LATVIJAS LAUKSAIMNIECĪBAS UNIVERSISTĀTE
INFORMĀCIJAS TEHNOLOĢIJU FAUKULTĀTE
DATORU SISTĒMU KATEDRA

Vitālijs Komašilovs

HETEROGĒNAS ROBOTU KOLONIJAS SPECIFIKĀCIJAS OPTIMIZĀCIJAS UZDEVUMA RISINĀŠANAS PROCEDŪRA

Promocijas darbs doktora grāda ieguvei
Informācijas tehnoloģiju nozarē
(Dr.sc.ing.)

IEGULDĪJUMS TAVĀ NĀKOTNĒ

EIROPAS SAVIENĪBA

Promocijas darba izstrāde un noformēšana līdzfinansēta no Eiropas Savienības Sociālā fonda

Promocijas darba vadītājs: ..... Dr.sc.ing., asoc. profesors
Egils Stalidzāns

Promocijas darbu izstrādāja: ..... Mg.sc.ing., doktorants
Vitālijs Komašilovs

JELGAVA
2013

LATVIA UNIVERSITY OF AGRICULTURE
FACULTY OF INFORMATION TECHNOLOGIES
DEPARTMENT OF COMPUTER SYSTEMS

Vitalijs Komasilovs

PROCEDURE FOR RESOLVING SPECIFICATION OPTIMIZATION TASK OF HETEROGENEOUS ROBOT COLONY

Thesis for the acquisition of doctoral degree
in the field of Information Technologies
(Dr.sc.ing.)

EUROPEAN SOCIAL FUND

INVESTMENT IN YOUR FUTURE

EUROPEAN UNION

Thesis prepared with financial support of European Social Fund

Scientific Adviser of the Thesis: ..... Dr.sc.ing., Associate professor
Egils Stalidzans

Author of the Thesis: ..... Mg.sc.ing., Doctorate
Vitalijs Komasilovs

JELGAVA
2013

PARTICULARS

Research was executed at: Latvia University of Agriculture, Faculty of Information Technologies, Department of Computer Systems, Liela st. 2, Jelgava, Latvia

Experimental research was executed at: Latvia University of Agriculture, Faculty of Information Technologies, Computer Systems Department, Liela st. 2, Jelgava, Latvia

Scientific Advisor of the Doctoral Thesis: Dr.sc.ing. Egils Stalidzans, Associate Professor, Latvia University of Agriculture

The research results are presented in seven scientific publications:

1. Komasilovs, V., Stalidzans, E. (2010) Simulation of Real-Time Robot Control Systems Using Player/Stage Software. In: *Industrial Simulation Conference 2010*. Budapest, Hungary: Eurosys-ETI, p. 39–41.
2. Komasilovs, V., Stalidzans, E. (2011) Functional decomposition method reveals the number of possible specifications of multi-robot system. In: *2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI)*. Budapest, Hungary: IEEE, p. 161–165.
3. Komasilovs, V. (2012a) Investment and running cost estimation for heterogeneous multi-robot system. In: *5th International Scientific Conference on Applied Information and Communication Technology*. Jelgava, Latvia, p. 118–122.
4. Komasilovs, V. (2012b) Investment costs optimization of multi-robot system using genetic algorithm. In: *Annual 18th International Scientific Conference "Research for Rural Development 2012"*. Jelgava, Latvia, p. 229–232.
5. Komasilovs, V., Stalidzans, E. (2012a) Genetic algorithm used for initial evaluation of specification of multi-robot system. In *13th International Carpathian Control Conference*. High Tatras, Slovakia: IEEE, p. 313–317.
6. Komasilovs, V., Stalidzans, E. (2012b) Procedure of Specification Optimization of Heterogeneous Robotic System. In: *IEEE 10th Jubilee International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. Herl'any, Slovakia: IEEE, p. 259–263.

7. Komasilovs, V. (2013) Software modules for optimization of specification of heterogeneous multi-robot system. In: *12th International Scientific Conference Engineering for Rural Development*. Jelgava, Latvia, p. <in–press>.

The research results were presented at the following conferences and seminars:

1. Simulation of Real-Time Robot Control Systems Using Player/Stage Software. *Industrial Simulation Conference*. Budapest, Hungary. June 7-9, 2010.
2. Concept of Functional Decomposition Method Used for Optimization of Specification of Heterogeneous Multi-robot System. In *RTU 52nd International Scientific Conference*. Riga, Latvia. October 13-15, 2011.
3. Complexity analysis of decomposition approach used for optimization of specification of multi-robot system. In *Joint 3rd World Congress of Latvian Scientists and 4th Letonica Congress "Science, Society and National Identity"*. Rīga, Latvija. October 24-27, 2011.
4. Functional decomposition method reveals the number of possible specifications of multi-robot system. In *2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI)*. Budapest, Hungary. November 21-22, 2011.
5. Procedure of Specification Optimization of Heterogeneous Robotic System. In *IEEE 10th Jubilee International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. Herl'any, Slovakia. January 26-28, 2012.
6. Investment and running cost estimation for heterogeneous multi-robot system. In *5th International Scientific Conference on Applied Information and Communication Technology*. Jelgava, Latvia. April 26-27, 2012.
7. Investment costs optimization of multi-robot system using genetic algorithm. In *Annual 18th International Scientific Conference "Research for Rural Development 2012"*. Jelgava, Latvia. May 16-18, 2012.
8. Genetic algorithm used for initial evaluation of specification of multi-robot system. In *13th International Carpathian Control Conference*. High Tatras, Slovakia. May 28-31, 2012.
9. Specification optimization of heterogeneous multi-robot systems. In open seminar of "*Development of intellectual multi-agent robotic system"*. Riga, Latvia. January 11, 2013.

The thesis was approved at the expanded academic session of the Computer Systems Department of the Faculty of Information Technologies of the Latvia University of Agriculture on 30 November, 2012. Minutes No. 5

The doctoral thesis was produced with the assistance of the **Europenian Social Fund (ESF)** project “Atbalsts LLU doktora studiju īstenošanai” (agreement No. 2009/0180/1DP/1.1.2.1.2/09/IPIA/VIAA/017).

ABSTRACT

PhD thesis of **Vitalijs Komasilovs** on the **Procedure for Resolving Specification Optimization Task of Heterogeneous Robot Colony** was developed at the Department of Computer Systems of Latvia University of Agriculture during the period from September 2009 through February 2013.

The PhD thesis consists of 194 pages, comprising 8 tables, 39 pictures, 17 formulae, 4 annexes. 247 literature sources were used.

The goal of the PhD thesis is to improve the specification development for heterogeneous multi-robot systems during design stage by analyzing the full solution domain instead of testing only a part of possible solutions.

In order to achieve the goal of the thesis the list of objectives were defined as follows:

1. perform analysis of specification development methods applied for heterogeneous multi-robot systems;
2. define specification optimization task and its solution concept for heterogeneous multi-robot systems;
3. develop the procedure for finding optimal specification of heterogeneous multi-robot system in full solution domain;
4. develop mission definition technique and its decomposition approach for heterogeneous multi-robot systems;
5. perform the analysis of the size of feasible solution domain of the specification optimization task;
6. implement and experimentally test heuristic search algorithm for initial evaluation of specifications of multi-robot system;
7. analyze possibility to use simulation techniques for fine evaluation of specification of multi-robot system.

The content of PhD thesis is structured according to the goal and the tasks of PhD thesis consisting of 6 sections.

Section I gives a general overview of multi-robot systems, discusses the state-of-the-art of the multi-robot research domain and indicates specification selection problem for the heterogeneous multi-robot system.

Section II defines specification optimization task for multi-robot system, describes the concepts used for optimization and gives an overview of specification optimization procedure developed within the thesis.

Section III describes the first and the second steps of the specification optimization procedure, defines mission decomposition approach into components and tasks.

Section IV refers to the third and the fourth steps of the specification optimization procedure, provides analysis of the domain of feasible solutions, defines formulas for estimating the size of the domain, and introduces *CoMBot-Gen* software used for the analysis.

Section V refers to the fifths step of the procedure and defines initial evaluation of solution candidates using heuristic methods, introduces *GAMBot-Eva* software implementation of genetic algorithm based heuristic search, describes development of genetic representation of the solution domain, and defines the model for the estimation of total costs of ownership.

Section VI refers to the last steps of the procedure and defines simulation based evaluation of solution candidates, describes the setup of the simulation environment, and introduces *SiMBot-Ctr* control framework and its application peculiarities.

The main results, conclusions and future development prospects are described in conclusion of the PhD thesis.

ANOTĀCIJA

Vitālijs Komašilovs promocijas darbs **Heterogēnas robotu kolonijas specifikācijas optimizācijas uzdevuma risināšanas procedūra** izstrādāts Latvijas Lauksaimniecības universitātē, Informācijas Tehnoloģiju fakultātē, Datoru sistēmu katedrā laika periodā no 2009. gada septembra līdz 2013. gada februārim.

Darba apjoms ir 194 lapaspuses, tas ietver 8 tabulas, 39 attēlus, 17 formulas, 4 pielikumi. Darbā izmantoti 247 literatūras avoti.

Promocijas darba mērķis ir uzlabot heterogēno daudz-robotu sistēmu specifikācijas izstrādi projektēšanas stadijā analizējot pilnu risinājumu telpu nevis pārskatot tikai daļu no iespējamiem risinājumiem.

Pētījuma mērķa sasniegšanai tika izvirzīti vairāki uzdevumi:

1. analizēt specifikācijas izstrādes metodes, kas ir pielietojamas heterogēnām daudz-robotu sistēmām;
2. definēt specifikācijas optimizēšanas uzdevumu un tā risinājuma konceptu heterogēnām daudzrobotu sistēmām;
3. izstrādāt procedūru optimālās heterogēnās daudzrobotu sistēmas specifikācijas meklēšanai pilnā risinājumu telpā;
4. izstrādāt heterogēno daudzrobotu sistēmu uzdevuma uzdošanas tehniku un tā dekompozīcijas paņēmienu;
5. analizēt iespējamo risinājumu telpas izmēru specifikācijas optimizēšanas uzdevumam;
6. implementēt un praktiski pārbaudīt heiristiskās meklēšanas metodi daudzrobotu sistēmas specifikācijas pirmās kārtas novērtēšanai;
7. analizēt iespēju izmantot imitācijas tehnikas daudz-robotu sistēmas specifikācijas otrās kārtas novērtēšanai.

Promocijas darba saturs un struktūra veidota atbilstoši mērķim un uzdevumiem un sastāv no 6 nodaļām.

Pirmajā nodaļā ir sniegts pārskats par daudz-robotu sistēmām, ir diskutēts par jaunākām tendencēm daudzrobotu izpētes sfērā un ir norādīta specifikācijas izvēles problēma heterogēnām daudzrobotu sistēmām.

Otrā nodaļā tiek definēts daudzrobotu sistēmas specifikācijas optimizēšanas uzdevums, tiek aprakstīti tā koncepti un tiek dots pārskats par specifikācijas optimizēšanas procedūru, kas ir izstrādāta disertācijas ietvaros.

Trešajā nodaļā ir aprakstīti specifikācijas optimizēšanas procedūras pirmais un otrais soļi, ir definēts uzdevuma dekompozīcijas paņēmiens komponentēs un apakšuzdevumos.

Ceturtajā nodaļā tiek aprakstīti specifikācijas optimizēšanas procedūras trešais un ceturtais soļi, tiek veikta iespējamo risinājumu telpas analīze, tiek definētas formulas risinājumu telpas izmēra novērtēšanai, tiek aprakstīta risinājumu telpas analīzes programmatūra *CoMBot-Gen*.

Piektajā nodaļā ir definēta risinājumu kandidātu pirmās kārtas novērtēšana izmantojot heiristiskās metodes, ir aprakstīta *GAMBot-Eva* programmatūra kas implementē uz ģenētiskā algoritma balstītu heiristisko meklēšanu, ir aprakstīta risinājumu telpas ģenētiskā attēlojuma izstrāde, ir definēts kopējo izmaksu novērtēšanas modelis.

Sestajā nodaļā tiek aprakstīti pēdējie optimizēšanas procedūras soļi, tiek definēta uz imitācijas modeļiem balstītā risinājumu kandidātu novērtēšanas, tiek aprakstīta imitācijas vides konfigurācija, tiek aprakstīts *SiMBot-Ctr* vadības karkass un tā izmantošanas īpatnības.

Galvenie disertācijas rezultāti, secinājumi un nākotnes perspektīvas ir aprakstītas disertācijas secinājumu nodaļā.

АННОТАЦИЯ

Докторская диссертация **Виталия Комашилова «Процедура оптимизации спецификации гетерогенной колонии роботов»** разработана на Кафедре вычислительных систем Латвийского сельскохозяйственного университета в период времени с сентября 2009 года до февраля 2013 года.

Объем работы – 194 страниц, включая 8 таблиц, 39 иллюстраций, 17 формул, 4 приложения. В работе использовались 247 литературных источников.

Цель докторской диссертации – усовершенствовать разработку спецификаций для гетерогенных мультиагентных роботизированных систем во время этапа проектирования, анализируя полный домен возможных решений вместо тестирования только части возможных решений.

Для достижения цели докторской диссертации были выдвинуты следующие задачи:

1. анализировать методы разработки спецификации для гетерогенных мультиагентных роботизированных систем;
2. формулировать задачу оптимизации и концепты её решения для гетерогенных мультиагентных роботизированных систем;
3. разработать процедуру для нахождения оптимальной спецификации гетерогенной мультиагентной роботизированной системы в полном домене возможных решений;
4. разработать технику дефиниции задания и её декомпозиции для гетерогенной мультиагентной роботизированной системы;
5. анализировать размер домена возможных решений для задачи оптимизации спецификации гетерогенной мультиагентной роботизированной системы;
6. реализовать и экспериментально проверить эвристический алгоритм поиска для первичной оценки спецификации мультиагентной роботизированной системы;
7. анализировать возможность использовать технику симуляций для точной оценки спецификации мультиагентной роботизированной системы.

Содержание докторской диссертации: работа структурирована в соответствии с целью и задачами и состоит из 6 глав.

В первой главе дан общий обзор мультиагентных роботизированных систем, обсуждаются современные тенденции в сфере исследований мультиагентных роботизированных систем и указана проблема выбора спецификации для гетерогенных мультиагентных роботизированных систем.

Во второй главе формулируется задача оптимизации спецификации мультиагентной роботизированной системы, описаны концепты оптимизации и дан обзор процедуры оптимизации разработанной в рамках данной диссертации.

В третьей главе описаны первый и второй шаги процедуры оптимизации спецификации, определен метод декомпозиции задания роботизированной системы в компоненты и подзадачи.

В четвертой главе описаны третий и четвертый шаги процедуры оптимизации спецификации, дан анализ домена возможных решений, разработаны формулы для расчета размера домена, описано программное обеспечение *CoMBot-Gen* использованное для анализа домена.

В пятой главе формулирована первичная оценка кандидатов решений, используя эвристические методы, описано *GAMBot-Eva* программное обеспечение, которое реализует эвристический поиск основанный на генетическом алгоритме, формулировано генетическое представление домена решений и модель для определения совокупной стоимости владения роботизированной системой.

В шестой главе описаны последние шаги процедуры и формулирована оценка кандидатов решения на основе симуляций, описана конфигурация среды симуляций, разработан *SiMBot-Ctr* программный каркас контроля и описаны особенности его использования.

Основные результаты, выводы и дальнейшие перспективы развития изложены в конце докторской диссертации.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

During the last decade multi-robot systems research field has become one of the most actual directions in the robotics and many researchers have focused on it. Early researches laid the foundation of the multi-robot system functional principles and therefore provided a solid base for the following investigations. Most of the researches in multi-robot systems field have a trend to focus on the development of working solution for a particular task. For now there are available a lot of different control architectures, communication strategies and other approaches developed to be used in multi-robot system. From the other side there are relatively few formal models and analytical solutions that support decision making during design stage.

Despite increased complexity in design and development of multi-robot systems, it has various advantages over single-robot systems. The list of advantages of multi-robot systems includes aspects and applications as follows:

- ✓ robustness or fault tolerance of the system which is achieved by additional redundancy;
- ✓ tasks which are beyond the limits of single robot like moving the large and heavy objects, assembly of complex structures;
- ✓ tasks which are too complex to be cost effective to perform by single multi-purpose robot;
- ✓ rapid task execution due to massive parallelism in multi-robot system.

Relatively less explored are the multi-robot systems that are composed from heterogeneous robots, which means that at least one member of such system differs from others by mechanical, sensing or processing hardware, or by internal control architecture. Heterogeneous multi-robot systems potentially have larger fault tolerance degree and are capable for redundant solutions of a problem, as well as more versatility in performing complex tasks.

For a user of multi-robot system implemented to perform certain task one of the major indicators are the costs of the system. The number of robot classes, as well as the specification of functions of each class and the number of instances of each class in the system are the parameters of the system that could be adjusted in order to optimize the costs of the system. In practice mentioned parameters are usually predefined and optimization potential is not assessed. As a result multi-robot system becomes

unattractive for the customer because of lack of clear calculations in all positions of costs and predictable results of adjusting the parameters of system.

The thesis refers to the formal identification and quantification of the characteristics of multi-robot systems and it is devoted to the approach of optimization of various parameters of multi-robot systems to reach the most efficient set-up of system for a particular task assigning necessary functionality to instances of robot classes.

Goal and objectives of the thesis

The goal of the thesis is to improve the specification development for heterogeneous multi-robot systems during design stage by analyzing the full solution domain instead of testing only a part of possible solutions.

In order to achieve the goal of the thesis the list of objectives were defined as follows:

1. perform analysis of specification development methods applied for heterogeneous multi-robot systems;
2. define specification optimization task and its solution concept for heterogeneous multi-robot systems;
3. develop the procedure for finding optimal specification of heterogeneous multi-robot system in full solution domain;
4. develop mission definition technique and its decomposition approach for heterogeneous multi-robot systems;
5. perform the analysis of the size of feasible solution domain of the specification optimization task;
6. implement and experimentally test heuristic search algorithm for initial evaluation of specifications of multi-robot system;
7. analyze possibility to use simulation techniques for fine evaluation of specification of multi-robot system.

Research methods

Custom software is developed for analysis of solution domain of specification optimization task for multi-robot systems. This includes modules for defining the

missions, components and their properties, for generating solution candidates, for filtering incomplete combinations and for estimating the total number of possible solutions. The software is developed using *Java* programming language.

Custom methods of combinatorial analysis are applied to assess the solution domain of the problem. Initial evaluation of specification candidates of multi-robot system is implemented using genetic algorithm, the kernel for genetic processing is provided by *JGAP* framework. Practical experiments are executed on dedicated processing hardware available in university (*IBM 3850*).

Simulation based evaluation of specification of multi-robot system is implemented using *Player* device interface and network server for robot control (a hardware abstraction layer for robotic devices) and *Stage* simulation package for population of mobile robots.

Theses

- ✓ There is a lack of formal methods for the development of optimal specification of heterogeneous multi-robot systems.
- ✓ Introduced level of the component primitives allows formal mission definition for the multi-robot systems.
- ✓ The size of the solution domain of the specification development problem for heterogeneous multi-robot systems depending on the number of components grows nonlinearly.
- ✓ It is possible to develop full solution domain scale specification optimization procedure for heterogeneous multi-robot systems.

Scientific novelty and practical value

- ✓ Specification optimization task for a heterogeneous multi-robot system is defined using detailed concepts for the solution including component and agent primitives.
- ✓ Full solution domain covering heterogeneous multi-robot system specification optimization procedure is developed defining the workflow from the business requirements specification to the preferred specification of the multi-robot system.

- ✓ Formulas for determination of the number of solutions in the specification domain of multi-robot system are developed.
- ✓ Genetic algorithm based heuristic search is adapted for the specification optimization task adapting techniques for implementation of the genetic representation, the fitness function and the evolution processing.
- ✓ Signal based processing is implemented within the framework for control of multi-robot system in the simulated environment.

The developed specification optimization procedure enables formal analysis of the business requirements and provides a framework for finding the optimal setup of the heterogeneous multi-robot system. Optimal specification aims to apply appropriate agents and the increase utilization of their components in industrial applications. That leads to the increased efficiency of the production system, which in turn lowers the maintenance costs of the system and increases industrialist's income.

The author sees the possibility to use the robotic system implemented using a real hardware for fine tuning of the specification optimization procedure.

Practical hardware implementation of grass mowing agents is started within the master's thesis supervised by the author. Field experiments with the working prototype of autonomous grass mower with steering and GPS system are running.

Acronyms and definitions

The list of acronyms and their definitions used within the thesis is as follows:

Acronym	Definition
CoMBot-Gen	Combination Generator of Multi-Robot system specification
CPU	Central Processing Unit
DBMS	Database Management System
DNS	Domain Name Service
GAMBot-Eva	Genetic Algorithm based Evaluation of Multi-Robot System Specification
GPS	Global Positioning System
JDBC	Java Database Connectivity
JGAP	Java Genetic Algorithms and Genetic Programming Package
SiMBot-Ctr	Simulated Multi-Robot System Control Framework
TCO	Total Costs of Ownership
TCP	Transmission Control Protocol
XML	eXtensible Markup Language

1. DEFINING THE PROBLEM OF SPECIFICATION SELECTION FOR MULTI-ROBOT SYSTEM

From the very beginning of civilized society humans where endeavored to use various tools and mechanisms in order to facilitate their labor and increase level of production. Technological advances contributed to the development of more complex machines that in turn boosted the progress. Modern history shows great examples of achievements which changed everyday life of human beings such as steam and combustion engines, industrialization, and electrical power.

Second half of second millennium is known for man intention to build complex automates. The list of great names includes Leonardo Da Vinci and his mechanical knight; musical automates by Jacques de Vaucanson, Hisashige Tanaka's mechanical toys. They laid the foundation of mechanical techniques afterwards used by modern inventors in combination with electrical components (e.g. Nikola Tesla).

In the early XX century word "*robot*" was introduced by Czech writer Karel Čapek in his play R.U.R. (Rossum's Universal Robots). The play shows artificial people called "robots" that are able to think for themselves and happy to serve. Later Isaac Asimov used the word "robotics" to describe this field of study. Robotics is the field of science or technology that deals with the design, construction, operation, structural disposition, manufacture and application of robots (Robotics, 2011). It is closely related to the science of electronics, engineering, mechanics and information technologies. In popular culture, the term "robot" generally implies some human-like appearance of artificially created machines made up of mechanical parts.

The term "robot" does not have unified definition, contrary, many authors suggest different interpretation. There are wide diversity in definitions available in dictionaries: a machine that resembles a human and does mechanical, routine tasks on command; a person who acts and responds in a mechanical, routine manner, usually subject to another's will; any machine or mechanical device that operates automatically with humanlike skill (Robot, 2011).

Together with the definition of robot terms intelligent and autonomous are often used. Murphy (2000a) defines an intelligent robot as it is a mechanical creature which can function autonomously. In opposite to factory automation, "intelligent" denotes that robot does not act mindlessly, in repetitive manner. The "mechanical creature" portion

of definition is used to emphasize the difference between a robot and a computer. Usually robot uses a computer as a building block, but the robot is able to interact with its world as opposed to computer, which does not do it. Autonomous functioning means that robot operates self-contained, without requiring input from human operator. Autonomy refers to systems capable of operating in the real-world environment without any form of external control for extended periods of time. It indicates that robot can adapt to changes in its environment or itself and continue to carry out its mission. The author of this work uses aforementioned definition as most suitable for the scope of the research.

First robots used to exhibit biological behaviors and were used to understand the principles of robot control. William Grey Walter was the pioneer in this field; he demonstrated electronic autonomous robots that were able to respond on light stimulus, by which they used to find recharging station (Sabbatini, 1999). Further advances in robotics showed wide advantages of this field. In the end of 1950s the first industrial robot Unimate was created using original patents of George Devol. Machine was used for automation of dangerous tasks for human workers.

Eventually robot utilization on production sites for automation of various tasks has become common practice. Modern robotic systems are used widely in such areas as machinery, assembly-line production or medicine, where they perform repeatable and precise actions in more efficient way than human. Some habitual industries are unimaginable without the utilization of robotic systems; these include food processing, microchip production.

Notable application of the robotic systems, that should be mentioned, is the use of the robots for dangerous service. Remotely operated military robots are used for mine clearance and intelligence that lowers risks for humans being injured or even killed. Robots are well suited for the operation in the hazardous environments. In 2011 various types of robots worked to contain the meltdown at the Fukushima Daiichi nuclear plant. They were designed to operate at radiation levels too high for humans. Meanwhile the area above the Fukushima plant was a no-fly zone for manned aircraft, but a Global Hawk drone has been providing imagery information (Hambling, 2011).

Perhaps the best example of intelligent autonomous robot is the Terminator shown in films by James Cameron. It exhibits extreme adaptability and autonomy. More real examples include space exploration robots. Great examples of engineering achievements are *Mars Exploration Rovers* (see figure 1.1), which operated on Mars

more than 5 years, while planned length of mission was only 90 day (Bajracharya et al., 2008).

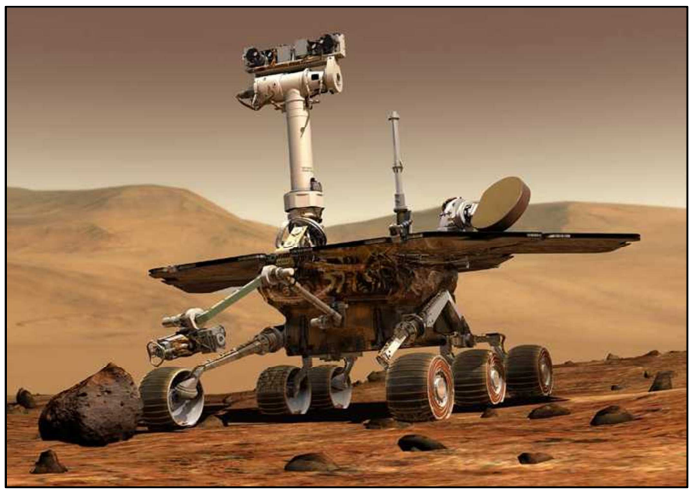


Figure 1.1. **Mars Exploration Rover**
Source: http://en.wikipedia.org/wiki/Mars_Exploration_Rover (accessed 2012.11.20)

1.1. Robot control principles

The definition of robot used in previous section points toward the intelligent machine that operates autonomously. However term “control” is used frequently in scope of robotic systems, which appears to be a contradiction to autonomy. In order to clarify this trait deeper insight to the robotics field is required.

Robotics field is closely related to and even sometimes considered as a part of mechatronics. Mechatronics is the multidisciplinary field, combination of mechanical engineering, computing, and electronics, as used in the design and development of new manufacturing techniques (Mechatronics, 2011); the approach aiming at the synergistic integration of mechanics, electronics, control theory, and computer science within product design and manufacturing, in order to improve and/or optimize its functionality (see figure 1.2). The word Mechatronics itself came from "mecha" for mechanical and "tronics" for electronics.

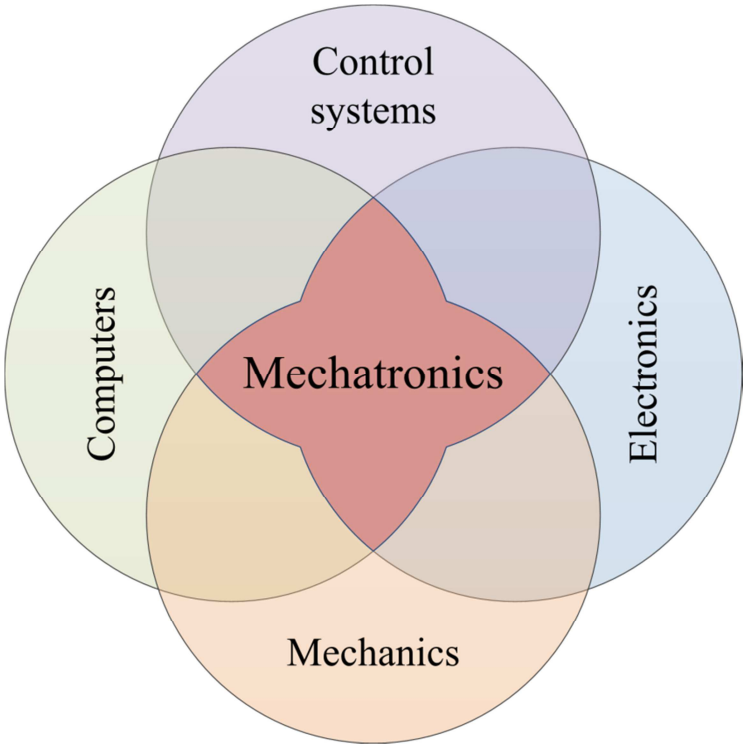


Figure 1.2. **Concept of mechatronics**

The main difference between mechatronic and robotic systems is that the first gets required inputs from external sources, while the second one produces such inputs by its own.

Robot consists of various types of elements and because of that it is usually considered as a system. System is a group or combination of interrelated, interdependent, or interacting elements forming a collective entity; a methodical or coordinated assemblage of parts, facts, concepts, etc.; any assembly of electronic, electrical, or mechanical components with interdependent functions, usually forming a self-contained unit; computer system, including input/output devices, the supervisor program or operating system and possibly other software (System, 2011).

With the exception of mechanical construction of chassis robot consists of three main blocks of elements that are as follows:

- ✓ actuators are mechanical devices that operate by the source of energy, usually in the form of an electric current, and convert that energy to some kind of motion. Actuators are usually used for robot motion as well as for manipulating physical objects in environment;
- ✓ perception facilities are used to perceive the information about robot's environment as well as to identify the state of its internal physical components. It is usually implemented using various types of electronic sensors;

- ✓ control system is an element that links aforementioned components into single unit and directs it in order to perform required actions.

By the definition, control system is a system for controlling the operation of another system (Control System, 2011). Control theory distinguishes wide variety of control approaches with deep analysis of their strengths and weaknesses, but the very basic approach is a control system with feedback loop (see figure 1.3).

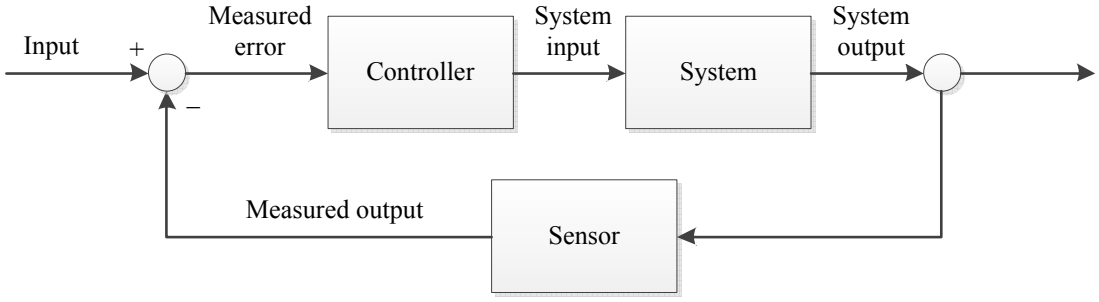


Figure 1.3. **Basic control system**

It is used to control output of the system according to inputs and taking into account the output of system (feedback). The elements showed on picture are as follows:

- ✓ input is a desired system's output, which is usually represented by any kind of measurable parameter, such as velocity, temperature or humidity;
- ✓ sensor is a feedback element, which is used to detect actual output of the system in terms of measurable parameters;
- ✓ the actual output value is compared to the input value using a comparator, thus producing an error signal – the deviation from desired value ("negative feedback");
- ✓ the error signal drives a controller, which processes and amplifies it, and finally produces a control signal for the controlled system;
- ✓ the controlled system is a device (or a set of devices) that directly produces the output of overall system (rotates wheels, affects environment).

Conceptually robot control system also corresponds to the described principles of control systems, where the input of the robot control system is a desired behavior of robot, the feedback is received through the perception and the set of actuators affecting the environment while being controlled by controller (hardware or software).

However robot control concept has several different meanings in robotics. To some designers it refers exclusively to ensuring that movement of robot remains stable

and performs according to control system design criteria. This means that the parameters of robot actuators are maintained on at desired values. Such aspect of control is usually called “low-level control”. The design of such control systems follows the principles of control and systems theories. Other robot designers understand robot control as the ability of the robot to follow instructions toward the goal of a mission. The abilities such as path planning and execution, mapping and navigation or task allocation are considered as “high-level control”.

Low-level control is frequently reactive, that is there is very close coupling of sensing and action: as soon as sensor detects a change in the attribute of the robot, signals are sent to actuators to perform appropriate actions. On the other hand, high-level control often requires reasoning, so sensory information has to be processed in order to send deliberated signals to actuators.

Matarić defines robot control as follows: Robot control is the process of taking information about the environment, through the robot sensors, processing it as necessary in order to make decisions about how to act, and then executing these actions in the environment (Matarić, 2002).

Following sections provide deeper insight into various aspects of robot control systems.

1.1.1. Insight into history of robot control

Science of robotic control systems takes origins from cybernetics, which is the interdisciplinary study of the structure of regulatory systems. Cybernetics is closely related to control theory and systems theory. The roots of cybernetic theory where placed in XVIII – XIX centuries, when first artificial regulatory systems where created. Contemporary cybernetics began as an interdisciplinary study connecting the fields of control systems, electrical network theory, mechanical engineering, logic modeling, evolutionary biology and neuroscience in the first half of XX century. Early applications of negative feedback in electronic circuits included the control of gun mounts and radar antenna during World War II. During the second half of XX century the field of cybernetics followed a boom-bust cycle of becoming dominant because of various actual research directions and falling when these researches are completed.

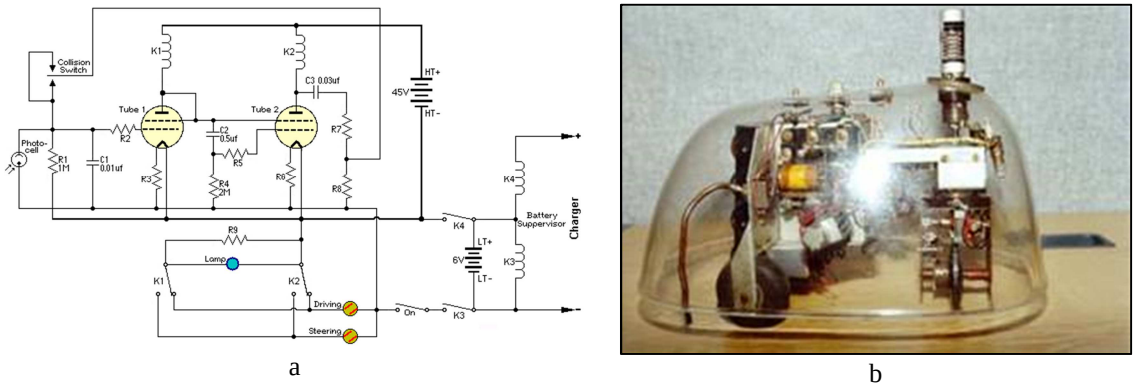
Cybernetics is an earlier but still-used generic term for many types of subject matter. These subjects also extend into many others areas of science, but are united in

their study of control of systems. Theoretical concepts from cybernetics are used in such areas as systems biology, computer science, engineering, management.

Perhaps the earliest robot control system was used by William Grey Walter in the design of his *Tortoise* in 1953. He applied natural behavior principles developed by Ashby and Wiener (1952) in form of mathematical feedback control systems. Some of the principles that were captured in his design are described below.

- ✓ Parsimony: simple is better. Simple reflexes can serve as the basis for behavior.
- ✓ Exploration or speculation: The system never remains still except when feeding (recharging). The constant motion is adequate under normal circumstances to keep it from being trapped.
- ✓ Attraction: the system is motivated to move towards some environmental object.
- ✓ Aversion: the system moves away from certain negative stimuli, for example avoiding heavy obstacles and slopes.
- ✓ Discernment: the system has the ability to distinguish between productive and unproductive behavior, adapting itself to the situation at hand.

Tortoise was constructed as an analogue device, consisted from two sensors, two actuators, and two vacuum tubes. The directional photocell for detecting the light and a bump contact sensor provided the required environmental feedback. One motor steered the single from driving wheel. The photocell always pointed to the direction of this wheel and thus could scan the environment. The driving motor powered the wheel and provided locomotion (see figure 1.4).



a b

Figure 1.4. ***Tortoise* robot by William Grey Walter**
a – electrical circuit, b – design
Source: http://www.rutherfordjournal.org/article020101.html (accessed 2012.11.20)

The tortoise exhibited the following behaviors:

- ✓ seeking light: the sensor rotated until a weak light source was detected while drive motor continuously moved the robot to explore the environment at the same time;

- ✓ head toward weak light: once a weak light was detected, the tortoise moved in its direction;
- ✓ back away from bright light: an aversive behavior repelled the tortoise from bright light sources;
- ✓ turn and push: used to avoid obstacles, this behavior overrode the light response;
- ✓ recharge battery: when the onboard battery power was low, the tortoise perceived a strong light as weak. Recharging station had strong light over it, thus the robot moved toward it and docked.

The behaviors were utilized in order of their priority. Walter's *Tortoise* exhibited moderately complex behavior: moving safely around a room and recharging itself as needed (Sabbatini, 1999).

A quite different control system characterized a robot constructed at the Stanford Research Institute (*SRI*) in 1969. This robot, *Shakey*, inhabited an artificial world, an office area with objects specially colored and shaped to assist it in recognizing an object using vision. It was constructed of two independently controlled stepper motors and had a television camera and optical range finder mounted on top of it (significantly more complex than Walter's photocell). Bump sensors where mounded at the periphery of robot for protection. However, the sensor outputs were not directly connected with the drive motors. Rather, they formed inputs to "thinking" layer that uses an artificial-intelligence planner known as the *STRIPS*. It was theorem proving system developed in *SRI* that used first-order logic to develop a navigational plan. The planner used information stored within symbolic world model to determine what actions to take to achieve the robot's goal at given time. Thus the operation of the robot consisted of the "sense-plan-act" sequence.

Around 1977 robot *HILARE* was created at *LAAS* in Toulouse, France. Its world contained the smooth flat floors found in typical office environment. It was equipped with three wheels, video camera, fourteen ultrasonic sensors, and a laser range finder (see figure 1.5). Planning was conducted within multi-level representational space: geometric models represented the actual distances and measurements of the worlds, and a relational model expressed the connectivity of rooms and corridors.

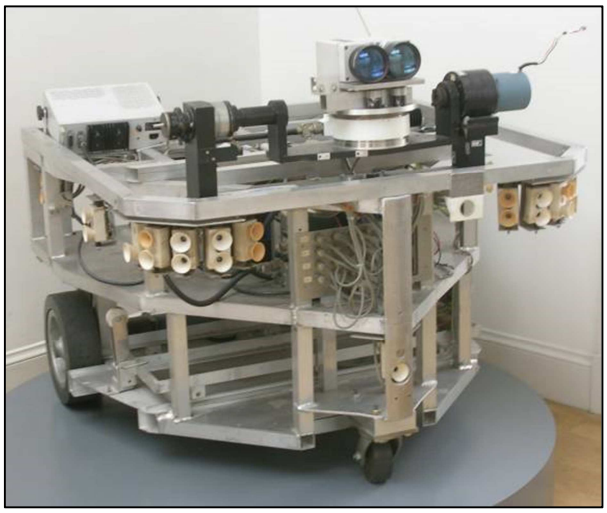


Figure 1.5. ***HILARE* robot**
Source: http://maisonetranger.wordpress.com/2010/07/28/musee-des-arts-et-metiers/
(accessed 2012.11.20)

The Stanford Cart was a minimalistic robot platform used by Moravec to test stereo vision as a means for navigation (Moravec, 1977). The cart successfully navigated fairly complex twenty meter courses, avoiding visually detected obstacles as it went. Obstacles where added to its internal world map as detected and were represented as enclosed spheres. The cart used a graph search algorithm to find shortest path through this abstract model.

These and other robotic precursors set the stage for the advances and controversies to come as paradigms of robot control.

1.1.2. Primitives of robot control paradigms

A paradigm is a philosophy or set of assumptions and/or techniques which characterize an approach to class of problems; a very general conception of the nature of scientific endeavor within which a given enquiry is undertaken (Paradigm, 2011b). In an intellectual discipline a set of assumptions, concepts, values, and practices that constitutes a way of viewing reality for the community that shares them (Paradigm, 2011a).

No one paradigm is right; rather, some problems seems better suited for different approaches. Applying the right paradigm makes problem solving easier. Therefore, knowing the paradigms of robotics is one key to being able to successfully control a robot for a particular application.

There are currently three paradigms for organizing intelligence of robots: hierarchical, reactive, and hybrid. There are two ways of describing the paradigms. The

first approach is to describe the relationship between the three commonly accepted primitives of robotics. The primitives themselves are as follows:

- ✓ SENSE – the functions taking information from the robot's sensors and producing an output useful to other functions;
- ✓ PLAN – the functions taking information either from other primitives or using own knowledge about world, and producing one or more tasks for the robot to perform;
- ✓ ACT – the functions that produce output commands for physical actuators.

The second approach is to describe the paradigm by the way sensory data is processed and distributed through the system. In some paradigms, sensor information is restricted to being used in a specific, or dedicated, way for each function of robot. Other paradigms expect all sensor information to be first processed into single model of world.

The very first step in design of robot control system is determination of most suitable paradigm for particular application. Selection of the paradigm engages to use the tools and common approaches associated to the paradigm, usually called as architectures.

Arkin (1998) describe several definitions of robot architecture. One of them is the derivative from the definition of computer architecture, and it states that robot architecture is the discipline devoted to the design of highly specific and individual robot from a collection of common software building blocks. Hayes-Roth (1995) refers architecture to the set of structural components in which perception, reasoning and action occur; the specific functionality and interface of each component, and the interconnection topology between components.

Matarić (1992a) provide another definition, stating that architecture provides a principled way of organizing a control system. However, in addition to providing structure, it imposes constraints on the way the control problem can be solved.

Russell and Norvig (2009) give definition in their textbook of artificial intelligence as follows: the architecture of a robot defines how the job of generating actions from percepts is organized. Thus the definition of architecture concerns the practical structure of robot's control system – the software.

Following sections describe three aforementioned robot control paradigms in details and consider various architectures used in particular paradigms.

1.1.3. Hierarchical and deliberative robot control paradigm

The hierarchical paradigm is historically the oldest method of organizing robot control. Its inception is related to the first AI robot – the *Shakey* made at *SRI*. The hierarchical paradigm was prevalent from 1970's up to late 1980's when reactive paradigm took over the initiative. Under it, the robot operates in a top-down fashion, mostly relying on planning the actions. It was based on view how people think.

Under the hierarchical paradigm, the robot senses world, plans the next action, and then acts. At each step the robot explicitly plans the next move. Therefore each action of robot is deliberated by control system (e.g. planning module, AI). It processes primitives of robotics in sequential and ordered way as it is showed on figure 1.6.

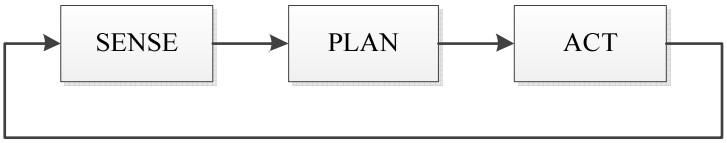


Figure 1.6. **Hierarchical robot control paradigm**

Another distinguishing feature of the hierarchical paradigm is that all the sensing data is used to create or update global world model – a single data structure, which the planner accesses. Term world model is very broad; it means both outside environment and whatever meaning the robot refers to it. Typical world model in hierarchical paradigm contains following features:

- ✓ a previously acquired representation of the environment the robot operates in, such as a map of building or relations between actions and impact;
- ✓ sensing information, such as a position of robot in environment;
- ✓ any additional knowledge that might be required to accomplish a task, such as goals of its mission.

Design and development of generic global world models is very challenging and ungrateful task, therefore various assumptions are used in order to simplify the model. The other drawback of hierarchical paradigm is its requirements for processing power. In case if the robot operates in the environment, which is close to real world, the complexity of model arises dramatically. As a result the response time of control system is too high to direct the robot in such environment.

