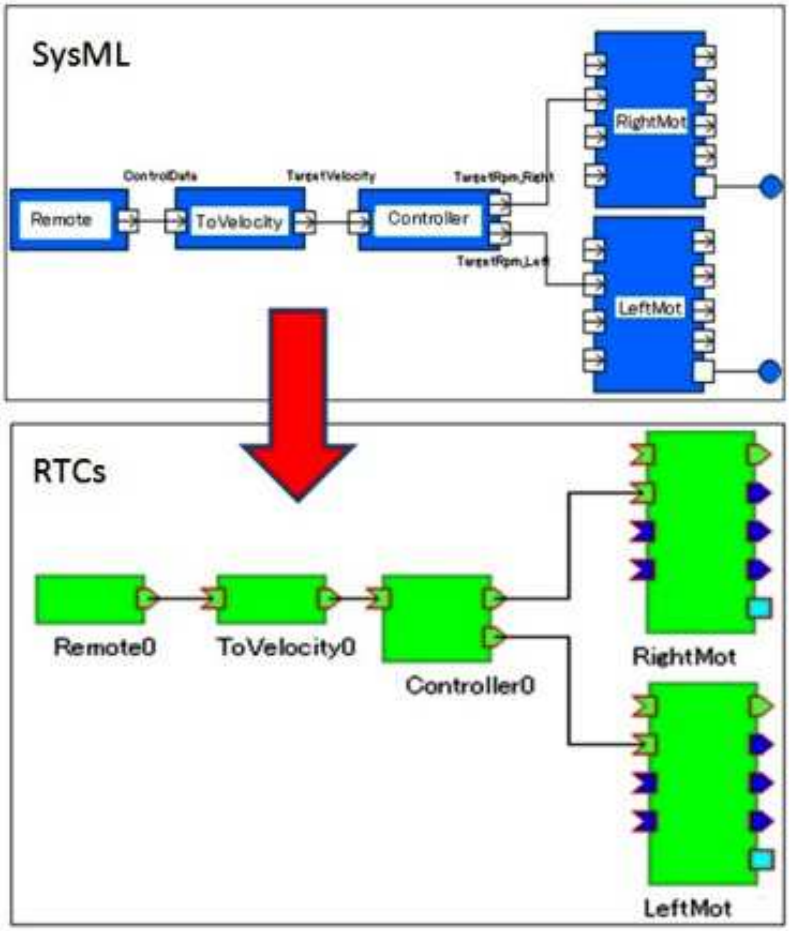


Fig. 13. The SysML to RT-Cs mapping.

4. Lessons learned

This section summarizes the lesson learned from the study where we successfully applied the model-based approach to developing mobile robot software modules for reusability purposes. As a result, we provide a modeling process to develop those modules.

4.1 The usage of SysML constructs for robotics

Through the study, we found that the OMG SysML standard was fit as a notation for specifying the requirements, modeling the system views (e.g. both structural and behavioral), decomposing into objects and/or blocks, and defining relationships between objects especially in a complex embedded system like a robotics system. Some diagrams and notations provided in the SysML were particularly important for abstracting information, analyzing the structure and behavior, designing the functionality, as well as

modeling robot systems. In the case of context and requirement diagrams, functions (e.g. functional requirements) which robot system should perform can be identified and defined in terms of actors who are users of the robot system and use cases. Context and requirement diagrams define the information of some aspect of the robot system without knowing its internal structure.
Additionally, block definition diagram (BDD) notation is used to compose the structure model, which focuses on the static structure of a robot system (i.e. both software and hardware). The BDD also shows the blocks of objects in the system, their internal structure that include attributes, operations, parts, and their relationships to other blocks in terms of associations and generalization.
For internal block diagram (IBD), as its name, is used to show each part of the blocks that participate in software or hardware modules interface with each other by defining ports and flows in order to send and/or receive signals (see Fig.9 and Fig.11). Although SysML's behaviour diagrams, like sequence diagram (SD) and state machine diagram (STM) are not specifically described in this paper, we believe that its roles in modeling the behavior of our robot system are significantly important. SD plays its main roles in designing object interaction arranged in time sequence and also could be used to describe the task event sequencing logic. This is a common activity in designing real-time embedded systems especially in robotics. As for STM, it can be used to describe how state-dependent aspects of the system are defined by a finite state machine and can help in designing and in developing highly state-dependent systems. As depicted in Fig.12, the SysML specification (OMG SysML, 2010) defines an allocation notation using stereotypes to show how various elements are allocated to and from other elements. Such allocations may be used to show deployment or more generally to relate different parts of a model as the design progresses. Allocation models help us to map the SysML software modules to the compatible RTCs based on the C++ language.
In addition, by using the SysML notation for system engineering analysis, software and system engineers not only can communicate among themselves to develop and integrate specific components for providing various functions but also will encourage interest in model-driven engineering approach.

4.2 The future of robotics software development

To our best knowledge, not much effort has been done especially in robotics software and system development that use model-driven engineering (MDE) method by adopting a general modeling language like SysML. Both MDE and MDA are interconnected in terms of technology approaches to a broad range of systems and software engineering. The former tends to propel the development process while the latter is the separation of the operation of a system from the details of its platform in the development process.
According to (Bruyninckx, 2008), the robotics community is seriously neglecting the progress in MDE, hence discarding lots of opportunities to let software engineering and practice mature in the domain of robotics. Therefore, creating SysML profile for robotics is one of the solutions. For instance, the robotics literature includes bunch of articles about architecture which most of them use graphical models with boxes and arrows, but the meaning of these models has never been standardized, and the practical constraints on real-world implementations are implicit. In spite of that, standardization of SysML and its real-time and embedded specialization (MARTE) provides exceptional ways to start with reusable and semantically well-defined designs of complex software systems.