The first studies on the Shakey robot, which control system was built according the hierarchical paradigm, showed several design issues: the closed world assumption and the frame problem. The closed world model assumption states that the world model

contains everything the robot needs to know in order to successfully operate, that means there are no unexpected cases. If closed world assumption is violated, the robot may not be able to function correctly. On the other hand, the completeness of model details depends on designer of robot: how well the human is aware of the details of the robot's environment.

Experiments of moving the Shakey robot between two rooms showed that world model is likely to be huge. Further investigations in hierarchical paradigm produced various approaches aimed on world model simplification. One of solutions tried to divide the problem into multiple layers of abstraction that is to solve the problem on a coarse level, and then refine solution in detailed level. Such approach is closely related to an area of AI called planning, which become even separate field in scientific community by the 1980's. During the 1970's and 1980's many scientists worked on either computer vision related issues, trying to get the robots to be able to better sense the world, or on path planning, computing most efficient route through robot's world. Both were directed by the demand of the hierarchical paradigm which was relevant in that time.

As mentioned above architecture is a method of implementing a paradigm, of embodying the principles in some concrete way. Perhaps the two most known architectures of the hierarchical paradigm are the Nested Hierarchical Controller developed by Meystel and the NIST Real-time Control System originally developed by Albus. A close inspection of these architectures suggests that they are well suited for semi-autonomous control. The human operator could provide the world model, decide the mission, decompose it into a plan, and then into actions. The lower level controller (robot) would carry out the actions. As robotics advanced, the robot could replace more functions and "move up" the autonomy hierarchy.

The primary advantage of the hierarchical paradigm was that it provides an ordering of the relationship between sensing, planning and acting. The primary disadvantage was planning. Every update cycle, the robot had to update a global world model and then do some type of planning. The sensing and planning algorithms of the day were extremely slow, so this introduced a significant bottleneck. Also the sensing and acting are always disconnected. This effectively eliminated any stimulus-response types of actions that are seen in nature.

Another issue that was never really handled by architectures in the hierarchical paradigm was uncertainty. It comes in many forms, such as semantic, sensor noise, and

actuator errors. Another important aspect of uncertainty is action completion: did the robot actually accomplish the action? Therefore additional control is required to check to see if the action was successful and then restart the action if it was not.

1.1.4. Reactive and behavior based robot control paradigm

The reactive paradigm grew out of dissatisfaction with the hierarchical paradigm and with an influx of ideas from nature (ethology). Although various reactive systems may or may not strictly adhere to principles of biological intelligence, they generally mimic some aspect of biology. (Brooks, 1986) summarized the dissatisfaction with the hierarchical paradigm and characterized those systems as having a horizontal decomposition as shown on figure 1.6.

Instead, an examination of the ethological literature suggests that intelligence is layered in a vertical decomposition, shown in figure 1.7. Under a vertical decomposition, an agent starts with primitive survival behaviors and evolves new layers of behaviors which reuse the lower, older behaviors, inhibit the older behaviors, or create parallel tracks of more advanced behaviors. The parallel tracks can be thought of layers, stacked vertically. Each layer has access to sensors and actuators independently of any other layers. If anything happens to an advanced behavior, the lower layer behaviors would still operate. This return to lower level mimics degradation of autonomous functions of the brain. Functions of the brain stem (such as breathing) continue independently of higher order functions (such as counting, face recognition, task planning.

The reactive paradigm was initially met with stiff resistance from traditional customers of robotics, particularly the military and nuclear regulatory agencies. These users of robotic technologies were uncomfortable with the imprecise way in which discrete behaviors combine to form a rich emergent behavior. In particular, reactive behaviors are not amenable to mathematical proofs showing they are sufficient and correct for a task. In the end, the rapid execution times associated with the reflexive behaviors led to its acceptance among users.

The fundamental attribute of the reactive paradigm is that all actions are accomplished through behaviors. By the definition, behavior is the aggregate of responses to internal and external stimuli, a specific response of a certain organism to a specific stimulus or group of stimuli (Behavior, 2011). In ethological systems,

behaviors are a direct mapping of sensory inputs to a pattern of motor actions that are then used to complete a task. From a mathematical perspective, behaviors are simply a transfer function, transforming sensory inputs into actuator commands. In robotics behavior usually treated as a schema that consist of at least one algorithm for generating the pattern of action in a physical actuator (motor schema) and one algorithm for extracting the percept and its strength (perceptual schema).

The reactive paradigm literally threw away the PLAN component of the robot primitives, as shown on figure 1.7. The SENSE and ACT components are tightly coupled into behaviors and all robotic activities emerge as the result of these behaviors operating either in sequence or concurrently. The SENSE-ACT organization does not specify how the behaviors are coordinated and controlled; this is addressed by architectures.

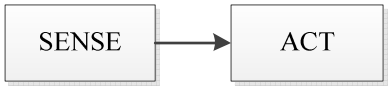


Figure 1.7. **Reactive robot control paradigm**

Sensing in the reactive paradigm is local to each behavior, or behavior-specific. Each behavior has its own dedicated sensing. In many cases, this is implemented as one sensor and perceptual schema per behavior. But in other cases, more than one behavior can take output from a sensor and process it differently. One behavior literally does not know what another behavior is doing or perceiving. Figure 1.8 graphically shows the sensing style of the reactive paradigm.

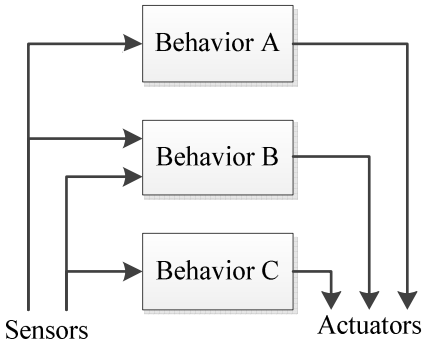


Figure 1.8. **Behavior-specific sensing**

Aforementioned aspect is fundamentally opposite of the global world model used in the hierarchical paradigm. Sensing is immediately available to the behavior's perceptual schema, which can do as little or as much processing as needed to extract the relevant percept. If computationally inexpensive processing is used, then the sensing portion of the behavior is nearly instantaneous and action is very rapid.

In early implementations of the reactive paradigm, the idea of “one sensor, one behavior” worked well. For more advanced behaviors, it become useful to fuse the output of multiple sensors within one perceptual schema to have increased precision or a better measure of the strength of the stimulus. This type of sensor fusion is permitted within the reactive as long as the fusion is local to the behavior.

There are five characteristics of almost all architectures that follow the reactive paradigm.

1. *Robots are situated agents operating in an ecological niche.* This means that robot is an integral part of the world. A robot has its own goals and intentions. When a robot acts, it changes the world, and receives immediate feedback about the world through sensing. What the robot senses affects its goals and how it attempts to meet them, generating a new cycle of actions.
2. *Behaviors serve as the basic building blocks for robotic actions, and the overall behavior of the robot is emergent.* Behaviors are independent, computational entities and operate concurrently. The overall behavior is emergent: there is no explicit “controller” module which determinates what will be done, or functions which call other functions. There may be a coordinated control program in the schema of behavior, but there is no external controller of all behaviors for a task. Since the overall behavior or a reactive robot emerges from the way its individual behaviors interact, the major differences between reactive architectures is usually the specific mechanism for interaction.
3. *Only local, behavior-specific sensing is permitted.* The use of explicit abstract representational knowledge in perceptual processing, even though it is behavior-specific, is avoided. This eliminates unnecessary processing to create a world model, then to extract information from it.
4. *These systems inherently follow good software design principles.* The modularity of these behaviors supports the decomposition of a task into component behaviors. The behaviors are tested independently, and behaviors may be assembled from primitive behaviors.
5. *Animal models of behavior are often cited as a basis for these systems or a particular behavior.* Unlike in the early days of AI robotics, where there was a conscious effort to not mimic biological intelligence, it is very acceptable under the reactive paradigm to use animals as a motivation for a collection of behaviors.

Constructing a robotic system under the reactive paradigm is often referred to as programming by behavior, since the fundamental component of any implementation is a behavior. Programming by behaviors has a number of advantages, most of them consistent with good software engineering principles. Behaviors are inherently modular and easy to test in isolation from the system. Behaviors also support incremental expansion of the capabilities of a robot. It becomes more intelligent by having more behaviors. The behavioral decomposition results in an implementation that works in real-time and usually computationally inexpensive.

In order to implement a reactive system, the designer must identify the set of behaviors required for the task. The behaviors can either be new or use existing behaviors. The overall action of the robot emerges from multiple, concurrent behaviors. Therefore a reactive architecture must provide following functionality:

- ✓ the ability to trigger behaviors;
- ✓ the capability to determinate what happens when multiple behaviors are active at the same time.

Another distinguishing feature between reactive architectures is how they define a behavior and any special use of terminology. There are many architectures which fit in the reactive paradigm. The two most known and most formalized are the subsumption (Brooks, 1986; Brooks, 1987) and potential field methodologies. Subsumption refers to how behaviors are combined. Potential field methodologies require behaviors to be implemented as potential fields. Other approaches include fuzzy methods and auction based decision making, but these tend to be implementation details rather than architectural features.

Under the reactive paradigm, systems are composed of behaviors, which tightly couple sensing and acting. Sensing in the reactive paradigm is local to each behavior. A behavior may create and use its own internal world representation, but there is no global world model as with hierarchical paradigm. As a result, reactive systems are the fastest executing robotic system possible.

In terms of support of modularity, representative architectures decompose the actions and perceptions needed to perform a task into behaviors. Behaviors are combined to layers and allow incremental development, as well as provide high level of robustness.

Reactive systems are limited to applications which can be accomplished with reflexive behaviors. They cannot be transferred to domains where robot needs to do

planning, reasoning about resource allocation, etc. In practice, very few of the subsumption levels can be ported to new applications without changes. As with animals, a reactive robot will also do something consistent with its perception of the world, but not always the right thing.

1.1.5. Hybrid deliberative reactive robot control paradigm

The two robot paradigms described in the previous sections developed independently. The hierarchical SENSE-PLAN-ACT approach came earlier, with the Shakey robot being among the first of this species, reactive approach came later. As would be expected, it was only a matter of time before researchers began to investigate hybrid versions of the two types of paradigms. Among the first was (Matarić, 1992b) who added a planning layer on top of the Brooks's multiple reactive layers. From this point of view, the simple structure of reactive architecture can be modified to include higher-level layers that correspond to planning and deliberation. However a number of advocates of reactive architectures dispute the need for such planning and reasoning, suggesting that the robot's environment is the only source of information it needs.

The new challenge for AI robotics at the beginning of the 1990's was how to put the planning and deliberation into robots, but without disrupting the success of the reactive behavioral control. The consensus was that behavioral control was the best way to do low level control because of its pragmatic success, and its elegance as a computational theory for both biological and machine intelligence. At first, hybrids were viewed as an artifact of research, without any real merit for robotic implementations. Some researchers recommended using reactive paradigm if a robot was being designed to operate in an unstructured environment. If the task was to be performed in a knowledge-rich environment, then hierarchical paradigm was preferable. Hybrids where believed to be worth of both, involving the fast execution times of reactivity with the difficulties in developing hierarchical models.

The current thinking in the robotics community is that hybrids are the best general architectural solution for several reasons. First, the use of asynchronous processing techniques allow deliberative functions of execute independently of reactive behaviors. A planner can be slowly computing next goal for a robot to navigate to, while the robot is reactively navigating toward its current goal with fast update rates. Second, good

software modularity allows subsystems of objects in hybrid architectures to be mixed and matched for specific applications.

The organization of hybrid deliberative reactive system can be described as: PLAN, then SENSCE-ACT (see figure 1.9). The PLAN component includes all deliberation and global world modeling, not only navigation or task planning. The robot would first plan how to accomplish a mission (using global world model) or a task, then instantiate or turn on a set of behaviors (SENSE-ACT) to execute the plan or a portion of it. The behaviors would execute until the plan was completed, then the planner would generate a new set of behaviors, and so on.

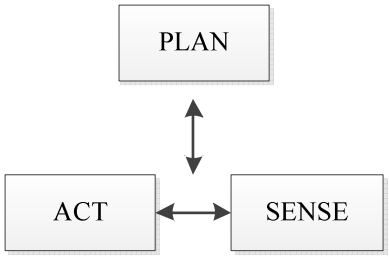


Figure 1.9. **Hybrid deliberative reactive robot control paradigm**

The idea of PLAN, then SENSE-ACT evolved from two assumptions of hybrid paradigm. First, planning covers a long time horizon and requires global knowledge, so it should be decoupled from real-time execution on the software design level. Planning and global modeling algorithms are computationally expensive, so they should be decoupled from real-time execution just from a standpoint of practicality because they would slow down the reaction rate.

The organization of sensing in hybrid architecture is more complex. In the behaviors sensing remains as it was for the reactive paradigm, it is local and behavior specific. But planning and deliberation requires global world models. The model is constructed by processes independent of the behavior-specific sensing. However, both the perceptual schema for the behaviors and the model making process can share the same sensors. Furthermore, the model making process can share the percepts created by perceptual schemas of behaviors or it can have sensors which are dedicated to providing observations which are useful for world modeling but are not used for any active behaviors.

The hybrid paradigm is an extension of the reactive paradigm: their behavioral components are similar. However that is not true. In the reactive paradigm behavior was referred to purely reflexive operation. In the hybrid paradigm behavior includes reflexive, innate and learned actions. Hybrid implementations tend to use assemblages

of behaviors sequenced over time, rather than primitive behaviors, there is more diversity in methods for combining the output from concurrent behaviors.

The deliberative portion of hybrid architecture contains modules and functions for things which are not easy to represent in reactive behaviors. Some functions require a global world model, e.g. mapping and path planning. But other activities require global knowledge of a different sort. Planning which behaviors to use – behavioral management – requires knowing something about the current mission and the current state of the environment. Likewise, performance monitoring uses additional sources of information to see if the robot actually making progress to its goal.

While hybrid architectures vary significantly in how they implement deliberative functionality, what they implement is fairly similar. Generally hybrid architecture has the following modules or objects:

- ✓ a sequencer agent which generates the set of behaviors to use in order to accomplish a subtask, and determinates any sequences and activation conditions;
- ✓ a resource manager which allocates resources to behaviors, including selecting from libraries of schemas, for example, manager selects fittest sensor among IR, sonar or stereo vision devices present on a robot in order to get reliable distance reading for current state of robot;
- ✓ a cartographer is responsible for creating, storing and maintaining map or spatial information, plus methods for accessing the data, which often contains a global world model and knowledge representation, even if it is not a map;
- ✓ a mission planner interacts with the human, transforms the commands into robot terms, and constructs a mission plan;
- ✓ a performance monitoring and problem solving agent allows the robot to notice if it is making progress or not, as well as provides recovery hints.

The primary contribution of a hybrid paradigm is to provide a template for merging deliberation and reaction. Robotic architectures designed according to hybrid robot control paradigm are highly modular. Most are divided into layers, which are then subdivided into modules.

Hybrids tend to have high degree of niche targetability. The addition of the deliberative component allows hybrids to be used for applications not appropriate for purely reactive systems.

Another attractive aspect of hybrid paradigm is that its representative architectures often explicitly attempt to ensure robustness. Modules within the various deliberative

components attempt to monitor the performance of the reactive behaviors and either replace or adapt the configuration as needed.

Major drawback of hybrid paradigm is processing overhead needed for robot control due to planning activities in comparing with purely reactive systems, however this limitation is eliminated by progress in processing hardware.

1.2. Heterogeneous multi-robot systems research domain

Previous chapter describes various aspects of the robot control organization, its paradigms and architectures. However these approaches are applicable for the control of single robot. Robot development methods have evolved further and there appeared suggestions to use several robots simultaneously in order to complete more complex tasks.

During the last decade multi-robot systems research field become one of the most actual directions in the robotics and many researchers have focused on it. Early researches laid the foundation of the multi-robot system functional principles (Balch, Parker, 2002; Parker et al., 2005) and therefore provided a solid base for the following investigations.

The idea to utilize multiple robots comes from biological systems like social insects. Complex social organization and decentralized behavior attended many researchers from various scientific directions, including philosophy, sociology, management, etc. (Oster, Wilson, 1978) Social insects like ants, bees or wasps demonstrate high degree of self-organization and ability to adopt to environment and solve complex problems. Many investigations have been made in order to understand basis of such social behavior in terms of work organization (Jeanne, 1986; Gordon, 1996) or structural parameters (Pamilo, 1991a; Pamilo, 1991b). These features have long been an inspiration and subject of study with aim to apply same approaches in variety of economic sectors.

The word “sociobiology” was introduced to describe the field of social behavior of insects (Wilson, 1971). Insect societies are governed by rigid instincts and appear to have small number of built-in rules for behavior. (Wilson, 2000) draw a conclusion: “Each insect colony is an assemblage of related organisms that grow, competes, and eventually dies in patterns that are consequences of the birth and death schedules of its members.” Despite the fact that there is little if any autonomy and learning, such

societies display amazing collaborative behavior. It is important to note that insect societies are self-controlled and self-regulating systems; there is no manager that issues orders to subordinates. The queen in some insect societies is in fact an egg-bearing machine, with no real authority.

One of significant domains directly inspired by social insects is swarm intelligence, a subfield of artificial intelligence. The roots of swarm intelligence are deeply embedded in the biological study of self-organized behaviors in social insects. Swarm intelligence, as a scientific discipline including research fields such as swarm optimization or distributed control in collective robotics, was born from biological insights about the incredible abilities of social insects to solve their everyday-life problems (Garnier et al., 2007).

The complexity of collective behaviors and structures does not reflect at all the relative simplicity of the individual behaviors of an insect. However, the complexity of an individual insect in terms of cognitive or communicational abilities may be high in an absolute sense, while remaining not sufficient to effectively supervise a large system and to explain the complexity of all the behaviors at the colony scale (Seeley, 2002). In most cases, a single insect is not able to find by itself an efficient solution to a colony problem, while the society to which it belongs finds “as a whole” a solution very easily (Camazine et al., 2003).

Swarm intelligence algorithms are being successfully applied for such tasks as computational optimization (Poli et al., 2007; Hu et al., 2003), routing of traffic in telecommunication networks (Kassabalidis et al., 2001), business management (Bonabeau, Meyer, 2001), distributed sensing systems (Hackwood, Beni, 1992) and many others.

Research on multiple mobile robots has lagged behind research on single robots. Major reason for it is that for many years robot hardware and software was very unreliable and required huge amount of effort to keep single robot working. Over time, robotic systems have become more available and much cheaper. There has been increased research interest in systems composed of multiple autonomous mobile robots exhibiting cooperative behavior.

First in this area is considered to be biologist William Grey Walter (1950), who constructed two electromechanical *Tortoises* (see. 1.1.1). In the 1980s there was significant interest in the control of multiple manipulators, where two robot arms grasp

the same object, such as large panel for automobile assembly (Zheng, Luh, 1985; Hayati, 1986; Koivo, Bekey, 1988).

In the mid-1990s multiple robot control direction began to change quickly. Researchers inspired by the phenomena of social insects focused on development of various algorithms for cooperative control of multiple robots (Beni, J. Wang, 1993; Kube, H. Zhang, 1993). Groups of mobile robots were constructed, with an aim to study such issues as group architecture, resource conflict, origin of cooperation, learning, and geometric problems (Cao et al., 1997).

Arkin and Balch has studied multi-robot communication and navigation during that period and have developed fundamental principles used nowadays (Arkin, 1992; Arkin, Balch, 1998; Balch, Arkin, 1994; Balch, Arkin, 1998). Pioneering work in behavior based multi-robot systems was done by Matarić and her collaborators (Matarić, 1992a; Matarić, 1994; Matarić, 1997). Parker have studied approaches for multi-robot cooperative control (Parker, 1994b; Parker, 1997; Parker, 1999b) and successfully developed multi-robot control architecture (see 1.2.1).

Despite increased complexity in design and development of multi-robot systems, they have various advantages over single-robot systems. Groups of autonomous robots are able to perform tasks that may be difficult, undesirable, or impossible for single robot. Some of them are as follows (Bekey, 2005):

- ✓ explorations in hazardous environments where failure of one robot should not lead to failure of the entire mission and where redundancy may increase the fault tolerance of the colony;
- ✓ tasks beyond the limits of single robots, like cooperative lifting or pushing large and heavy objects or assembly of complex structures;
- ✓ tasks that can be completed more rapidly by multiple robots than possible is for a single robot due to massive parallelism in multi-robot system;
- ✓ complex tasks that may be less expensive with a group of specialized, simpler vehicles that with single, multipurpose robot;
- ✓ highly distributed sensing, in which large colonies of simple and inexpensive robots are used as mobile, communicating sensors.

As shown in previous chapters, control of autonomous robots requires integration of principles from biology, control theory, kinematics, dynamics, computer engineering, and other disciplines. Control of robot group to achieve collective performance requires additional consideration of issues from animal ethology, social psychology,

organization theory, economics, and others. Following chapters describe features of control of multi-robot systems showing specific control problems and architectures and algorithms for solving them.

1.2.1. Control features of robot colonies

Control approaches for multi-robot system are inspired by behavior of social insects. Insects of the society display at least three common characteristics:

- ✓ members of the society collaborate in caring for the young;
- ✓ those members with highest reproductive potential have a higher standing in the society, so workers that are sterile tend to work for the benefit of their more fertile fellows;
- ✓ there is an overlap of at least two generations in the work being done for the colony, so the offspring help their parents during some time in their lives.

These aspects of behavior are not directly applicable to robot societies. There are, however, two additional aspects of organization of insect societies that can serve as models for robot societies as well:

- ✓ insect societies have evolved a large degree of specialization among their workers; there may be as many as ten distinct specializations;
- ✓ to coordinate their activities, insects have developed a surprisingly rich repertoire of communication methods.

The issue of specialization is an important factor for a group robotics. In general, multi-purpose robot will be significantly more expensive than those designed for special task. Detailed discussion on this topic is provided in 1.2.2.

The control of robot colonies required completely different approach in comparison with individual robot control. In general, control of each member of a colony by an external controller is possible only with small groups; it is almost impossible as the colony grows in size, just as it is impossible for a military general to control actions of each individual soldier on a battlefield.

Multi-robot research domain in some sense is similar to another direction of artificial intelligence – multi-agents. Both of these domains fall into a research area of Distributed Artificial Intelligence. Most of issues in organizing teams of robots apply to software agents as well. Murphy (2000a) classify most often cited problems organizing teams of robots.

- ✓ Designing teams is hard. How does a designer recognize the characteristics of a problem that make it suitable for robot colony? How does the designer or the agents themselves divide up the task? Are there any tools to predict and verify the social behavior?
- ✓ There is a “too many cooks spoil the broth” effect. Having more robots working on a task or in a team increases the possibility that individual robots will unintentionally interfere with each other, lowering the overall productivity.
- ✓ It is hard for a team to recognize when it, or members, is unproductive. One solution to the “too many cooks spoil the broth” problem is to try engineering the team so that interference cannot happen. But this may not be possible for every type of team or the vagaries of the open world may undermine that engineering. To defeat itself, the team should be capable of monitoring itself to make sure it is productive. This in turn returns to the issue of communication.
- ✓ It is not clear when communication is needed, and what to say. Many animals operate in flocks, maintaining formation without explicit communication. Formation control is often done simply by perceiving the proximity to or actions of others; for example, schooling fish try to remain equally close to fish on either side. But robots and modern telecommunications technology make it possible for all agents in team to know whatever is in the mind of the other robots, through at a computational and hardware costs. How this unparalleled ability be exploited? What happens if telecommunications links goes bad? Is there is a language for multi-agents that can abstract the important information and minimize explicit communication?
- ✓ The “right” level of individuality and autonomy is usually not obvious in a problem domain. Agents with a high degree of individual autonomy may create more interference with group goals, even to the point of seeming “autistic”. But agents with more autonomy may be better able to deal with open world.

There were made a lot of studies on aforementioned questions and many of them are not answered at this time. In general, researchers are working on control architectures that make application of robot team more and more productive. Likewise in multi-agents, the aim of architecture is to provide a control framework, which will allow each robot to perform concurrent but independent actions, which in turn will lead to emergent social behavior.

The number of most successful multi-robot control architectures is reviewed in next chapters in details. But before particular architectures the general control strategies should be indicated. There are three major types of control strategies (Bekey, 2005) described below.

- ✓ Centralized and hierarchical control. By analogy with control of an army, factory, or an enterprise, each individual robot in this type of control strategy is responsible for small number of other individuals, who in turn control alike number of others, to the bottom level where the actual physical work is done. In this strategy the global goal of the entire team may be known only to the top level of hierarchy. There is clearly a problem if a high-level supervisor is disabled. A lower-level individual must in such case take the place of the disabled leader, just like it happens in military situations. In the group of robotics, this implies that each robot has an internal model of a supervisor sufficient to allow it to take over the work of supervisor who has failed. At present, there is no theoretical basis for enabling a robot to take over the work of another using only internal models, the external input is required.
- ✓ Decentralized and local control. A completely opposite approach is based on allowing each individual to operate on local information while accomplishing global goals. This strategy requires that these goals be contained implicitly within the rules of behavior of each individual. Cooperation is an emergent property of the robot colony; it is consequence of the way in which the robots interact with environment. As biological example, consider a colony of ants building an anthill. It is clear, that individual ants caring leaflet and needles to the anthill do not have a global blueprint of the structure in their brains. Rather, they follow simple rules, such as “Move toward home, go as high as you can, and deposit your load.”
- ✓ Hybrid structures. Some organizations may incorporate both types of aforementioned control strategies, in which some groups are highly autonomous and decentralized, while others operate under central authority.

A lot of information on these strategies is available from a study of human work structures such as factories and enterprises. The organization and control of such groups is the subject of management disciplines (Daft, 2009).

Following chapter describe in details most successful and widely spread architectures for multi-robot systems.

Nerd Herd architecture

One of the earliest studies on the organization of team of robots was performed by Maja Matarić. She investigated both theoretical and experimental aspects of group behaviors. Her experimental work was performed on a collection of twenty identical robots manufactured by *ISX/IS Robotics*.

Since these robots did not possess a lot of intelligence, they were sometimes referred to as the *Nerd Herd*. The robots (see figure 1.10) were *Ackerman* steered bases about 13" long with "forks" that could be used to pick up and stack objects. They had IR and contact sensors on the ends of the forks, bump sensors on the sides and back, and a radio-sonar positioning and communication system. They were controlled by a network of 68HC11 processors programmed in the Behavior Language.

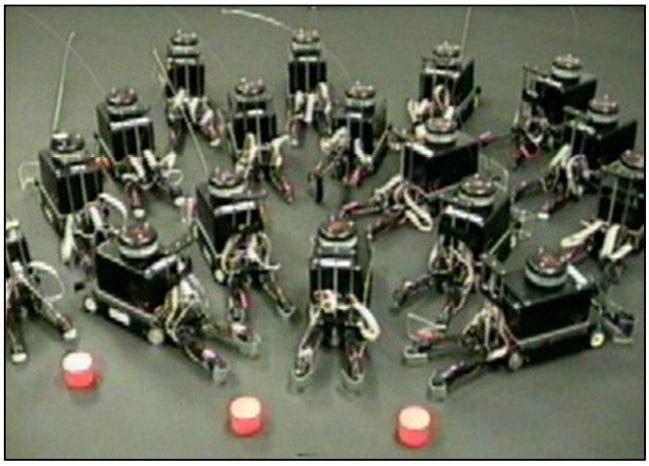


Figure 1.10. **The *Nerd Herd* colony** (Matarić, 2010)

These robots were used to demonstrate large scale group behavior. A number of the interaction primitives was used in experiments:

- ✓ homing – each robot strives to move to common home base;
- ✓ aggregation – robots try to gather together while maintaining a spatial separation;
- ✓ dispersion – robots cover a large area, establishing and maintaining minimum separation between robots;
- ✓ following – robots follow each other;
- ✓ collision avoidance – robots move around while avoiding collisions with obstacles and each other.

These interaction primitives were implemented using rule-based encoding (“if-then-else” syntax). Two different coordination mechanisms are used: direct combination – which is vector summation process; and temporal combination – which sequences through a series of behavior states. Perceptual information is encoded as a series of

predicates (e.g. at-home, have-puck) used to encode the sensory data required to activate the relevant behaviors.

In later research some of the basic behaviors were combined to implement more complex social interactions, such as foraging. In foraging behavior the robots begin to search for food by dispersion. Once robot has located “food”, it begins homing. During homing phase, it may avoid other robots not carrying food but follow others who have it. The group of food-carrying robots then forms a flock. In this way, primitive behaviors can be combined to produce more complex ones. Some other examples are as follows:

- ✓ flocking, consisting of collision avoidance, aggregation and dispersion;
- ✓ surrounding, consisting of collision avoidance, following and aggregation;
- ✓ herding, consisting of surrounding and flocking;
- ✓ foraging, consisting of collision avoidance, dispersion, following, homing and flocking.

A unique feature of research was the use of animal models to develop very simple algorithms for the above behaviors. This simple strategy was successful for *Nerd Herd* because all the robots were alike; it might require modifications for colonies of heterogeneous robots (Matarić, 1992b).

The MissionLab architecture

Another group of researches was led by Ronald C. Arkin, who performed at the University of Michigan. Their topic was focused on application of reactive behavior-based architectures for robot team control.

Behavior-based architectures decompose a robot’s control program into a collection of behaviors and coordination mechanisms, but the visible behavior of robot comes from emergent interaction of these behaviors. The decomposition process supports the maintenance of a library of reusable behaviors.

The MissionLab is a multi-agent mission specification system developed at Georgia Tech, uses an agent-oriented philosophy as the underlying methodology, permitting the recursive formulation of societies of robots. It includes a graphical configuration editor, multi-robot simulation system, and two different architectural code generators. The software system embodies the Societal Agent Theory (Mackenzie, 1996), which describes a society as an agent that consists of a collection of either homogeneous of heterogeneous robots. Each individual robot consists of assemblies of

behaviors, coordinated in various ways. Temporal sequencing affords transitions between various behavioral states is represented as a finite state machine. Coordination of parallel behaviors can be implemented as vector summation, action selection, priority or other coordination operators as necessary. These individual behavioral assemblages consist of groups of primitive perceptual and motor behaviors, represented by physical sensors and actuators of the robot.

Creating a configuration of multi-robot system involves three steps: determining an appropriate set of skills for each of the robots; translating those mission-oriented skills into sets of suitable behaviors (assemblages); and constructing or selecting suitable coordination mechanisms to ensure that the correct skill is deployed for the mission's duration.

The MissionLab architecture has several important features that make suitable for wide range of applications of multi-robot system.

- ✓ The binding to a particular behavioral architecture is delayed until the desired mission behavior is specified. Binding to a particular physical robot also occurs after specification. This permits the design to be both architecture and robot independent.
- ✓ The system has separate software libraries for abstract behaviors, specific architectures and various robots. This allows non-expert users to specify robot missions and automatically configure the software required for coordination and control of a group of robots, either in simulation or in hardware.
- ✓ The system provides multiple levels of abstraction, each of which can be targeted to different specialists. They range from entire robot mission configurations down to the low-level language for a particular behavior.

The MissionLab architecture was developed for effective design and implementation for a mission for a team of robots. Several less effective approaches were developed within the research program (Mackenzie et al., 1997).

ALLIANCE architecture

Another offshoot of the behavior based approach is the *ALLIANCE* architecture developed by Lynne Parker (Parker, 1994b; Parker, 1994a; Parker, 1999a). The goal of the architecture was the development of a fault-tolerant, adaptive, distributed, behavior-based software system for control of teams of robots. Architecture implies special consideration for control of heterogeneous teams of robots. The robots were to

accomplish a mission in a dynamic environment and in the presence of failures in their action selection mechanisms and with noises both in perception and in actuation. *ALLIANCE* has been successful at accomplishing its goals and has inspired other approaches to the control of multiple robots.

The architecture has several features described below. One of the most notable features is that there no centralized control used for coordination of robot team. All the robots are fully autonomous and have the ability to perform useful actions even in presence of failures of other teammates. This makes the robot team very flexible upon to composition of the team.

Another feature of the architecture is that the robots of the team can detect the effects of their own actions and those of other members of the team. Robots detect own actions using variety of sensors and feedback control. The actions of others are detected via an explicit communication. The robots also are able to select appropriate actions during mission, taking into account the environment, their own internal state, and the actions of other robots.

In comparison with single robot behavior based architectures the *ALLIANCE* introduces behavior sets and motivation model that enables particular robot to perform tasks only when those actions are expected to have the effect. Behavior sets correspond to some high-level task-achieving functions. Sets enable different groups of behaviors to be active together or to hibernate, permitting configuration atypical for subsumption architectures. Lower-level behaviors correspond to survival behaviors; higher-order behaviors may correspond to exploration of map-building (see figure 1.11). Layer 0 represents reactive survival actions and is active all the time; Layer 1 may correspond to collision avoidance and also be active continually. Layer 2 corresponds to higher-level competences that are turned on or off by the appropriate behavior sets.

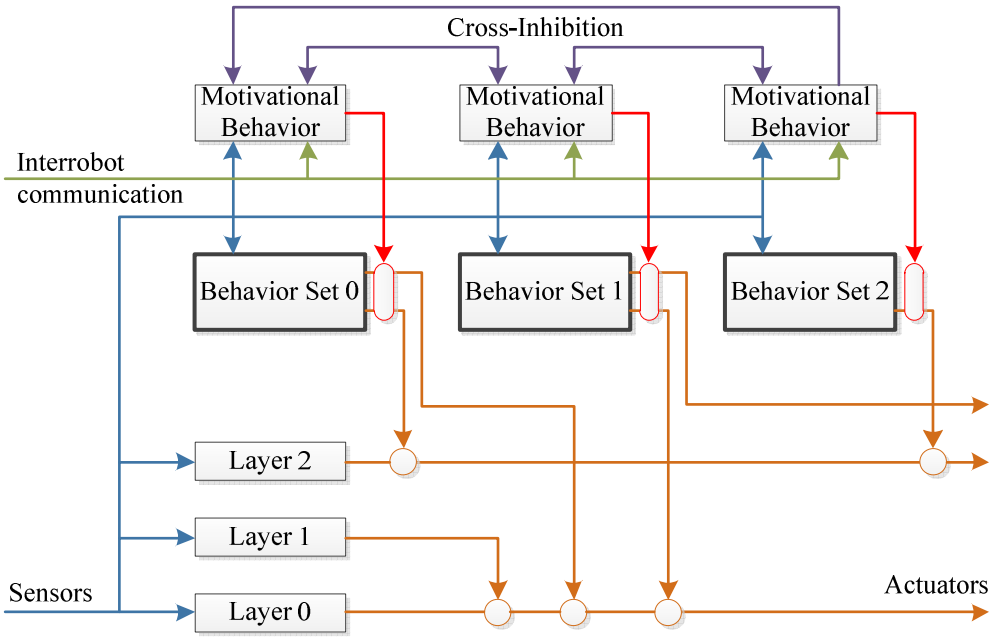


Figure 1.11. **The *ALLIANCE* architecture** (Parker, 1994a)

Motivational behaviors enable or disable these behavior sets. They operate by accepting, in addition to the inputs from sensors and other behaviors, information from communication with other teammates and from own internal motivational state. Specially, robot impatience and acquiescence is modeled. For example, if other robots do not perform some task needed by current robot, it becomes increasingly impatient to take over the needed task and to perform desired actions itself. Similarly, when robot becomes aware that it is not completing its tasks adequately, it will try to give up current task and find other actions to perform.

According to the *ALLIANCE* architecture motivation model is implemented using rate functions. Each robot's (r_i) overall motivation for a behavior set a_{ij} is computed using formula (1).

$$\begin{aligned} & m_{ij}(0) = 0, \\ & m_{ij}(t) = \left[m_{ij}(t-1) + impatience_{ij}(t)\right] \times sensory\_feedback_{ij}(t) \times \\ & \qquad \times\ activity\_suppression_{ij}(t) \times impatience\_reset_{ij}(t) \times \\ & \qquad \times acquiescence_{ij}(t) \end{aligned} \tag{1}$$

where

$impatience_{ij}(t)$ is the impatience rate function that determinates how quickly the robot becomes impatient;

$sensory\_feedback_{ij}(t)$ is a binary predicate that indicates whether the preconditions for the behavioral set are satisfied or not;

$activity\_suppression_{ij}(t)$ is a binary predicate indicating whether or not another behavioral set $a_{ik}, j \neq k$ is active at time t;

$impatience\_reset_{ij}(t)$ is a binary predicate that is 0 when another robot is making progress on the task that the robot is waiting on, and otherwise 1;

$acquiescence_{ij}(t)$ is a binary predicate that determinates whether to give up on a task or not.

Thus, the motivation for a behavior set will continue to grow unless sensor data indicates that it is not needed, another competing behavior is active, another robot has taken over the task, or the robot gives up on the task. When motivation value grows up to an arbitrary defined threshold, the behavioral set a_{ij} becomes active in robot r_i. The robot then periodically broadcasts to all other robots the fact that a_{ij} is active.

The *ALLIANCE* has been used for a wide range of mission scenarios, which include hazardous waste cleanup missions, cooperative multi-robot observation of multiple moving targets (Parker, 1997; Parker, 1999b) and others (Parker, 1998a). To reduce the need for parameter tuning, the architectures has been extended to include a learning mechanism, and the modified version is referred as *L-ALLIANCE* (Parker, 1996; Parker, 1998b). The learning mechanism requires the robots to monitor and evaluate their performance in the changing environment and then update parameters as required.

The pheromone architecture

The pheromone architecture refers to biology where pheromones are chemical markers used by insects for communication, coordination and sexual attraction. Insects follow a pheromone trail in a zigzag fashion, moving across the trail in one direction and then another.

The architecture for coordination of multiple robots inspired by pheromone trail following was developed by David Payton and his associates (Payton et al., 2001). They use so called virtual pheromones, which preserve essential features of biological pheromones as follows:

- ✓ pheromones are locally transmitted, thus there is no need for unique identities that are impractical in large groups;
- ✓ pheromone diffusion gradients provide important navigational signals and also encode useful information about barriers in the environment that block pheromone propagation;

- ✓ pheromones decay over time, which reduces obsolete or irrelevant information.

Virtual pheromones are implemented as simple beacons and directional sensors mounted on top of each robot. They facilitate simple communication and coordination and require little on-board processing. Robot teams coordinated by this architecture are able to perform complex tasks. Some examples are as follows (Payton et al., 2003):

- ✓ gradient following which enables a dispersed robot swarm to guide one or more robots to a particular area or object of interest;
- ✓ "go hide", allowing the entire swarm to run for cover to avoid detection and/or injury;
- ✓ cooperative sensing intended for positive identification of some objects, which requires agreement between two or more robots, either to provide redundancy for fault tolerance, or to raise confidence through cross-correlation.

The robot collective becomes a computing grid embedded within the environment while acting as a physical embodiment of the user interface. The virtual pheromone approach externalizes the map, used in other path planning and terrain analysis methods, spreading it across a collection of simple processors, each of which determinates the terrain features in its locality. Required terrain-processing algorithms are then spread over the population of simple processors. The user interface for this distributed robot collective is itself distributed, and entire collective works cooperatively to provide a unified display embedded in the environment (Payton et al., 2002).

The Ranger-Scout architecture

Another approach for cooperative robot control, inspired by biological marsupials, was developed by Paul Rybski and his associates (Rybski et al., 2000; Stoeter et al., 2002). Approach is based on so-called marsupial robots, in which, by analogy with kangaroos and other marsupials, a larger robot carries one or more smaller robots and then deploy them as appropriate. Researches use heterogeneous team of robots that consists of two types of robots (see figure 1.12).

- ✓ *Scout* – mobile sensor platform, have cylindrical shape (40 mm in diameter and 110 mm in length), uses unique combination of rolling and jumping locomotion.
- ✓ *Ranger* – platform based on *ATRV-Jr*™, can carry payload up to 25 kg, transfers scouts to the deployment site. Ranger acts as communication and coordination hub for scouts.

a b c

Figure 1.12. **Ranger-Scout team members**
a – Scout (http://distrob.cs.umn.edu/scout.php), b – Ranger (http://lrm.isr.ist.utl.pt/rescue/), c – team composition
Source: http://www.wired.com/science/discoveries/multimedia/2004/04/63099?slide=6&slideView=6 (accessed 2012.04.15)

The architecture presented several benefits over other architectures. The small size of scout robots makes them it suitable for several challenging reconnaissance and surveillance tasks. They can be deployed manually or launched to desired environment. Larger utility platform (ranger) carries large number of processing, communication and control. This makes whole system an effective reconnaissance tool.

1.2.2. Heterogeneity in robot colonies

Previous chapters briefly indicate heterogeneity feature of robot colonies. Heterogeneity refers to the degree of similarity between individual robots that are within colony. In general, any robotic system consisting of multiple robots can be classified either as homogeneous of heterogeneous colony. Heterogeneous colony has at least two members with different hardware or software capabilities, while in homogeneous colony the members are all identical. Differences between members of colony could be in any mechanical, sensing or processing hardware, or in internal control architecture (G. S. Sukhatme, 1999). Different robot types are called robot classes. Depending on the level of heterogeneity robots in a colony are classified as weakly or strongly heterogeneous. This feature is referred as diversity of robot colony and there are several metrics proposed for evaluating it, for instance, hierarchic social entropy (Balch, 2000). Moreover, the diversity of the colony can change dynamically, for instance, homogeneous colony can behave according to one model, and then it becomes heterogeneous if several members change the behavioral model.

Homogeneous robot colonies are investigated more widely and are usually referred as robot swarms. Each robot of such colony is identical, which simplifies both

the manufacturing and the software development. The biological model for such systems is social insects, such as ants or bees.

An example of competition for homogeneous teams of robots is *RoboCup* (Kitano, Asada, Kuniyoshi, 1997; Kitano, Asada, Kuniyoshi, et al., 1997). A lot of researchers were challenged by it. As a result many successful approaches were demonstrated during this competition, which in turn were used as models for real industrial solutions.

Heterogeneous robot colonies are relatively new trend in multi-robot systems. Active researchers assume that within next few decades robot colonies will consist of heterogeneous robots because of various advantages over homogeneous swarms (Kiener, Stryk, 2010). Also interaction between household devices from different manufacturers and different generations becomes a usual practice (e.g. PC, TV, mobile phone, climate control, etc.). This refers to so-called ambient intelligence paradigm, which describes electronic environments that are sensible and responsible to the presence of people. In an ambient intelligence world, devices operate collectively using information and intelligence that is hidden in the network connecting the devices. Lightning, sound, vision, domestic appliance, and personal health care products all cooperate seamlessly with one another to improve the total user experience through the support of natural and intuitive user interfaces (Aarts, Wichert, 2009).

A common heterogeneous colony arrangement is to have one member with more expensive computing hardware. That robot serves as the colony leader and can direct others, less intelligent robots, or it can be used for special situations. The drawback of such design is that failure or destruction of the leader will prevent the team mission from being accomplished.

Aforementioned drawback is eliminated in highly distributed robotic colonies, where each individual robot is independent, although highly specialized, unit which selects tasks to perform by itself according to its goals, internal state and/or environment. Such colonies are capable to establish workgroups for particular tasks and then rearrange the groups when other tasks are more relevant.

Despite increased design and production costs (each robot class should be designed and produced separately) heterogeneous robot colonies have several advantages over homogeneous colonies described below.

- ✓ Fault tolerance – the ability of the robot colony to respond to individual robot failures or failures in communication that may occur at any time during the

mission. The colony as a whole is able to complete its mission to the greatest extent possible in spite of any single-point failure (Parker, 1998b). A critical phase of fault tolerance is the ability of the system to diagnose the correct failure state. (Parker et al., 2004). Fault tolerance usually improved using such methods as dynamic task allocation (Duffie et al., 1988) or self-stabilizing (J. El Haddad, S. Haddad, 2004).

- ✓ Robustness refers to its ability of the heterogeneous robot colony to complete the mission even in cases of certain failures. Robustness is obtained because of highly redundant solutions available in such colony. Critical functionality of the system is distributed over many simple units, thus any of them is easily replaced in case of failure. Also missing functionality can be obtained by combining functions of other units. The mission itself could be solved in totally different way in case if some functionality is unavailable (Hazon, Kaminka, 2008). Heterogeneous robot colonies demonstrate high degree of adaptively.
- ✓ High versatility, which demonstrate heterogeneous robot colonies in performing complex tasks. Due to dynamic task allocation among the individual robots, the colony as a whole is able to perform wide range of complex tasks (Simmons et al., 2001). Emergent behavior of the colony is not limited to the functionality of particular robots, but is obtained by combining simple functions.
- ✓ Increased utilization of particular components of the robotic system is the aim of several optimization investigations (Stoeter et al., 2002; Dias, Stentz, 2003). The required capability of the colony to perform certain function can be precisely calculated based on mission specification and redundancy requirements. Thus it is possible to plan behavior of the colony in such way, that particular functions are utilized almost for 100% of time, contrary to homogeneous colonies where the functions (components) are always available for robots, but are used only on demand.
- ✓ High performance of the colony in finding the solution is achieved due to massive parallelism. The colony of robots can dynamically compose subgroups working on particular tasks, many of which can be performed in parallel. Thus a whole system can adopt its capabilities to demands of the current stage of the mission.

Various aspects of heterogeneous robot colonies were already briefly described in previous chapter. This includes control architectures, suitable both for homogeneous and heterogeneous colonies. One of the most investigated combination of robots is

autonomous air and ground vehicles (Grocholsky et al., 2006). Another special case of a cooperative, heterogeneous colony of robots are marsupial robots (Murphy, 2000b).

1.3. Formal analysis problem of specification of multi-robot system

Previous sections describe various control aspects of robotic systems and their development peculiarities. This section is intended to show current research directions in the field of robotic colonies and address the problem solved in current thesis.

1.3.1. State-of-the-art of multi-robot research domain

Multi-robot research domain has progressed since first investigations aimed to develop approaches to control multiple robots at the same time. Leading groups of researchers have distinguished several research sub-directions in multi-robot domain (Arai et al., 2002; Garnier et al., 2007). This chapter provides a review of most recent researches done in these directions and in the multi-robot domain as a whole.

Biological inspirations

A lot of features of multi-robot systems are inspired by biological analogues. This includes behavior and communication models for individual robots, as well as emergent cooperative performance of a whole robot colony.

The most common application of biological knowledge is the use of the simple local control rules of biological societies (e.g. ants, bees, birds, etc.) to develop similar behaviors in cooperative robot systems. Researches in this direction have demonstrated the ability of multi-robot system to flock, disperse, forage, and follow trails. The dynamics of ecosystems has also been applied to the development of multi-robot teams that demonstrate emergent cooperation. Behavior of higher animals such as wolf packs also has been used in researches, especially for predator-prey systems modeling (Madden et al., 2010).

Most recent researches inspired by biological systems demonstrate advanced methods used for cooperative control of robot colonies. One of the widely investigated topics is self-organization of distributed multi-robot systems. When the number of robots becomes large, traditional approaches that rely on a centralized management of the robots' activities and on excessive information exchange rapidly reach limits, for

instance, because of the risk of individual failure or of limits in the communication bandwidth.

Nouyan and his colleagues investigate the conditions under which multi-robot system can “emerge” in an intelligent system (Nouyan et al., 2009). The research is aimed to development of effective teamwork organization using self-organizing processes, which is typically demonstrated by vertebrates.

Ducatelle and his colleagues address self-organization approach to heterogeneous robot swarms (Ducatelle et al., 2010). The key component of their approach is a process of mutual adaptation, in which some robots execute instructions given by others, and at the same time regulatory robots observe the behavior of “workers” and adopt the instruction they give. The result of research show that that this process allows the system to find a solution in a cluttered environment.

There are researches related to so-called digital hormones (Hamann et al., 2010; X. Li et al., 2010). The bio-inspired reaction-diffusion mechanism of hormones is used to control actions of single robot in cooperative environment. By the regulation of hormones, robots adjust their behaviors in time and improve the ability of self-organization.

As there is an increasing interest for household robots observed, the development of sociable, communicative humanoid robots becomes actual research topic. Researchers endow robots with expressive non-verbal behaviors, such as gestures, which are typical for high animals (Salem et al., 2010). Results demonstrate the ability of humanoid robot to produce synthetic speech and expressive gesture at runtime, while not being limited to a predefined repertoire of motor actions in this.

Another example of inspiration from high animals (humans in fact) is RoboCup challenge. The official objective of it is as follows: “By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rules of the *FIFA*, against the winner of the most recent World Cup.” (RoboCup, 2012). The challenge promotes new researches in the directions of robotics and AI, by offering publicly appealing, but formidable challenge.

Communication

Communication in multi-robot systems has been studied extensively since the inception of distributed robotics. Early researches demonstrated univocal benefit of the communication for the performance of the system (Balch, Arkin, 1994). Distinctions

between implicit and explicit communication are usually made, in which implicit communication occurs as a side-effect of other actions, whereas explicit communication is designed to convey information to other robots on the system. Various interaction approaches has been studied and compared in the context of multi-robot systems: interaction through the environment, interaction through sensing, and interaction through communication (C. Jones, Matarić, 2005). Since the benefit of the communication has been proved, researchers focus on novel approaches for improvement of communication effectiveness.

One of the actual directions in multi-robot communication focuses on representations of languages and the grounding of these representations in the physical world. Fundamental aspects of communication and language are described in details (Nolfi, Mirolli, 2010). Novel researches focus on abstraction from communication realization aspects to the goals of communication (Agüero, Veloso, 2012).

Another significant direction is oriented on improvement of fault tolerance in multi-robot communication. This includes various self-stabilizing communication protocols (J. El Haddad, S. Haddad, 2004) and algorithms (Cornejo, N. Lynch, 2010). Researches related to topics of limited communication also could be referred as fault-tolerance. Investigations demonstrate novel approaches in communication link chains (Jung et al., 2010; Arrichiello et al., 2010) and reactive robot behavior to communication availability (De Hoog et al., 2010).

Because of increasing distribution of robots in everyday life very wide research direction becomes essential – robot-human interaction – which also could be referred to communication issues. Alami and his colleagues indicated actual topics and challenges in this direction (Alami et al., 2006). Researchers investigate human-robot collaboration and focus on such fundamental aspects as “what” and “when” to communicate (Kaupp et al., 2010). The problems of multi-robot/multi-human collaboration also has been studied (Whetten, Goodrich, 2010; Lackey et al., 2011). Another novel researches are focused on such aspects as safety (Haddadin, 2011) and implicit communication topics in multi-robot/multi-human communication (Clair et al., 2011).

Multi-robot communication research direction has been grown widely, more detailed analysis of present situation in in this direction, as well as future development demands are described in (Y.-Q. Zhang, 2010).

Architectures, task allocation and control

There is a direction in distributed robotics research which focuses on the development of architectures, task planning capabilities, and control. It addresses the issues of action selection, delegation of authority and control, the communication structure, heterogeneity versus homogeneity of robots, achieving coherence amidst local actions, resolution of conflicts, and other related issues.

Fundamental researches in this field were presented in section 1.2.1, which describes basic control principles of multi-robot systems. First researches were oriented more on conceptual realization of multi-robot control, while novel investigations cover more general aspects of intelligent control.

One of the widely investigated aspects is fault-tolerance of multi-robot system. Novel approaches are based on sensory information analysis for fault detection (Xingyan Li, Parker, 2008). Methods from the field of artificial intellect are widely used for information processing (Azuma, Karube, 2010; Portugal, Rocha, 2010).

Due to the increasing processing power and finer sensing capabilities of robots, researchers are able to deploy robot colonies into unknown, dynamic environments. The exploration of an unknown environment itself is challenging task, especially if the terrain have complex morphology (Renzaglia et al., 2011). Researchers also consider environments that are hazardous for robots (Schwager et al., 2011), which means that there are adversarial agents in the environment trying to disable the robots or that some regions of the environment tend to make the robots fail. Also world modeling facilities have been progressed. Novel methods allow robots to build the model of dynamic environment in real-time using their own limited sensing, known models of actuation, and the communicated information from others (Coltin et al., 2010).

The issues related to task allocation within robot colonies were actualized because of increased processing power and capabilities to implement more complex behavior. Novel researches in this direction include such aspects as decomposition of global mission into tasks for individual robots (Francesca et al., 2011) and subsequent synchronization of these tasks (Karimadini, H. Lin, 2010), task allocation strategies for unknown and dynamic environments (Jeon et al., 2011; L. Jain et al., 2011).

Successful application of robot colony depends on effective planning of its tasks. Novel researches demonstrate applications of various planning approaches for such facilities as path and task allocation planning (Chuang et al., 2010; Jolly et al., 2010),

teamwork and learning planning (Veloso, 2012), planning in large scale colonies (Velagapudi et al., 2010).

Another important aspect of robot colonies investigated in recent years is maintenance features. Many researchers have focused on autonomous mobile robots because in the field of industrial automation each robot is required to work autonomously in response to demands given on site. However, robot failures and maintenance might affect fault tolerance and performance of robotic systems (Satoshi Hoshino et al., 2011). Also the behaviors of individual robots of the colony are changing taking into account maintenance activities (S. Hoshino et al., 2010).

Cooperative performance

An essential and at the same time important direction in multi-robot research is cooperative performance related features. It includes variety of algorithms, methods and approaches for cooperative localization, mapping and exploration, object transportation and manipulation, motion coordination, etc. This direction is widely represented by researches and novel results, which consider both improvements of existing methods as well as completely new approaches.

Localization, mapping and exploration field is represented by such researches as indoor and outdoor localization (D. Kim, Choi, 2011), algorithms for retrieved localization information processing (Elor, Bruckstein, 2010; Prorok, Martinoli, 2011), approaches for massively distributed exploration (Dellaert et al., 2010) and others. Also there are investigations focused on improving of intelligent localization (Pinheiro, Wainer, 2011) and on novel paradigm development (Haumann et al., 2010).

Multi-robot manipulation and transportation tasks usually require high degree of cooperation between robots and coordination of colony becomes nontrivial if more than 2 robots are involved into process. Recent researches demonstrate novel methods for intelligent control of manipulation and transportation tasks (Simzan et al., 2011; Y. Wang et al., 2011). Also relatively new direction is 3D transportation which is usually implemented using aerial vehicles (Michael et al., 2011). Also improved design of actuators is proposed by several researchers (Campbell et al., 2011).

Motion coordination field includes various methods for navigation (M. Yang et al., 2010) and formation control (Beer et al., 2010; P. Chen et al., 2011; de Denus et al., 2011) of multi-robot system. Also researches focus on fundamental problems such as describing the formation rigorously in a mathematical manner (Ma et al., 2011).

Self-reconfigurable robots

Relatively new direction of modular robotics has advanced from proof-of-concept systems to physical implementations and simulations. The field of modular self-reconfigurable robotic systems addresses the design, fabrication, motion planning, and control of autonomous kinematic machines with variable morphology. Beyond conventional actuation, sensing, and control typically found in fixed-morphology robots, self-reconfigurable robots are also able to deliberately change their own shape by rearranging the connectivity of their parts in order to adapt to new circumstances, perform new tasks, or recover from damage (Yim et al., 2007).

Technological exploitation of self-reconfigurable robots provides different practical advantages not only for advanced robotics, but also for autonomous and adaptive systems in general. Three most important advantages are extended reliability, advanced adaptivity and self-evolving properties. Kernbach and his collegues describe most actual research dirrections in the field of self-reconfigurable robots and introduce current challenges (Kernbach et al., 2010).

Researchers consider advantages of homogeneous and heterogeneous self-reconfigurable robotic systems (Kernbach et al., 2011). Homogeneity and heterogeneity provide different advantages and represent two opposite points on the scale of universality and specialization. For instance, homogeneous elements can be easily replaced, such systems are more scalable. However a heterogeneous system benefits in computational and energetic aspects, as well as in reliability of the whole system.

Papers are presenting practical application concepts for self-reconfigurable robotic systems. For instance, Sprowitz and his collegues worked on Roombots project, which aims on development of interractive, shape changing furniture (Sprowitz et al., 2010). They deal with distributed control implementation fundamentals for such systems.

Robot swarms

Swarm robotics could be selected into separate research field, which focus on a coordination of large numbers of relatively simple robots, in contrast to ordinary multi-robot systems. Swarm robotics is a novel approach which takes its inspiration from social insects.

There are scientific papers reporting results in various aspects of the swarm robotics, like emergent behavior of the swarm (Trianni et al., 2010; Şahin, Winfield, 2008), physical design of swarm robots (Seeni, Schafer, 2010) or reconfigurable

systems called multi-robot organisms (Kernbach et al., 2010). Robot swarms are being applied in various fields (Şahin, 2005), including medicine (Davies, 2010; Rogozea et al., 2010), everyday life (Hansen et al., 2010; Swangnetr et al., 2010), urban search and rescue (Wong et al., 2011) and others.

1.3.2. Development trends in multi-robot system research domain

Most of the researches in multi-robot systems field had a trend to focus on the development of working solution for a particular task. For now there are available control architectures, communication strategies or approaches developed to use in multi-robot system (Burgard et al., 2005; Nouyan et al., 2009; Rybski et al., 2007). From the other side there are relatively few formal models and analytical solutions that describe specific type of problem (Gerkey, 2003).

Because of aforementioned assumptions the analysis of economic benefit and/or structural design of a multi-robot system are not performed. Next chapters show several adjacent research directions dealing with these aspects.

Mission implementation using heterogeneous robot group can reduce costs by increasing utilization of particular components of robotic system. In this case the space of possible solutions expand dramatically due to new dimension of parameters – types of robots – added to the scope of choice of robot specification and their number in a homogenous group. The same combinatorial explosion is typical for almost all combinatorial tasks, like chess solving or travelling salesman problem. Therefore often just several intuitive solutions are analyzed and the best of them is considered as optimal. The author aims to search the optimum in the full space of solutions applying formalization of the specification, feasibility analysis and computational power. Through proposed procedure optimum is found in full solution domain eliminating application of suboptimal solutions. The optimization procedure is divided in eight consecutive steps.

1.3.3. Industrial robot selection problem

There are relatively few formal approaches intended to analyze various aspects of robotic systems from economical and/or industrialist's point of view. Although, these aspects are being developed in the field of industrial automation and the researches are focused on an industrial robot selection problem.

The base for this direction is laid in the field of multiple-criteria decision analysis, which is a sub-discipline of operations research that explicitly considers multiple criteria in decision-making environments. Typically it deals with conflicting criteria that need to be evaluated in making decisions. Multiple criteria decision making discipline started in early 1960s and there have been important advances in developing algorithms and tools for decision making (Zionts, 1979; Oppenheimer, 1978; Yakowitz, 1993). Modern multiple criteria decision analysis field is independent research direction providing various approaches to other fields (S Zanakis, 1998).

Economic benefit of an industrial company depends on forethought deployment of an industrial production system. Robotic systems are used to increase effectiveness of the production system providing variety of automation approaches. Industrial robots are used extensively in advanced manufactures to perform repetitious, hazardous tasks with precision. The successful deployment of industrial robotic system increases effectiveness of production system and opens potential for improving economic benefit of company. Industrial robot selection problem stands for the process of selecting the most suitable robot among many alternatives involves robots' performance in a number of key areas. Various quantitative methods have been proposed as an aid to selection decision on the choice of robots (Khouja, 1995; Parkan, 1999).

Robot selection problem become actual at the same time when robotic systems have spread in the industries. In 1980s number of investigations was performed in order to develop evaluation procedures for industrial robots (P. Y. Huang, Ghandforoush, 1984). Advanced decision support and robot evaluation models were proposed during that period (Imany, Schlesinger, 1989; Knott, Robert, 1982).

Special systems were developed to support decision making in robot selection (M. S. Jones et al., 1985). Offodile, Lambert (1987) developed computer aided procedure for industrial robot selection, which opened an opportunity for application of complex and more intellectual evaluation models. Successful application of these procedures was demonstrated in various fields (Offodile, Johnson, 1990).

Aforementioned computer aided decision support systems and evaluation models evolved further and utilized various approaches from adjacent research direction. Some of them are described below.

The operational competitiveness of a production unit where resources are transformed into outputs of goods and services is a very important component of its overall competitiveness. The efficiency with which these activities are carried out

determines the operational competitiveness of a production unit. (Parkan, 1994) indicated the necessity of a reliable rating system in order to control and improve operational competitiveness. He worked on the procedure for transparent and robust evaluation, which could reflect prevailing managerial perspectives and competitive priorities.

Parkan & M.-L. Wu (1999b) have made a valuable effort to this direction and proposed the operational competitiveness rating analysis (OCRA) method for evaluation of the performance of production units. They demonstrate the application of this method in variety of industries such as finances (Parkan, M.-L. Wu, 1999a), software development (Parkan et al., 1997) and others. In their method, ratings are used for measuring the performance of production unit. Ratings depend upon the assumptions made by the modelers for the performance evaluation model, and thus can be somewhat subjective. The problem with the OCRA method is that all the costs (inputs) and revenues (outputs) must be measured in a single measurement since the cost/revenue ratios must be known in this model. The OCRA method assumes that the category with a higher cost will receive a higher weight, other things being equal. In contrast, (S. Wang, 2006) contradicts their results and using several examples show that the premise and assumptions of OCRA method are flawed and invalid.

Neely (1999) show the importance of performance measurement in the management of any organization. He contributed in investigation of the design of performance measurement system (Neely et al., 2000) and offered a research agenda (Neely et al., 2005). The analysis and practical applications of performance measurement methods is widely known in research community (Tangen, 2004).

Aforementioned performance or competitiveness evaluation methods are used in combination of decision making tools in order to interpret evaluation results in an appropriate way. Decision-making problem is the process of finding the best option from all of the feasible alternatives. In almost all such problems, the decision maker wants to solve a multiple criteria decision making problem. Multiple criteria decision making may be considered as a complex and dynamic process including one managerial level, which defines the goals, and chooses the final “optimal” alternative. One of the multiple attribute decision making tools widely used in industrial robot selection is called technique for order preference by similarity to ideal solution (TOPSIS). The main principle of TOPSIS states, that the chosen alternative should be as close to the ideal solution as possible and as far from the negative-ideal solution as possible. The ideal

solution is formed as a composite of the best performance values exhibited by any alternative for each attribute. The negative-ideal solution is the composite of the worst performance values. TOPSIS is being constantly improved and new applications of its extensions are being developed in variety of fields such as fuzzy environments (C.-T. Chen, 2000; Izadikhah, 2009; Jahanshahloo, Lotfi, 2006), interval data processing (Jahanshahloo, Lotfi, 2006), robot selection (Chu, Y.-C. Lin, 2003).

Despite solid research base described before, robot selection problem is in the agenda of current researches, and new methods, approaches and improvements are proposed. Researchers give special attention to real-time automation systems and robot selection for them (Chatterjee et al., 2010). Another direction of researches deals with fuzzy methods, which bring more intellectuality to robot selection domain. Papers propose robot selection method based on fuzzy digraphs (Koulouriotis, Ketipi, 2011), decision models based on fuzzy linear regression (Karsak et al., 2011). Others develop extensions that allow application of various methods in fuzzy environments (Devi, 2011).

The development of decision making procedures is performed within the researches related to robot selection. Novel methods are being proposed for decision making support in general manufacturing environments (Rao, Patel, 2011; Chakraborty, 2010), as well as for specific robot selection problem, taking into account subjective preferences (Rao et al., 2011) or applying nonstandard approaches (R. Kumar, Garg, 2010).

Aforementioned analysis of the literature shows that robot selection problem is actual research domain. Taking into account peculiarities of the industry the investigations consider selection of the best solution among the options available on the market. Within the thesis the author threats this as a drawback and supposes that for the industrial robots the level of details of analysis can be refined down to the selection of functional components of robots like mechanic arm, wheel drive, vision system and similar ones instead of ready-made solutions. The analysis and development of the author's proposed approach is described in next sections.

1.3.4. Coalition formation problem in multi-agent research domain

Robot selection problem domain is not the only research direction dealing with complex multi-attribute optimization and decision making. Another similar direction

dealing with complex combinatorial considerations is coalition formation problem in multi-agent research domain.

The coalition formation is a term from sociology meaning the situation where coalition parties join their resources in order to gain outcomes (Gamson, 1961). Coalitions occur when three or more persons are involved, two or more act as a unit against at least one other, and the joint action produces a result superior to any result possible by individual action (Shaw, 1971).

Originally coalition formation between agents was studied in the scope of economic processes (Kelso Jr, Crawford, 1982; Kirman et al., 1986). Advances in computational intelligence brought new approaches and methods. Among them was agent based systems used for advanced computation, and the coalition formation between agents become one of the priority directions.

Many researchers focused on coalition formation as on a control approach for multi-agent systems (Zlotkin, Rosenschein, 1994; Sandholm, Lesser, 1995; Shehory et al., 1998). The number of methods was developed for task allocation via coalition formation (Shehory, Kraus, 1995; Shehory, Kraus, 1998). Eventually coalition formation has become a key topic in multi-agent research domain, as a result various classifications were proposed (Lau, L. Zhang, 2003).

Scientific papers reported application of intelligent multi-agent systems in variety of fields. Multi-agent systems in general and coalition formation of the agents were widely used in economics for trading applications (Yeung et al., 1999; Lerman, Shehory, 2000) . Another application direction of multi-agent systems is optimization and planning in various domains. The examples include the applications for power transmission planning (Yen et al., 1998; Contreras et al., 1998; Zolezzi, Rudnick, 2002).

There is a direction in research area of multi-agent systems which aims to finding optimal coalition of the agents for particular mission (Service, Adams, 2010). Similar approaches can be transformed to robotics problem domain (Vig, 2008). However the coalition formation is performed on operational (run-time) level, while authors propose the optimization approach for design stage of a robotic system.

1.3.5. Optimization of specification of heterogeneous multi-robot system

For a customer of multi-robot system implemented to perform certain task one of the major indicators are costs of such system. In case of other criteria they usually can

be transformed to be measured as costs or benefits in units of money. The number of robot classes, as well as the specification of functions of each class and the number of instances of each class in the system are the parameters of the system that could be adjusted in order to optimize the costs of the system. In practice mentioned parameters are usually predefined and optimization potential is not assessed. As a result multi-robot system becomes unattractive for the customer because of lack of clear positions of costs and predictable results of adjusting the parameters of system.

Clearly, the required behavioral performance in a given application dictates certain constraints on the physical design of the robot team members. However, it is also clear that multiple choices may be made in designing a solution to a given application, based upon costs, robot availability, ease of software design, flexibility in robot use, and so forth. Designing an optimal robot team for a given application prior to deployment requires a significant amount of analysis and consideration of the tradeoffs in alternative strategies. The idea of the optimal team design is to engineer the best robots for a particular application in advance, and then apply those robots to the application with a certain solution strategy in mind (Parker, 2003).

1.4. Summary of the section

The analysis of actual trends in multi-robot research field reveals that multi-robot systems are not being investigated in the context of its formal design and evaluation. The author clearly sees potential for improving multi-robot system at the design stage of the system if its specification is evaluated and selected according to defined criteria. Adjacent fields of research are identified dealing with industrial robot selection problem and with coalition formation problem in multi-agent systems.

There is a lack of investigations aimed to analysis and prediction of utilization of various functions of the multi-robot system. Properly designed system requires fewer investments from a customer and at the same time it is capable to demonstrate performance and fault tolerance required for completing the task.

The behavioral decomposition used within reactive robot control architectures improves results and if properly implemented leads to decreased response time of the control system and opportunity to use it for real-time applications.

2. PROCEDURE FOR OPTIMIZATION OF SPECIFICATION OF MULTI-ROBOT SYSTEM

As it was concluded in previous chapter members of multi-robot system and their functions usually are selected intuitively from available options, without explicit proofing of the choice. The author found that economic efficiency of the multi-robot systems is barely investigated in general and the configuration optimization problem in particular (see 1.3).

This chapter introduces the author's proposed approach for solving aforementioned problem – specification optimization procedure. First of all fundamental terms and the optimization task are defined, including parameters, criteria and constraints. Then conceptual model of solution is presented in details. Finally the steps of specification optimization procedure are introduced, which are described in details in subsequent chapters.

2.1. Definition of multi-robot system specification

As stated before the research made within the thesis is related to analysis of configuration of multi-robot system. The author also uses term specification with the similar meaning. So first of all the object of research should be explicitly and clearly defined.

Consider an autonomous robot system, which consists of single mobile device that is able to get some information about the environment and is able to interact with it. Such systems have a bunch of different parameters that could be used to describe it. Mechanic engineer would describe it by using such parameters as wheelbase, track, type of gearbox, bearings, etc. Electrician would describe types of used links, boards, amperages, voltages, etc. From IT specialist's point of view most important features would be architecture of used software, algorithms, special know-hows, etc.

Now consider the system that consists of several mobile robots. In addition to recent parameters the system could be described in terms of composition and organization of a robot group. The system could consist of identical robots (homogeneous) or of different types of robots (heterogeneous). The control of the system could be organized rather in centralized or distributed manner. Also the behavior

of members of the systems could be cooperative or competitive against each other depending of application peculiarities.

All aforementioned parameters are precisely describe the system, but most of them are highly specialized and are useless for specialists from adjacent fields. Also many of them depend on each other, for instance, if one parameter is defined, that the other one should also be defined, or should be ignored. Such situation complicates the creation of formal approaches for processing and analysis of the system. The author uses simplified model of parameters used for definition of robotic system that is described in details in section 3.1 of the thesis.

The set of parameters of the system uniquely define the system at a given level of details. The author uses a term specification of a system to state such set of parameters. Specification is a detailed description or assessment of requirements, dimensions, materials, etc., as of a proposed building, machine, bridge, etc.; a particular item, aspect, calculation, etc., in such a description; a detailed precise presentation of something or of a plan or proposal for something; a document describing how some system should work (Specification, 2011). In the scope of robotic systems this term is used to name a process of robot's task definition – mission specification (Ulam et al., 2010; Endo et al., 2004). Others use the term to declare detailed formal description of robot controllers (Zieliński, Winiarski, 2010).

The author uses definition matching aforementioned options. Thus, within the scope of the thesis a *specification* of robotic system is a set of parameters that uniquely specify the system. If any of parameters from the set changes, then such specification of system is considered as different in comparison to initial.

Thereby different specifications are obtained varying parameters of the system. Since a specification is a set of all relevant parameters of the system, it could be used to formally analyze the system as a single entity.

In the scope of heterogeneous multi-robot systems the specification defines types of robots (classes) as well as a number of instances of each class of robots in the system. An important feature should be mentioned: if two specifications define the same types of robots but the numbers of instances of these robots are different, then the specifications (and the systems themself) are considered as different.

2.2. Optimization task definition

According to the definition, an *optimization* is a mathematical technique for finding a maximum or minimum value of a function of several variables subject to a set of constraints, as linear programming or systems analysis (Optimization, 2012).

Optimization task stands for the task of finding the best solution, which includes selection or development of mathematical description (model) of possible solution domain, mathematical definition of optimization criteria and finding the optimal solution.

Within the scope of the thesis the optimization task is aimed to find best specification of a multi-robot system maximizing an objective function. This means the searching for an optimal solution in a full space of possible solutions. There are no limitations regarding heterogeneity of the specification. Therefore the optimal specification can correspond to homogenous or heterogeneous multi-robot system. For a heterogeneous multi-robot system possible solutions include all combinations of robot types and number of their instances.

Aforementioned objective function is analyzed in details in following chapters. In general it depends on a mission for robotic system and on results that user is expecting from the optimization. *Mission* stands for a global goal of a robotic system, which performs certain tasks thereby achieving the goal (Mackenzie et al., 1997). Thus optimization of a specification of the robotic system is performed subject to defined mission. It is worth mentioning that the robotic system should be able to complete the mission, because otherwise it is not reasonable to consider the optimization of the specification.

The goal of optimization is defined by user and it depends on his desire. Following chapters explain this aspect in details. In general user may have an aim to reduce the expenses of a robotic system, or to increase productivity (mission fulfillment speed) or even lower ecological impact of production.

The introduction of current section stated that the aim of the thesis is related to analysis of various specifications of a multi-robot system and to selection of most suitable option for particular case. In other words the aim of thesis is to develop a method for specification optimization of multi-robot system. The author uses the term “specification optimization” in the meaning of the process by which the optimal specification of multi-robot system is selected among other less suitable specifications.

Thus the specification itself is not optimized in conventional sense of function optimization. It is closer to meaning of the selection of a best element from some set of available alternatives and therefore the solution cannot be obtained analytically. Various specifications of multi-robot system are analyzed and the most suitable is selected in the process of optimization.

At the same time the procedure does not guarantee finding global optimal solution due to its complexity. Although, various approaches are used through the procedure in order to iteratively guide search towards better solution.

An optimization process implies that **optimization criterion**, **parameters** and **constraints** are defined. A *criterion* of an optimization task is a mathematical function, which is used to evaluate values of variables with intent to assess solution candidate according to defined goal.

Before selecting criterion for specification optimization of multi-robot system the usage peculiarities of the proposed method should be described. Robotic system has a wide range of parameters that are about to be optimized. It includes almost all aspects of robotic system, such as physical design, navigation, task allocation, communication, planning, control strategy, and a lot of others. There are separate research directions for each of these, and a lot of successful results were already achieved (see 1.3.1 for more details). Any of these directions has actual topics for separate thesis. However these topics are extremely technical and are barely could be useful for end user of the system.

In opposite, the thesis is aimed to the analysis of the multi-robot system from customer's perspective. Because of that several assumptions are declared. First of all, implementation of particular components of robotic system is not considered in the scope of specification optimization. For instance, if the system is capable to perform task planning, then it is assumed that the used planning algorithm is suitable to perform the planning as well as that the algorithm is the best for the particular case in comparison with other algorithms. In other words the control and coordination including communication are ideal and works without delay. Such approach allows the analysis to be focused on parameters of the whole robotic system, rather than on parameters of system's components.

The optimization of specification is aimed on mathematical evaluation. Thus non-quantitative parameters of the robotic system are not considered in scope of the thesis. For instance, an appearance of robots or their esthetical view is out of the scope.

Thereby there are a limited number of optimization criteria which correspond to aforementioned scope. In general, the parameters of the robotic system could be associated with a number of categories.

- ✓ Time related parameters, which include any indicators, which could be expressed as a time. This category includes such parameters as time, required to perform a mission, downtime, task switching speed, productivity, etc.
- ✓ Parameters related to energy consumption indicate any values that are consumed by robotic system during the mission. This includes electricity and fuel consumption, as well as raw material consumption for production and others.
- ✓ Other parameters include such quantitative parameters as CO_2 output, soil packing, noise, etc.

As it is shown above, multi-robot system has a bunch of parameters subject to the optimization with quite different measurement units. In order to demonstrate specification optimization method in the scope of the thesis the author tend to select simple and understandable criterion, while at the same time ensure its universality.

Bearing in mind all mentioned features the author has decided to use costs based optimization criterion. It is confirmed by several advantages. First of all costs are quite univocal indicator which is also understandable for non-technical user. For a customer (businessman), ordering robotic system, the expected costs for purchase and running the system are almost always the most significant among other indicators. If the customer does not have enough funds for the system, then it is unreasonable to evaluate the system by other criteria.

Secondly, the costs are very universal criteria and almost any other indicators of the system could be expressed as costs. For example, the time required to carry out the mission can be expressed as costs by introducing expenses rate. A fault tolerance of the system is converted to costs by estimating the expenses which will be present if the system would fail.

Costs are poorly suitable for evaluating non-technical aspects of a robotic system. This includes ethical, esthetical, ecological issues. For instance, it is impossible to express working environment impact on employee’s health in terms of costs. It is inhumanely to evaluate such indicators. However the scope of the thesis is limited to autonomous robotic systems, thus employment issues are not considered as corresponding. A robot is a piece of hardware which can be easily replaced in case of defects without any ethical implication. Also the thesis is focused on industrial robotic

systems, and they are not considered to be used in such areas as healthcare, hazardous production, nuclear power generation and others. Thus the ecological impact is not expected to be greater than conventional industrial approaches. Among other things the ecological impact of the robotic system could be evaluated (e.g. soil packing), but it directly affects the benefit of the customer not a global ecological situation (if a farmer have packed the soil a lot by heavy machines, then he well get less crop, as a result less income).

There are several positions which should be considered when evaluating a robotic system in terms of costs. These include the expenses required to purchase the system, the expenses required to run the system, as well as any indirect expenses, such as fault recovery expenses, maintenance, depreciation, etc. The author uses the total costs of ownership as a universal criterion for demonstrating specification optimization method. By the definition, total costs of ownership is the real costs of owning and using a piece of equipment such as a computer, taking into account the price of the hardware, software, maintenance, training, and technical support that may be needed (TCO, 2012). Total costs of ownership (TCO) is a financial estimate whose purpose is to help consumers and enterprise managers determine direct and indirect costs of a product or system. It is a management accounting concept that can be used in full costs accounting or even ecological economics where it includes social costs (Total-cost-of-ownership, 2012). Detailed analysis of TCO estimation for heterogeneous multi-robot system is provided in sections 5 and 6.

Parameters of the optimization stand for model values which are about to be optimized. In other words these values are set for such values, which maximize / minimize an objective function (criterion), which is TCO in this case. In case of specification optimization for multi-robot system the specification itself is the parameter for optimization. In particular number of robot types (classes), their functions and number of their instances used in the multi-robot system are being optimized.

Optimization *constraints* define boundaries for objective function in which it should be optimized. Constraints also could be interpreted as conditions that should be fulfilled in order to consider an optimization successful. For specification optimization of multi-robot system constraints are primarily defined by user according to expected application peculiarities of the system. These could include such conditions like time limits, mobility requirements or even communication strategies. Level of details of constraints depends on the background of user and can vary from abstract description of

system down to detailed specification of required implementation. This topic is discussed in details in chapter 3.1.

2.3. Concept of specification optimization solution

Previous chapters define a specification of heterogeneous multi-robot system and analyze optimization criterion, parameters and constraints. This chapter provides an analysis of conceptual model used for specification optimization.

The model proposed within the thesis is based on decomposition approach. By the definition, decomposition is a process of decomposing, which in turn means to separate or resolve into constituent parts or elements (Decompose, 2012). In mathematics it means to express in terms of a number of independent simpler components, as a set as a canonical union of disjoint subsets, or a vector into orthogonal components (Decomposition, 2012).

An implication of using decomposition approach within the thesis hides in the idea of decomposing requirements for heterogeneous multi-robot system into simpler units, which are subsequently analyzed in formal manner. The system is built of the units according to the results of analysis (composed from them). Thus the approach intents that the requirements are decomposed, analyzed and then composed back to single system. This corresponds to mathematical method called functional decomposition, which is used to resolve complex functional relations into its constituent parts in such a way that the original function can be reconstructed.

Before proceeding to description of concepts several general terms used within the thesis should be defined. First of all the *mission* is defined as a set of goals to be achieved by heterogeneous multi-robot system. It could be defined as simple or complex goals. The mission could define some global target for the system (e.g. eliminate pollution on site) as well as simple tasks (e.g. move to pollution origin, perform cleaning, and return back to service site for unloading / recharging). User of the specification optimization procedure defines the mission and the level of details is selected by him taking into account application peculiarities of the robotic system as well as desired optimization precision.

Proposed specification optimization approach for heterogeneous multi-robot systems is based on several concepts. The mission for the system is defined using the list of components. *Component* stands for an abstract definition of ability (function) of

the system without an explicit specification of its realization. It is an abstract entity capable to perform some specific functionality. Within the real system component could be implemented rather as hardware element or software module, as well as a combination of both.

Thus the mission is defined by specifying the abilities, which are required for multi-robot system to successfully perform the tasks and complete the mission. In order to facilitate user of the optimization approach as well as to allow computational analysis the components should be formalized. Definition of formal entity comes from mathematics and it pertains to manipulation of symbols without regard to their meaning (Formal, 2012). For that purpose an advanced classification of components is used.

The list of components is used for defining the requirements for particular multi-robot system in formal manner. Next, components are grouped together in order to form agents. In general, *agent* is a functional unit of the system. Within the thesis agents are considered to be mobile robots (e.g. transporter, observer) or stationary units (e.g. communication unit, warehouse). The term stands away from definitions used in agent based software systems and is focused on robotic domain. It is fair to mention that agent based software could be specified for mission of multi-robot system but as a single component without detailed description of its implementation features.

Also the thesis is focused on such specifications of multi-robot systems which imply that there is at least one mobile robot. This assumption is caused by two reasons. Systems consisting of many stationary units are widely investigated in the domain of multi-agent systems, where software agents are interacting each other in order to realize some desired functionality. Small number of mobile robots along high number of stationary units defines research direction which is different from the scope of the thesis.

Finally a set of agents is selected to form a solution. *Solution* is a specification of heterogeneous multi-robot system, it defines types of agents (classes) and a number of their instances used to carry out a mission. It is necessary clarify that agent concept described before actually defines distinct types of agents, not a real instances of robots and units. Graphical representation of the conceptual model is shown on figure 2.1.

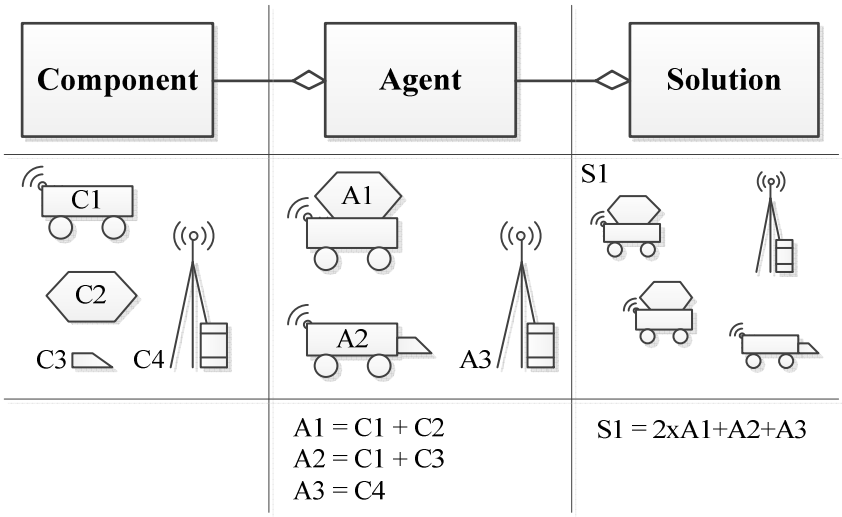


Figure 2.1. **Conceptual model of solution**

In addition to relations between concepts figure 2.1 shows a simple example of them. Components represent required functions of a system without distinguishing simple or complex ones. For example mobile base for robots (C1) and communication hub could implement quite complex functionality, while structural components (C2 and C3) are very simple.

Agents could be represented by any possible combination of components. The figure shows two mobile robots (A1 and A2) and stationary unit (A3) which is composed from a single component. In general, any component could be used as standalone agent, of course, if it is reasonable. For example, implementing transporting container as standalone stationary unit is doubtful arrangement.

Solution defines types of agents and the number of their instances. Figure shows single example of solution, which consists of two instances of A1 agent and of single instance of both A2 and A3 agents.

In comparison to other similar researches (see 1.3.3 and 1.3.4) the thesis introduces additional level of details for specification analysis. New dimension of parameters appears – components – which are added to the scope of selection of specification of multi-robot system. Because of that the space of possible solution expand dramatically and combinatorial explosion is observed, which is also typical for almost all combinatorial tasks, like chess solving or travelling salesman problem. However the thesis is aimed on optimal specification searching in full solution domain using advanced optimization procedure in contrast to comparison of limited number of intuitively selected solutions. Refined level of details of considered parameters of the

system opens potential to reach unique (well suited) and applicable solutions for particular mission.

Solution forming from agents (and from components, in general) implies application of various rules and constraints. Main purpose of introducing the rules is to allow formal processing as well as to limit scope of the thesis away from marginal cases (Komasilovs, Stalidzans, 2011). Next paragraphs describe the rules in details.

Rule 1 – Components of solution

The first rule defines the components of solution. It states that all components defined for specification optimization procedure should be used in the solution at least once. The rule is derived from the concept of decomposition approach which implies that the mission is defined using components which are required for its fulfillment. Thus if the solution (specification of the system) does not contain any of defined components, then such system is not satisfy all functional requirements (it is unable to fulfill the mission because of lack of some functionality). Therefore such solution is considered inappropriate.

This rule also affects the mission definition approach. There is no doubt that only mandatory components should be defined for the mission and it also corresponds to overall idea of specification optimization procedure: the mission is defined without designating its realization. As a result it is possible to cover wider scope of possible solution within optimization process.

Rule 2 – Number of components within solution

The second rule defines the number of components within the solution. Conceptual model of specification optimization procedure imply, that the agents are composed from the components but does not specify the number of components. Thus each defined component could be used in any number of different agent classes. Based on this the number of applications of components is not limited within the solution.

From the other side the components define only functionality required for particular agent without specifying its realization. There is no sense to define multiple components of the same type for the single agent. Thereby each component can be used only once in particular agent.

Rule 3 – Instances of agents

The third rule deals with instances of agents. As it was stated before, solution consists of multiple agents, and these agents are not required to be of unique class.

Thus, multiple instances of any agent class could be used within solution. Also it is worth to mention that it is not mandatory to use all possible agent classes within the solution. This aspect is discussed in details in section 4.

Furthermore, during optimization procedure it is possible to get solutions, which consist of same agent classes but differ by the number of their instances. Such solutions are considered as different and are analyzed (evaluated) independently.

2.4. Specification optimization development approach

Previous chapters define terms which are used within the thesis. Conceptual entities used within specification optimization are defined as well. This chapter briefly defines the most important steps of the developed optimization procedure and introduces the structure of the thesis, which, in general, describes the steps of the procedure.

By the definition, *procedure* is the sequence of actions or instructions to be followed in solving a problem or accomplishing a task (Procedure, 2012). Optimization procedure stands for the sequence of actions which are needed to find the optima or close to that value. Besides running an optimization the procedure includes preparatory steps and detailed analysis of results of optimization runs. A formal approach is proposed within the thesis, which is used for analysis of the functional and structural parameters of heterogeneous multi-robot system (its specification), as well as for the optimization of its costs taking into account customer's criteria and peculiarities of the multi-robot system.

The procedure provides a framework for finding best specification of the heterogeneous multi-robot system. It aims to optimal solution searching in full solution domain and provides methods to eliminate non-optimal solution domain branches on early stages of optimization. For combinatorial problems such approaches are required in order to get results in feasible time.

Figure 2.2 shows basic flowchart of the specification optimization procedure (Komasilovs, Stalidzans, 2012b). It consists of 8 consecutive steps and can be executed iteratively.

Step 1. First of all business requirements for multi-robot system are defined by customer (user). In other words he defines the mission for the system (see 3.1).

Step 2. Then mission decomposition process takes place which imply the mission definition decomposed in components, selection of optimization criteria, adjustable parameters and constraints (see 3.2 and 3.3). Also compatibility analysis between components is performed on this stage (see 3.5).

Step 3. Solution domain is analyzed in order to assess the total number of possible solutions (see 4.2).

Step 4. If the number of solutions is too high to evaluate all of them, then proceed to the step 5 (see 4.3). Otherwise one should go towards step 6.

Step 5. Heuristic algorithms are used to narrow the scope of considerable options to the bunch of fittest solutions (see 5).

Step 6. Fine evaluation is performed using simulations in order to select optimal solution for particular mission (see 6).

Step 7. The results of the optimization process are analyzed in a simulation environment to find out the differences between forecasted fitness in step 5 and step 6 (see 6.3)

Step 8. If the differences are not acceptable then parameters of initial evaluation is tuned to meet the requirements (step 5) or it is possible to apply different decomposition of the mission (step 2) and to execute the procedure again.