5. Conclusion and future work

In this paper, we have presented the basic specification of our mobile robot. The proposed mobile robot software system is fully based on model-driven engineering methodology to overcome the current difficulties of mobile robot development. The highlighted features of our approach are really straightforward, mobile platform independent, reusable model design, reduced development efforts, and also customizable system configuration.
Based on this specification, future work will focus on the detailed design for each module as well as extending SysML profiles that ideally suit to the robotics domain. Then the suitable RTCs will be selected from the RTC Centre of Intelligent RT Software Project. The reusability of the platform independent component, such as mobile control will be further extended by extracting the platform-independent parts.
As we believe that SysML can easily provide information models for capturing system aspects (i.e. both structure and behavior), we continue our effort to integrate some software tools for model-based design (e.g. Simulink and Modelica) in order to fully utilize SysML as a platform of model integration.

6. Acknowledgment

This work is supported by NEDO Japan (New Energy and Industrial Technology Development Organization) project on "Intelligent RT Software Project".

7. References

Ando, N.; Suehiro, T. ; Kitagaki, K. ; Kotoku, T., & Yoon, W.-K. (2005). RT-Middleware: Distributed Component Middleware for RT (Robot Technology), *Proceedingsof IEEE/RSJ 2005 International Conference on Robots and Intelligent Systems (IROS 2005)*, pp. 3555-3560, ISBN 0-7803-8192-3, Edmonton, AB, Canada, Aug 2-6, 2005

Ando, N.; Suehiro, T. & Kotoku, T. (2008). A Software Platform for Component Based RT-System Development : OpenRTM-Aist, *Proceedingsof the 1st International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2008)*, pp. 87-98, ISBN 987-3-540-89075-1, Venice, Italy, Nov 3-7, 2008

Bruyninckx, H. (2008). Robotics Software: The Future Should Be Open. *IEEE Robotic & Automation Magazine,* Vol.5, No.1, (March 2008), pp. 9-11, ISSN 1070-9932

Model Driven Engineering, MDE (2008). Model-Driven Engineering. Available at http://www.theenterprisearchitect.au/archive/2008/model_driven_engineering/

Okano, K. & Yasukawa, K. (2009). NEDO: Overview of Intelligent RT Software Project, *Proceedingsof RSJ 27th Annual Conference on The Robotics Society of Japan*, RSJ2009 AC1D1-01 (in Japanese), Tokiwadai, Hodogaya, Yokohama, Sept 15, 2009

OMG MDA, Version 1.0.1 (2003). Model Driven Architecture Guide. Available from http://www.omg.org/cgi-bin/doc?omg/03-06-01

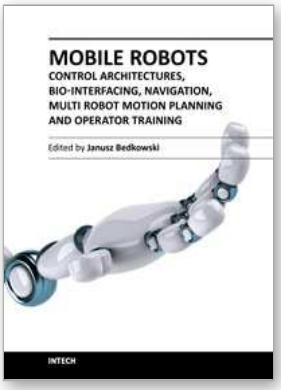
OMG Robotics-DTF, (2010). The OMG Robotics Domain Task Force. Available from http://www.robotics.omg.org/

OMG Systems Modeling Language, Version 1.2 (2010). SysML Specification. Available from http://www.omg.org/spec/SysML/2.3/

OMG Unified Modeling Language, Version 2.3(2010). UML Specification. Available from http://www.omg.org/spec/UML/2.3/

SEGWAY, (2009).Tohoku University team completes Tsukuba Challenge. Available from http://rmp.segway.com/2009/11/16/team-from-tohoku-university-completes-tsukuba-challenge-09/

Yuta, S.; Mizukawa, M.; Hashimoto, H.; Tashiro, H. & Okubo, T. (2010). Tsukuba Challenge 2009- Towards Robots Working in the Real World: Records in 2009. *Journal of Robotics and Mechatronics,* Vol.23, No.2, (April 2011), pp. 201-206, ISSN 1883-8049



Mobile Robots - Control Architectures, Bio-Interfacing, Navigation, Multi Robot Motion Planning and Operator Training

Edited by Dr. Janusz Będkowski

ISBN 978-953-307-842-7
Hard cover, 390 pages
Publisher InTech
Published online 02, December, 2011
Published in print edition December, 2011

The objective of this book is to cover advances of mobile robotics and related technologies applied for multi robot systems' design and development. Design of control system is a complex issue, requiring the application of information technologies to link the robots into a single network. Human robot interface becomes a demanding task, especially when we try to use sophisticated methods for brain signal processing. Generated electrophysiological signals can be used to command different devices, such as cars, wheelchair or even video games. A number of developments in navigation and path planning, including parallel programming, can be observed. Cooperative path planning, formation control of multi robotic agents, communication and distance measurement between agents are shown. Training of the mobile robot operators is very difficult task also because of several factors related to different task execution. The presented improvement is related to environment model generation based on autonomous mobile robot observations.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821