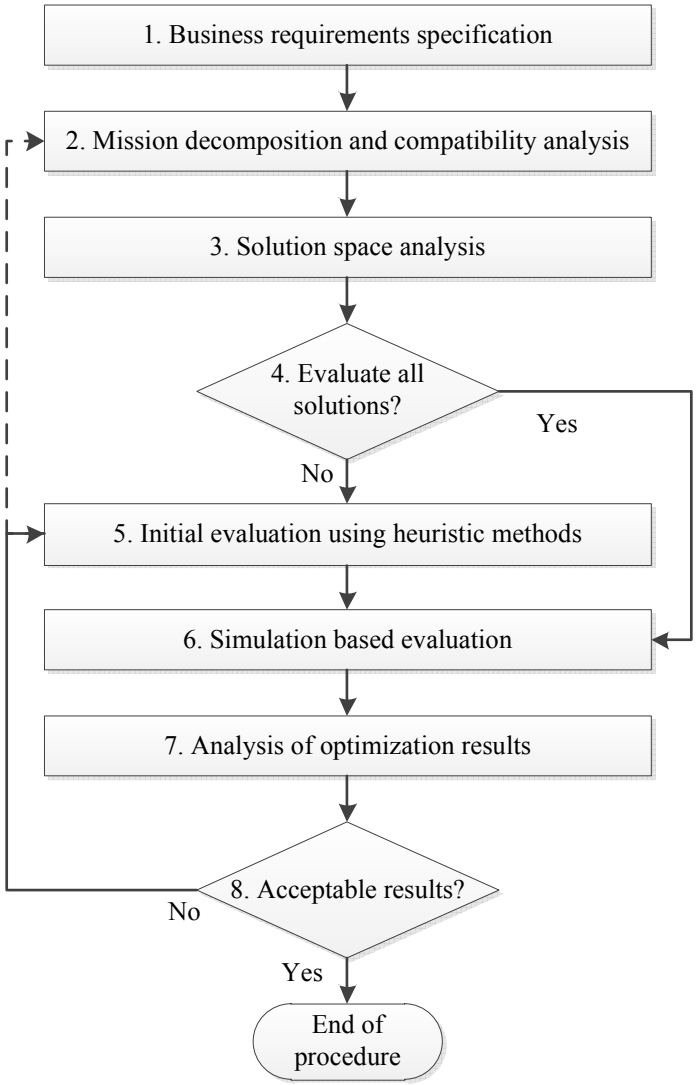


Figure 2.2. **Specification optimization procedure**

The steps of proposed specification optimization procedure are analyzed in details in following sections of the thesis.

2.5. Practical example for demonstration of specification optimization procedure

This chapter describes a mission for multi-robot system which is used within the thesis as a demonstration example for proposed specification optimization procedure. The mission was selected taking into account several features.

First of all it has to be suitable for multi-robot system in the sense that the mission has to be feasible both for single robot and for multiple robots. Also any conditions, that explicitly grant preference to any of solutions, are undesirable. For example, heavy object lifting task has to be implemented by the group of robots that is exactly capable

to handle heaviest object. Fewer robots (or single one) will not be able to handle the objects, while larger groups of robots will waste resources due to idle time.

Secondly, the mission has to be complex enough to make optimal solution not obvious. At the same time it has to be simple as much as possible in order to be implemented rather in simulation or hardware to test the procedure. Also the mission should provide sufficient dynamic environment in order to allow selection of solutions. For example, the same heavy object lifting task is quite simple and static, while Mars exploration mission is highly dynamic, but could be too complex and requires a lot of additional research for its implementation.

Finally the mission should be obvious and easy understandable and at the same time it should be practical enough to be able to show the benefit of the specification optimization approach. For example, ordering robots to move from point A to point B one after the other is simple enough, however the practical application is unclear. From the other side, automated fuel element replacement mission for nuclear power plant seems practical. However, most non-technical readers will not be familiar with problem specific features.

Practical example

Within the thesis lawn mowing problem is considered as the practical application example of specification optimization procedure. The mission for the robotic system is to mow defined lawn within a certain time. Lawns are common in territories of moderate climate and can be found in city parks, within private sectors as well as in country (farm grasslands). The mission consists of two tasks: the grass should be moved (1) and transported (2) outside the lawn.

The dynamics of the system changes depending on the number of agents. Single agent has to consequently follow the lawn in order to evenly cover the lawn. Contrary, multiple agents have to plant their navigation in order to minimize path overlay. In case of multiple types (classes) of agents the behavior of the system becomes even more unpredictable.

Due to biological origin of the mission object (grass is continuously growing) it is possible to extend the mission in time by repeating mowing cycle. This allows analyzing different specifications of the system some of which could be most effective at short time frame while others could reveal their advantages only in long periods. As an example of lawn for moving could be considered royal gardens (see figure 2.3). They

setup a complex environment with various obstacles and transportation paths that can be used to develop advanced strategy for behavior of the system.

Figure 2.3. **Garden of *Rundale Palace***
Source: http://www.vietas.lv/userfiles/image_gal/big/30/image-1630.jpg (accessed 2012 April 11)

The lawn mowing example is used as a case study in following sections when particular aspects of specification optimization procedure are analyzed. Additional features of considered lawn subject to specified step of the procedure are defined on demand in order to keep description simple and compact.

2.6. Summary of the section

The author defines optimization task for specification of multi-robot system. Total costs of ownership are selected as a preferable criterion. The author proposes detailed concept of optimization task solution, which is defined as a set of agents. Agents, in turn, are composed from components. Optimization parameters include types of agents and a number of their instances used within the solution. Various optimization constraints are defined by customer or are derived from mission analysis.

Custom procedure is proposed for the optimization of the specification of multi-robot system. It defines a workflow for resolving the optimization task and includes business requirement specification, mission decomposition into components, solution domain analysis, solution candidate evaluation using heuristic algorithms and simulated models.

Grass mowing task is defined as an example for demonstration of specification optimization procedure. The same task is used through the thesis for demonstrative purposes.

3. MISSION DECOMPOSITION AND SOLUTION DOMAIN DEVELOPMENT

The first step of the procedure (see figure 2.2) stands for specification of business requirements for the robotic system. In other words the mission is defined, which includes tasks and usage peculiarities of the system. This step corresponds to the similar stage of system analysis process and is described in next chapter (3.1).

The second step of the specification optimization procedure is aimed to formalization of system requirements obtained from previous step and preparing them (decomposing the mission) for consecutive optimization performed in next steps. The section is organized in several chapters which describe various aspects of mission decomposition and solution domain development process.

By the definition formal stands for being in accordance with the usual requirements, customs, etc.; conventional; in strict logical form with a justification for every step; correct in form; pertaining to manipulation of symbols without regard to their meaning (Formal, 2012). In computer science term formal methods is used to define particular techniques that are based on mathematical approaches and are used for the specification, development and verification of software and hardware systems (Butler, 2001). Formal methods allow improving reliability and robustness of a design due to appropriate mathematical analysis. From the other side application of formal methods leads to higher costs which in turn makes them unfavorable for small projects (Holloway, 1997).

Within the thesis a formalization approach is followed in specification optimization procedure and is used to reinforce design of heterogeneous multi-robot system with evaluation based on mathematical models. During development of the objective function formalization is applied to components. In other words formal definition of components allows processing them by general methods without focusing on features of a particular component.

3.1. Business requirements specification

By the definition a requirement can be thought of as something that is demanded or obligatory; a property that is essential for the system to perform its functions.

Requirements vary in intent and in kinds of properties. They can be functions, constraints, or other elements that must be present to meet the needs of the intended stakeholders. Requirements can be described as a condition or capability which is needed for a customer to be able to solve a problem or to achieve an objective. For clarification purposes, a description of the objective should always precede requirements; for example, business requirements, user requirements, system requirements, operational requirements, contract requirements, or test requirements (Ellis, 2012).

System requirements specification is a document or a set of documentation that describes the features and behavior of a system. A functionality required by different users of the customer is defined within the document. In addition to behavioral properties of the system, the specification also defines the main business process that will be supported by the system, as well as key performance features will need to be met by the system. In other words, requirements specification is a description of the features needed by all parties involved in using, implementing or maintaining the system.

Requirements specification as a document follows conventional recommendations and structure. Usually it describes such features as business model, use cases, business, functional and technical requirements, and various constraints. Development peculiarities of system documentation are studied within system analysis domain and are not considered in a scope of the thesis.

The expected result of current step of the specification optimization procedure is a detailed description of requirements for desired robotic system without going into details of its implementation. There is no goal to develop appropriate specification documentation and because of that description of requirements is provided in free form text.

Various system analysis methods are applied in order to develop successful requirements specification. Usually a set of interviews with a customer is performed in order to obtain initial information about the desired system. Customer describes his vision of the system in business terms, while system analyst is responsible to catch the most important information related to implementation of the system and transform it into functional requirements understandable for developers of the system.

Requirements specification process tends to be highly iterative, especially in agile development methodologies. Initial requirements obtained from customer are discussed with developers, who probably request additional details for functional requirements.

They, in turn, are again discussed with customer and so on until the design of the system is acceptable for all the parties. Figure 3.1 shows the process and the role of system analyst in it. Current step of the specification optimization procedure imply very similar process. As a studies on system analysis process belong to different domain it is not considered in details within the thesis.

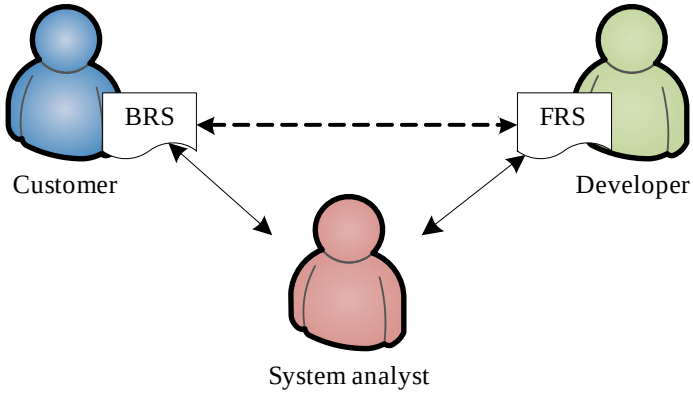


Figure 3.1. **System analyst role in requirements specification process**

Practical example

The lawn mowing mission is considered within the thesis as a practical example. An industrialist (customer) wants to use autonomous robotic system to mow the lawn in the garden near royal castle (see schematic map on figure 3.2).

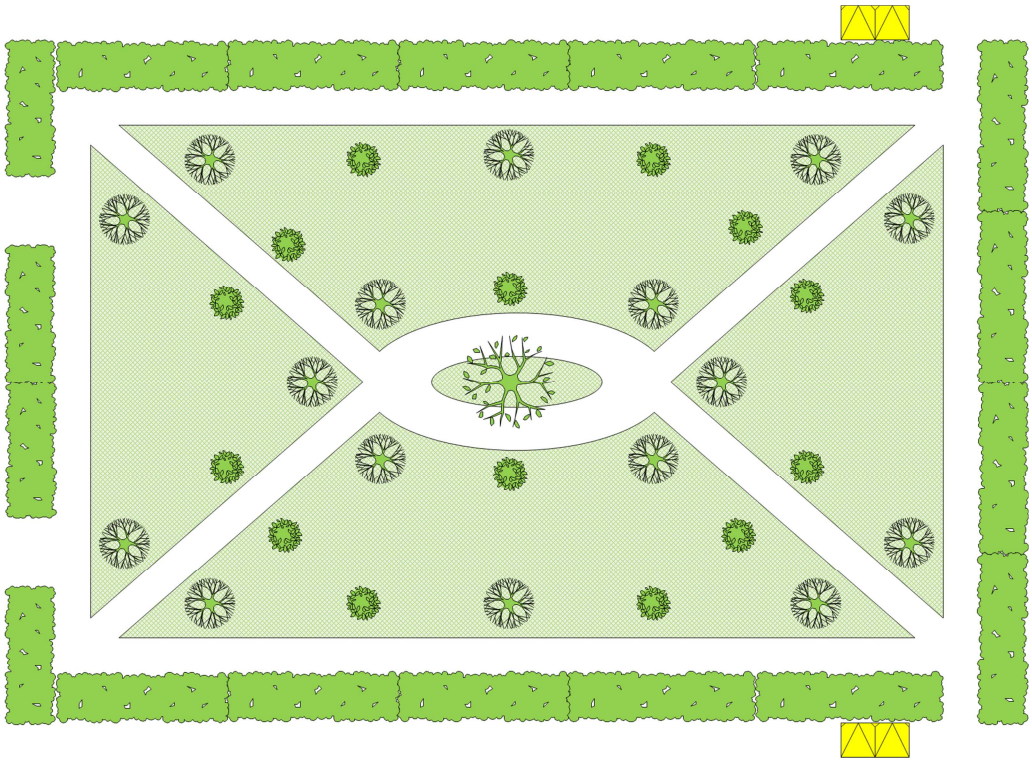


Figure 3.2. **Schematic map of the garden**

The customer provides detailed description of object's parameters as follows:

- ✓ the size of the garden is 150 per 120 meters;
- ✓ walkways are along the border of the garden and another walkways connecting corners and a central square of the garden;
- ✓ there is a plain terrain in the garden, the slopes are up to 5 ‰;
- ✓ trees and flowerbeds are located on the lawn and they should be avoided;
- ✓ minimum distance between adjacent objects (e.g. flowerbeds) is 2 meters.

The customer defines the task for the robotic system as follows:

- ✓ any spot of the lawn should be mowed at least once every four days;
- ✓ mowed grass should be transported to any of two dumpsters located outside the garden.

Additional parameters of the garden and requirements for the system are specified in next chapters in context of considered step of specification optimization procedure. When the customer has introduced his business model and specified requirements for the system it is time to formalize them and develop objective function for optimization procedure.

3.2. Formal classification of components

According to conceptual model of proposed specification optimization procedure the mission is defined using the list of components. And the components should be suitable for formal analysis methods which imply that they have certain structure and set of properties used in analysis. In order to support user of the optimization procedure a list of possible components should be developed. Such activities are usually referred as taxonomy or classification.

In general, taxonomy is the science or technique of classification; a classification into ordered categories (Taxonomy, 2012). Classification stands for systematic assignment of objects to categories. Within the scope of the thesis classification is applied to robotic components.

Due to the large variety of different methods developed in multi-robot systems domain a lot of approaches have been proposed to classify the state-of-the-art in different research aspects of multi-robot systems. Authors discuss open research topics within multi-robot domain and emphasize theoretical issues that arise in study of cooperative robotics. For instance, Cao and his colleagues (Cao et al., 1997) provide a

review of actual research axes. First of all they define canonical task domains for multi-robot systems as follows:

- ✓ traffic control, which deals with multiple agents that are moving in common environment and attempting to avoid collisions;
- ✓ cooperative manipulation, which focus on task allocation, fault tolerance, learning and communication organization issues;
- ✓ foraging, which addresses such issues as achieving performance gain in cooperative behavior, biological inspirations and group architectures;
- ✓ other task domains, which include such fields as multi- robot security systems, landmine detection and clearance, robotic structural support, map making, and assembly of objects using multiple robots.

Authors classify actual open research topics within cooperative robotic systems domain into several axes as follows:

- ✓ group architecture which provides the infrastructure upon which collective behaviors are implemented, determines the capabilities and limitations of the system, and encompasses such concepts as robot heterogeneity/homogeneity, the ability of a given robot to recognize and model other robots and communication structure;
- ✓ resource conflicts, which arise when multiple robots request some indivisible resource, are resolved by variety of mechanisms being proposed in actual researches;
- ✓ origins of cooperation refers to how cooperative behavior is actually motivated and achieved and deals with biological parallels, game theories and concepts of emergence;
- ✓ robot learning which is aimed on adjustment of control parameters of multi-robot system in order to optimize their task performance, and to adapt to changes in the environment;
- ✓ geometric problems which covers research issues that are tied to the embedding of robot tasks in a two or three dimensional world. These issues include multi-agent path planning, moving to formation, and pattern generation.

Arai and his colleagues (Arai et al., 2002) provide classification of research directions and identify primary topics as follows:

- ✓ biological inspirations;
- ✓ communication;

- ✓ architectures, task allocation, and control;
- ✓ localization, mapping, and exploration;
- ✓ object transport and manipulation;
- ✓ motion coordination;
- ✓ reconfigurable robots.

Some of these topics are already discussed in details in previous chapters (see 1.3.1). Other researchers provide classifications of various aspects of the systems. For example Dudek and his colleagues (Dudek et al., 2002) provide taxonomy of robot collectives taking into account following features:

- ✓ size of the collective;
- ✓ communication range, topology and bandwidth;
- ✓ collective re-configurability;
- ✓ processing ability of each unit;
- ✓ collective composition (homogeneous, heterogeneous).

Other researchers focus on classification of specific problems related to multi-robot systems. For instance, Gerkey provide classification of task allocation problems in the scope of multi-robot systems (Gerkey, Matarić, 2004). Yanco report a classification of human-robot interaction taking into account such categories as task type and criticality, people to robot ratio, interaction rules and other (Yanco, Drury, 2005). Next chapter provide analysis of custom classification of robotic components proposed within the thesis.

3.2.1. Building classification tree of robotic components

As it was stated before component stands for an abstract definition of function of the robotic system without reference to its realization. Also the proposed specification optimization procedure is not limited to any specific domain, and is aimed to be a universal method. Because of huge number of robotic components (functions) utilized by modern industry it is unfeasible to define all possible components by the classification.

The specification optimization procedure uses custom component classification approach which implies application of component categories instead of components in specification processing. The classification of components was developed taking into account a number of requirements. It has to have following features:

- ✓ user oriented – the aim of the classification is to support user of the procedure thus it should provide him clear and intuitive categories of components;
- ✓ formal – in general, components are used to define a mission for robotic system, which is processed using the number of analytical methods, which in turn impose formal definition of the components;
- ✓ universal – the classification of components should be universal in order to support specification optimization procedure which intended to be domain independent method;
- ✓ extendable – as the number of possible components is huge and grows over time the classification is expected to be extendable on demand in order to allow its actualization and/or specialization for user's domain.

Taking into account aforementioned features the classification model of components was proposed. It is inspired by biological classification of species and follows tree structure. Tree structure has several benefits over plain list classification. First of all, trees are usually more compact and as a result, more user-friendly in comparison with lists. Secondly, tree structure is easily extendable by adding new branches.

Proposed component classification implies that each element of classification tree is a taxon, which describes certain component (functionality). The level of details of particular component increases going deeper through the tree. Root elements of the tree define categories and logical groups of components, deeper elements of tree (branches) are the specific components of robot system. Leafs of the tree (deepest elements) can specify even implementation of components by particular vendor depending on the level of details of the specification optimization task.

An important feature of proposed classification is that any node of the tree (rather, category, group or implementation of component) can be used to define the mission for the specification optimization procedure. This makes the procedure highly universal and allows applying the same methods both to optimize multi-robot specification for conceptual mission, defined using general categories of components, as well as for detailed mission, defined by vendor specific components. In case if rather more detailed mission specification is required user can easily add new elements to the tree.

The proposed classification of components is not intended to be complete and absolute. In opposition, the thesis demonstrates universal approach for component classification which can be easily extended and adopted for specific domain. The

essential tree elements have been created based on taxonomies provided in (Bekey, 2005).

In general, defining some functionality for robotic system usually implies also certain structural changes – new component is added which implements the functionality. Thereby primary categories of the classification tree define structural elements of the robotic system, which include also sensing, actuation and control elements.

The very basic feature of any mobile robot is **locomotion**, which defines various methods used to transport them within the environment. Locomotion type is usually selected taking into account such indicators as energy efficiency, control simplicity, type of environment, impact of adjacent domains (such as biomechanics). There could be distinguished several types of locomotion depending on application environment.

For the on-ground locomotion the most common and efficient in terms of energy consumption are wheeled robots. The mobile base is equipped with the number of wheels which are used to support the base. Usually each wheel has its own motor and differential steering is used for control of locomotion. Other option includes car-like construction where steering and drive wheels are controlled independently. The first option is easier to implement and it is more reliable while the second option is suitable for more high-speed locomotion. In advanced robotic systems (e.g. space missions) advanced suspension systems are implemented.

Another on-ground locomotion type is track-driven, which imply that mobile base of the robot is equipped with pair (or several pairs) of caterpillar tracks – a system of vehicle propulsion in which a continuous band of treads is driven by two or more wheels. Typically such robots are used for military purposes, for planetary exploration, or for hazardous environments. From a kinematic point of view, a tracked vehicle can be considered as differentially driven vehicle. The importance of tracked vehicles arises from the ability to climb over obstacles not passable with wheels.

Some other on-ground locomotion types are less distributed and include such options as legged, hopping and serpentine locomotion. Legged (especially bipedal) locomotion has attracted new investigations in recent decade because of advances in hardware production as well as in biomechanics.

Besides on-ground robotics there is wide domain of underwater robots. Robots primary are used for exploration of areas which are not directly accessible for humans. Underwater locomotion usually implies special control algorithms (due to non-linear

dynamics) which allow the vehicle to move in three coordinate directions. Two major classes of underwater locomotion can be distinguished. The first is distinguished to submarines and other underwater vehicles, while the second class is inspired by biology and is focused on fish-like locomotion.

Another category of locomotion is related to aerial vehicles. To some extent this type of locomotion is similar to underwater locomotion in sense that vehicles have to move in three dimensional medium. There are two types of structural implementation of flying robots could be distinguished: fixed-wing vehicles (unmanned planes in general) and rotary-wing vehicles (helicopters). Nowadays many investigations are aimed on micro unmanned aerial vehicles, which are used to collectively perform such tasks as construction or exploration.

- ✓ Locomotion
  - ○ Wheeled
    - Differential
    - Car-like
  - ○ Tracked
  - ○ Other on-ground locomotion
    - Legged
    - Hopping
    - Serpentine
  - ○ Underwater
    - Submarine
    - Fish-like
  - ○ Aerial
    - Fixed-wing
    - Rotary-wing

Next important feature of robots next to their locomotion is **sensing**. Without any sensing robot will not be able to appropriately react to external stimulus and will fail its mission even in slightly changing environment. There are two classes of sensors can be distinguished. First is related to sensing of robot's internal environment – proprioception. In general this relates to feedback from various internal devices and includes such categories as position sensing usually used for subsequent navigation planning; velocity, acceleration and load used for advanced locomotion control; power source state (for instance, charge of the battery or fuel level) used for intellectual task allocation and mission planning. Another important featured are the sensing of internal temperature and indication of failures from internal devices, which allows earlier identification of hardware issues and appropriate mission planning.

The second class of sensors is related to sensing of robot's external environment – exteroception. This includes any possible device which is capable to perceive some information from the environment. Several groups of sensors can be distinguished. One of the most important is vision, which usually includes a camera and software designed for low-level image processing, such as edge and color detection, object recognition.

Another very important group is proximity sensing, which allows distance perception to adjacent objects which in turn is used for obstacle identification and avoidance. Implementation of proximity sensors can vary from ultrasonic devices to laser based meters. Touch sensors, which are somewhat similar to proximity sensors, are used to sense the physical world through direct contact. The simplest sensors just close a switch on contact with an external object. More advanced sensors provide indication of contact force. Other similar sensors are designed to measure slippage, especially important in grasping.

Also many other sensors are available which are used in domain specific mission and devices for acquiring such information as audition used for alarm detection or response to voice commands; olfaction used to detect of particular hazardous compounds; temperature used for remote environment probing and others.

- ✓ Sensing
  - ○ Proprioception
    - Position
      - ◊ GPS
      - ◊ Odometer
    - Velocity
    - Acceleration
    - Load
    - Power source state
      - ◊ Battery charge
      - ◊ Fuel level
    - Internal temperature
    - Failures
  - ○ Exteroception
    - Vision
    - Proximity
      - ◊ Ultrasonic
      - ◊ Laser
      - ◊ Touch
    - Slippage
    - Audition
    - Olfaction
    - Temperature

The third important feature of robotic systems is their ability to interact with environment. Various types of **manipulators** and actuators are utilized for this. Number of degrees of freedom (DOF) is used within the thesis for classification of manipulators. Single DOF manipulators stand for simplest transitional actuators like dumpers. Two DOF manipulators are usually represented by various grippers capable to hold and lift the objects. Three DOF manipulators are widely spread on production sites for loading and unloading various types of cargo. Manipulators with four and more DOFs are typical robotic arms capable to perform large number of tasks defined within mission. Different group is distinguished for various end effectors, which are usually connected at the end of kinematic system of manipulator, and are capable to perform domain specific tasks, like boring, welding or painting.

- ✓ Manipulation
  - ○ 1 DOF
  - ○ 2 DOF
  - ○ 3 DOF
  - ○ 4+ DOF
  - ○ End effector

In order to allow more detailed definition of mission using the components two additional features of robotic systems are added to classification tree. One of them is **communication** capabilities of agents. As a scope of the thesis is limited to multi-robot systems it is implied that agents can communicate with each other, no matter, explicitly or implicitly. Moreover communication components are considered as mandatory for any mission definition because study about emergent behavior of robot group with no communication is out of scope of the thesis. Communication components can be divided into two groups depending on their effective range. Local communication allows information exchange between agents on relatively small distances and includes such technologies as infra-red transducers, Bluetooth networks as well as sign based communication, like color led, gestures and others. Range of global communication is limited only by specification of utilized technology. It includes such technologies as wired and wireless networks, radio signals, GSM networks and others.

- ✓ Communication
  - ○ Local
    - IR
    - Bluetooth
    - Sign based
  - ○ Global
    - Wi-Fi

- Wired
- Radio
- GSM

The other feature is related to **computation** capabilities of the robots which are required in order to fulfill the mission. These include various components for image processing, mapping and localization, navigation and path-planning, task allocation, fault-recovery and many others. Most often these components are implemented as software module for robot controller.

- ✓ Computation
  - ○ Image processing
  - ○ Navigation
  - ○ Path planning
  - ○ Task allocation
  - ○ Localization
  - ○ Mapping
  - ○ Fault recovery

Provided classification categories are intended to demonstrate the approach proposed within the thesis. The classification tree should be extended by adding more detailed categories of components or even their vendor specific implementation in order to adopt the specification optimization procedure for specific domain of industry.

3.2.2. Defining properties of components

One of the reasons building the classification of components in the form of three is its intuitive structure and possibility to find earlier defined and to add new categories. The other important feature of tree structure is discussed in this chapter and it is related to properties of components.

As it was described before the mission is defined using the components from classification tree and then processed using various computational methods. Also the whole specification optimization procedure is intended to be formal, that is not limited to specific application domain. Thereby in order to successfully process defined mission in formal manner the components require additional properties to be specified for them, which in turn are used by processing routines.

Conceptual features of the properties themselves should be described before assigning specific properties for the components. The properties of the components are designed similarly to methods in object-oriented paradigm of software development. First of all tree structure of component classification allows inheritance of properties. It is designed in such way that all properties are inherited from parent components down

to their children. This simplifies extension of the classification tree because most of the properties are defined on ancestor nodes, thus only most detailed properties are additionally required for new categories.

Secondly, core properties can be overridden by child nodes. That is individual properties inherited from a whole branch of the tree can be redefined in child node by setting up new values. Such approach makes the classification tree flexible and highly adoptable for application domain.

Also some properties are allowed to be defined without their value, just as placeholder for real components. The values for these properties are mandatory and should be set at mission definition stage just before formal processing of specification. An example of such property is price of particular components, which depends on industrial domain of desired multi-robot system. As a result it could be defined only at practical application of the specification optimization procedure.

Practical example

Finalizing the discussion, the list of mandatory properties of components should be developed for the example used within the thesis. As it was stated before, primary criteria for evaluating specification used within the thesis are the total costs of ownership. Thereby properties required for calculating the TCO are mandatory. The list of proposed properties is as follows:

- ✓ price of the component, which stands for expenses required for buying the component;
- ✓ power consumption, which influences the costs required for operation of component;
- ✓ complexity index, which indicates relative difficultness of assembling the agent that consists from particular component.

Complexity index is a relative value used to compare agent components with each other and as a result its bounds are highly dependent from the subject area peculiarities. For example, CPU and motor controller mounted onto a printed circuit board could have complexity indexes equal, respectively, to 5.0 and 1.0, meaning CPU is 5 times more complex to mound in comparison with motor controller. But in case of CPU, motor controller and a led mounted onto the same printed board the complexity indexes could have values equal, respectively, to 1.1, 1.0 and 0.1, meaning that both CPU and

controller mounting is about ten times more complex in comparison with a led. For the grass mowing example the author used values within range from 0.5 up to 1.5.

These properties are discussed in details in following sections related to costs estimation model (see 5.2). Also the application name of the component is recommended to be specified in order to distinguish same components used for different functions.

3.2.3. Implementation model of classification tree

Technical implementation of component classification tree follows classical approaches for parent-child models. It was stated before that any node of the tree can be selected for mission definition, thus there is only single type of nodes in the tree – components. Also the number of children of each node is not known in advance. Moreover new child nodes can be added after a while, so the structure should be dynamic.

Dynamic trees are a usually implemented using reverse reference that is each child object has reference to its parent. Only the root of the tree does not have defined parent. Such object is usually hidden in user interface and is not permitted to be selected. Also parent objects sometimes maintain the list of their children using special lists. Each node of the tree (component) has a varying size array containing references to its unique properties. It means that inherited properties are not directly assigned to each child node, but instead they are obtained by requesting such information from parent node using recursive procedure call. Conceptual model demonstrating implementation of classification tree is available on figure 3.3.

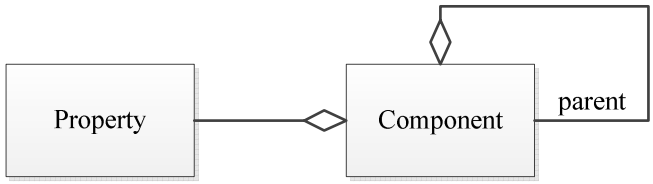


Figure 3.3. **Classification tree implementation model**

3.3. Mission tasks

Previous chapters describe the classification tree of components, which is used to define mission objectives for multi-robot system. Components allow specifying only structural features of the mission, for instance, the requirements for the robotic system

to be able to perform certain functionality. However dynamic features of the mission remain undefined and the application of certain component within the context for the mission is unclear.

A concept of task is introduced because of above described concept of mission. In addition to the list of components, global mission is defined in terms of tasks. They stand for simple independent missions, which can be univocally performed by robotic system. Such missions are usually used as test beds for robotic systems in investigational projects.

Thereby the mission for the robotic systems considered within the thesis is defined using the list of components (i.e. required functions of the system) as well as the list of conceptual tasks, which define behavior of the system. Additionally each component is assigned to at least one task. This is required for evaluation of the solution candidate described in details in following sections (5.2.4). In general, such assignment allows predicting overall performance of the system on the mission. For the same reason the amount of work to be performed by the robotic system within certain task is defined. Units of measurement depend on application domain and can vary from time to production entities.

The classification of the tasks is not considered within the scope of the thesis because it strongly depends on application domain of the proposed specification optimization procedure. As it was stated before, tasks should be simple enough to be obviously performed by robotic system. Theoretically any task can be decomposed into simpler tasks upon reaching very general actions. However this will result in a high number of the tasks defining the global mission and as a result will impede application of the specification optimization procedure. The recommended number of tasks considered within the thesis is up to four.

3.4. Practical mission definition for multi-robot system

A practical example used through sections of the thesis is grass mowing mission defined in chapter 2.5. This chapter formally defines the mission in terms of components and mission tasks. The result of formalization is used in next sections in order to demonstrate the following steps of the proposed specification optimization procedure. Also the optimization criterion was defined in 2.2 and that are the total costs of ownership of the robotic system designed to perform grass mowing mission.

Practical example

First of all the list of required components is defined. Very basic function of the desired robotic system is its ability to move over the lawn – the **locomotion**. Wheeled car-like mobile base was selected because several reasons. Firstly, mowing mechanisms are relatively small and lightweight thus wheels are suitable for handling such vehicle. Secondly, car-like steering mechanism has least impact on grass and soil in comparison with tracks and differential steering.

Communication facilities are mandatory for multi-robot systems. Global Wi-Fi communication was selected for this purpose because of its distribution on the market as well as it offers adequate communication range suitable for the scale of the lawn.

Next group of components is related to performing mission specific tasks. First of all the **mowing machine** is required. In general it could be considered both as manipulator and as end-effector for manipulator. For demonstrative purpose 1-DOF manipulator was selected representing mowing machine, which could be in two states: idle (lifted up) and working (lifted down). Also the size of mowing machine is defined according to business requirements. They define that minimum distance between adjacent objects is 2 meters, thus the mowing robot should be able to maneuver in such spots. In order to simplify demonstrative calculations the size of mowing machine is considered to be 1 meter.

Secondly, a **manipulator for collecting grass clippings** into container is required. Its implementation highly depends on composition of other components and can be implemented as a part of mowing machine as well as separate device. In order to allow any of these options to be considered in specification optimization procedure the manipulator was defined as an abstract end-effector without selecting its implementation details. The third mandatory **manipulator is related to unloading** the grass clippings into dumpsters after transporting them outside the garden. For demonstrative purpose a dump track is defined to perform unloading, as a result another 1-DOF manipulator is selected from classification tree.

Finally the **sensing and computation** facilities are defined for the robotic system. For the demonstrative purpose sensing features include such components as proximity sensing implemented as laser distance meter for avoiding the obstacles and GPS positioning for global navigation. Also load meter is used in order to perform transportation of grass clippings in appropriate frequency. Computational features required for the mission include such facilities as navigation for aiming the robots on

the lawn as well as task allocation for distributing task among the robots (in case if solution candidate consists of multiple robots).

The next aspects to be defined are mission **tasks**. For the example mission considered within the thesis two tasks could be distinguished. The first stands for grass mowing task, which in general can be described as a traveling on the lawn in such manner that a path goes through uncovered area of the lawn. Within a scope of the thesis such task is called as **area coverage task**. The essence of the task is to evenly cover (travel through) the defined area. A robotic system is successfully performing the task if the size of the area which was covered more than once tends to zero. Other practical examples of area coverage task include such mission as floor cleaning, aerial photomapping, mine detection, most of agricultural procedures and others.

The second task of the mission is to transport grass clippings from lawn to the dumpsters. Within the thesis it is called **transportation task**. This task is essentially different from previous because it does not require even covering of the area but instead requires traveling from one point to another. Successful implementation of this task imply that the length of traveling path is as short as possible which in turn will minimize the costs for robotic system performing such task. Also other path evaluation criteria are possible, for instance, traveling time or smoothness of the path. This type of task is common for logistics and its complexity rapidly increases depending on the number of points to be visited. A generalized definition of the problem is known as travelling salesman problem, which in turn is NP-hard combinatorial optimization problem.

All the previous definitions of the components are summarized in table 3.1, which includes also additional properties defined before as mandatory for specification optimization procedure. The units used for price and power consumption definition of the components are referred as abstract units of cost respectively for buying the component and for running the component per time unit. Also area coverage task is referred as the first (I), and transportation task as the second (II).

Table 3.1. **Detailed definition of mission components**

Name	Class	Price	Power	Additional properties
Mobile base	Wheeled car-like	60	4.0	Speed: 0.5 m/s
Network	Wi-Fi	30	2.0	Complexity index: 1.1
Mowing machine	1-DOF manipulator	40	5.0	Only I task
Loader	End-effector	40	4.0	Only II task
Dumper	1-DOF manipulator	20	3.0	Only II task; max 20 kg
Laser	Proximity	30	2.0	n/a
GPS	Position	25	1.5	n/a
Load	Load	20	0.5	Only II task
Navigation	Computation	50	1.0	Complexity index: 1.3
Task allocation	Computation	50	1.0	Complexity index: 1.2

Additionally the amount of work within particular task should be defined. These values are derived from the business requirements of the mission. As the size of the garden is considered to be 150 x 120 meters and the size of mowing machine is 1 meter it is can be concluded that mowing robot at least have to cover 18 000 square meters of lawn. Considering additional consultations with customer this amount is reduced by 40% because of walkways and flowerbeds. Thus there are 10 800 square meters of the lawn to be mowed, which are also can be considered as units of work to be performed by the robotic system.

The amount of work for second task is not so trivial to determinate. It is known that dumper can handle 20 kg of grass clippings. Also, considering consultations with the customer, the density of the grass is equal to 0.25 kg per square meter. This means that the dumper becomes fully loaded after collecting grass clippings from 80 square meters of the lawn. Thus it is expected to have 10 800 / 80 = 135 trips from lawn to dumpsters outside the garden. Another interpretation of the obtained value is that it shows the number of stacks to be transported. Thereby, the value can be used to define the amount of work to be performed for the second task.

Of course aforementioned assumptions are very rough, but because of their simplicity they can be easily used for processing in specification optimization procedure. Also some of them are refined on demand in following steps of the procedure described within next sections.

The general rule, which was followed during selecting the components, is to specify the mission as simple as possible. Such approach allows fast progressing through the steps of proposed specification optimization procedure. If the results are satisfactory then the mission could be refined and procedure executed iteratively. Such

approach is much practical then initially defining the mission with fine details and then after long processing to obtain negative results.

3.5. Iterative analysis of mission definition

According to specification optimization procedure the mission definition provided in previous chapters can be explicitly used in next steps of the procedure. However by the analogy with business requirements specification, the mission definition can be performed in an iterative manner, which means that the definition of the mission is refined after regular consultations with customer. This chapter provides demonstrative analysis of the mission definition with aim to develop additional constraints for solution domain. This in turn will decrease computational complexity of subsequent steps of specification optimization procedure.

As the number of solution candidates depends on the number of components (for more details see section 4), primary aim of the analysis is to reduce the number of components or at least to reduce the number of combinations considered by optimization procedure. Constraint development for component combinations is based on logical derivations and can be partially automated.

Practical example

The very first example of constraint refers to **mobility of the agents**, which are composed from the components. A feature of the mission specifies that the grass mowing machine should be transported over destination area (e.g. lawn, grassland). Because of that any agent (combination of components), which contains mowing machine and does not contains mobile base are initially irrational. Such agents will not be able to perform one of mandatory functions. The same logics can be applied for loader of grass clippings, laser proximity sensor and GPS positioning device. All these components are useless if placed on stationary unit within the scope of the current mission.

Contrary the dumper manipulator used for unloading the transportation container can be placed on a stationary unit. For instance, mobile robot with a fixed container can drive up to stationary unloading mechanism, which will turn the container around.

Formally such rule can be implemented using logical implication function. It returns false only when first operator is true, but the second operator is false (see table 3.2).

Table 3.2. **Truth table of logical implication function**

$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{A \rightarrow B}$
0	0	1
0	1	1
1	0	0
1	1	1

An interpretation of such Boolean operator is as follows. Considering component A requiring component B for its proper function (2). If an agent does not contain component A then it is out of scope of the current rule (it bypasses the validation because the result of logical implication is true). If an agent contains component A then it bypasses the validation only if component B is also present in the agent.

$$compA \rightarrow compB \tag{2}$$

Next type of applicable constraints is based on **performance capacity** calculations of robotic system. These constrains are derived from business requirements of the mission and from properties of the components. The most obvious constraint could be derived from the size of lawn and traveling speed of mowing machine. It was stated before, that there are 10 800 square meters of the lawn to be mowed during the mission. According to specification of components the mobile base of mowing machine moves with speed of 0.5 meters per second. Thereby the robot needs at least 21 600 seconds or 6 hours to mow a whole lawn taking into account an assumption that the robot constantly works during this time.

According to current business requirements the system is capable to complete the mission in time (the limit is set to 4 days). However if the dimensions of lawn are increased five times up to 750 meters per 600 meters, then the time required for single robot to complete the mission becomes more then 6 days. Thus in order to fulfill four day limit the system has to contain of at least 2 mobile mowing machines taking into account that they are working 24 hours per day, which also is not desirable.

This chapter demonstrated general approach followed for constraints development on the second step of the proposed specification optimization procedure. A constraints development process highly depends on application domain and it mimics system analysis processes used for software architecture development.

3.6. Summary of the section

This section describes mission decomposition process which is used for solution domain development. The author proposes formal classification of robotic components which supports the decomposition process. A flexible classification structure is proposed and populated with the definitions of the general purpose robotic components. The mandatory properties are defined for the practical example used within the thesis.

The concept of the mission task is introduced for supporting definition of dynamic aspects of the mission for a multi-robot system. Area coverage and transportation tasks are defined for the demonstrative example.

The iterative analysis of constraints is described used to eliminate initially irrational solution candidates. This allows decreasing the number of processed solution candidates during the next steps of the procedure, which leads to faster processing in general.

4. ANALYSIS OF DOMAIN OF FEASIBLE SOLUTIONS

The third step of proposed specification optimization procedure (see figure 2.2) stands for solution domain analysis, which is applied on formal definition of the mission developed within previous step of the procedure. The main goal of this step is to recognize the complexity of the specification optimization problem and predict the number of feasible solutions. The results of the analysis are used for selecting appropriate optimization and evaluation approach for subsequent steps of the proposed procedure.

4.1. Calculating a number of unique agents

According to conceptual model described in chapter 2.3 the specification of multi-robot system (solution) is formed from agents, which in turn are composed from components. Thus very first step of complexity analysis is to obtain the number of agents which are possible to combine from defined components. Taking into account the fact that components define only a type of agents, not the actual instance of the agent, the number of unique agents is considered at this stage.

Proposed conceptual model does not specify any limitation upon agent composition, thus any combination of components is allowed and is considered as unique agent. Among the number of possible options there are two extreme combinations of components: an agent formed from single component, and an agent combined from all defined components.

Agent combination from components is demonstrated on a generalized example. Consider an abstract mission which is defined using only single component. The only possible agent is composed from that component. So the total number of possible agents equals to 1. Now consider a mission defined using two components. In such case it is possible to combine agents from each single component, as well as from both components together. So the total number of unique agents equals to 3. Furthermore consider a mission defined using three components. Just like in previous cases it is possible to combine agents from every single component (three unique agents). Next it is possible to combine agents from pairs of components (another three unique agents).

And finally the agent composed from all three components is also feasible. Thus the total number of unique agents equals to 7.

Generalizing agent composition rules it is possible to distinguish clear algorithm for generating all possible agents from the given components. First agents are composed from every single component, then from all possible pairs of components. Next agents are composed from possible triples, quads of components and further more until the total number of components is reached. The total number of unique agents equals to the sum of number of combinations for each of aforementioned positions. The demonstration of recently considered cases is provided in table 4.1.

Table 4.1. **Agents combined from various numbers of components**

Components	**Agents**	**Number of agents**
A	[A]	1
A, B	[A], [B] [AB]	3
A, B, C	[A], [B], [C] [AB], [AC], [BC] [ABC]	7
A, B, …, x	[A], [B], …, [x] [AB], …, [Ax], ..., [Bx], …, [x-1, x] [ABx], …, [x-2, x-1, x] … [AB…x]	?

Finally the analysis of number of unique agents is reinforced by mathematical considerations and equations. First of all exact number components is known at mission definition stage, thus it is known the number of positions (singles, pairs, triples) to be summed up. Next, the number of combinations within each position can be calculated according to definitions from set theory (3).

$$C_n^k = \frac{n!}{k!\,(n! - k!)} \tag{3}$$

where

n – total number of elements within set;

k – number of selected elements for combination.

Thereby the final equation for calculating number of unique agents which can be produced from defined components is developed (4) and simplified.

$$f(n) = \sum_{k=1}^{n} C_n^k = \sum_{k=1}^{n} \frac{n!}{k!\,(n! - k!)} = 2^n - 1 \quad (4)$$

where

n – number of defined components for the mission;

$f(n)$ – total number of unique agents.

As it can be seen, the obtained equation is of exponential type and, as a result, the number of unique agents grows rapidly depending on the number of components, used to define the mission. This effect is known as combinatorial explosion where small increase of the number of components brings huge increase of possible combinations.

4.2. Calculating the number of feasible solutions

When the number of possible unique agents is obtained it becomes possible to analyze the number of feasible solutions. According to the conceptual model of the proposed specification optimization procedure the solution candidate is considered as a set of agents which is capable to fulfill the mission. Very basic criterion of capacity of the system to complete the mission specifies that every defined component should be used within the solution at least once.

Additionally conceptual model allows multiple instances of agents. Moreover solutions which are composed from the same types of agents but differ with each other only by the number of instances are also considered different. A logical conclusion arises that the number of solutions is infinite even with single defined component because the number of instances is infinite (conceptual model does not define any constraints). However such considerations are not practical, because infinite number of agent instances leads to infinite expenses for such system. Within a scope of the thesis total costs of ownership is used as a primary optimization criterion of a specification of a robotic system. The criterion implies minimization of its value, which in turn will also lead to decreasing number of instances of agents.

An artificial limit for the number of agent instances (5) is introduced for practical reasons. It is used in order to protect computational algorithms from overflowing during evaluation steps of the proposed specification optimization procedure. The value of the constraint used within the thesis equals to 10.

$$A \in N^* \quad (5)$$

where

N^* – positive natural numbers.

Previously introduced limit of agent instances appears as multiplier for total number of solutions. For illustrative purposes of further solution domain analysis performed within current chapter the constraint is limited to $A = 1$ and is omitted until final conclusions.

Solution composing from agents is demonstrated on a generalized example. Consider an abstract mission, which is defined using single component. Only single unique agent is possible to compose from the component. And as a result only single solution is possible: to utilize the only available agent.

In case if mission is defined using two components A and B, three unique agents are feasible: [A], [B] and [AB]. Taking into account the limitation of agent instances it is possible to combine these agents in seven different ways (see table 4.2).

Table 4.2. **Demonstrative combinations of agents and feasible solutions**

Agents			**Comments**
[A]	**[B]**	**[AB]**	
✓			Invalid solution, missing component B
✓	✓		Two simple agents
✓	✓	✓	All possible agents
✓		✓	Simple and complex agents
	✓		Invalid solution, missing component A
	✓	✓	Simple and complex agents
		✓	Only complex agent

As it is seen from the table, not all possible combinations of agents can be considered as solutions (marked rows). Some combinations cannot be solutions because they do not contain all defined components, which are required by conceptual model of multi-robot specification. It is worth to mention that this validity criterion does not depend on the number of agents selected for solution. For instance, both invalid combinations are composed only from one agent, but at the same time valid solutions is composed from single complex agent. The same situation appears for missions defined by three and more components.

The number of possible combinations of agents is easily obtained using similar approach used for calculation of component combinations (Komasilovs, Stalidzans, 2011). Thus the number of valid solutions is calculated using equation (6), which also can be simplified.

$$g(n) = \sum_{l=1}^{f(n)} C_{f(n)}^{l} - r(n) = 2^{2^n - 1} - 1 - r(n) \quad (6)$$

where

n – number of defined components for the mission;

$g(n)$ – number of solutions;

$f(n)$ – number of unique agents (4);

$r(n)$ – number of combinations, which are not solutions.

Equation (6) contains an unknown function $r(n)$ which represents the number of such combinations, which cannot be considered as solutions because of missing components. In order to obtain its value it is required to analyze a content of the combination, which is hard to perform using analytical approaches. Because of that it was decided to obtain its values experimentally.

Special software was developed by the author in order to support experimental calculations of the function – CoMBot-Gen, which stands for **Co**mbination **Gen**erator for **M**ulti-Ro**bot** system specification problem. The main feature of the software is the use of a special algorithm for generation of solutions (described below). The combinations are organized in tree structure, which in turn allow generation of combinations on demand. According to algorithm the list of possible agents is sorted at first, and then added to the tree structure on root level. Next, child nodes are populated recursively on demand, when particular parent node is expanded. Only such agents are considered as s child nodes, which are positioned behind current agent within ordered list. This rule ensures that tree does not contain repeating branches. The solution (the combination of agents) is obtained by selecting particular node the tree and recursively fetching agent information from its parent. Figure 4.1 shows an example tree for mission defined using three components. Yellow nodes indicate valid solutions, while grey nodes stay for incomplete agent combinations.

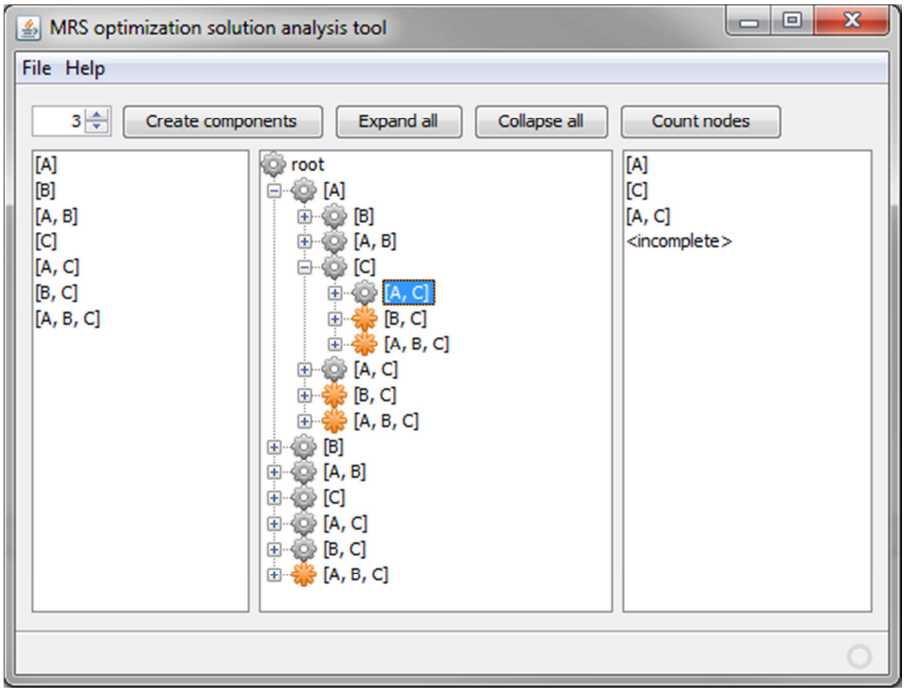


Figure 4.1. **Agent combinations analysis software**

Table 4.3 demonstrates the total number of combinations of agents calculated using aforementioned equations, as well as the values experimentally obtained using solution analysis software.

Table 4.3. **Number of components, agents and their combinations**

Components	**Agents,** $f(n)$	**Combinations,** $g(n)$	**Invalid combinations,** $r(n)$	$\boldsymbol{r(n)/g(n)}$, %
1	1	1	0	0.00
2	3	7	2	28.571
3	7	127	18	14.173
4	15	32 767	470	1.434
5	31	2.15×10^{9}	162 630	0.008
6	63	9.22×10^{18}	$\approx 3.36 \times 10^{11}$ *	≈ 0
7	127	1.70×10^{38}	$\approx 5.94 \times 10^{15}$ *	≈ 0

* – values extrapolated using exponential regression.

As it can be seen from the table, the total number of combinations of agents grows extremely fast and reaches computational limits of modern systems rapidly. The same is also confirmed by its diagram. It has exponential trend line when plotted on logarithmical axis (see figure 4.2).

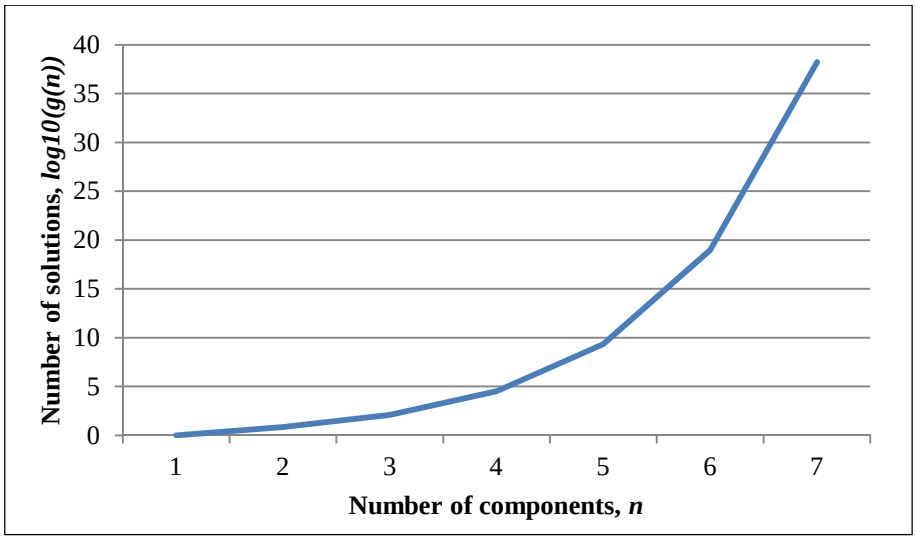


Figure 4.2. **Number of solutions plotted on logarithmical axis**

The values of $r(n)$ function were experimentally obtained for cases with up to 5 components. Greater parameter values lead to overflowing errors on computation hardware used within the experiments. However the trend of the function is obtained. Absolute values of $r(n)$ function grow slightly faster than pure exponential function (see exponential trendline on figure 4.3), which means that the function lags behind $g(n)$ function.

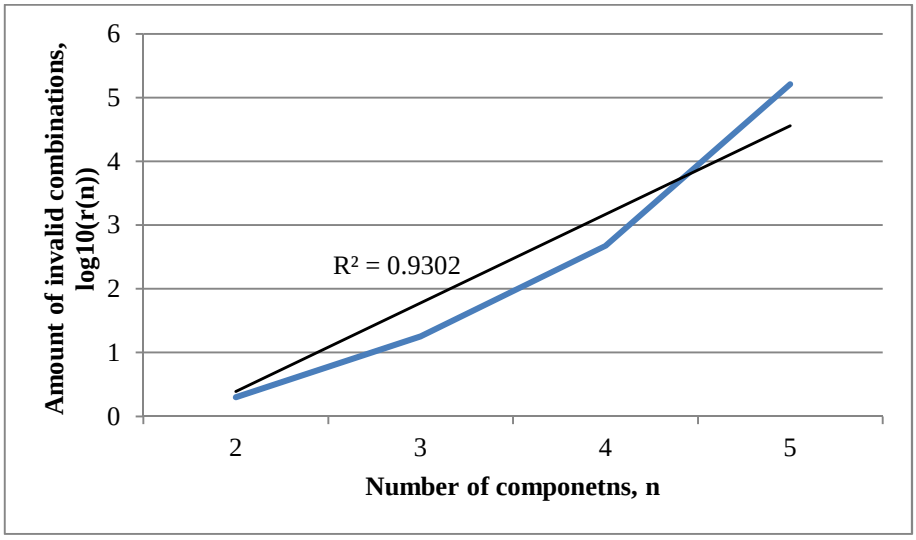


Figure 4.3. **Absolute values of *r(n)* function plotted on logarithmical axis**

Relative amount of invalid combinations was analyzed in the context of all combinations: $r(n) \,/\, g(n)$. Despite the fact that absolute amount of invalid combinations grow exponentially, the amount of invalid combinations compared to total number of combinations converges to zero percent. Thereby relative value of $r(n)$ function goes delow 0.01 % at five components (see figure 4.4).

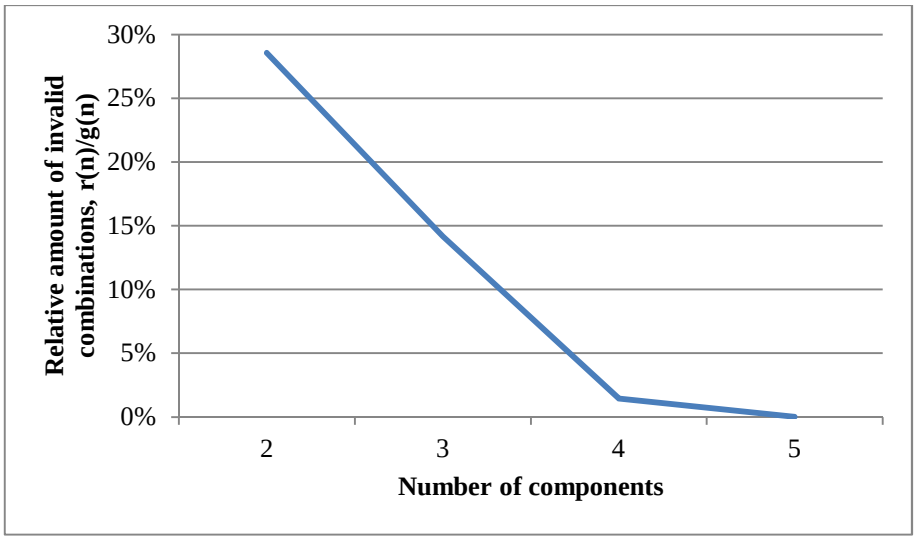


Figure 4.4. **Relative value of *r(n)* function**

Taking into account aforementioned considerations and the fact that values of $r(n)$ function can be obtained only experimentally, it is reasonable to ignore it in final conclusions about solutions space of specification optimization problem. Also previously provided analysis implied that there is a limitation of single instance per any agent ($A = 1$). Adding it to final equation will grant additional exponential degrees for the function and will let it grow even more rapidly. Also it is not reasonable form practical point of view because it leads to use of large numbers.

Completing analysis the final equation used for evaluation of solution domain for specification optimization problem is as follows (7):

$$G(n) = 2^{2^n - 1} - 1 \tag{7}$$

where

$G(n)$ – considerable number of solutions;

n – number of defined components for the mission.

Practical example

Returning back to grass mowing example, it is worth to obtain an evaluation of the expected solution domain. There are 10 explicitly defined components for the grass mowing mission. The number of possible agents equals to $f(10) = 1023$, which result in $G(10) \approx 8.99 \times 10^{307}$ possible combinations.

Taking into account developed constraints (see 3.5) the number of considerable agents reduced down to $f'(10) = 211$ agents. The number of solution candidates passed to the next steps of the specification optimization procedure is estimated to $G'(10) \approx 3.29 \times 10^{63}$ combinations.

4.3. Selection of suitable optimization approach

Previous chapter provided analysis of solution domain for specification optimization problem and demonstrated that the number of feasible solutions usually is beyond computation capacity of modern hardware. Similar circumstances are met by most of other combinatorial optimization problems. In general, combinatorial optimization is a topic in applied mathematics which stands for finding an optimal object in the finite set of other objects. Applications of combinatorial optimization include such fields as logistics, production industries, military and others. Common practice used for solving combinatorial optimization problems implies application of heuristic methods (Hromkovic, 2010; Korte, Vygen, 2012).

In general, heuristic stands for a rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood. Unlike algorithms, heuristics do not guarantee optimal, or even feasible, solutions and are often used with no theoretical guarantee (Heuristic, 2012; SH Zanakis, Evans, 1981; Helsgaun, 2000).

According to the specification optimization procedure the solution domain analysis is performed with aim to reveal number of expected solutions and to select appropriate optimization method (see 2.4). Steps 5 and 6 of the procedure are intended for evaluation of solution candidates and for selection the best options in terms developed criteria. In other words, defined optimization task is being resolved during these steps of the procedure.

As it was demonstrated on flowchart of the procedure (see figure 2.2, step 4) there are available two options of steps performed after the space of solutions is analyzed. First, it is possible to proceed directly to fine evaluation of solution candidates using simulations (step 6). Second option implies further analysis of solution domain with the aim to reduce it (step 5). The selection of next step depends on user expectations as well as from computational capacities available for processing.

In terms of accuracy of final results it is reasonable to process all possible solutions using fine evaluation, which is based on simulations. However this will require enormous computational resources and time, which is usually undesirable. Because of that additional step is added to specification optimization procedure, which is intended for initial evaluation of solution candidates. Taking into account the analysis of solution domain, performed within this section, it is not recommended to perform

fine evaluation on full space of solutions for missions, which are defined using four or more components.

4.4. Summary of the section

During the analysis performed within this section the scale of solution domain was estimated. A structural analysis of the solution domain is described providing fundamental considerations on solution generation process.

Custom formulas were developed for calculating the number possible agents combined from the components. Formulas for analytical estimation of the number of feasible combinations of the agents were proposed.

Custom utility software *CoMBot-Gen* was developed by the author used for analysis of the solution domain and aimed to the generation of possible combinations of specified components and agents. Near double exponential growth of number of solutions as a function of the number of defined component is found.

An expediency of application of the constraints developed in step 3 of the procedure was proved for the practical example. The constraints allowed narrowing the solution search space and reducing number of possible combinations by 244 orders.

5. INITIAL EVALUATION USING HEURISTIC METHODS

The fifths step of proposed specification optimization procedure stands for initial evaluation of solution candidates using heuristic methods. According to the procedure this step is optional and it is reasonable to skip it if an analysis of all the number of solution candidates is feasible. The main goal of this step is to apply initial (rough) evaluation of solution candidates and eliminate worse of them from future processing thus narrowing the space of considerable options. A genetic algorithm is utilized within the thesis for aforementioned purpose as a heuristic search method (Eiben, J. E. Smith, 2003).

The genetic algorithm as well as other evolutionary algorithms is inspired by organization and concepts of biological processes. The genetic algorithm is a search algorithm based on the conjecture of natural selection and genetics. The features of a genetic algorithm are different from other search techniques in several aspects. First, the algorithm is a multipath one that searches many peaks in parallel, hence reducing the possibility of local minimum trapping. Second, the genetic algorithm works with a coding of parameters instead of the parameters themselves. The coding of parameter will help the genetic operator to evolve from the current state into the next state with minimum computations. Third, the genetic algorithm evaluates the fitness of each candidate to guide its search instead of the optimization function. The genetic algorithm only needs to evaluate objective function (fitness) to guide its search. There is no requirement for derivatives or other auxiliary knowledge. Hence, there is no need for computation of derivatives or other auxiliary functions. Finally, the strategies employ genetic algorithm explores the search space where the probability of finding improved performance is high (K. Y. Lee, El-Sharkawi, 2008).

Optimization is the basic concept behind the application of genetic algorithms to any field of interest. Traditional optimization techniques begin with a single candidate and search iteratively for the optimal solution by applying static heuristics. On the other hand, the genetic algorithm approach uses a population of candidates to search several areas of a solution domain, simultaneously and adaptively.

Genetic algorithms have been most commonly applied to solve combinatorial optimization problems. Combinatorial optimization usually involves a huge number of possible solutions, which makes the use of other optimization techniques hopeless. In

problems of this kind, the number of possible solutions grows exponentially with the problem size. Therefore, the application of analytical optimization methods to find the optimal solution is computationally impracticable. Heuristic search techniques are frequently employed in this case for achieving high-quality solutions within reasonable run time. Also genetic algorithm has been applied successfully to real world problems, several of their crucial parameters have been selected empirically. Theoretical knowledge of the impact of these parameters on convergence is still an open problem.

Genetic algorithms (Holland, 1975) operate on a population of individuals called genotype. Each individual is a solution candidate to a given problem and is typically encoded as a fixed-length binary string, which is an analogy with an actual chromosome. After an initial population is randomly or heuristically generated, the algorithm evolves the population through sequential and iterative application of three operators: selection, crossover, and mutation. A new generation is formed at the end of each iteration. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified to form a new population. The new population is then used in the next iteration of the algorithm.

Application of genetic algorithm requires special attention on two challenging aspects, which affect overall results of heuristic search. First of all, a genetic representation of the solution domain should be developed. It should cover complete solution domain and be stable against local extremum. Secondly, a fitness function is required in order to evaluate the individuals. Quality of fitness function directly affects the results of optimization because solution candidates are selected for next generations according to their fitness. Next chapters provide detailed analysis of both genetic representation of solution domain and development of fitness function. Further chapters describe design of custom heuristic optimization software developed for initial evaluation for multi-robot system specification, which utilizes genetic algorithm in its kernel. The final chapter provides analysis of optimization results and conclusions.

5.1. Genetic representation of solution domain

As it was mentioned before, correct genetic representation of solution domain is vital for successful application of genetic algorithm. In biology, chromosomes of the cell contain all information of its genotype required for reproducing similar cells, for

protein generation, etc. In other words chromosome is a mapping of existing entity properties onto medium.

Computational genetic algorithms mimic biological processes in sense, that real solution is also mapped onto artificial chromosome – the sequence of bytes which is processed by computer. The creation of such a mapping requires some creative thinking because proteins and computer programs are very different things. Thankfully, human-built computer programs are much easier to understand than proteins and it is not necessary to know, for instance, the rules that determine the dimensional structure of proteins to create a simple genotype/phenotype system capable of evolving computer programs (Ferreira, 2002).

Genetic representation of solution domain for genetic algorithm stands for development of such data structure, which is suitable for computational processing and at the same time capable to encode solution candidate. Genetic representation for particular solution domain is characterized by several features. First of all, it has to cover full solution domain. In other words, data structure of chromosome has to be capable to encode any possible solution. This ensures that whole heuristic search occur on whole solution domain. Contrary, if chromosome is capable to encode only particular subset of solutions, then resolution of optimization task could be considered only as local extremum.

Next, genetic representation of solution has to ensure intrinsic class membership. This means, that solution candidate of particular class always produce valid structures for respective class. In other words, chromosome has to be stable against application of genetic operators and reproduce valid solutions through generations.

Final requirement is related to the previous requirement and also it depends on configuration of a genetic processor, described in next chapters. Genetic representation of solution domain has to ensure that evolution occur smoothly and efficiently. If chromosomes suppress application of genetic operators, or allow only partial evolution, then heuristic search will occur only within subset of feasible solutions. This in turn will result in local solutions which contradicts purposes of genetic algorithm itself.

Taking into account aforementioned conclusions a genetic representation of solution domain is defined for multi-robot specification optimization task. According to the proposed concepts, the specification of multi-robot systems stands for a set of agents, which in turn are composed of components. Solution of this optimization task is a specification of multi-robot system. Therefore tuning to the genetic representation of

solution domain it is possible to conclude, that chromosome (solution candidate) should define a set of agents.

In order to define data structure for such set of agents several options were considered. First of all, the question arises whether it is necessary to encode solution as fine as on level of components. From the theoretical point of view, providing finer details for genetic processing allows to reach finer results. From the practical point of view, taking into account peculiarities of multi-robot specification optimization task, such level of details is useless. Components are not used in solution directly; instead they define agent types which, in turn, are static. Thus it is computationally effective to define all possible agent types before actual genetic processing. Next, in case if components are involved into genetic processing, an additional validation is required to avoid the situation, when the same agent type is evolved multiple times instead of multiple instances of the single type.

Therefore genetic representation is developed on the level of details of agent types. Deeper analysis reveals additional benefits of this approach. Agent types are generated before actual genetic processing and this provides possibility to build additional level of constraints. For specification optimization task this level is used to eliminate pointless combinations from genetic processing, for instance, stationary agents with mobile actuators, etc.

The next decision is related to encoding of agent instances. Usual application of genetic algorithm implies use of binary genes within the chromosome. For the optimization of specification of multi-robot system this would mean that only agent types are encoded within chromosome. An additional encoding dimension is needed for the number of instances of agents.

Taking into account aforementioned requirement and the framework of genetic processing used by the author (see 5.3) the integer genes are used instead of bit genes. This type of genes allows storing integer values within defined range and is well suited for application of genetic operators. Also range definition corresponds to the limit of agent instances introduced in previous chapter (see formula (5)) and the set of values of agent genes is defined using formula (8).

$$v \in [0; A] \tag{8}$$

where

v – is a value of agent gene;

A – is artificial limit of agent instances used to avoid overflow.

The value of gene equal to 0 (zero) is considered as if particular agent type is not used within the solution. Considering integer genes in the context of genetic operators, for crossover operations the value of the gene is directly transferred to related chromosome, while during mutations the value of the gene is changed to random value within allowed range.

Finalizing analysis of genetic representation for specification optimization task of multi-robot system, the number of genes within chromosome is calculated based on the number of valid combinations of components (9).

$$f(n) = 2^{n-1} - C \tag{9}$$

where

$f(n)$ – is the number of valid combinations of components;

n – is the number of components;

C – is the number of combinations (agents), which do not pass initial level of constraints (depends on the definition of particular mission).

Schematic view of developed chromosome is provided on figure 5.1.

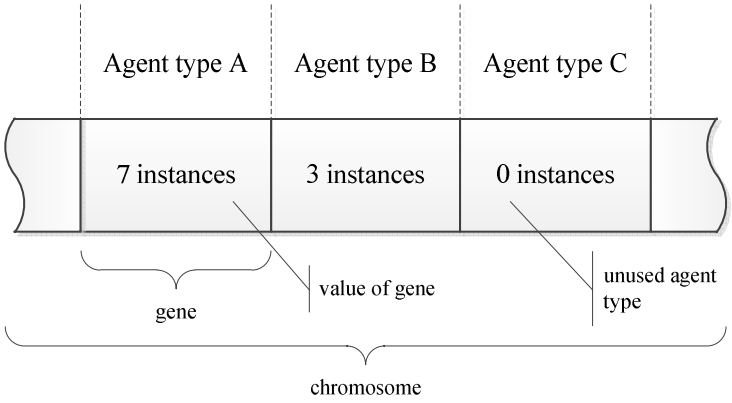


Figure 5.1. **Genetic representation of solution domain**

As it was confirmed by analysis developed genetic representation covers whole solution domain. Also it allows smooth evolution and application of the genetic operators.

5.2. Fitness function development

As it was stated before, another challenging aspect of genetic algorithm application is development of fitness function. The quality of fitness function directly affects the fidelity of the optimization, its performance and reliability. Well-designed fitness function may substantially increase the chance of finding a global solution and reaching higher solution domain coverage. It can happen that the customer (the one who

develops business requirements) is unable to represent the needs in form of fitness function correctly. Therefore a help of a modeler would be useful.

Fitness function for genetic algorithm stands for a numeric figure of merit, which is applied to each solution candidate within the genotype. The fitness value expresses the performance of an individual with regard to the current optimum so that different individuals can be compared. From the other side fitness value has to be explicitly calculated and should depend only on values modeled within the optimization task. For example, relating the fitness of the grass mowing solution to stock indices of customer's company is not rational.

Usually a spread of solutions exists ranging in fitness from very poor to good. The notion of fitness is fundamental to the application of evolutionary algorithms; the degree of success in their application may depend critically on the definition of a fitness that changes neither too rapidly nor too slowly with the design parameters of the optimization problem. The fitness function must guarantee that individuals can be differentiated according to their suitability for solving the optimization problem (Baresel et al., 2002).

The fitness function has several important features which ensure good applicability of the function. First of all, the fitness function has to cover whole domain of input parameters. In other words, the fitness function is designed improperly if it can't provide a fitness value for some solution candidate.

Next, it is advised that the fitness function has to be continuous. This means, that "small" changes in input parameters are reflected in "small" changes in fitness value. In other words, interruptions in the fitness function are not recommended. For the most of optimization tasks an example of absolutely invalid fitness function is binary function, which returns same value for all solution candidates except the optimum. In this case, genetic algorithm is unable to direct its evolution towards optimum, and only occasional application of genetic operator can lead to changes in fitness value. In this case evolution can take a lot of computational time, because of dominating random factor.

Final feature is related with the previous and states, that the fitness function has to guide heuristic search. Genetic algorithm works alike multiple gradient descent optimization algorithms applied on multi-dimensional solution domain. Thus, fitness function has to provide a guide for evolution to evolve towards better solutions. Good fitness function is featured by multiple distinct fitness values assigned to different

solution candidates. Identical fitness values within the single genotype lead to dependence of random factor and to longer processing times.

As it was described in previous sections (see 2.2), the author uses total costs of ownership (TCO) as universal optimization criterion for specification of multi-robot system. It is obvious that initial evaluation of proposed optimization procedure is also based on TCO. Thereby fitness function for genetic algorithm is also uses TCO as its core criterion for evaluation of solution candidates.

Next chapters describe various aspects of fitness function development process and provide deeper analysis on proposed positions.

5.2.1. Fitness function simplifications

Within proposed multi-robot system specification optimization procedure genetic algorithm is used for initial evaluation of solution candidates and for reducing the number of options for final evaluation. It is implied that such evaluation is relatively fast and rough. Therefore several assumptions and simplifications are used in order to increase computational throughput of genetic algorithm.

First of all, two positions of TCO are distinguished: 1) investment costs and 2) operating costs. Investment costs estimate the amount of investments, required to purchase multi-robot system. This includes purchase of hardware, costs for design and production of system agents. Operating costs estimate the amount of resources required for performing the mission. It includes resources for running the agents as well as maintenance costs. These positions are described in details in next chapters.

Next assumption is related to a core of costs estimation models. A special function $q(n)$ is used to calculate a price for some entity, which is dependent on number of sub-entities (n). An example of such entity is circuit board mounting costs which, obviously, depend on a number of the elements being mounted. The list of the entities related to the specification evaluation task is available in next chapters. This chapter provides only general analysis of this function.

According to the author's assumption, the function $q(n)$ should grow nonlinearly depending on the parameter n. This assumption is enforced by a derivation of costs for same type operations. It is obvious that any operation takes some resources no matter how complex this operation is. For example, mounting an element onto circuit board takes some time on production line (manipulating, soldering, etc.). Similarly, it can be

assumed that mounting another element of the same complexity on the board will take the same time for production line. Thereby, there is at least linear growth of $q(n)$ function which corresponds to costs of each operation.

Further analysis reveals derivation of another source of costs. It is hard to imagine an operation on production line except the simplest ones, which would take the same amount of resources without dependence on the total number of operations performed on the same object. For example, mounting single element onto the circuit board costs one unit of money. It would be wrong to assume that mounting 100 elements on single board would cost 100 units of money. Most probably different mounting technology and production line should be used for boards of such complexity. Therefore, in this case another overhead of costs appears for compensating complexity of the entity. The author assumes that this overhead grows nonlinearly, because linear growth does not correspond to industrial trends. In the example of a circuit board mounting linear growth would mean that mounting 10 elements instead of 1 will increase the costs of production by the same amount as increasing number of elements from 100 to 110 (in second case technology upgrade may be required).

Thereby, a nonlinear nature of $q(n)$ function is argued by peculiarities of industrial production. The author has analyzed several types of nonlinear functions which correspond to aforementioned description. The list of analyzed functions includes hyperbolic, logarithmical, power, exponential and other regressions. The power and exponential regressions where selected for deeper analysis according to their graphs. Hyperbolic regressions is also applicable for certain range of parameters, however it was eliminated from analysis because of other more suitable candidates.

Initially the author proposed to use an exponential regression as $q(n)$ function (Komasilovs, 2012a), which has two coefficients (see formula (10)).

$$q_{exp}(n) = b_0 * {b_1}^n \tag{10}$$

where

n – number of entities (input parameter),

$b_{0,1}$ – adjustment coefficients.

Exponential $q(n)$ functions were used for initial testing of genetic algorithm and demonstrated acceptable results. However the author faced an issue while defining coefficients in order to adjust this function according to desired costs model. The main difficulty appears because the coefficients do not correspond to any costs position and can be specified only by analyzing the full range of input parameters.

As a result, the author switched his proposal to power function which has four coefficients (see formula (11)).

$$q_{pow}(n) = b_0 + b_1 * n + b_2 * n^k \tag{11}$$

where

n – number of entities (input parameter),

$b_{0,1,2}$ – adjustment coefficients,

k – power coefficient.

These two functions have similar trend on the range of interest of the input parameter (see figure 5.2). Power regression (*Pow K*) shown on the plot uses following coefficient values: $b_0 = 10,\ b_1 = 5,\ b_2 = 0.02,\ k = 2$. Exponential regression uses following coefficient values: $b_0 = 15, b_1 = 1.16$.

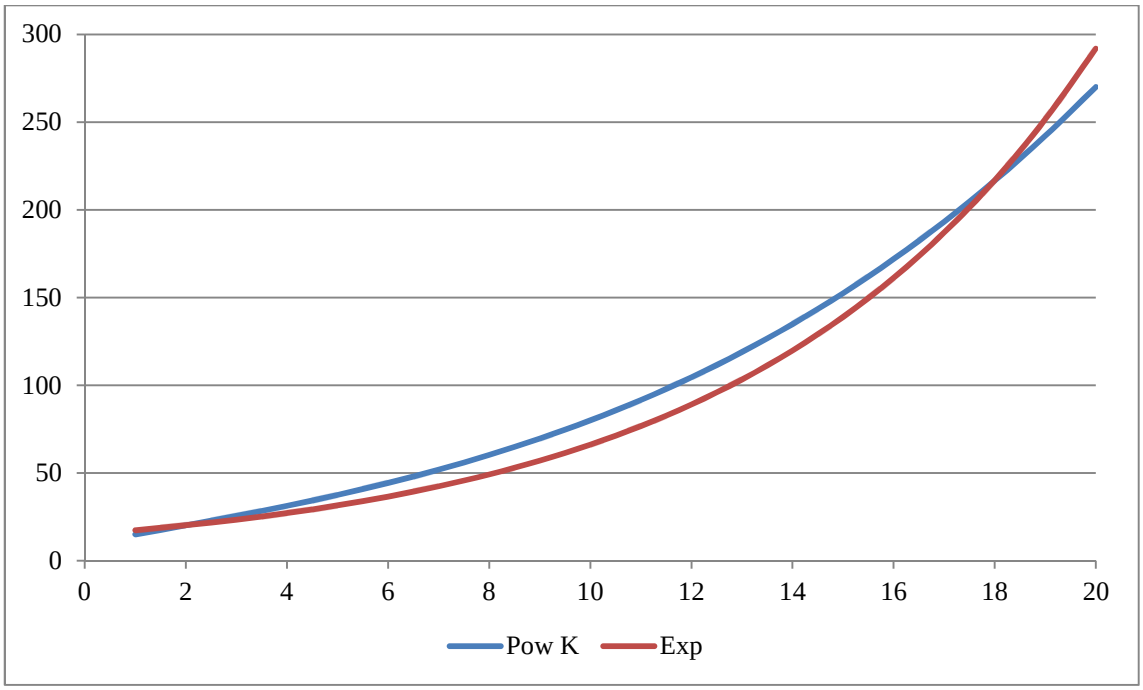


Figure 5.2. **Graphs of power and exponential regressions**

The main benefit of using power regression as a $q(n)$ function is that its coefficients correspond to costs positions, which facilitates their definition. Thereby, b_0 coefficient corresponds to minimal costs of an entity, or costs required for starting production. Coefficient b_1 corresponds to net costs of each entity. Coefficient b_2 correspond to overhead (in percent), which is required for every additional entity. Coefficient k is used to adjust the growth trend of the overhead.

Aforementioned $q(n)$ function, which is based on power regression, is used to model various positions of costs of multi-robot system. These positions are described in

details in next chapters, related to costs estimation models. For simplification of equation notation the function is supplied with a label and is written as $q_{<label>}(n)$.

5.2.2. Investment costs estimation

Investment costs concept used within the thesis stands for such spending which is required to create a robotic system for particular mission. Investment costs defines all expenses required to design, implement and deploy multi-robot system from the scratch into production environment and do not include expenses related to the operation of the system.

Since the purpose of the initial evaluation step is to process large number of solution candidates, proposed costs model is highly simplified. The author assumes that investment costs could be divided into several positions analyzed below.

Several concepts should be agreed before the actual analysis of costs estimation models. According to conceptual model of multi-robot system specification the mission for such system is defined using a list of components (see 2.3). Proposed costs estimation models assume that there are predefined special properties for the components which are used as basis for derived costs positions.

According to proposed estimation model investment costs of the whole robotic system consists of such positions as design costs of the system and investment costs for all agent types of the system (12). Moreover, design costs of the system grow depending on the number of agents in the system.

$$Q_{inv} = q_{sysDsgn}(n) + \sum_{k} Q_{invAgent} \tag{12}$$

where

Q_{inv} – investment costs of the system;

$q_{sysDsgn}$ – system design costs estimation function;

n – total number of agents in the system;

$Q_{invAgent}$ – investment costs of a particular agent type;

k – number of distinct agent types in the system.

It is assumed that the investment costs of a particular agent type consist of design costs of the agent type and of production costs for each instance of the agent type. The author assumes that agents are produced on the same enterprise, because of that each agent type should be designed only once, while production costs are relevant to each instance of the particular agent type.

Production costs include assembly costs which are estimated taking into account the number of components within agent. Additionally assembly costs is adjusted by maximum complexity index of components within the agent. The author considers that assembly of complex components costs more than assembly of simple ones. Also production costs includes a sum of prices of components which should be purchased from vendors (13). Complexity index and price of the components are defined by user at mission decomposition step (see 3.4).

$$Q_{invAgent} = q_{agDsgn}(n) + k \times \left(q_{agAssy}(n) \times \max_{n} P_{cplx} + \sum_{n} P_{price} \right) \qquad (13)$$

where

$Q_{invAgent}$ – investment costs of particular agent type;

q_{agDsgn} – agent design costs estimation function;

n – number of components within the agent;

k – number of instances of particular agent type in the system;

q_{agAssy} – agent assembly costs estimation function;

P_{cplx} – component complexity index (property);

P_{price} – component price (property).

Graphical representation of positions of investment costs estimation model and their dependencies for multi-robot system is available on figure 5.3. In general the model has tree-like structure but the figure shows only single branch expanded.

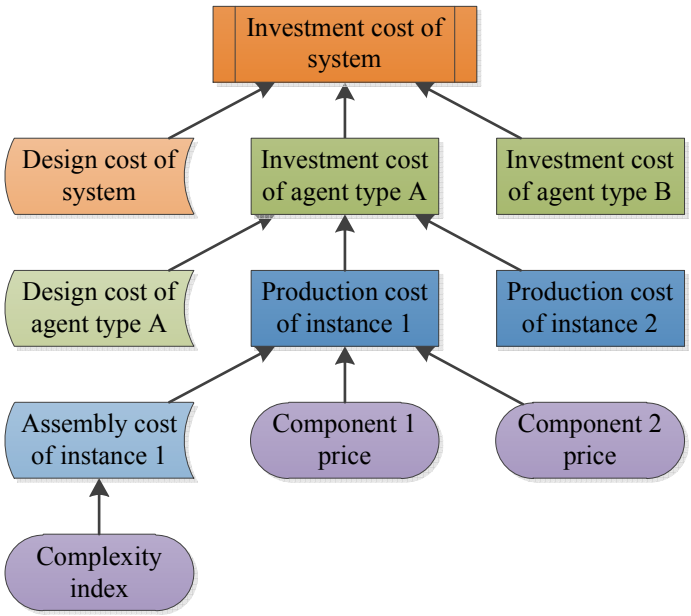


Figure 5.3. **Graphical representation of investment costs estimation model**

Investment costs estimation model described in this chapter is designed for high calculation performance and because of that has various simplifications. At the same time, proposed estimation model is based on analysis of real production peculiarities and therefore provide reliable results.

Practical example

For the practical example the author define coefficients for $q(n)$ functions as demonstrated in table 5.1. These functions are used to estimate various positions of investment costs as described above.

Table 5.1. **Coefficients of estimation functions for investment costs positions**

Coefficients	**Costs position**		
	System design $q_{sysDsgn}(n)$	Agent design $q_{agDsgn}(n)$	Agent assembly $q_{agAssy}(n)$
b_0	280.00	40.00	10.00
b_1	20.00	10.00	5.00
b_2	2.00	0.50	0.02
k	2.00	2.00	3.00

The coefficients are selected keeping in mind that design of an entity requires deep analysis of application peculiarities. At the same time assembly of an entity refers to repeating actions which are performed according to provided instructions and therefore is less expensive.

5.2.3. Operating costs estimation

Another position of TCO proposed for multi-robot system specification evaluation is operating costs. In general, operating costs can be described as the expenses which are related to the operation of a business, or to the operation of a device, component, and piece of equipment or facility.

Within the scope of the thesis the author defines operating costs as the expenses which are needed to perform particular mission specified for specification optimization procedure. Operating costs of the multi-robot system is highly dependent on application peculiarities of the system because of relations between mobile robots of different types and random deviations which affect the overall performance of the system.

The most precise method to estimate operating costs is to reproduce the operating environment in the costs model. However the aim of initial evaluation step of the proposed procedure is to reach high performance and process large number of solution candidates. Because of that some simplifications are assumed.

First of all it is assumed that agents have no downtime. It means that agents switch from one task to another instantly, and they are utilized for 100% of time until there are mission tasks to perform. In addition, this assumption leads to derivation that performance of the whole system equals to sum of performances of each agent. For example, if an agent is able to complete a task within 30 minutes, then it is assumed that two agents will complete the same task within 15 minutes, three agents – within 10 minutes and so on. In real life production total performance of the system is usually lower for loosely coupled agents, in contrast with highly coupled systems which demonstrate higher performance.

The second assumption is used to simplify calculations and it defines that particular agent can perform only one task at time. For output optimization of the whole robotic system this assumption is not acceptable. For example, a robot could perform communication session while traveling to different working site and such improvement could result in reduced working time and costs. However, taking into account the aim to evaluate large number of solution candidates this assumption reduces calculation complexity and speeds up processing.

According to proposal operating costs include such positions as energy, maintenance and eventual replacement expenses (14). Energy expenses are generalization of spending related directly to operation of agents which includes fuel, electricity, for some applications time, etc.

Maintenance expenses include spending which are not directly calculated from agent actions but still required for completing the mission. These include staff salaries, regular service, infrastructure, deprecation and others costs. Maintenance costs per time unit are estimated depending on the number of agents within the system. The author assumes that maintenance of a complex system is more expensive than maintenance of a simple system. The total operating time of the whole system also affects the amount of maintenance expenses and is equal to the maximal operating time of agents of the system.

Eventual replacement costs stand for unplanned expenses, which could occur during system operation. This includes agent faults, deprecation, disasters, accidents and other. Because of occasional nature of this costs position the author propose to apply risk justification installments on regular basis. The amount of installment depends on investment costs of whole system which is modified by eventual replacement rate

coefficient. These considerations are confirmed by assumption that within long enough period of time every element of the system will be replaced.

$$Q_{oper} = Q_{enrg} + \left(q_{mnt}(n) \times t_{sys}\right) + \left(Q_{inv} \times c_{rate} \times t_{sys}\right) \quad (14)$$

where

Q_{oper} – operation costs of the system;

Q_{enrg} – energy expenses of the system;

q_{mnt} – maintenance costs estimation function;

n – number of agents in the system;

t_{sys} – total operating time of the whole system;

Q_{inv} – investment costs of the system;

c_{rate} – eventual replacement rate, defined by mission.

Energy expenses form major part of operating costs of the system. Also it is obvious that this position is highly dependent on operating time of each agent of the system. In order to simplify further analysis an abstract power and time units are used for defining energy expenses. Thereby considered concepts have operating power consumption and operating time indicators.

According to the concept of the specification of multi-robot system, there are one or more tasks defined for global mission of the system. Each task define features and, which is more important, the amount of work to be performed by the system. Each agent of the system may be suitable to perform one or multiple tasks. Thereby, the energy expenses of the whole system depend on time which particular agent spends doing particular task.

Additionally each agent has its own power consumption indicator which depends on power consumption properties of its components. The author assumes that some amount of power is lost for maintaining the agent inner facilities, like infrastructure, wiring, etc. The amount of loss is estimated based on the number of components within the agent. Power consumption properties of the components are defined by used at mission definition stage.

Final energy expenses are calculated using formula (15) which defines sum of all aforementioned derivations.

$$Q_{enrg} = \sum_{n}\sum_{m}\left(t_{agent} \times \sum_{k}\left(P_{pow} + q_{powLoss}(k)\right)\right) \quad (15)$$

where

Q_{enrg} – energy expenses of the system;

n – number of agents within the system;

m – number of tasks within the mission;

t_{agent} – agent operating time doing particular task;

k – number of components within the particular agent;

P_{pow} – power consumption indicator of the component (property);

$q_{powLoss}$ – power loss estimation function of the agent.

Graphical representation of positions of operating costs estimation model and their dependencies is available on figure 5.4. Some repeated branches of the tree are collapsed.

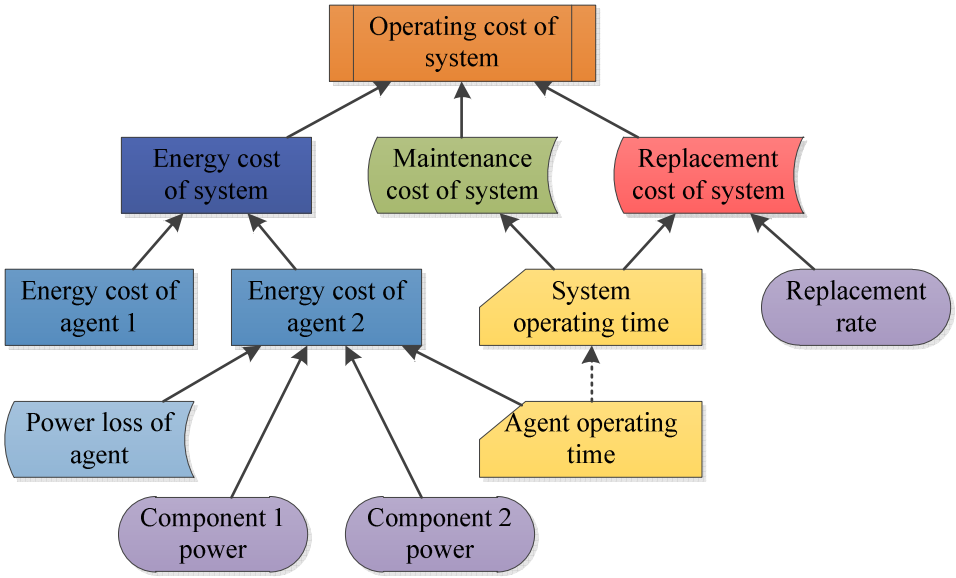


Figure 5.4. **Graphical representation of operating costs estimation model**

Operating costs estimation model described in this chapter has various simplifications, but at the same time provides reliable results. Developed operating costs estimation model has several input parameters defined as component properties. However one unknown variable is left uncovered. That is operating time of an agent doing particular task. Estimation of this parameter is not trivial and is described in next chapter.

Practical example

For the practical example the author define coefficients for $q(n)$ functions as demonstrated in table 5.2. These functions are used to estimate various positions of operating costs as described above.

Table 5.2. **Coefficients of estimation functions for operating costs positions**

Coefficients	**Costs position**	
	System maintanence $q_{mnt}(n)$	Agent power loss $q_{powLoss}(n)$
b_0	8.00	0.00
b_1	2.00	1.00
b_2	0.10	0.01
k	2.00	2.00

System maintenance and agent power loss is estimated per time unit. Because of that the coefficients are relatively small, but during working time of the system these positions obtain value.

An eventual replacement rate c_{rate} was selected equal to 0.005 for practical example. This results in moderate eventual replacement costs for relatively long missions for robotic system.

5.2.4. Agent operating time estimation

As it was clarified in previous chapter the model for operating costs estimation of multi-robot system requires the time estimation for each agent in the system. This estimation becomes non-trivial for missions that define more than one task and has multiple agent types. However the proposed specification optimization procedure for multi-robot systems is expected to be universal up to some degree. Because of that more complex combinations of mission tasks and agent types should be considered.

The main question in agent time estimation is how to assign working time for particular agent keeping in mind the operating costs of the whole system. The simplest case implies a mission defining a single task for a single agent. Obviously the agent will spend as much time as it needs to complete the defined amount of work. Here appears parameter of agent which defines the speed of doing the work – agent performance.

In more complex scenario the system of two agents of the same type will be able to complete the same amount of work twice as fast. More complex example implies that system contains agents of multiple types. In this case each agent will complete amount of work proportional to its performance while the operating time will be even for all

agents. The author assumes that highest time utilization of an agent is one of the key factors for cost effective system.

Contrary to the previous example, a mission with multiple tasks requires more complex agent time estimation methods. In case if there are dedicated agent types for each task the estimation of operating time could be performed within multiple iterations of simple approach. However an ambiguity appears for agents which are capable to perform multiple tasks. It is assumed that agents can perform only one task at time, therefore performance and operating costs of agents has to be considered while assigning tasks to agents.

The operating time estimation for agents could be investigated as classical transportation planning problem. From the one side there are mission tasks which should be completed. In terms of transportation these are consumers who request certain amount of goods to be delivered. From the other side there are agents capable to perform one or multiple tasks. These correspond to suppliers of various goods. Each route from supplier to consumer has its delivery price (distance) and capability (car type). In terms of agents these are operation costs and working performance of an agent per time unit. The aim transportation is to deliver all requested goods in cheapest way.

Transportation planning problems are widely investigates within a field of logistics and there is a bunch of methods used to solve this type of problems. The main problem of their application is that time estimation for agents should be performed for all specification solution candidates. In terms of genetic algorithm such operating time estimation should be done for all chromosomes through all generations of evolution. There is number of approaches tested by the author described below with supplementary analysis.

The most common and simplest way to solve aforementioned transportation planning problem is using **linear programming** (LP). The problem is converted into a set of mathematical inequalities (constraints) and objective function. The author hasused Simplex (Nelder, R. Mead, 1965) method for finding solution. This approach produces fast and reliable results and in most cases is the best selection for problems of similar type.

However there is an issue in application of LP for operating time estimation of agents. LP uses only operational costs of particular agent doing certain task and produce solution, where cheapest agent is exclusively assigned to the particular task. However the total operating costs of the whole system contains also maintenance and replacement

positions. These positions depend on the operating time of the system which in turn is calculated from operating time of agents. Therefore solution found by LP is optimal in terms of energy costs but not in terms of operating costs of whole system.

In order to overcome aforementioned drawback additional positions are added to objective function thus transforming it to **non-linear programming** problem (NLP). These types of problems are more complex and selection of solving methods depends on peculiarities of problem definition. The author has tested Generalized Reduced Gradient (Lasdon et al., 1978) method for NLP problem solving. The obtained solution had high accuracy, time estimation was made taking into account operating time of the whole system. As a result the tasks were distributed among agents in order to reach higher utilization of their functions and to lower total operating time of the whole system. However the performance of NLP was not acceptable, time estimation for single solution candidate took about 15 seconds in average on modern PC (CPU x4 @ 3.3 GHz, 4GB RAM). Taking into account peculiarities of the genetic algorithm and the fact that each solution candidate has to be evaluated on every generation the processing would require unfeasible about of time even if executed on computational clusters (about 95 CPU-years).

Another approach tested by the author implied application of **another instance of genetic algorithm** for working time estimation (Komasilovs, Stalidzans, 2012a). For each solution candidate separate optimization task is defined for genetic processing. The setup of secondary genetic algorithm software is aimed on fast calculation therefore size of the population and number of the generations is set to minimal values. An accuracy of obtained results is comparable with NLP methods while processing speed is faster. However the overall performance of the approach is still unacceptable.

The author proposed own algorithms for operating time estimation inspired by **greedy approach**. The results produced by NLP and secondary genetic algorithm could be considered as optimal. These results had a strongly marked feature: operating time was distributed among agents fairly evenly. Particularly this is explained by difference between agent energy expenses and maintenance costs of the system, which is usually several times bigger than the first position. Therefore the author has used this feature as inspiration for development of fast time estimation algorithm.

The straightforward greedy approach tested by the author implies assignment of operating time to the cheapest agent-task combination. Taking into account total system operating time affected by operating time of each agent the algorithm has iterative

nature. In each iteration only single unit of time is assigned to cheapest agent-task combination. Next iteration assigns operating time to agent-task combination with minimal current operating time. In other words operating time is assigned to such positions which lead to minimal increase (delta) of total operating costs. As it is usual for greedy algorithms such operating time assignment leads to locally optimal solution. Using this approach the author was able to reach 135% mark of NLP results on the same testing sets.

The author has tested another, more **intellectual greedy approach** for the estimation of the operating time. At the beginning the algorithm assigns all required operating time to the cheapest agent just like LP approach does. Then iteratively distributes the time among other agents taking into account decrease of the total operating costs. Reverse optimization direction (decrease of costs) demonstrates fine result ($\pm 2\%$ difference from NLP results on the same testing sets). Remarkably, on certain testing sets this approach demonstrated better results than NLP algorithm.

However aforementioned algorithm leads to poor performance of the whole evaluation system. The main reason for this is iterative nature of the operating time estimation algorithm. This algorithm is well suitable for simple missions with relatively small number of time units required for completing them. But for practical example considered within the thesis amount of work for the robotic system is calculated in thousands of time units. Processing time increases depending on the expected amount of time required to complete the mission.

Taking into account operating time estimation peculiarities and **even distribution** of operating time among the agents the author proposed to use average values. Thereby, an average amount of time was calculated based on amount of required work and total performance of the whole system. Then rounded up values are assigned to each agent (16).

$$t_{agent} = \left\lceil W_{goal} / \sum_{n} p_{agent} \right\rceil \tag{16}$$

where

t_{agent} – operating time of an agent for particular task;

W_{goal} – required amount of work within particular task;

n – number of agents in the system;

p_{agent} – performance of an agent for particular task.

Tests demonstrate good performance and at the same time the approach provide fine results. The author reached mark of 115% of NLP results which could be considered as acceptable taking into account processing speed and amount. This approach provides fine results for common mission definitions. The difference from optimal result increases in case if there are multiple types of agents suitable for the same task but with very different performance indicators. In fact such case is not common because agent performance is calculated based on features of its components and, according to the concept of solution, it is unlikely that the agents composed from totally different components will be suitable for the same task.

Current chapter provides analysis of various approaches for estimation of operating time of agents. The author considered linear programming, nonlinear programming and secondary genetic algorithm as global optimization methods, two types of greedy algorithms. Finally mathematical time estimation was proposed based on average performance of agents. This operating time estimation approach is embedded into fitness function used by genetic algorithm.

5.2.5. Additional fitness value adjustment positions

In addition to aforementioned investment and operating costs positions fitness function used for genetic algorithm includes several justifications which are not directly related to expenses.

Fitness value of a chromosome is used to evaluate particular solution candidate and to rank it among other candidates. According to its fitness the solution candidate is selected (or not) for next generations. Thereby evolution is guided towards global solution using fitness value of its individuals. Keeping in mind aforementioned considerations the author proposes to use special adjustment positions in fitness function in order to eliminate blind evolution branches on early stages.

There is a challenge to develop such indirect indicators for solutions candidates which could be used to predict blind branch of evolution. The author has performed an analysis and found one very trivial indicator.

According to solution concept there are several rules which should be fulfilled by solution (see 2.3). First of them define that all defined components should be used within the solution at least once. Developed genetic representation of solution covers full solution domain (see 5.1). However, it also allows combinations with certain

components missing. For example, there is a possible valid solution candidate with only single agent. And if this agent is not composed from all defined components, then all other components will be missing in a solution as whole. Therefore such solution candidate could be considered as invalid and can be eliminated as early as possible.

In order to implement aforementioned derivations the author proposes to increase costs of such system artificially thereby decreasing its fitness and a probability that such solution candidate will be selected for next generations. From the other side it is possible that there are multiple incomplete solutions within the population and they should be compared to each other.

In order to ensure comparison of incomplete solution candidates the author proposes to use a numeric indicator showing the level of incompleteness of particular individual. The number of missing components is suitable for role of such numeric indicator: the larger number of components is missing the more incomplete the solutions are. This leads to situations, that solutions candidates with missing single component are considered better than the candidate with missing ten or more components. Finally, the adjusted fitness value for incomplete solutions is calculated using formula (17).

$$Q_{inc} = n \times C_{inf} \tag{17}$$

where

Q_{inc} – fitness value for incomplete solution candidate;

n – number of missing components;

C_{inf} – infinite costs constraint.

Aforementioned fitness value adjustment is embedded into fitness function used by genetic algorithm for initial evaluation of specifications of multi-robot system.

Practical example

For the practical example C_{inf} constant was selected equal to 10^{12}. This value ensures that for incomplete solutions costs will be always higher than for valid but not very successful solutions.

5.3. *GAMBot-Eva* software

Specialized software was developed by the author within current research for initial evaluation of specifications of multi-robot system – *GAMBot-Eva*, which stands for **G**enetic **A**lgorithm based **Eva**luation of **M**ulti-Ro**bot** System Specification. The

software provides functionality which supports execution of 5th step of the proposed specification optimization procedure and performs heuristic search in wide solution domain. There is an implementation of genetic algorithm within a kernel of the system which carries out all required processing.

Software is developed using *Java SE* platform. Therefore it is available on any operating system which supports *Java Runtime Environment*. Software uses *JGAP – Java Genetic Algorithms and Genetic Programming Package* (Meffert et al., 2012) as a kernel for genetic processing. *MySQL* database and its client side JDBC drivers are used for persistence storage facilities. Software source code is available on public project site (Komasilovs, 2012c).

Next chapters describe design of *GAMBot-Eva* software in details and provide analysis of various aspects of its implementation.

5.3.1. Architecture of *GAMBot-Eva* software

This chapter describes general requirements and architectural design of the *GAMBot-Eva* software. The main aim of the software is to provide universal processing package for the initial evaluation of specification of multi-robot system. Another important requirement defines that the software should allow batch processing on dedicated server.

Taking into account aforementioned considerations an architectural design of the system includes three concepts as follows (Komasilovs, 2013):

- ✓ processing module, which executes genetic algorithm and manages evolution of its population;
- ✓ presentation module, which provides an user interface for viewing processing results;
- ✓ storage module, which ensures persistent data storage and exchange between first two modules.

The software is designed in a way to allow asynchronous processing of multiple populations of genetic algorithm. This is inspired by island model for advanced processing of genetic algorithm (Whitley et al., 1999). The software also provides access to the intermediate results. Thus user is able to follow the processing of genetic algorithm online.

In order to increase the versatility of the system it is designed to be parameterized. It allows evaluation task definition via parameters file. This makes the system very flexible and enables execution of multiple different evaluation processes at the same time in parallel. Graphical representation of the architecture of the *GAMBot-Eva* software is available in figure 5.5.

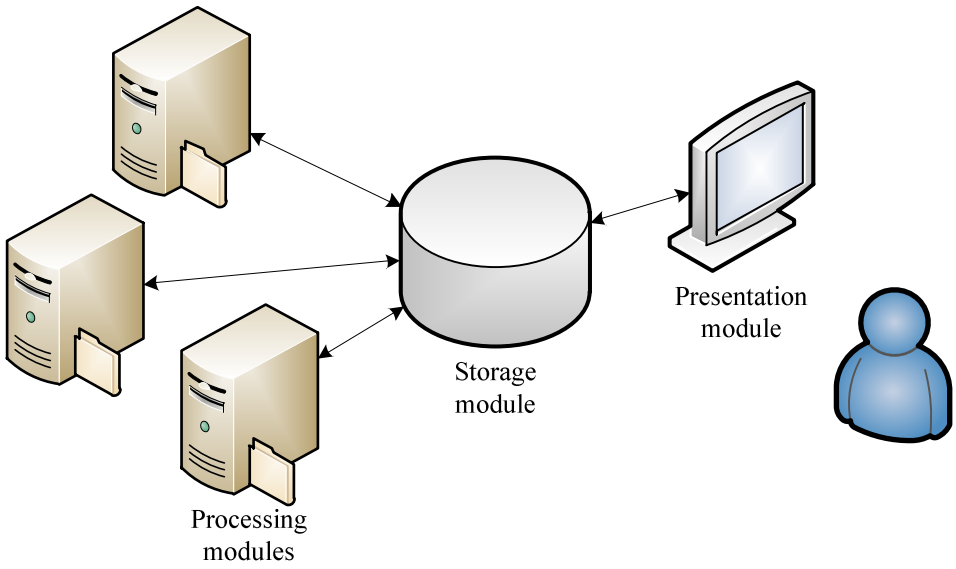


Figure 5.5. ***GAMBot-Eva* software modules**

The design of processing and presentation modules is described in details in the next chapters. Storage module is implemented using relational database. The software is not limited to any specific database management system. Instead it utilizes JDBC interfaces and drivers to interact with the database. For testing purposes the author has used *MySQL* 5.5 database management system which provide acceptable performance and at the same time do not require dedicated server (it was executed on developer level PC, CPU x4 @ 3.3 GHz, 4GB RAM).

5.3.2. Processing module of *GAMBot-Eva* software

The aim of the processing module of the software is to coordinate the execution of the genetic algorithm used for initial evaluation of the specification of the multi-robot system. Obviously, the kernel of this module is organized by the genetic algorithm engine. The author uses *JGAP* framework which provides tools for implementation of genetic algorithm engine in the custom software.

One of the most important concepts of *JGAP* is the configuration of genetic algorithm. Special singleton object is created for this purpose and it holds all information required to start genetic processing.

First, configuration defines the use of technical and low level objects in genetic processing. This includes such concepts as breeders, which are used to create and maintain the population of chromosomes, random number generators, which could use various randomization approaches, event managers, which manages event notification in the system.

Next, various parameters of genetic algorithm are defined in the configuration. The list includes such parameters as population size to be used by genetic algorithm, relative amount of chromosomes which are transferred to next generations, allowed variations in population size, chromosome pooling options and other parameters. These parameters directly affect the performance and capabilities of the genetic algorithm.

The next group of parameters defines genetic functions applied during evolution of genetic algorithm. One of the most important functions is the natural selection, which transfers the fittest individuals to the next generations. For genetic algorithm this function is mandatory. Also there should be at least one genetic operator for modifying individuals during evaluation. Any genetic operator is allowed in *JGAP* framework. The most common operators are mutation and crossover. The configuration defines also application parameters of genetic operators (frequency, rate, etc.).

Another important configuration parameter is the sample chromosome. This parameter is highly important because it defines genetic representation of the optimization problem. A single chromosome is created with all relevant genes. During genetic processing this chromosome is used as a sample for breeding population.

Finally, the fitness function is defined in the configuration. Technically fitness function is an object which implements special abstract class with protected evaluation method. Thereby separate class is defined for fitness function, and an instance of it is passed to the configuration of genetic algorithm. The fitness function directly implements the costs estimation model described in previous chapter (see 5.2). However in default implementation of *JGAP* framework chromosomes with larger fitness values are considered better (maximization task). But for the specification evaluation task smaller fitness values mean better solution (costs are about to be minimized). Because of that special fitness evaluator with inverted logics was implemented.

As it is shown above the configuration object of genetic algorithm holds all required information required for genetic processing. Thereby a genotype (population) is created according to configuration. Initially it consists of randomly generated individuals. Next, evolution takes place which is executed iteratively or in batch. During

the evolution the information of the whole population is available, however only the fittest chromosome is usually retrieved from the population. It is considered as solution at particular stage of evolution (generation).

In addition to *JGAP* facilities special concepts are defined in order to support genetic algorithm processing and to store results according to solution design. These concepts are defined by the author and are relevant only within the scope of the thesis.

First, the author defines a project for each execution of initial evaluation of specification of multi-robot system. The project is used as wrapping entity for all other concepts and it defines the evaluation task for genetic algorithm. In addition to various parameters described below, the project stores parameters of genetic algorithm configuration and costs estimation model.

According to solution concept, the mission for robotic system is defined using components. Thus the project holds a list of components and their properties which are relevant to defined mission. In addition the inter-component requirements are defined removing initially invalid solution candidates from processing.

By analogy with the components the project holds definition of mission tasks. The main reason of defining mission tasks at initial evaluation stage is because operating costs estimation model requires such parameters as amount of work to be done and performance of agents doing particular task. Definitions of mission tasks include general parameters, references to mandatory components and their performance indicators.

The concepts described above are defined by user and are provided as an input to processing module using XML configuration file (see annex 1). There are additional technical concepts used for various aspects of processing (Komasilovs, 2012b). Thereby, the list of agents is generated from components taking into account their inter requirements. Special type of genes is defined which combines integer gene properties and agent specific facilities. These genes are used in genetic processing. Caching technique is used for derived values in order to speed up processing.

Graphical representation of conceptual design of processing module is shown on figure 5.6. Orange boxes correspond to the concepts of specification optimization solution (see 2.3). Blue boxes correspond to technical objects developed exclusively for *GAMBot-Eva* software. Green boxes indicate external objects imported from *JGAP* framework.

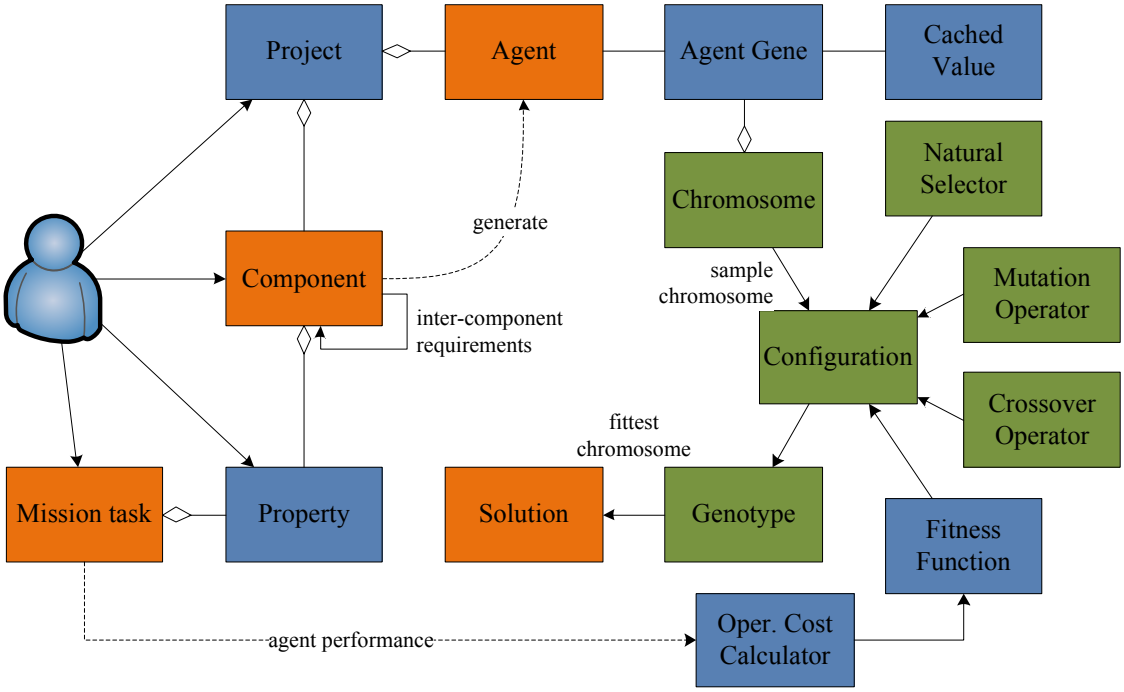


Figure 5.6. **Conceptual design of processing module**

In addition a lot of technical objects were created in order to implement certain programming templates and to support desired functionality of the module. These objects are not shown on conceptual diagram but they are used to make the implementation of the module flexible and highly extendable. In total there are 34 new *Java* objects are defined which include classes, interfaces, desktop forms and internal libraries.

Aspects of persistent storage of processing results are also worthy of special attention. The main aim of the data persistence is to allow deep analysis of evaluation results on later stages. Thus the system should store all data obtained during the heuristic search. In order to reproduce the evaluation process the author stores initial conditions before starting genetic processing. Basically this means recording of full project definition including the list of components and mission tasks, their properties, derived agents, coefficients of costs estimation model and other information.

Next, information about genetic evolution should be stored. The best option would imply recording the state of all chromosomes on every generation. However, taking into account the amount of data to be stored the author found this approach unpractical (the time spent for persisting information about single generation extends the time required to calculate it).

The author has tested several data storage approaches and found that information about all chromosomes of population is nearly useless except the information about the

fittest chromosome. Next, the author has noticed that the fittest chromosome (the solution) is changing frequently at the start of genetic processing and almost stops changing later during the optimization. It is unreasonable to store the identical best chromosomes of every generation. Therefore the implemented approach imply recording only of such generations when the fittest chromosome changes.

Moreover, in order to speed up processing of genetic algorithm multiple generations are executed in batches. The size of batch (the number of generations to execute as single entity) changes dynamically. Initially the size of batch equals to one generation. During the evolution if the fittest chromosome is not changing then the size of batch is increased by one up to pre-defined maximum batch size. This approach ensures small batch size at the start of evolution when the fittest chromosome changes frequently and at the same time speeds up processing when the changes are rare.

Practical example

During practical testing the author hasused various combinations of parameters provided to processing module using XML configuration file. The most important practical experiments and their results are analyzed in chapter 5.4. The general setup and parameters of the processing module is described in the following paragraphs.

The size of population of genetic algorithm is set to 200 species. This value is found as a compromise between processing speed and the diversity of solution candidates in the population. Larger populations are processed much longer, while smaller populations evolve slowly to global optimum and tend to stack in local extreme. The limit of generations is set to 5×10^5, for some experiments it is extended up to 1.5×10^6. The maximal size of evolution batch (the step) is set to 10^3.

During evolution a number of genetic operators is applied. The crossover is applied to 35% of chromosomes. Mutation is applied to about 7% of chromosomes in average. The natural selection transfers 95% of chromosomes to the next generation depending on their fitness value.

In addition duplicate chromosomes within the population are disabled. That means that only unique solution candidates are maintained. If two equal individuals are found in the population, one of them is replaced with randomly initialized chromosome. This option highly improves solution searching speed because identical chromosomes tend to fill up whole population with locally optimal solutions and stall the search process.

5.3.3. Presentation module of *GAMBot-Eva* software

The second module of *GAMBot-Eva* software is presentation module, which provides a graphical interface for user to view and analyze the results of the initial evaluations. In general it uses the main concepts of specification optimization solution and graphically presents the data which is generated by the processing module of the software.

The main feature of the presentation software is to provide a tool for monitoring and analysis of asynchronously running evaluation processes. The module fetches data from persistent storage and presents it graphically to user. Keeping in mind defined application peculiarities three different views are defined in user interface of presentation module.

The first is the process view (see figure 5.8), which provides an overview of the running or already finished processes on computational server. In general this view presents the information from the project concept defined in the previous chapter. This includes the *Id* of the project, start and end time of execution. Also this view specially indicates currently running processes. The latest information about these processes is loaded when it is available in persistent storage.

The second view is the evolution view. It provides a graphical representation of evaluation process and shows the chart of solutions (fittest chromosomes) of particular generation. The evolution view uses external *JFeeChart* engine for low level graphical processing and chart drawing. Several special approaches are used to increase readability of the charts. One of them is the logarithmical scaling of axis. As it was described above the frequency of changes in fittest chromosome is decreasing as it approaches the end of evolution. Therefore it is reasonable to show finer details at the beginning of the process when changes are frequent and to show less details when the changes are rare.

In addition the evolution view supports special analysis supporting features. One of them is the possibility to show multiple processes on the same plot. This enables comparison of the results, processing dynamics and parameters of different processes. The other feature is the zooming facility which allows analyzing only particular spots of the graph in much finer details.

The third view is the solution view. It displays the details of a particular solution selected in the evolution view. In general it shows the specification of the multi-robot

system. This includes the types of agents used in the system, their components and the number of agent instances of each particular type.

Graphical representation of conceptual design of presentation module is shown on figure 5.7. The orange boxes correspond to the concepts of specification optimization solution (see 2.3). The blue boxes correspond to technical objects developed exclusively for GAMBot-Eva software. The green boxes indicate objects of external framework.

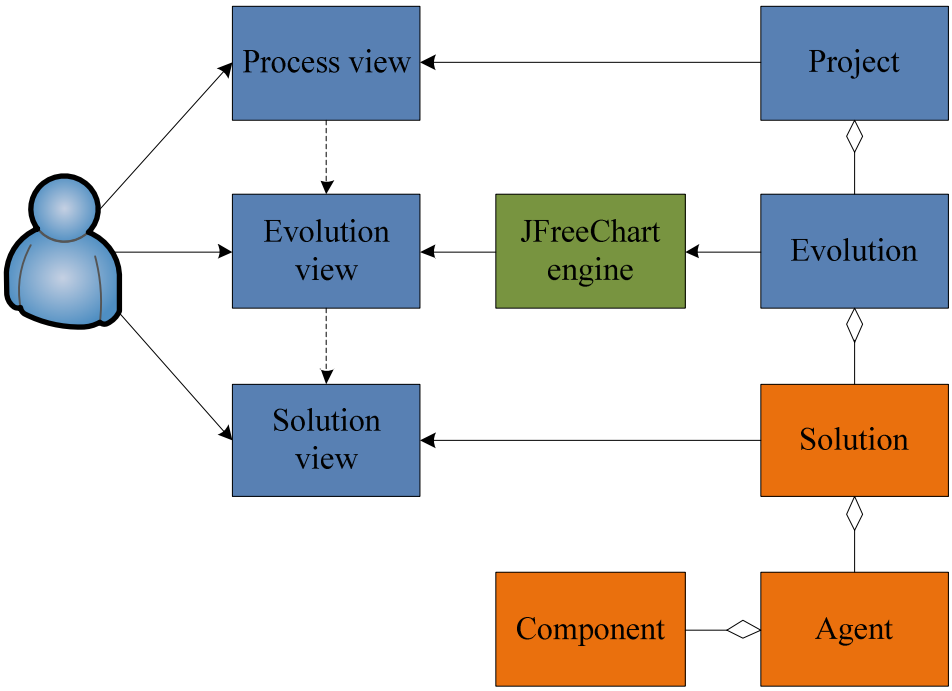


Figure 5.7. **Conceptual design of presentation module**

A screenshot of user interface of presentation module is available on figure 5.8. There are all three views of the module demonstrated on the figure. Two projects with different parameters are selected for chart. Detailed analysis of these projects and their parameters is provided in next chapter.

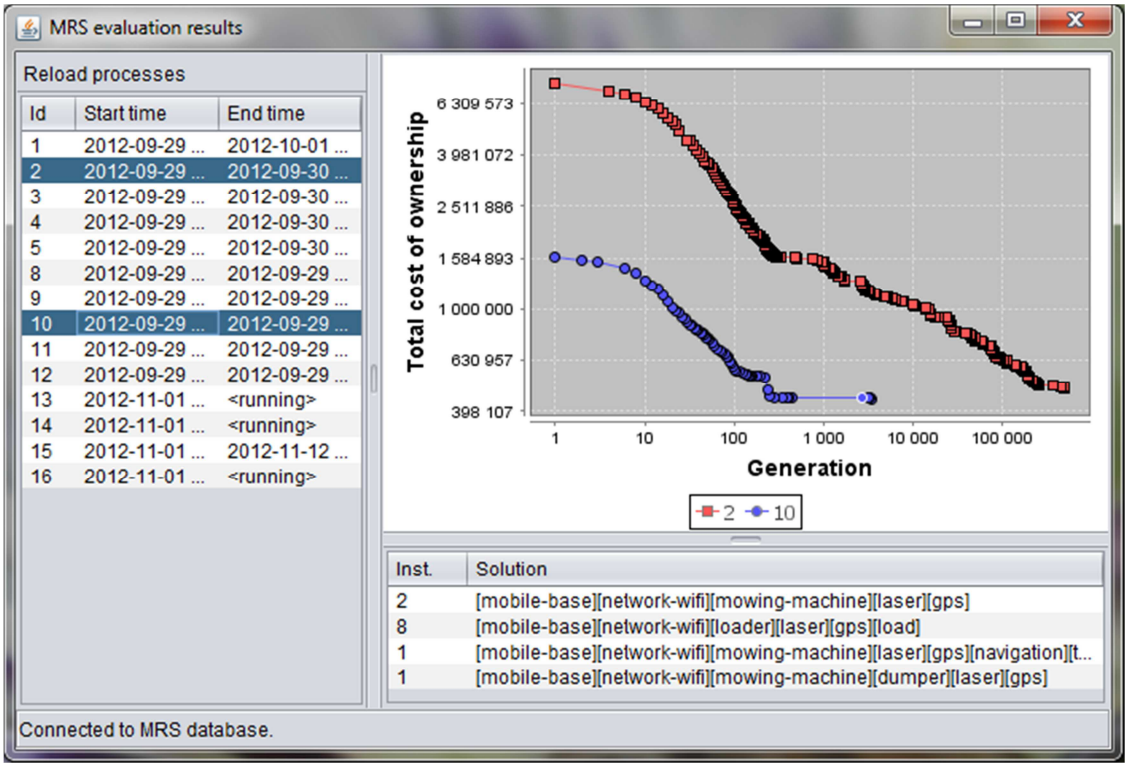


Figure 5.8. **Screenshot of user interface of *GAMBot-Eva* software**

GAMBot-Eva software described in this chapter is developed as a tool for the implementation of step 5 of the specification optimization procedure which implies initial evaluation of a specification using heuristic methods. The kernel of the software is an implementation of genetic algorithm which is a heuristic search approach. The software has modular design and it is main processing module is intended to be executed on dedicated computational server.

5.4. Analysis of initial evaluation results

The implementation of *GAMBot-Eva* software used for initial evaluation of the specification of multi-robot system is described in previous chapters. This chapter provides an analysis of results obtained using this software.

The experimental mission is described in details in previous sections (see 3.4). In general the mission is defined using 10 components and 2 tasks. There is a number of inter-component requirements defined as well in order to eliminate invalid component combinations from a processing in early stages.

For the first experiments during debugging the software the author used strict constraints in order to speed up processing. The number of possible agent types combined from the defined ten components is equal to 1024 (see 4.1). Because of

compatibility constraints the number of valid agent types was reduced down to 51, while 973 agent types were considered as invalid. In fact the number of agents equals to the length of the chromosome (genes). For the late experiments the constraints were looser allowing more variance in solutions. Thereby, the number of valid types of agents was equal to 211 (see 4.2). Additional experiments are presented in annex 2.

5.4.1. Parallel execution of multiple evaluation processes

The very first experiment is intended to test the behavior of the software executing multiple parallel processes in general, and to recognize the overall dynamics of genetic evolution in particular. The author expected heavy variations in processing results which are usually common for genetic algorithm due to random mutation and crossover factors. However, obtained results demonstrate amazingly similar results for all five processes. The evolution tends to optimum solution rapidly and all processes reached stable solution within 500 generations (see figure 5.9). As it is seen from the chart further evolution up to 10 000 generations resulted only in few changes in solution with minor decrease of costs.

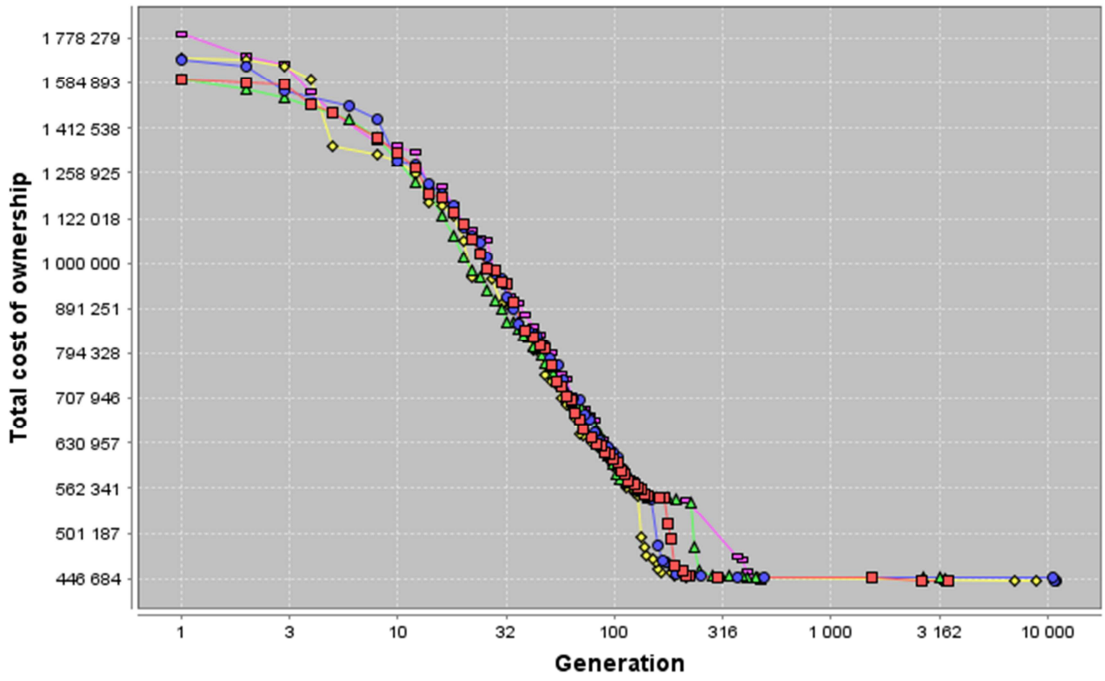


Figure 5.9. **Five identical evaluation processes executed in parallel**
Source: *GAMBot-Eva* software

During this experiment the modular design of the *GAMBot-Eva* software demonstrated its advantages. First, it was possible to execute multiple calculation processes in parallel. The only common aspect was persistent storage module, which is

implemented using *MySQL* DBMS and it supports parallel transactions. In other aspects the processes are completely independent and can be started or ended without any impact on others.

Secondly, it was possible to analyze the results of evaluations which are still in calculation process. This approach allows receiving early results and based on them it is possible to terminate particular process in case if its behavior does not correspond to expected ones.

And finally, the modular approach, especially the possibility to execute multiple calculation processes in parallel, is very well suited for dedicated computation hardware. The author executed the experiments on hardware with relatively slow but multiple CPUs (in this case x16 @ 2.0 GHz), which resulted in very good performance in parallel processes.

5.4.2. Comparison of strict and loose compatibility constraints

The second experiment performed by the author was intended to compare results of evaluation of simple and complex missions with, respectively, strict and loose inter-component compatibility constraints. In addition the behavior of the *GAMBot-Eva* software was tested for mission definition close to real application. As it was stated, for simple mission there were used 51 valid types of agents while for complex mission there were defined 211 valid types of agents. The results of evaluation of these missions are presented in figure 5.10.

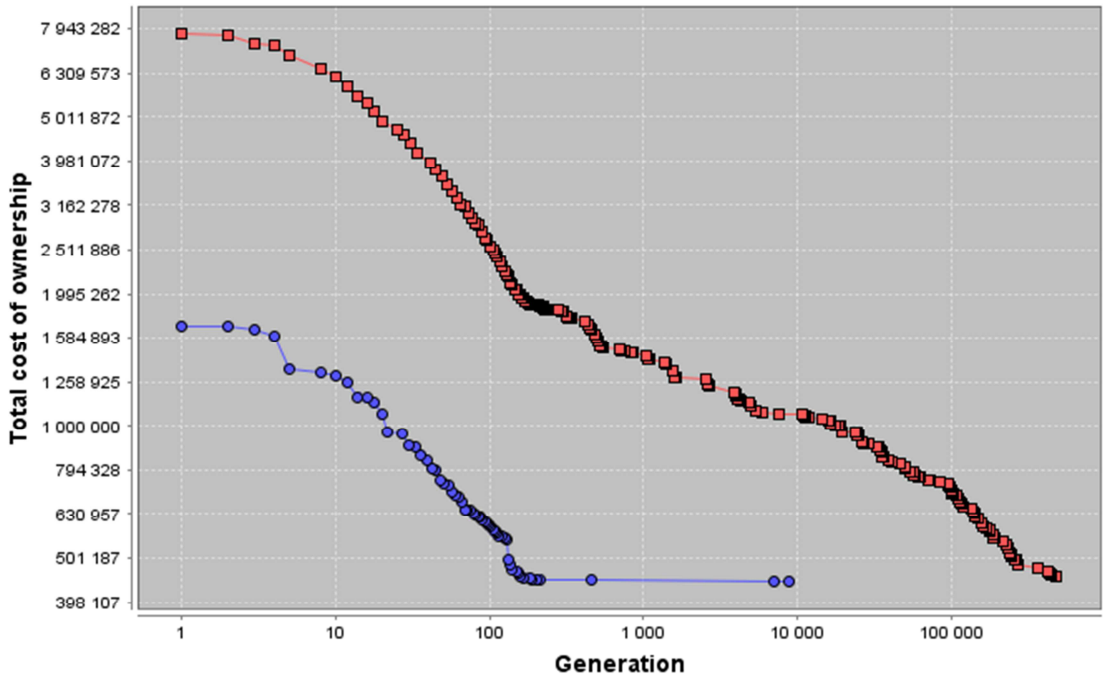


Figure 5.10. **Comparison of simple and complex mission evaluations**
Source: *GAMBot-Eva* software

According to the obtained optimization results it can be concluded that the behavior of the software is stable on complex mission definitions similarly as on simple ones. But the dynamics of the evolution is different for complex missions. As it is seen from the chart solution is improved through whole evolution. The evolution is executed up to 500 000 generations and according to the trend the solution would improve further. This indicates a proper implementation of fitness function which guides the evolution towards the global solution and does not stack in local extremes.

The processing time greatly increased for the complex mission in comparison with the simple mission. The full evolution of the simple mission was processed within an hour, while the stable solution was reached within first five minutes. Contrary, the full evolution of the complex mission took two days on the same hardware. Despite increased processing time the author believes that this approach is much more suitable for such kind of evaluation in contrast to testing every possible configuration of the system on real hardware or in simulations.

There is another notable conclusion made upon this experiment. Despite the fact that the initial solution is much worst for complex mission, the evolution tends to the same final solution. This could be explained by the similar mission definitions for both processes which have the only differences in the inter-component requirements. However this indicates the stability of the implementation of the genetic algorithm.

5.5. Results of the initial evaluation

The results of initial evaluation (the solution candidates) are being stored in the persistence module of the *GAMbot-Eva* software and are available for analysis and interpretation. In theory any solution candidates could be selected for model based evaluation (step 6 of the optimization procedure). However the author recommends selecting the fittest candidates obtained during evolution of the genetic algorithm.

In general, the software provides a description of the solution candidates in terms of solution concepts. In other words, the list of agents (their components) and the number of their instances is obtained. System analyst or modeler is responsible for the interpretation of the definition because it highly depends on peculiarities of targeted industrial area.

Practical example

For the demonstrative grass mowing example the author has selected several solution candidates with representative features. Differences of the estimated total costs of ownership (the fitness) among these solution candidates are within 10% of the total amount. Thereby the solution candidates are near equally feasible (see table 5.3).

One of the selected solution candidates (A) defines **highly distributed and redundant solution**. There are 8 agent types and 18 agent instances in total within the solution candidate. The components are distributed unevenly between the agents and overlap each other. There are 13 agents capable for grass mowing, 16 agents are capable for transportation and only 3 of them are capable for unloading the cargo. Global processing facilities are distributed among the different types of agents, while for navigation processing stationary agent is dedicated.

The other solution candidate (B) defines **near homogeneous** specification for multi-robot system. There are 10 universal agents capable to perform the grass mowing as well as the transportation tasks. In addition 4 agents are dedicated for the transportation task only and single agent performs a utility function and is capable for the transportation task and for unloading (unloads containers of other agents). Processing facilities are evenly distributed among universal agents.

Another solution candidate (C) selected by the author defines clearly **heterogeneous specification** for multi-robot system. There are 9 agents dedicated for grass mowing task alongside with 5 transportation agents. In addition one of the

transportation agents is equipped with unloading facilities. Notably that the processing facilities are distributed among the transporting robots only.

Table 5.3. **Number of agent instances for selected solution candidates**

Agent type	Solution candidate		
	A	**B**	**C**
Mowing robot	–	–	9
Mowing robot with processing facilities	1	–	–
Transportation robot	3	4	2
Transportation robot with processing facilities	1	–	2
Transportation robot with unloading and processing facilities	1	1	1
Mowing and transportation robot	9	8	–
Mowing and transportation robot with processing facilities	–	2	–
Mowing and transportation robot with unloading facilities	1	–	–
Mowing and transportation robot with unloading and processing facilities	1	–	–
Stationary processing unit	1	–	–
Total costs of ownership (monetary units)	**488 419**	**462 179**	**455 026**

According to the estimation of total costs of ownership the third (C) solution candidate is selected for fine evaluation using simulated models. Of course, the results of the initial evaluation are highly dependent on the estimation model of the total costs of ownership. The parameters of the model should be tuned for a targeted industrial domain before application in a production environment. Aforementioned fine-tuning is out scope of the thesis. Although, achieved results show successful application of the heuristics based evaluation approach for the demonstrative grass mowing example considered within the thesis.

5.6. Summary of the section

This section provides analysis of heuristics based initial evaluation step of the proposed specification optimization procedure (step 5). Genetic algorithm has been adopted by the author for the specification optimization task.

The model for estimating the total costs of ownership has been developed by the author. Multiple techniques are used for the estimation of the operating time of the each agent within the system.

The genetic representation of the solution domain is developed using integer type of genes instead of bit genes used in classical implementation of the genetic algorithm.

The fitness function for the genetic algorithm is based on the total costs of ownership estimation model and adds special penalties for improper solution candidates. Population of the genetic algorithm is being evolved within the batches with the variable size, which is being adopted during the execution. This approach improves processing performance of the genetic algorithm.

Aforementioned features are implemented within custom *GAMBot-Eva* software developed by the author, which supports execution of the initial evaluation step of the procedure. The software allows execution and monitoring of multiple parallel evaluation processes as well as provides user interface for accessing evaluation results.

The author has experimentally tested various aspects of the software and the heuristics based initial evaluation step of the procedure in general. The results obtained for the demonstrative example have been analyzed and interpreted.

6. SIMULATION BASED EVALUATION

The final computational phase (step 6) of the proposed specification optimization procedure (see figure 2.2) stands for simulation based evaluation of multi-robot specification solution candidates. This step implies that the set of considered solution candidates is relatively small and it can be processed within reasonable period of time. Simulation based evaluation is proposed as precise evaluation method which is applied to solution candidates obtained during preceding heuristic evaluation step.

The main goal of this step is to reproduce an environment close to real-world situation and to test selected solution candidates in it. The simulated environment is intended to avoid the development of a real robotic system for tests. Each solution candidate would require full amount of investment and operating costs for testing it which is not acceptable for an industrialist. In general, simulation techniques have been used for similar evaluation and prediction tasks also in other research domains of multi-robot systems (Dawson et al., 2010).

Modern simulation tools in combination with available computational hardware allow high fidelity of reproduction of real-world conditions and acceptable performance. Also usual practice implies reproduction of only representative features of the real world problem. Because of that the simulated environment is simplified, and an interpretation of results is translated to real-world situation.

Another general goal of the simulation based evaluation step is to test the optimization procedure itself. As it was described before the procedure has iterative nature. The results of both evaluation steps are analyzed (step 7 of the procedure) and further decisions are made. In general, the results of heuristic evaluation and simulation based evaluation are compared. If the difference between these evaluations extends requirements of acceptable fidelity (step 8), the criterion estimation model of heuristic evaluation has to be modified and the next iteration of the heuristic evaluation takes place. Thereby both the optimization results and underlying models are improved iteratively.

There is another aspect of the simulation based evaluation step which is worth to mention. The main goal of the current research is to propose and demonstrate the specification optimization procedure. Therefore, a practical implementation of the

simulation based evaluation is not universal and is closely related to the demonstrative mission.

Following chapters describe the simulation based evaluation step in details. They provide analysis of simulation environment and its setup for practical example, special control software for simulated robots and well as analysis of results and discussion.

6.1. Simulation environment setup

The sixth step of the specification optimization procedure stands for precise simulation based evaluation. Generally, the implementation of simulation based evaluation step could be divided into two aspects: model development for considered multi-robot system using tools provided by simulation environment and execution of simulations to assess the performance of a particular specification of multi-robot system. This chapter provides analysis of setup of simulation environment (development of the model). Next chapters describe the peculiarities of its execution.

The setup of the simulation environment implies development of models for all agents of the considered multi-robot system as well as a model for the environment of the mission. In addition the agent models should be developed in the way which allows independent gathering of various performance parameters, like consumed power or operating time. Environment model has to reflect the features of the real environment of the mission and has to respond to the actions of the agents.

There is a wide spectrum of simulation software available for general models as well as for models specific for robotics domain. There are high-end simulation packages that provide maximized accuracy of simulation, for instance *Webots* developed by *Cyberbotics Ltd*. Other packages offer great usability and utility tools, for instance *Microsoft Robotics Developer Studio*. There are also open-source simulation packages offering various interesting solutions which are not available in commercial software. To compensate poor support open-source packages often provide higher degree of customization and control over the package.

The author has used *Player/Stage* software bundle as a simulation package. The examples considered within the thesis are developed within this package. *Player/Stage* is an open-source software set developed within *The Player Project* is widely used for multi-robot and distributed sensor research (Gerkey et al., 2003). The author has

selected this software for final model-based evaluation step taking into account several advantages discussed below.

First of all, *Player/Stage* software bundle has highly flexible and extendable architecture. The core of this bundle is *Player*, the cross-platform robot device interface and server, also called robotics infrastructure. It is installed on operating system of a robot and it publishes robot devices and facilities to the network. The control system (client) communicates with *Player* (server) using TCP or UDP based protocol, while server acts like a proxy between control program and real devices – it provides data from sensors and forwards commands to actuators. In addition *Player* provides pluggable system of drivers which are used to access robot devices and for controlling specific hardware. Such architecture detaches the control code from the underlying hardware.

Stage is a simulation package for a population of mobile robots, sensors and objects in a two-dimensional bitmapped environment. *Stage* is designed to support research into multi-agent autonomous systems, so it provides fairly simple, computationally cheap models of lots of devices rather than attempting to emulate any device with great fidelity. *Stage* uses pre-compiled plug-in modules for controlling device models during simulation which ensures high performance and stability. Also *Stage* simulation package provide advanced tools for user like environment visualization, sensors' data rendering, time control and others.

In addition *Stage* provides special driver for *Player* server which allows combination of these two systems. *Stage* simulated devices are added to *Player* configuration and become accessible via regular network interface just like any other device. This feature allows easy prototyping by combining real and simulated devices and ensuring independence of control system from underlying hardware. Architecture of *Player/Stage* software bundle is schematically shown on figure 6.1.

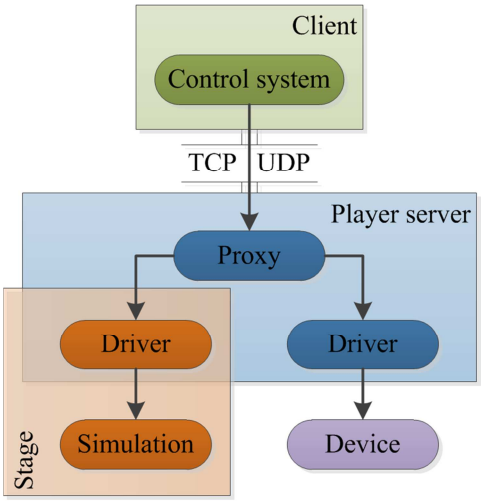


Figure 6.1. ***Player/Stage* architecture**

Another attractive feature of the *Player/Stage* software bundle is simple configuration based on model primitives. Simulated world is built from primitives derived from simple model concept. Typical model have attributes defining its appearance, such as size, position, color, and behavioral attributes such as sensor detection indicators. More complex models like sensors or actuators are inherited from the basic model and add specific attributes. The list of available models includes mobile base of the robot, range and object perception sensors, gripper actuators and other. Taking into account simplified physics model (2.5D space) implemented in *Stage*, the definition of the simulated world is easy and straight forward activity.

Player server provides direct access to simulated or real robot devices. Such approach allows low-level control over the system and produces minimal computational overhead. The author sees advantages in such low-level features because they grant relative freedom on design of control system.

Another important advantage of *Player/Stage* software bundle is derived from its architecture. The control system is executed on remote system and as a result it can be implemented using any development tool. There is only requirement for basic network communication and realization of *Player* protocol. Examples considered within the thesis are implemented using *Java* programming language and protocol library based on socket concepts.

Practical example

The author has tested *Player/Stage* software bundle (Komasilovs, Stalidzans, 2010) and found it suitable for model-based evaluation of a specification of a multi-robot system. Setup of the simulation model is made in two steps. First of all the

simulated environment is created. This implies configuration of simulated world within *Stage*, its objects and robots. Next active objects of the simulated environment are published through *Player*.

Initially the author has planned to create the simulated model for practical example described in 3.1. For better simulation configuration management *Stage* supports several concepts usually used in programming languages. First, *Stage* allows prototyping and inheritance, which allows user to define own types of models and to instantiate them on demand. This feature ensures minimal code overlapping within the configuration and makes the configuration more readable and manageable. Secondly, *Stage* supports file inclusions, which allows even better code management and modularity. Several types of models or their instances are defined in separate files which are later referenced in main configuration file on demand.

Taking aforementioned considerations, the author defined several types of models for the simulation. First, environmental model types were defined, including trees, bushes, flowerbeds and park boundaries. These models are stationary and they are considered as obstacles for any active model. In opposite, dumpsters were defined as zones providing no obstacles, but configured in a way to provide information about themselves for the active models (robots).

Special attention was given to grass modeling. For visual feedback dummy models were created for grass areas. But in order to calculate performance of the robots additional computational model is required which would store information about processes areas, where grass is already mowed. For performance evaluation of robots the author made an assumption that any spot of the lawn should be mowed only once, and any future processing on the same spot is considered as idle time.

The author found that creation of such in-memory model is not handy. The approach used within the thesis is based on simulated model instead of computational model. General idea is to define each blade of grass as separate model within the simulation. Due to computational limitations of modern hardware this is possible only up to some assumed fidelity. The author founded that suitable density of grass models is about 5 blades of grass per area of the active robot.

Setting up grass model instances within the configuration of simulation could be done manually. Taking into account number of such models the author looked for automated approach, however. Within first trial grass model instances were placed according to mathematical model of the lawn. But the result of simulation was not very

impressive because grass pieces were placed in clearly recognizable pattern. Because of that the author developed special utility, which generated random population of grass models according to given lawn and density parameters. An implicit benefit from such approach is that it models uneven distribution of grass mass through the lawn, which corresponds to real world situation.

Grass instances were defined as non-obstacle models, allowing robots to pass through them. In addition every instance was supplied with unique id for mowing process visualization and future performance tracking. Figure 6.2 shows complex simulated model of the garden (individual grass models are not visible on particular figure).

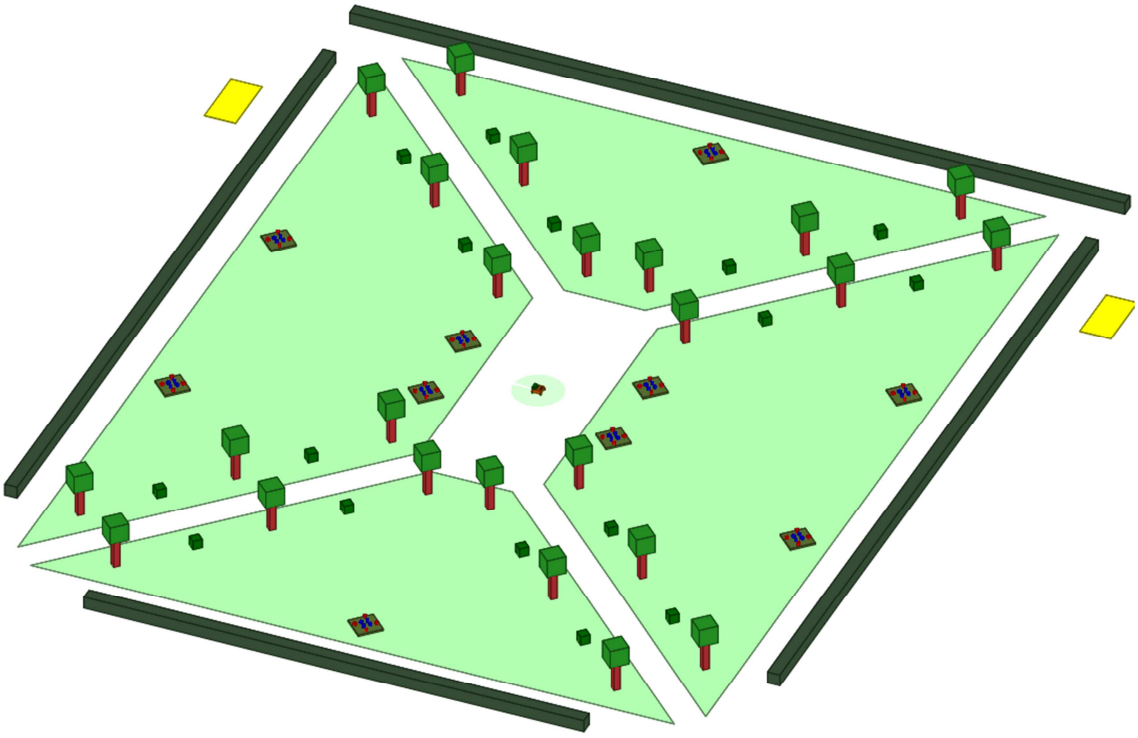


Figure 6.2. **Simulated garden model for grass mowing task**

The first modeling trials revealed several disadvantages of complex simulated environment. First of all, the scale of the simulated model of the garden influences the performance of the simulation package. Secondly, large open environment decrease the chance for robots to interact with each other. And finally, debugging of control system within complex environment is not handy.

As a result the author decided to create simplified simulated model for grass mowing task. The basic setup was taken from harvesting task usual for most of farmers. Plain lawn was defined and several large buildings were created as the only obstacles.

Dumpster was placed at the edge of the lawn. No additional requirements were defined (e.g. walkways or boundaries). Figure 6.3 shows simplified simulated model of environment for grass trimming task.

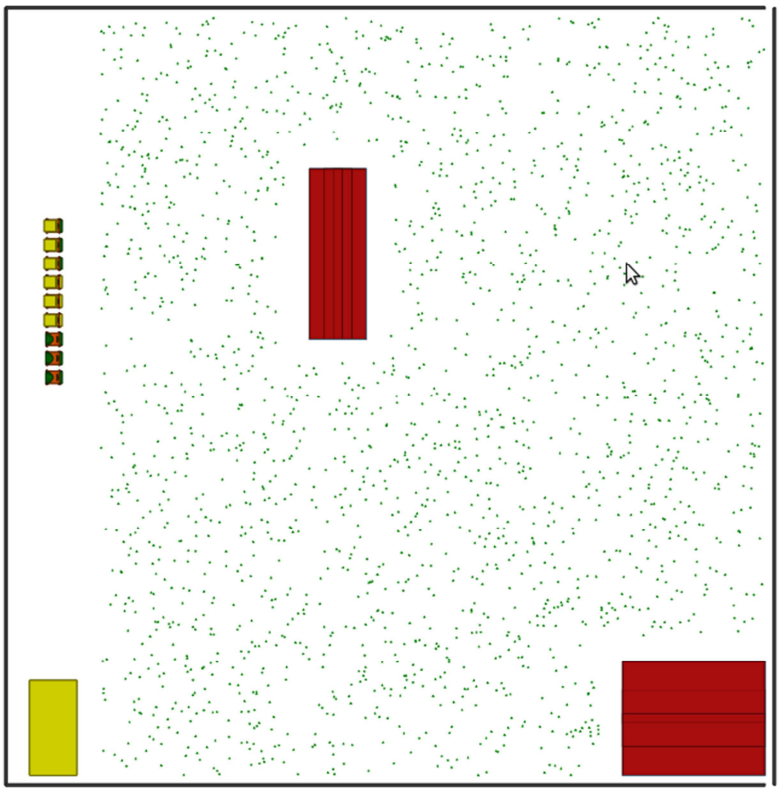


Figure 6.3. **Simplified simulated environment for grass mowing task**

For both simulated environments described before the same robot models were used. First of all common mobile base was defined, which included differential locomotion, distance measurement sensors for short range obstacle avoidance and camera for long range navigation and perception. Communication and processing facilities are considered as included in the mobile base or provided via external services.

Next the author defined reusable facilities required for grass mowing task. The list includes mowing device, grass packing device, stack collecting device and transporting container. These devices were mounted on mobile base in order to get model of particular robot type.

Taking into account simplification considerations the author defined three types of robots: the mowing robot which, in addition, packs the grass into the stacks and unloads them on the ground; the transporting robot which collects the stacks of the grass and transports them to the dumpster; and the robot combined from previous two, which mows the grass, collects it to own container and unloads it on demand.

Special attention is required for grass mowing and stack loading device models. As there is very simplified physics model implemented within *Stage* package these devices are not allowed to directly affect other models. Because of that the devices are equipped with short range sensor which identifies other models. As a result obtained information is used in control system for proper mowing task simulation. More details about its implementation are provided in 6.2.

As *Stage* package does not allow dynamic model creation in runtime there were implemented several technical work-around solutions for achieving desired behavior within simulation. For example, grass stacks are defined in advance and are placed to hidden place. After that they are moved to desired position on demand.

Finally, when configuration of simulated world is done (see annex 3), the network interface of the *Player* package is configured. Its configuration is straight forward and does not require special solutions (see annex 1). First of all the simulation itself is published to the network, and then all active models (robots) are listed. According to best practices each published entity is assigned to separate network port.

When *Stage* and *Player* packages are both configured the setup becomes accessible via network and robots could be controlled using any custom control software. For testing and debugging purposes *Player* package provides special viewer utility for rendering values perceived by sensors and for manual control of robots.

6.2. *SiMBot-Ctr* control framework for simulated robots

The previous chapter described the setup of simulated environment used within the thesis. This chapter describes control peculiarities of the simulated robots. As it was described before, the simulated models are accessible through the network. *Player* package uses a special protocol for communication with the control system. The author used *Java* based implementation of the protocol (Simon, Rusu, 2013), which is slightly different from the original *C++* implementation in terms of threading.

In general the protocol is based on asynchronous messaging between *Player* server and its client. First, client connects to server and bounds to particular simulated entity (usually robot). Then client subscribes to the particular devices of the simulated entity. This includes actuators and sensors, which are present on the simulated entity. The client regularly sends commands to devices and reads perception data from them.

Due to asynchronous approach several considerations should be kept in mind. First of all, when client sends a command to the device it does not guarantees immediate action on the robot. Besides of network latency, *Player* server receives a command and places it into the command stack for particular device. Therefore it is possible that the device is busy with other activities when particular command is received.

Secondly, perception data from sensors is not always available. Different types of sensors require additional time for collecting and publishing their readings. *Player* protocol specifies that a client has to request data from particular device, and when it is ready *Player* sends an answer for the request. Taking into account the dynamic environment usual for robotic applications the client software should not wait until data is ready but has to proceed with other tasks. In *Java* based implementation of the *Player* protocol a special background thread is used for continuous data requesting from the server, while user control code has to check data readiness in order to properly access it.

The same workflow is used also for accessing real robots with the only difference that usually real robots have different IP addresses and the same port, while simulated devices are usually located on the same host.

Taking into account previous considerations it could be concluded that control system uses *Player* protocol provided facilities for controlling simulated robots and other devices, which imply exchange with sensor data readings and low level commands. This provides great freedom for developer of the control system. For more intellectual control such low-level access is not very handy, however. Because of that various types of architectures are implemented in order to keep control system well-structured and easily maintainable (see 1.1).

Keeping in mind the simplification of the lawn mowing task the author assumes reflexive behavior of the robots without deep operational planning. The subsumption architecture was selected as suitable architecture for such robots. It is relatively simple to implement and, at the same, time it provides acceptable level of control. Also the control systems based on subsumption approach are easy to extend and add new levels of intellectual control above reflexes critical for survival.

In order to support the development of control system the author proposes special **framework *SiMbot-Ctr***, which stands for **Si**mulated **M**ulti-Ro**bot C**on**tr**ol System and uses *Java* implementation of the *Player* protocol. In general the framework is an abstract tool for creating control systems based on subsumption architecture. It uses a

number of concepts for creating internal structure of control system (Komasilovs, 2013).

One of the basic concepts is node. It defines a computational entity of the control system. Another important concept is signal, which is transmitted between various nodes. The signal contains of useful information and control indicator, which affects further processing of the signal and is used for signal prioritization. If a node wants to get control over the robot it sends signal with control indicator set to true. The final state of the signal depends on the topology of the control graph: either the signal is transmitted to output of the system or it is converted or decayed within the internal structure of the control system.

Basic node type used in control graph is behavior. It defines unified action scenario which is performed by the robot. According to subsumption architecture there are multiple behaviors within a robot running in parallel and propagate signals over the control graph. Depending on the topology of the graph a behavior might obtain exclusive control over particular device of the robot for a given time frame.

A special type of node is the input node, which transfers actual readings of the sensor data to subscribing nodes. Another special type of node is the output node, which receives a signal from internal nodes of the control system and transmits it to the external subscribers. These types of nodes form interface for the control system, which is used to interact with underlying levels of the system or directly with hardware.

There are two special internal nodes implemented within the framework: subsumption and inhibition nodes. Both of them have two input signals and one output signal. The behavior of these nodes depends on control indicator of the input signals. Subsumption node outputs second input signal if its control indicator is set to true, otherwise first input signal is sent to output. Inhibition node outputs the first input signal only if the second input signal has false control signal. Otherwise, dummy signal is transmitted with false control flag, which is interpreted as “no signal”.

The framework allows quick and easy setup of a control graph using generic and abstract classes. Also it provides parameterized implementations of most common behaviors of robots like cruising, wandering, obstacle avoidance, homing and others. A control system is constructed from ready to use blocks, thereby supporting quick prototyping. Figure 6.4 demonstrates conceptual model of the framework.

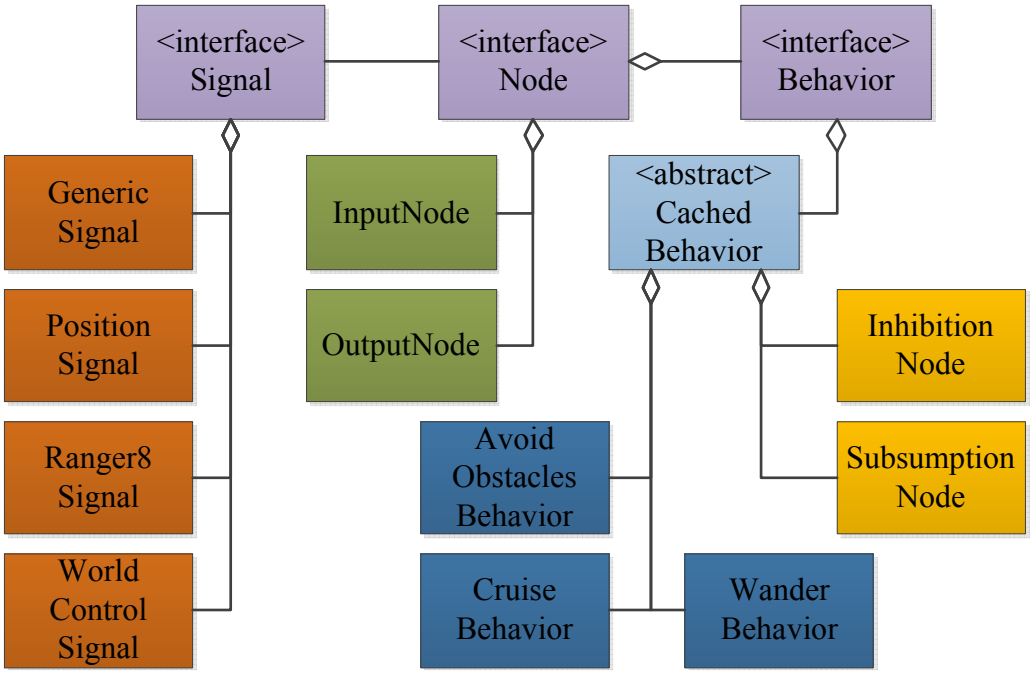


Figure 6.4. **Conceptual design of *SiMBot-Ctr* framework**

Another aspect of the control system is its execution. Previous paragraphs describe construction of the control graph, but not its execution. In the theory such control systems has to be implemented using analogue devices, which would allow parallel processing and best response time. A special solution for processing is required taking into account that control system is executed on discrete digital computers. The straight-forward solution is to execute each node of the control graph in its own thread and to try to synchronize their outputs in some usable way. This approach is usually applicable for multi-agent systems with a heavy computational workload. The control graph consists of many simple and computationally cheap nodes which would waste memory if executed in separate thread. Also response time of multi-threaded approach is longer as it needs a sort of synchronization for the output.

The author has used a control graph processing approach inspired by Domain Name Service (DNS) behavior used in networking. An execution part of the framework is designed in such way that the outputs are requested directly from the system without prior processing. The internal nodes automatically propagate processing requests if their predecessor nodes are still not processed. This approach benefits in such way, that the output of the control system is obtained by single call, and at the same time only nodes, which are required for obtaining the output are processed, leaving unnecessary lower importance nodes unprocessed. Taking into account stateless nature of most of

behaviors such processing approach is well suitable for examples considered within the thesis.

Practical example

The author used *SiMBot-Ctr* framework to build control systems for simulated robots for grass mowing task. As it was mentioned above, low-level access to simulated devices is obtained through *Player* protocol facilities. Also at the lower level an infinite loop is executed which repeatedly requests readings from sensor data and sends command to the actuators. On top of this skeleton control system is build. Its inputs are regularly populated with latest sensor readings and output commands are requested.

The control system created for the practical example has multiple input nodes (see figure 6.5). Distance measurement input used for local navigation and obstacle avoidance. A special short range object detection sensor is used for modeling mowing machine and stack loading facility. A camera is used for visual orientation and detection of object of interest. A load sensor is used to determine when unloading process should start.

The control system consists of several internal behaviors. The lowest priority has wandering behavior which is activated only when no other behavior requests control. Wandering behavior sets commands for locomotion drives of the robot to slightly change mowing direction while maintaining constant cruise speed. Next priority has homing behavior which tries to direct robot towards object of the interest (grass to mow or stack of grass to collect). The highest priority has obstacle avoidance behavior mainly because it is critical for survival. It tries to direct robot away from the obstacle. However triggering distance is suspended for obstacle avoidance behavior when homing behavior is active.

Among locomotion control behaviors there is number of other behaviors with act in more independent way. One of such behaviors is grass mowing or pack collecting behavior. It is triggered when object of interest is perceived by close range sensor and it sends a notification for simulation engine to model mowing of particular grass object. Another independent behavior is involved in a kind of short term planning. It receives notifications from previous behavior and counts the number of processed objects (in fact it models loading process). When particular amount of objects is reached it sends a command to unload the pack or to change strategy and try to go to dumpster. Although,

these behaviors are highly related to the simulated environment, its peculiarities and limitations, the overall performance of the system corresponds to the desired one.

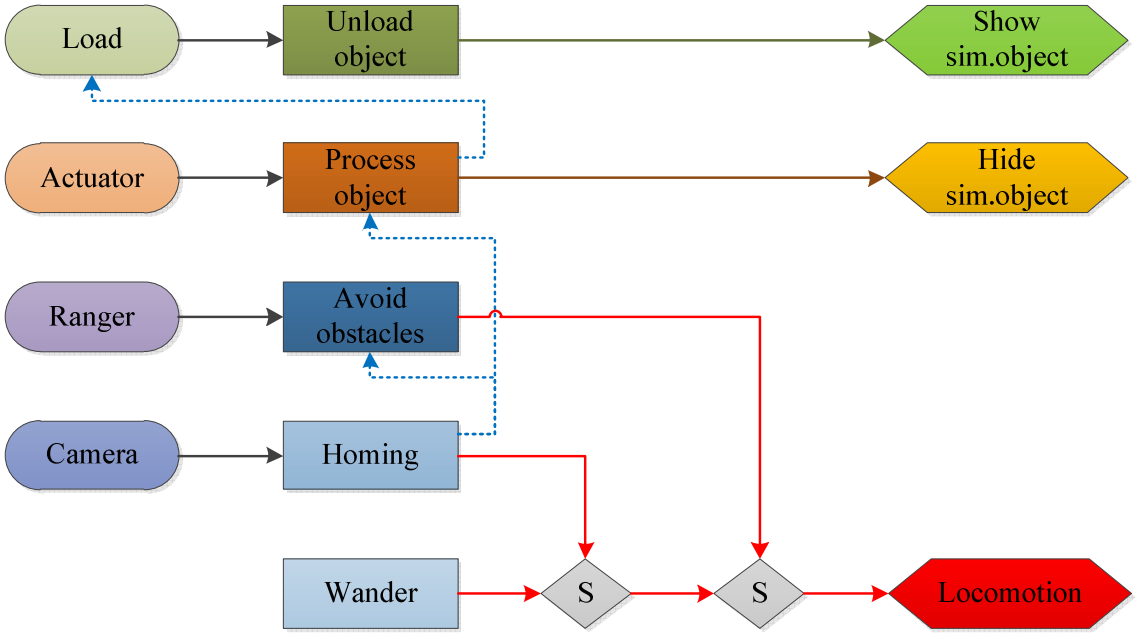


Figure 6.5. **Design of control system for simulated grass mowing robots**

Another aspect of implemented control system, which is worth to mention, is custom solution for modeling environmental changes in simulated world. As it was described above, *Stage* package has poor facilities for simulating physical processes. The author has used special concepts for explicit control over the simulated objects. This mainly includes grass trimming, stack unloading and collecting activities.

During the experiments the simulation package was executed on dedicated host while control system for all robots was built as a single application with multiple control threads for each robot. Taking into account aforementioned consideration it is easy to implement inter-thread signaling and data exchange. For the control of the simulated world a special control system was implemented and executed in separate thread apart from regular robot control thread. The robot control systems inform the simulation control system about their activities within the simulated environment. For example, the grass mowing robot eventually sends notifications about the mowed grass objects. For performance considerations, the simulation control system collects notifications into a queue, which is repeatedly being processed. This includes hiding and showing grass or stuck objects depending on the actions of the robots.

6.3. Simulation results and proposals

This chapter provides the analysis of the grass mowing task simulation results and simulation based evaluation step of the optimization procedure in general. In addition a list of proposals for future development is defined and discussed.

In general, the simulated model based evaluation of the specification should be considered in the context of the expected benefit from such evaluation. Taking into account that development of the model itself is an expensive and complex process, the rationality of the requested fidelity should be analyzed. There are might be cases when the development of the precise simulated model is more expensive than the implementation of the system itself.

Also there is a direction in robotics research domain aimed to analysis and development of the precise simulated models for multi-robot systems. Many researchers find this task highly challenging and non-trivial (Trianni, Marco Dorigo, 2006; Vaughan, 2008; Dawson et al., 2010). This topic is considered as out of the scope of the current thesis.

The very first experiments of the simulation based evaluation revealed the need for the highly intelligent control system for the active agents. The reactive architecture of the robot control system described in previous chapters demonstrated acceptable performance. However, a need for the additional system-wide intelligent planning was clearly identified. During experiments robots acted on their "own", without explicit cooperation. This resulted in a poor overall performance of the whole robotic system.

From the other side development of the intellectual control system for the multi-robot system is highly challenging task and might be the subject of another thesis (e.g. (Parker, 1994b)). Because of that the intellectual control system development aspect is considered as out of the scope of the current thesis.

Practical example

For illustrative purposes the author has implemented only one solution candidate C selected during the previous step of the procedure. Additional simplifications were assumed. Thereby, different types of transportation robots were simulated as a single type of agent capable for unloading its own container. Processing facilities were eliminated from the simulation as they are modeled as virtual devices.

The author has used two different strategies for the control of the robots (see figure 6.6). The first strategy grant relative freedom for the robots and they are allowed

to select their actions independently. As a result the robots near chaotically tried to move around the lawn in order to find the object of interest (e.g. the grass to mow). The second approach implies semi-autonomous control of the robots used to plan their actions in a more rational way. As it was stated before, the additional solution is required for the intelligent operational planning in order to make the system fully autonomous.

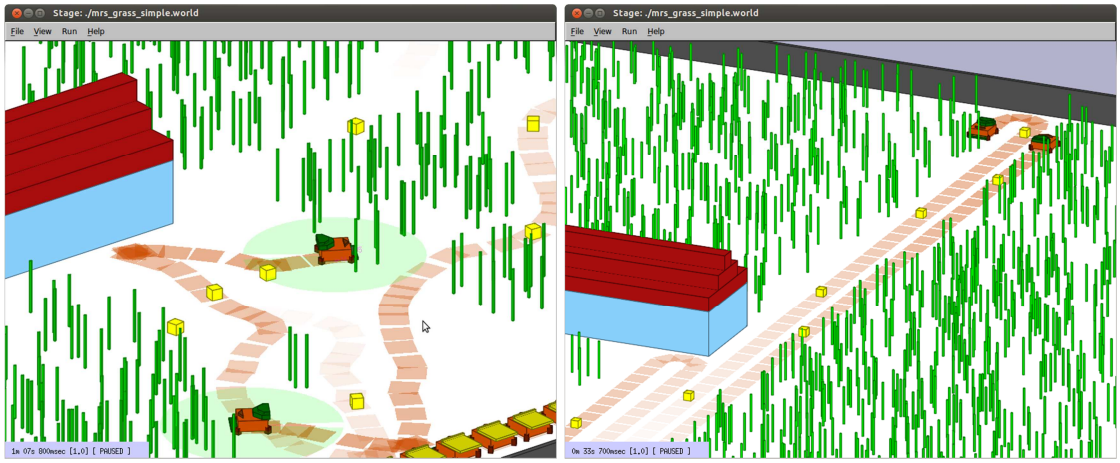


Figure 6.6. **Wandering and semi-autonomous simulated robots**

Due to lack of the operational planning within the control system the author used the time required for simulated agents to complete the mowing task as a primary indicator for costs estimation. The author repeated execution of the simulation experiment for 10 times and recorded simulation time when the mission was completed.

For the first experiment the operating time of the simulated robotic system varied from 348 seconds up to 492 seconds, 410.7 seconds in average. Taking into account the operating costs of each agent of the system the estimated costs are equal to 20 946 monetary units.

For the semi-automatic control approach the author have measured the average performance of the agents and used it for the analytical operation time estimation. According to results the system completes the mission within 360 seconds, which equals to 18 360 monetary units.

During the step 7 of the specification optimization procedure the estimation of the total cost of ownership obtained from the step 6 of the procedure is compared with the value of the initial estimation (step 5). Taking into account the simplifications assumed by the author for the illustrative example scaling coefficients should be applied. This includes time unit conversions, the scale of the environment, acceleration coefficients used within simulated environment and others.

For illustrative example the author has defined 2.8 times smaller lawn area, 5 time faster robots and double simulated time acceleration. Taking into account these coefficients the estimated costs of the robotic system for the complex mission equal to 586 488 monetary units for randomly wandering robots and 514 080 monetary units for semi-automatically controlled system.

The relative difference between the initial and simulation based estimations is used to draw a conclusion on the results of the optimization. For the illustrative example of the specification the initial estimation of the total costs of ownership was equal to 455 026 monetary units. The estimations obtained during the simulation based approach are 29 % and 13 % larger. The author assumes that the difference between estimations is within the acceptable threshold and additional iteration of the procedure is not required.

In general, the author concludes that custom control system based on *SiMBot-Ctr* framework allows precise calculation of the utilization time of the each particular component within the system. The obtained statistics are converted to expenses according to the operational parameters of the defined components.

The author has supervised development of master thesis of Janis Strods. The aim of his work is to develop a hardware prototype (see figure 6.7) of the autonomous robot for grass mowing task, which includes also realization of advanced control system. The analysis and adaptation of task planning algorithms is performed within the master thesis.

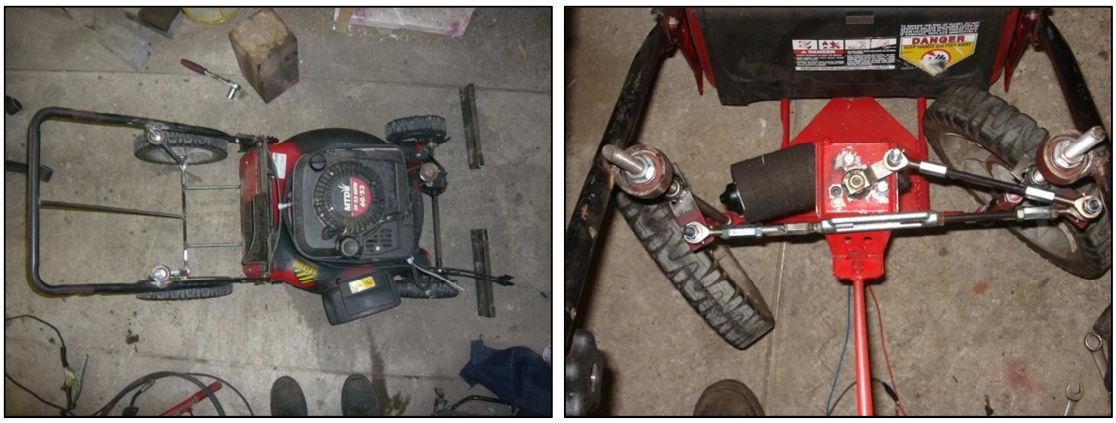


Figure 6.7. **Prototype of hardware autonomous grass mowing robot**
Author**:** Janis Strods

The author sees possibility to used developed hardware prototype for validating and improving the results obtained within the current thesis. Measurements obtained

from the real hardware could be compared to the values produced via the specification optimization procedure. However particular analysis is out of scope of the thesis.

6.4. Summary of the section

The description and the concepts of the model based evaluation step are provided within the section. The author defines general peculiarities for application of simulations for the evaluation of the multi-robot system specification. The author selects *Player/Stage* software bundle for implementing the evaluation step.

Custom control framework *SiMBot-Ctr* for simulated agents was developed by the author. The framework follows reactive control paradigm and is designed as an abstract tool for the implementation of user control systems. The signal propagation technique for processing the control system is implemented within the framework. According to the technique the processing requests are automatically propagated through the control graphs.

The demonstrative grass mowing task was implemented within the simulated environment. The control system for the task was implemented using *SiMBot-Ctr* framework. Various physical processes are modeled using direct access to simulation engine. In addition uneven distribution of grass mass on the lawn was modeled.

Practical experiments reveal the need for an intelligent control system with operational planning capabilities for the successful execution of the model based evaluation step. Development of such control system is out of scope of the thesis, however.

CONCLUSIONS

Major results of the thesis

A procedure for optimization of a specification of heterogeneous multi-robot system within the full solution domain is developed. The objectives of the thesis are fulfilled.

1. *There is performed an analysis of specification development methods applied for heterogeneous multi-robot systems.*

During the analysis actual research directions in the field of multi-robot systems were considered. The author revealed that most of modern investigations are aimed to development and fine tuning of working solution for particular task. However there are almost no researches focused on formal and universal method development for multi-robot systems.

The author found that high level design of multi-robot systems is not proved by any examination but instead is based on the facilities available during the research. Thereby, the author raised specification development problem as unsolved aspect of multi-robot system.

2. *Optimization task and solution concept of specification of heterogeneous multi-robot system is defined.*

The definition includes criterion, constraints and parameters for the optimization. The author has selected the total costs of ownership as the integral optimization criterion for practical experiments, taking into account application peculiarities of robotic systems.

The concept of solution for aforementioned optimization task was defined using three major terms: solution, agent, and component. The solution is defined as a set of agents. The agent is the unit of the system, either, mobile robot or stationary device. Agents are composed from the components, the indivisible concepts of the system which define particular functionality but not its implementation.

3. *The procedure for finding optimal specification of heterogeneous multi-robot system in full solution domain is developed.*

The procedure has iterative nature and consists of eight consecutive steps. First the business requirements are defined, which are then decomposed into the concepts of the solution. Next, the solution domain is analyzed and various evaluation methods are applied to solution candidates. Finally, the optimization results are analyzed and, if it is required, the definition of the mission is refined starting another iteration of the procedure.

4. *Mission definition technique for heterogeneous multi-robot systems and the approach for decomposition of the mission are developed.*

According to the proposed concept of the specification optimization procedure the mission for multi-robot system is defined using the list of components, required for the accomplishing the mission. Classification principles for components, their structural and dynamic properties are defined.

5. *The size of feasible solution domain of the specification optimization task is analyzed.*

Special methods are developed to find the number of unique agents and the number of their possible combinations. The custom *CoMBot-Gen* software was developed to perform analysis of agent combinations and eliminate invalid combinations. Near double exponential growth of number of solutions as a function of the number of defined component is found.

6. *Heuristic search algorithm for initial evaluation of specifications of multi-robot system is implemented and experimentally tested.*

Modular software *GAMBot-Eva* for genetic algorithm based heuristic search is developed. The software is used as a universal tool for the initial evaluation of the solution candidates. Implementation of the genetic algorithm includes development of the genetic representation of the solution domain, development of the fitness function used to estimate the total costs of ownership. Successful practical experiments are performed to test various aspects of the initial evaluation of solution candidates.

7. *Possibility to use simulation techniques for fine evaluation of specification of multi-robot system is analyzed.*

Practical experiments with the developed *SiMBot-Ctr* framework confirm the possibility for detailed evaluation by simulation techniques. It is important to assess the necessary level of details of the model because the development of the model and the simulation experiments are complicated and time consuming.

Conclusions and development prospects

Formal analysis and prediction of utilization of various functions of the multi-robot system are not being investigated within the multi-robot research domain. Properly designed multi-robot system requires fewer investments from a customer and at the same time it is capable to provide performance and fault tolerance required for completing the mission defined for the system.

The procedure is proposed for the optimization of the specification of multi-robot system. It defines a workflow for resolving the optimization task and includes business requirement specification, mission decomposition into components, solution domain analysis, solution candidate evaluation using heuristic algorithms and simulated models.

Analytical estimation of the number of feasible combinations of the agents reveal near double exponential growth of number of solutions as a function of the number of defined component for the specification optimization task.

There are improvements for proposed specification optimization procedure which are recommended for implementation but at the same time are out of scope of the thesis. The author recognizes following development prospects:

- ✓ to improve the model of components and their properties in order to allow more flexible definition of their investment and operating expenses;
- ✓ to implement the estimation model for the total costs of ownership using modern estimation and planning methods defined in the field of economics;
- ✓ to increase parameterization degree of the developed optimization software thus expanding application possibility;
- ✓ to developed universal concept for defining tasks of the mission for robotic system thereby making implementation of the optimization procedure more versatile;

- ✓ to introduce an additional fast simulation step to the optimization procedure alongside with final evaluation step;
- ✓ to extend the framework for the control of the simulated multi-robot system allowing operational planning facilities;
- ✓ to make the optimization process in general and its software implementation more interactive and continuous allowing greater control over optimization peculiarities and the final result;
- ✓ to analyze the application of proposed specification optimization procedure in the other areas to spread the approach and, vise-versa, to get knowledge and methods from external areas.

ACKNOWLEDGMENTS

This research project would not have been possible without the support of many people. The author wishes to express his gratitude to his supervisor, Associate professor, senior researcher, Dr.sc.ing. Egils Stalidzans, who offered invaluable assistance, guidance and encouragement.

Special thanks also to all colleagues and friends from Faculty of Information Technologies, especially, to Aleksejs Zacepins and Natalja Bulipopa for assistance in following the deadlines and inspiration. Not forgetting best friends who always been there.

The author would also like to convey thanks to the project of the European Social Fund (ESF) project “Atbalsts LLU doktora studiju īstenošanai” (agreement No. 2009/0180/1DP/1.1.2.1.2/09/IPIA/VIAA/017) for providing the financial support.

The author wishes to express his love and gratitude to his beloved family; for their understanding and endless love, through the duration of his studies.

REFERENCES

Aarts, E., Wichert, R. (2009) Ambient intelligence. In: *Technology Guide*. H.-J. Bullinger (ed.). Berlin, Germany: Springer-Verlag Heidelberg, p. 244–249.

Agüero, C., Veloso, M. (2012) Transparent Multi-Robot Communication Exchange for Executing Robot Behaviors. *Highlights on Practical Applications of Agents and Multi-Agent Systems*, Vol. 156, p. 215–222.

Alami, R. et al. (2006) Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In: *Procceedings IROS Workshop on Physical Human-Robot Interaction in Anthropic Domains*. Citeseer, p. 1–15.

Arai, T., Pagello, E., Parker, L.E. (2002) Guest editorial advances in multirobot systems. *IEEE Transactions on Robotics and Automation*, Vol. 18(5), p. 655–661.

Arkin, R.C. (1992) Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, Vol. 9(3), p. 351–364.

Arkin, R.C. (1998) *Behavior-based Robotics (Intelligent Robotics and Autonomous Agents)*. London, England: The MIT Press. 491 p.

Arkin, R.C., Balch, T. (1998) Cooperative multiagent robotic systems. *Artificial Intelligence and Mobile Robots*, p. 277–295.

Arrichiello, F., Das, J., Heidarsson, H., Pereira, A., Chiaverini, S., Sukhatme, G. (2010) Multi-robot collaboration with range-limited communication: Experiments with two underactuated ASVs. *Field and Service Robotics. Springer Tracts in Advanced Robotics*, Vol. 62, p. 443–453.

Azuma, T., Karube, T. (2010) Formation control with fault-tolerance based on rigid graph theory. In: *SICE Annual Conference 2010*. IEEE, p. 1137–1140.

Bajracharya, M., Maimone, M.W., Helmick, D. (2008) Autonomy for Mars Rovers: Past, Present, and Future. *Computer*, Vol. 41(12), p. 44–50.

Balch, T. (2000) Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous robots*, Vol. 8(3), p. 209–238.

Balch, T., Arkin, R.C. (1994) Communication in reactive multiagent robotic systems. *Autonomous Robots*, Vol. 1(1), p. 27–52.

Balch, T., Arkin, R.C. (1998) Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, Vol. 14(6), p. 926–939.

Balch, T., Parker, L.E. (2002) *Robot teams: from diversity to polymorphism*. Natick, MA, USA: A. K. Peters, Ltd. 425 p.

Baresel, A., Sthamer, H., Schmidt, M. (2002) Fitness function design to improve evolutionary structural testing. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 1329–1336.

Beer, B., Mead, R., Weinberg, J.B. (2010) A Distributed Method for Evaluating Properties of a Robot Formation. In: *Student Abstract and Poster Program of the 24th AAAI Conference on Artificial Intelligence*. Atlanta, Georgia.

Behavior (2011) *Dictionary.com Unabridged* [online] [accessed 18.06.2011]. Available at: http://dictionary.reference.com/browse/behavior.

Bekey, G.A. (2005) *Autonomous Robots: From Biological Inspiration to Implementation and Control (Intelligent Robotics and Autonomous Agents)*. London, England: The MIT Press. 577 p.

Beni, G., Wang, J. (1993) Swarm intelligence in cellular robotic systems. *Robots and Biological Systems: Towards a New Bionics?*, Vol. 102, p. 703–712.

Bonabeau, E., Meyer, C. (2001) Swarm intelligence. A whole new way to think about business. *Harvard Business Review*, Vol. 79(5), p. 106–114.

Brooks, R. (1986) A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, Vol. 2(1), p. 14–23.

Brooks, R. (1987) A hardware retargetable distributed layered architecture for mobile robot control. In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers, p. 106–110.

Burgard, W., Moors, M., Stachniss, C., Schneider, F. (2005) Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, Vol. 21(3), p. 376–386.

Butler, R.W. (2001) *What is Formal Methods?* [online] [accessed 13.04.2012]. Available at: http://shemesh.larc.nasa.gov/fm/fm-what.html.

Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E. (2003) *Self-organization in biological systems*. Princeton University Press. 560 p.

Campbell, E., Kong, Z.C., Hered, W., Lynch, A.J., O'Malley, M.K., McLurkin, J. (2011) Design of a low-cost series elastic actuator for multi-robot manipulation. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, p. 5395–5400.

Cao, Y.U., Fukunaga, A.S., Kahng, A.B., Meng, F. (1997) Cooperative mobile robotics: antecedents and directions. In: *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*. IEEE Comput. Soc. Press, p. 226–234.

Chakraborty, S. (2010) Applications of the MOORA method for decision making in manufacturing environment. *The International Journal of Advanced Manufacturing Technology*, Vol. 54(9-12), p. 1155–1166.

Chatterjee, P., Manikrao Athawale, V., Chakraborty, S. (2010) Selection of industrial robots using compromise ranking and outranking methods. *Robotics and Computer-Integrated Manufacturing*, Vol. 26(5), p. 483–489.

Chen, C.-T. (2000) Extensions of the TOPSIS for group decision-making under fuzzy environment. *Fuzzy Sets and Systems*, Vol. 114(1), p. 1–9.

Chen, P., Wan, J., Poo, A. (2011) Formation and Zoning Control of Multi-Robot Systems. *International Journal of Robotics and Automation*, Vol. 26(6), p. 35–48.

Chu, T.-C., Lin, Y.-C. (2003) A fuzzy TOPSIS method for robot selection. *The International Journal of Advanced Manufacturing Technology*, Vol. 21(4), p. 284–290.

Chuang, C.P., Chen, S.Y., Chen, W.H., Huang, Y.J. (2010) Applying Improved Ant Colony Algorithm for Multi-Robot Path Planning and Task Allocation in Educational Navigation Robot. In: *Proceedings of 2010 Third International Conference on Education Technology and Training (Volume 3)*. p. 143–151.

Clair, A.S., Atrash, A., Mead, Ross, Matarić, M.J. (2011) Speech, Gesture, and Space: Investigating Explicit and Implicit Communication in Multi-Human Multi-Robot Collaborations. In: *2011 AAAI Spring Symposium Series*. p. 11–13.

Coltin, B., Liemhetcharat, S., Meriçli, Ç., Veloso, M. (2010) Challenges of Multi-Robot World Modelling in Dynamic and Adversarial Domains. In: *Workshop on Practical Cognitive Agents and Robots, 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*. p. 36–42.

Contreras, J., Klusch, M., Yen, J. (1998) Multi-agent coalition formation in power transmission planning: a bilateral shapley value approach. In: *Proceedings of 2nd International Conference on Practical Applications of Multi-Agent Systems*. p. 19–26.

Control System (2011) *Dictionary.com Unabridged* [online] [accessed 23.04.2011]. Available at: http://dictionary.reference.com/browse/control+system.

Cornejo, A., Lynch, N. (2010) Fault-Tolerance Through k-Connectivity. *Workshop on Network Science and Systems Issues in Multi-Robot Autonomy: ICRA 2010*, Vol. 2, p. 47–72.

Daft, R.L. (2009) *Organization theory and design*. South-Western Pub. 649 p.

Davies, B. (2010) A review of the state-of-the-art of "smart" systems in surgery. *International Journal of Intelligent Systems*, Vol. 8(1/2/3/4), p. 423–433.

Dawson, S., Wellman, B.L., Anderson, M. (2010) Using simulation to predict multi-robot performance on coverage tasks. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, p. 202–208.

Decompose (2012) *Dictionary.com Unabridged* [online] [accessed 03.04.2012]. Available at: http://dictionary.reference.com/browse/decompose.

Decomposition (2012) *Dictionary.com Unabridged* [online] [accessed 03.04.2012]. Available at: http://dictionary.reference.com/browse/decomposition.

Dellaert, F., Fathi, A., Cunningham, A., Paluri, B. (2010) Local Exponential Maps: Towards Massively Distributed Multi-robot Mapping. *GVU Technical Report*, p. 1–7.

De Denus, M., Anderson, J., Baltes, J. (2011) Flexible Multi-Robot Formation Control: Partial Formations as Physical Data Structures. In: *2011 AAAI Spring Symposium Series*. p. 4–9.

Devi, K. (2011) Extension of VIKOR method in intuitionistic fuzzy environment for robot selection. *Expert Systems with Applications*, Vol. 38(11), p. 14163–14168.

Dias, M.B., Stentz, A. (2003) Traderbots: A Market-Based Approach for Resource, Role, and Task Allocation in Mulitrobot Coordination. *Robotics Institute*, Paper 154.

Ducatelle, F., Di Caro, G.A., Gambardella, L.M. (2010) Cooperative self-organization in a heterogeneous swarm robotic system. In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, p. 87–94.

Dudek, G., Jenkin, M.R.M., Milios, E. (2002) A taxonomy of multirobot systems. In: *Robot teams: From diversity to polymorphism*. T. Balch, L. E. Parker (eds.). Natick, MA, USA: A. K. Peters, Ltd, p. 3–22.

Duffie, N.A., Chitturi, R., Mou, J.-I. (1988) Fault-tolerant heterarchical control of heterogeneous manufacturing system entities. *Journal of Manufacturing Systems*, Vol. 7(4), p. 315–328.

Eiben, A.E., Smith, J.E. (2003) *Introduction to evolutionary computing*. Berlin, Germany: Springer Verlag. 415 p.

Ellis, K. (2012) *The Executive Briefing on the Issues Surrounding Getting Business Requirements Right* [online] [accessed 12.04.2012]. Available at: http://www.scribd.com/doc/6766319/Business-Requirements.

Elor, Y., Bruckstein, A.M. (2010) A thermodynamic approach to the analysis of multi-robot cooperative localization under independent errors. *Swarm Intelligence*, Vol. 6234, p. 36–47.

Endo, Y., Mackenzie, D.C., Arkin, R.C. (2004) Usability Evaluation of High-Level User Assistance for Robot Mission Specification. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, Vol. 34(2), p. 168–180.

Ferreira, C. (2002) Genetic representation and genetic neutrality in gene expression programming. *Advances in Complex Systems*, Vol. 5(4), p. 389–408.

Formal (2012) *Dictionary.com Unabridged* [online] [accessed 03.04.2012]. Available at: http://dictionary.reference.com/browse/formal.

Francesca, G., Pini, G., Brutschy, A., Pinciroli, C., Dorigo, M., Birattari, M. (2011) Task Partitioning in Swarm Robotics as a Multi-Armed Bandit Problem. *IRIDIA – Technical Report Series*, (September), p. 1–10.

Gamson, W.A. (1961) A theory of coalition formation. *American sociological review*, Vol. 23(3), p. 373–382.

Garnier, S., Gautrais, J., Theraulaz, G. (2007) The biological principles of swarm intelligence. *Swarm Intelligence*, Vol. 1(1), p. 3–31.

Gerkey, B.P. (2003) *On multi-robot task allocation*. Los Angeles, USA: University of Southern California. 117 p.

Gerkey, B.P., Matarić, M.J. (2004) A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, Vol. 23(9), p. 939–954.

Gerkey, B.P., Vaughan, R.T., Howard, A. (2003) The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: *Proceedings of the 11th International Conference on Advanced Robotics*. p. 317–323.

Gordon, D.M. (1996) The organization of work in social insect colonies. *Nature*, Vol. 380, p. 121–124.

Grocholsky, B., Keller, J., Kumar, V., Pappas, G. (2006) Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, Vol. 13(3), p. 16–25.

Hackwood, S., Beni, G. (1992) Self-organization of sensors for swarm intelligence. In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, p. 819–829.

Haddad, J. El, Haddad, S. (2004) A fault-tolerant communication mechanism for cooperative robots. *International Journal of Production Research*, Vol. 42(14), p. 2793–2808.

Haddadin, S. (2011) Safe Physical Human-Robot Interaction: Measurements, Analysis and New Insights. *Robotics Research*, Vol. 66, p. 395–407.

Hamann, H., Stradner, J., Schmickl, T., Crailsheim, K. (2010) A hormone-based controller for evolutionary multi-modular robotics: From single modules to gait learning. In: *IEEE Congress on Evolutionary Computation*. IEEE, p. 1–8.

Hambling, D. (2011) *Japan sends robots into Fukushima nuclear plant* [online] [accessed 05.04.2011]. Available at: http://www.newscientist.com/blogs/onepercent/2011/03/japanese-send-robots-into-fuku.html.

Hansen, S.T., Andersen, H.J., Bak, T. (2010) Practical evaluation of robots for elderly in Denmark: An overview. In: *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction*. ACM, p. 149–150.

Haumann, A.D., Listmann, K.D., Willert, V. (2010) Discoverage: A new paradigm for multi-robot exploration. In: *2010 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, p. 929–934.

Hayati, S. (1986) Hybrid position/Force control of multi-arm cooperating robots. In: *1986 IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers, p. 82–89.

Hayes-Roth, B. (1995) An architecture for adaptive intelligent systems. *Artificial Intelligence*, Vol. 72(1-2), p. 329–365.

Hazon, N., Kaminka, G.A. (2008) On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, Vol. 56(12), p. 1102–1114.

Helsgaun, K. (2000) An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, Vol. 126(1), p. 106–130.

Heuristic (2012) *Dictionary.com Unabridged* [online] [accessed 09.05.2012]. Available at: http://dictionary.reference.com/browse/heuristic.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, Michigan: University of Michigan Press. 211 p.

Holloway, C.M. (1997) Why engineers should consider formal methods. In: *16th DASC. AIAA/IEEE Digital Avionics Systems Conference. Reflections to the Future. Proceedings*. IEEE, p. 1.3: 16–22.

De Hoog, J., Cameron, S., Visser, A. (2010) Dynamic team hierarchies in communication-limited multi-robot exploration. In: *2010 IEEE Safety Security and Rescue Robotics*. IEEE, p. 1–7.

Hoshino, S., Seki, H., Naka, Y., Ota, J. (2010) Fault-tolerant multi-robot operational strategy for material transport systems considering maintenance activity. *Journal of Robotics and Mechatronics*, Vol. 22(4), p. 485.

Hoshino, Satoshi, Seki, Hiroya, Ota, Jun (2011) Optimal maintenance strategy in fault-tolerant multi-robot systems. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, p. 2314–2320.

Hromkovic, J. (2010) *Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics*. Berlin, Heidelberg: Springer-Verlag. 556 p.

Hu, X., Eberhart, R.C., Shi, Y. (2003) Swarm intelligence for permutation optimization: a case study of n-queens problem. *Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS'03)*, p. 243–246.

Huang, P.Y., Ghandforoush, P. (1984) Procedures given for evaluating, selecting robots. *Industrial Engineering*, Vol. 16(4), p. 44–48.

Imany, M.M., Schlesinger, R.J. (1989) Decision models for robot selection: a comparison of ordinary least squares and linear goal programming methods. *Decision Sciences*, Vol. 20(1), p. 40–53.

Izadikhah, M. (2009) Using the Hamming distance to extend TOPSIS in a fuzzy environment. *Journal of Computational and Applied Mathematics*, Vol. 231(1), p. 200–207.

Jahanshahloo, G.R., Lotfi, F.H. (2006) An algorithmic method to extend TOPSIS for decision-making problems with interval data. *Applied Mathematics and Computation*, Vol. 175(2), p. 1375–1384.

Jain, L., Aidman, E., Abeynayake, C., Dasgupta, P. (2011) Multi-Robot Task Allocation for Performing Cooperative Foraging Tasks in an Initially Unknown Environment. *Innovations in Defence Support Systems 2*, Vol. 338, p. 5–20.

Jeanne, R. (1986) The evolution of the organization of work in social insects. *Monitore Zoologico Italiano*, Vol. 20(2), p. 119–133.

Jeon, S., Jang, M., Park, Seunghwan, Lee, D., Cho, Y.-J., Kim, J. (2011) Task allocation strategy of heterogeneous multi-robot for indoor surveillance. In: *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, p. 169–173.

Jolly, K.G., Sreerama Kumar, R., Vijayakumar, R. (2010) Intelligent task planning and action selection of a mobile robot in a multi-agent system through a fuzzy neural network approach. *Engineering Applications of Artificial Intelligence*, Vol. 23(6), p. 923–933.

Jones, C., Matarić, M.J. (2005) Behavior-Based Coordination in Multi-Robot Systems. In: *Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications*. S. S. Ge, F. L. Lewis (eds.). New York, USA: Marcel Dekker, Inc., p. 549–569.

Jones, M.S., Malmborg, C.J., Agee, N.H. (1985) Robotics: decision support system for selecting a robot. *Industrial Engineering*, Vol. 17, p. 66–73.

Jung, J.H., Park, Sujin, Kim;, S.-L. (2010) Multi-robot path finding with wireless multihop communications. *IEEE Communications Magazine*, Vol. 48(7), p. 126–132.

Karimadini, M., Lin, H. (2010) Synchronized task decomposition for two cooperative agents. In: *2010 IEEE Conference on Robotics Automation and Mechatronics (RAM)*. IEEE, p. 368–373.

Karsak, E.E., Sener, Z., Dursun, M. (2011) Robot selection using a fuzzy regression-based decision-making approach. *International Journal of Production Research*, Vol. 50(23), p. 6826–6834.

Kassabalidis, I., El-Sharkawi, M.A., Marks, R.J., Arabshahi, P., Gray, A.A. (2001) Swarm intelligence for routing in communication networks. In: *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*. IEEE, p. 3613–3617.

Kaupp, T., Makarenko, A., Durrant-Whyte, H. (2010) Human–robot communication for collaborative decision making — A probabilistic approach. *Robotics and Autonomous Systems*, Vol. 58(5), p. 444–456.

Kelso Jr, A.S., Crawford, V.P. (1982) Job matching, coalition formation, and gross substitutes. *Econometrica: Journal of the Econometric Society*, p. 1483–1504.

Kernbach, S. et al. (2010) Multi-Robot Organisms: Stage Of The Art. In: *IEEE international conference on robotics and automation (workshop on modular robotics)*. Piscataway: IEEE Press, p. 1–10.

Kernbach, S. et al. (2011) Heterogeneity for Increasing Performance and Reliability of Self-Reconfigurable Multi-Robot Organisms. In: *International Conference on Intelligent Robots and Systems*. San Francisco, CA, p. 1–8.

Khouja, M. (1995) The use of data envelopment analysis for technology selection. *Computers & Industrial Engineering*, Vol. 28(1), p. 123–132.

Kiener, J., Stryk, O. (2010) Towards cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. *Robotics and Autonomous Systems*, Vol. 58(7), p. 921–929.

Kim, D., Choi, J. (2011) Multi-robot team outdoor localization using active marker and high frequency signal sources. In: *2011 11th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, p. 192–196.

Kirman, A., Oddou, C., Weber, S. (1986) Stochastic communication and coalition formation. *Econometrica: Journal of the Econometric Society*, p. 129–138.

Kitano, H., Asada, M., Kuniyoshi, Y. (1997) RoboCup: A challenge problem for AI. *AI Magazine*, Vol. 18(1), p. 73–85.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E. (1997) Robocup: The robot world cup initiative. In: *Proceedings of the first international conference on Autonomous agents - AGENTS '97*. New York, USA: ACM Press, p. 340–347.

Knott, K., Robert, D. (1982) A model for evaluating alternative robot systems under uncertainty. *The International Journal Of Production Research*, Vol. 20(2), p. 155–165.

Koivo, A.J., Bekey, G.A. (1988) Report of workshop on coordinated multiple robot manipulators: planning, control, and applications. *IEEE Journal Of Robotics And Automation*, Vol. 4(1), p. 91–93.

Komasilovs, V. (2012a) Investment and running cost estimation for heterogeneous multi-robot system. In: *5th International Scientific Conference on Applied Information and Communication Technology*. Jelgava, Latvia, p. 118–122.

Komasilovs, V. (2012b) Investment costs optimization of multi-robot system using genetic algorithm. In: *Annual 18th International Scientific Conference "Research for Rural Development 2012"*. Jelgava, Latvia, p. 229–232.

Komasilovs, V. (2012c) *Multi-robot system specification optimization tool* [online] [accessed 14.11.2012]. Available at: http://code.google.com/p/mrs-optimization/.

Komasilovs, V. (2013) Software modules for optimization of specification of heterogeneous multi-robot system. In: *12th International Scientific Conference Engineering for Rural Development*. Jelgava, Latvia, p. <in–press>.

Komasilovs, V., Stalidzans, E. (2010) Simulation of Real-Time Robot Control Systems Using Player/Stage Software. In: *Industrial Simulation Conference 2010*. Budapest, Hungary: Eurosys-ETI, p. 39–41.

Komasilovs, V., Stalidzans, E. (2011) Functional decomposition method reveals the number of possible specifications of multi-robot system. In: *2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI)*. Budapest, Hungary: IEEE, p. 161–165.

Komasilovs, V., Stalidzans, E. (2012a) Genetic algorithm used for initial evaluation of specification of multi-robot system. In: *13th International Carpathian Control Conference*. High Tatras, Slovakia: IEEE, p. 313–317.

Komasilovs, V., Stalidzans, E. (2012b) Procedure of Specification Optimization of Heterogeneous Robotic System. In: *IEEE 10th Jubilee International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. Herl'any, Slovakia: IEEE, p. 259–263.

Korte, B., Vygen, J. (2012) *Combinatorial optimization: theory and algorithms*. Vol. 21. Springer. 659 p.

Koulouriotis, D.E., Ketipi, M.K. (2011) A fuzzy digraph method for robot evaluation and selection. *Expert Systems with Applications*, Vol. 38(9), p. 11901–11910.

Kube, C.R.R., Zhang, H. (1993) Collective robotics: From social insects to robots. *Adaptive Behavior*, Vol. 2(2), p. 189–218.

Kumar, R., Garg, R.K. (2010) Optimal selection of robots by using distance based approach method. *Robotics and Computer-Integrated Manufacturing*, Vol. 26(5), p. 500–506.

Lackey, S., Barber, D., Reinerman, L., Badler, N.I., Hudson, I. (2011) Defining Next-Generation Multi-Modal Communication in Human Robot Interaction. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 55(1), p. 461–464.

Lasdon, L.S., Waren, A.D., Jain, A., Ratner, M. (1978) Design and testing of a generalized reduced gradient code for nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, Vol. 4(1), p. 34–50.

Lau, H.C., Zhang, L. (2003) Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity. In: *Proceedings of 15th IEEE International Conference on Tools with Artificial Intelligence.* IEEE, p. 346–350.

Lee, K.Y., El-Sharkawi, M.A. (2008) *Modern heuristic optimization techniques: theory and applications to power systems*. IEEE Press. 602 p.

Lerman, K., Shehory, O. (2000) Coalition formation for large-scale electronic markets. In: *Proceedings Fourth International Conference on MultiAgent Systems*. IEEE Comput. Soc, p. 167–174.

Li, X., Wang, X., Lei, Y., You, H. (2010) A self-organized algorithm based on digital hormone. In: *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*. IEEE, p. V2: 398–402.

Li, Xingyan, Parker, L.E. (2008) Design and performance improvements for fault detection in tightly-coupled multi-robot team tasks. In: *IEEE SoutheastCon 2008*. IEEE, p. 198–203.

Ma, H., Wang, M., Jia, Z., Yang, C. (2011) A new framework of optimal multi-robot formation problem. In: *2011 30th Chinese Control Conference (CCC)*. IEEE, p. 4139–4144.

Mackenzie, D.C. (1996) *A design methodology for the configuration of behavior-based mobile robots*. Georgia Institute of Technology. 267 p.

Mackenzie, D.C., Cameron, J.M., Arkin, R.C. (1997) Specification and execution of multiagent missions. In: *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*. IEEE Comput. Soc. Press, p. 51–58.

Madden, J.D., Arkin, R.C., MacNulty, D.R. (2010) Multi-robot system based on model of wolf hunting behavior to emulate wolf and elk interactions. In: *2010 IEEE International Conference on Robotics and Biomimetics*. IEEE, p. 1043–1050.

Matarić, M.J. (1992a) Behavior-based control: Main properties and implications. In: *Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*. Citeseer, p. 46–54.

Matarić, M.J. (1992b) Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, Vol. 8(3), p. 304–312.

Matarić, M.J. (1994) *Interaction and Intelligent Behavior*. Massachusetts Institute of Technology, Cambridge, Mass. 191 p.

Matarić, M.J. (1997) Behaviour-based control: examples from navigation, learning, and group behaviour. *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 9(2-3), p. 323–336.

Matarić, M.J. (2002) Situated robotics. *Encyclopedia of Cognitive Science*, Vol. 4, p. 25–30.

Matarić, M.J. (2010) *Retired Robots* [online] [accessed 15.02.2012]. Available at: http://robotics.usc.edu/?l=Robots:retired.

Mechatronics (2011) *Dictionary.com Unabridged* [online] [accessed 05.04.2011]. Available at: http://dictionary.reference.com/browse/mechatronics.

Meffert, K., Meseguer, J., Martн, E.D., Meskauskas, A., Vos, J., Rotstan, N. (2012) *JGAP - Java Genetic Algorithms and Genetic Programming Package* [online] [accessed 19.01.2012]. Available at: http://jgap.sf.net.

Michael, N., Fink, J., Kumar, Vijay (2011) Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, Vol. 30(1), p. 73–86.

Moravec, H.P. (1977) Towards Automatic Visual Obstacle Avoidance. In: *Proceedings of the 5th international joint conference on Artificial intelligence*. p. 584–592.

Murphy, R.R. (2000a) *Introduction to AI robotics (Intelligent Robotics and Autonomous Agents)*. London, England: The MIT Press. 466 p.

Murphy, R.R. (2000b) Marsupial and shape-shifting robots for urban search and rescue. *IEEE Intelligent Systems*, Vol. 15(2), p. 14–19.

Neely, A. (1999) The performance measurement revolution: why now and what next? *International Journal of Operations & Production Management*, Vol. 19(2), p. 205–228.
Neely, A., Gregory, M., Platts, K. (2005) Performance measurement system design: A literature review and research agenda. *International Journal of Operations & Production Management*, Vol. 25(12), p. 1228–1263.
Neely, A., Mills, J., Platts, K., Richards, H., Gregory, M., Bourne, M., Kennerley, M. (2000) Performance measurement system design: developing and testing a process-based approach. *International Journal of Operations & Production Management*, Vol. 20(10), p. 1119–1145.
Nelder, J.A., Mead, R. (1965) A simplex method for function minimization. *The computer journal*, Vol. 7(4), p. 308.
Nolfi, S., Mirolli, M. (2010) *Evolution of communication and language in embodied agents*. Springer-Verlag New York Inc. 313 p.
Nouyan, S., Grob, R., Bonani, M., Mondada, F., Dorigo, Marco (2009) Teamwork in Self-Organized Robot Colonies. *IEEE Transactions on Evolutionary Computation*, Vol. 13(4), p. 695–711.
Offodile, O.F., Johnson, S.L. (1990) Taxonomic system for robot selection in flexible manufacturing cells. *Journal of Manufacturing Systems*, Vol. 9(1), p. 77–80.
Offodile, O.F., Lambert, B.K. (1987) Development of a computer aided robot selection procedure (CARSP). *International Journal of Production Research*, Vol. 25(8), p. 1109–1121.
Oppenheimer, K.R. (1978) A Proxy Approach to Multi-Attribute Decision Making. *Management Science*, Vol. 24(6), p. 675–689.
Optimization (2012) *Dictionary.com Unabridged* [online] [accessed 26.03.2012]. Available at: http://dictionary.reference.com/browse/optimization.
Oster, G.F., Wilson, E.O. (1978) *Caste and ecology in the social insects*. Princeton, N.J.: Princeton University Press. 352 p.
Pamilo, P. (1991a) Evolution of colony characteristics in social insects. I. Sex allocation. *American Naturalist*, p. 83–107.
Pamilo, P. (1991b) Evolution of colony characteristics in social insects. II. Number of reproductive individuals. *American Naturalist*, p. 412–433.
Paradigm (2011a) *Answers.com* [online] [accessed 23.04.2011]. Available at: http://www.answers.com/topic/paradigm#ixzz1LU71VwI4.
Paradigm (2011b) *Dictionary.com Unabridged* [online] [accessed 23.04.2011]. Available at: http://dictionary.reference.com/browse/paradigm.
Parkan, C. (1994) Operational competitiveness ratings of production units. *Managerial and Decision Economics*, Vol. 15(3), p. 201–221.
Parkan, C. (1999) Decision-making and performance measurement models with applications to robot selection. *Computers & Industrial Engineering*, Vol. 36(3), p. 503–523.
Parkan, C., Lam, K., Hang, G. (1997) Operational competitiveness analysis on software development. *Journal of the Operational Research Society*, Vol. 48(9), p. 892–905.
Parkan, C., Wu, M.-L. (1999a) Measurement of the performance of an investment bank using the operational competitiveness rating procedure. *Omega*, Vol. 27(2), p. 201–217.
Parkan, C., Wu, M.-L. (1999b) Measuring the performance of operations of Hong Kong's manufacturing industries. *European Journal of Operational Research*, Vol. 118(2), p. 235–258.

Parker, L.E. (1994a) ALLIANCE: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*. IEEE, p. 776–783.

Parker, L.E. (1994b) *Heterogeneous multi-robot cooperation*. USA: Massachusetts Institute of Technology Cambridge. 227 p.

Parker, L.E. (1996) L-ALLIANCE: Task-oriented multi-robot learning in behavior-based systems. *Advanced Robotics*, Vol. 11(4), p. 305–322.

Parker, L.E. (1997) Cooperative motion control for multi-target observation. In: *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97*. IEEE, p. 1591–1597.

Parker, L.E. (1998a) Distributed Control of Multi-Robot Teams: Cooperative Baton Passing Task. *Synthesis (ISAS'98)*, Vol. 3, p. 89–94.

Parker, L.E. (1998b) ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, Vol. 14(2), p. 220–240.

Parker, L.E. (1999a) Adaptive heterogeneous multi-robot teams. *Neurocomputing*, Vol. 28(1), p. 75–92.

Parker, L.E. (1999b) Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing*, Vol. 5(6), p. 5–20.

Parker, L.E. (2003) The effect of heterogeneity in teams of 100+ mobile robots. *Multi-robot systems*, Vol. 2, p. 205–215.

Parker, L.E., Kannan, B., Bailey, M. (2004) Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. IEEE, p. 1016–1022.

Parker, L.E., Schneider, F., Schultz, A. (2005) *Multi-robot systems: From swarms to intelligent automata*. Springer Netherlands. 302 p.

Payton, D., Daily, M., Estkowski, R., Howard, M., Lee, C. (2001) Pheromone robotics. *Autonomous Robots*, p. 319–324.

Payton, D., Estkowski, R., Howard, M. (2002) Progress in pheromone robotics. *Intelligent Autonomous Systems*, Vol. 7, p. 256–264.

Payton, D., Estkowski, R., Howard, M. (2003) Compound behaviors in pheromone robotics. *Robotics and Autonomous Systems*, Vol. 44(3), p. 229–240.

Pinheiro, P., Wainer, J. (2011) Planning for multi-robot localization. *Advances in Artificial Intelligence - SBIA 2010*, p. 183–192.

Poli, R., Kennedy, J., Blackwell, T. (2007) Particle swarm optimization. *Swarm Intelligence*, Vol. 1(1), p. 33–57.

Portugal, D., Rocha, R. (2010) MSP algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, p. 1271–1276.

Procedure (2012) *Dictionary.com Unabridged* [online] [accessed 10.04.2012]. Available at: http://dictionary.reference.com/browse/procedure.

Prorok, A., Martinoli, A. (2011) A reciprocal sampling algorithm for lightweight distributed multi-robot localization. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, p. 3241–3247.

Rao, R.V., Patel, B.K. (2011) Novel method for decision making in the manufacturing environment. *Proceedings of the Institution of Mechanical Engineers. Part B. Journal of engineering manufacture*, Vol. 225(3), p. 422–434.

Rao, R.V., Patel, B.K., Parnichkun, M. (2011) Industrial robot selection using a novel decision making method considering objective and subjective preferences. *Robotics and Autonomous Systems*, Vol. 59(6), p. 367–375.

Renzaglia, A., Doitsidis, L., Martinelli, A., Kosmatopoulos, E.B. (2011) Multi-Robot 3D Coverage of Unknown Terrains. In: *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE Press, p. 2046–2051.

RoboCup (2012) *Objective < RoboCup* [online] [accessed 19.03.2012]. Available at: http://www.robocup.org/about-robocup/objective/.

Robot (2011) *Dictionary.com Unabridged* [online] [accessed 05.04.2011]. Available at: http://dictionary.reference.com/browse/robot.

Robotics (2011) *Dictionary.com Unabridged* [online] [accessed 05.04.2011]. Available at: http://dictionary.reference.com/browse/robotics.

Rogozea, L., Leasu, F., Repanovici, A. (2010) Ethics, robotics and medicine development. In: *Proceedings of the 9th WSEAS international conference on Signal processing, robotics and automation*. p. 264–268.

Russell, S., Norvig, P. (2009) *Artifcial Intelligence: A Modern Approach*. 3rd Edit. Englewood Cliffs, NJ: Prentice Hall. 1152 p.

Rybski, P.E. et al. (2000) Enlisting rangers and scouts for reconnaissance and surveillance. *IEEE Robotics & Automation Magazine*, Vol. 7(4), p. 14–24.

Rybski, P.E., Larson, A., Veeraraghavan, H., LaPoint, M., Gini, M. (2007) Communication Strategies in Multi-robot Search and Retrieval: Experiences with MinDART. In: *Distributed Autonomous Robotic Systems 6*. Rachid Alami, Raja Chatila, H. Asama (eds.). Springer Japan, p. 317–326.

Sabbatini, R.M.E. (1999) *Imitation of Life: A History of the First Robots* [online] [accessed 05.04.2011]. Available at: http://www.cerebromente.org.br/n09/historia/turtles_i.htm.

Şahin, E. (2005) Swarm robotics: From sources of inspiration to domains of application. *Swarm Robotics*, p. 10–20.

Şahin, E., Winfield, A. (2008) Special issue on swarm robotics. *Swarm Intelligence*, Vol. 2(2-4), p. 69–72.

Salem, M., Kopp, S., Wachsmuth, I. (2010) Generating multi-modal robot behavior based on a virtual agent framework. In: *Proceedings of the ICRA 2010 Workshop on Interactive Communication for Autonomous Intelligent Robots (ICAIR)*. p. 23–25.

Sandholm, T.W., Lesser, V.R. (1995) Coalition formation among bounded rational agents. In: *International Joint Conference on Artificial Intelligence*. Citeseer, p. 662–671.

Schwager, M., Dames, P., Rus, D., Kumar, V. (2011) A multi-robot control policy for information gathering in the presence of unknown hazards. In: *Proceedings of the International Symposium on Robotics Research*. Flagstaff, USA, p. 17–33.

Seeley, T.D. (2002) When is self-organization used in biological systems? *The Biological bulletin*, Vol. 202(3), p. 314–318.

Seeni, A., Schafer, B. (2010) Robot Mobility Systems for Planetary Surface Exploration – State-of-the-Art and Future Outlook: A Literature Survey. *Aerospace technologies advancements*, (January), p. 189–208.

Service, T.C., Adams, J. a. (2010) Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, Vol. 22(2), p. 225–248.

Shaw, M.E. (1971) *Group dynamics: The psychology of small group behavior*. New York: McGraw Hill. 414 p.

Shehory, O., Kraus, S. (1995) Task allocation via coalition formation among autonomous agents. In: *International Joint Conference on Artificial Intelligence*. Citeseer, p. 655–661.

Shehory, O., Kraus, S. (1998) Methods for task allocation via agent coalition formation. *Artificial Intelligence*, Vol. 101(1-2), p. 165–200.

Shehory, O., Sycara, K., Jha, S. (1998) Multi-agent coordination through coalition formation. *Intelligent Agents IV Agent Theories, Architectures, and Languages*, p. 143–154.

Simmons, R., Singh, S., Hershberger, D., Ramos, J., Smith, T. (2001) First results in the coordination of heterogeneous robots for large-scale assembly. *Experimental Robotics VII*, Vol. 271, p. 323–332.

Simon, J.S., Rusu, R.B. (2013) *Javaclient for Player/Stage* [online] [accessed 10.02.2013]. Available at: http://java-player.sourceforge.net/.

Simzan, G., Akbarimajd, A., Khosravani, M. (2011) A market based distributed cooperation mechanism in a multi-robot transportation problem. In: *2011 11th International Conference on Intelligent Systems Design and Applications (ISDA)*. IEEE, p. 1–5.

Specification (2011) *Dictionary.com Unabridged* [online] [accessed 09.08.2011]. Available at: http://dictionary.reference.com/browse/specification.

Sprowitz, A., Pouya, S., Bonardi, S., Den Kieboom, J., Mockel, R., Billard, A., Dillenbourg, P., Ijspeert, A. (2010) Roombots: Reconfigurable Robots for Adaptive Furniture. *IEEE Computational Intelligence Magazine*, Vol. 5(3), p. 20–32.

Stoeter, S.A., Rybski, P.E., Stubbs, K.N., McMillen, C.P., Gini, M., Hougen, D.F., Papanikolopoulos, N. (2002) A robot team for surveillance tasks: Design and architecture. *Robotics and Autonomous Systems*, Vol. 40(2-3), p. 173–183.

Sukhatme, G.S. (1999) Design and implementation of a mechanically heterogeneous robot group. In: *Proceedings of Mobile Robots XIV Conference*. Citeseer, p. 122–133.

Swangnetr, M., Zhu, B., Kaber, D. (2010) Meta-Analysis of User Age and Service Robot Configuration Effects on Human-Robot Interaction in a Healthcare Application. *2010 AAAI Fall Symposium*, p. 121–126.

System (2011) *Dictionary.com Unabridged* [online] [accessed 21.04.2011]. Available at: http://dictionary.reference.com/browse/system.

Tangen, S. (2004) Performance measurement: from philosophy to practice. *International Journal of Productivity and Performance Management*, Vol. 53(8), p. 726–737.

Taxonomy (2012) *Dictionary.com Unabridged* [online] [accessed 13.04.2012]. Available at: http://dictionary.reference.com/browse/taxonomy.

TCO (2012) *Dictionary.com Unabridged* [online] [accessed 29.03.2012]. Available at: http://dictionary.reference.com/browse/TCO.

Total-cost-of-ownership (2012) *Dictionary, Encyclopedia and Thesaurus - The Free Dictionary* [online] [accessed 29.03.2012]. Available at: http://encyclopedia.thefreedictionary.com/Total+Cost+of+Operations.

Trianni, V., Dorigo, Marco (2006) Self-organisation and communication in groups of simulated and physical robots. *Biological Cybernetics*, Vol. 95(3), p. 213–231.

Trianni, V., Tuci, E., Passino, K.M. (2010) Special issue on Swarm Cognition. *Swarm Intelligence*, Vol. 5(1), p. 1–2.

Ulam, P.D., Kira, Z., Arkin, R.C., Collins, T.R. (2010) Mission specification and control for unmanned aerial and ground vehicles for indoor target discovery and tracking. In: *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*. Georgia Institute of Technology, p. 28–40.

Vaughan, R.T. (2008) Massively multi-robot simulation in stage. *Swarm Intelligence*, Vol. 2(2-4), p. 189–208.

Velagapudi, P., Sycara, K., Scerri, P. (2010) Decentralized prioritized planning in large multirobot teams. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, p. 4603–4609.

Veloso, M. (2012) Teamwork Planning and Learning in Adversarial Multi-Robot Domains. *Advances in Autonomous Mini Robots*, p. 5–10.

Vig, L. (2008) *Multiple Robot Teaming: Coalition Formation: From Software Agents to Robots*. Saarbrucken, Germany: VDM VErlag Dr. Muller Aktiengesellschaft & Co. KG. 160 p.

Wang, S. (2006) Comments on operational competitiveness rating analysis (OCRA). *European Journal of Operational Research*, Vol. 169(1), p. 329–331.

Wang, Y., Siriwardana, P.G.D., Silva, C.W. de (2011) Multi-Robot Cooperative Transportation of Objects using Machine Learning. *International Journal of Robotics and Automation*, Vol. 26(4), p. 3486–3490.

Whetten, J.M., Goodrich, M.A. (2010) Specialization, fan-out, and multi-human/multi-robot supervisory control. In: *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, p. 147–148.

Whitley, D., Rana, S., Heckendorn, R.B. (1999) The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, Vol. 7, p. 33–48.

Wilson, E.O. (1971) *The insect societies*. Belknap Press of Harvard University Press (Cambridge, Mass). 548 p.

Wilson, E.O. (2000) *Sociobiology: The New Synthesis*. 25th anniv. Belknap Press of Harvard University Press. 148 p.

Wong, C.Y., Seet, G., Sim, S.K. (2011) Multiple-robot systems for USAR: Key design attributes and deployment issues. *International Journal of Advanced Robotic Systems*, Vol. 8(1), p. 85–101.

Yakowitz, D. (1993) Multi-attribute decision making: dominance with respect to an importance order of the attributes. *Applied Mathematics and Computation*, Vol. 54(2-3), p. 167–181.

Yanco, H. a., Drury, J. (2005) Classifying human-robot interaction: an updated taxonomy. In: *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*. IEEE Press, p. 2841–2846.

Yang, M., Yan, G., Tian, Y. (2010) A review of studies in flocking for multi-robot system. In: *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*. IEEE, p. 28–31.

Yen, J., Yan, Y.H., Wang, B.J., Sin, P.K.H., Wu, F.F. (1998) Multi-agent coalition formation in power transmission planning. In: *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*. IEEE Comput. Soc, p. 433–443.

Yeung, C.S.K., Poon, A.S.Y., Wu, F.F. (1999) Game theoretical multi-agent modelling of coalition formation for multilateral trades. *IEEE Transactions on Power Systems*, Vol. 14(3), p. 929–934.

Yim, M., Shen, W., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G. (2007) Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics]. *IEEE Robotics & Automation Magazine*, Vol. 14(1), p. 43–52.

Zanakis, S (1998) Multi-attribute decision making: A simulation comparison of select methods. *European Journal of Operational Research*, Vol. 107(3), p. 507–529.

Zanakis, SH, Evans, J. (1981) Heuristic "optimization": Why, when, and how to use it. *Interfaces*, Vol. 11(5), p. 84–91.

Zhang, Y.-Q. (2010) Present situation and future development of multiple mobile robot communication technology. In: *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*. IEEE, p. 597–601.

Zheng, Y.F., Luh, J.S. (1985) Control of two coordinated robots in motion. In: *1985 24th IEEE Conference on Decision and Control*. IEEE, p. 1761–1766.

Zieliński, C., Winiarski, T. (2010) General specification of multi-robot control system structures. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, Vol. 58(1), p. 15–28.

Zionts, S. (1979) MCDM: If Not a Roman Numeral, then What? *Interfaces*, p. 94–101.

Zlotkin, G., Rosenschein, J.S. (1994) Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In: *Proceedings of the National Conference on Artificial Intelligence*. Seattle, Washington: Alfred P. Sloan School of Management, Massachusetts Institute of Technology, p. 87–94.

Zolezzi, J.M., Rudnick, H. (2002) Transmission cost allocation by cooperative games and coalition formation. *IEEE Transactions on Power Systems*, Vol. 17(4), p. 1008–1015.

ANNEXES

Annex 1 – *GAMBot-Eva* software configuration XML file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<project>
    <components>
        <component>
            <code>mobile-base</code>
            <complexity>1.0</complexity>
            <family>wheeled car-like</family>
            <investmentCosts>60.0</investmentCosts>
            <name>Mobile base</name>
            <operatingPower>4.0</operatingPower>
            <required>
                <comment>Mobile base requires WiFi</comment>
                <refComponent>network-wifi</refComponent>
            </required>
        </component>
        <component>
            <code>network-wifi</code>
            <complexity>1.1</complexity>
            <family>Wi-FI</family>
            <investmentCosts>30.0</investmentCosts>
            <name>Wi-Fi networking</name>
            <operatingPower>2.0</operatingPower>
        </component>
        <component>
            <code>mowing-machine</code>
            <complexity>1.0</complexity>
            <family>1-DOF manipulator</family>
            <investmentCosts>40.0</investmentCosts>
            <name>Mowing machine</name>
            <operatingPower>5.0</operatingPower>
            <required>
                <comment>Mowing machine is useless on stationary agent</comment>
                <refComponent>mobile-base</refComponent>
            </required>
        </component>
        <component>
            <code>loader</code>
            <complexity>1.0</complexity>
            <family>End effector</family>
            <investmentCosts>40.0</investmentCosts>
            <name>Loader</name>
            <operatingPower>4.0</operatingPower>
            <required>
                <comment>Loader should be mobile</comment>
                <refComponent>mobile-base</refComponent>
            </required>
            <required>
                <comment>Loader should know the weight of cargo</comment>
                <refComponent>load</refComponent>
            </required>
        </component>
        <component>
            <code>dumper</code>
            <complexity>1.0</complexity>
            <family>1-DOF manipulator</family>
            <investmentCosts>20.0</investmentCosts>
            <name>Dumper</name>
            <operatingPower>3.0</operatingPower>
        </component>
        <component>
            <code>laser</code>
            <complexity>1.0</complexity>
            <family>Proximity</family>
            <investmentCosts>30.0</investmentCosts>
            <name>Laser</name>
            <operatingPower>2.0</operatingPower>
            <required>
                <comment>Laser is useless on stationary device</comment>
                <refComponent>mobile-base</refComponent>
            </required>
        </component>
        <component>
            <code>gps</code>
```

```
            <complexity>1.0</complexity>
            <family>Position</family>
            <investmentCosts>25.0</investmentCosts>
            <name>GPS</name>
            <operatingPower>1.5</operatingPower>
            <required>
                <comment>GPS is useless on stationary device</comment>
                <refComponent>mobile-base</refComponent>
            </required>
        </component>
        <component>
            <code>load</code>
            <complexity>1.0</complexity>
            <family>Sensing</family>
            <investmentCosts>20.0</investmentCosts>
            <name>Load</name>
            <operatingPower>0.5</operatingPower>
        </component>
        <component>
            <code>navigation</code>
            <complexity>1.3</complexity>
            <family>Computation</family>
            <investmentCosts>50.0</investmentCosts>
            <name>Navigation</name>
            <operatingPower>1.0</operatingPower>
            <required>
                <comment>Networking is required for controlling navigation</comment>
                <refComponent>network-wifi</refComponent>
            </required>
        </component>
        <component>
            <code>task-allocation</code>
            <complexity>1.2</complexity>
            <family>Computation</family>
            <investmentCosts>50.0</investmentCosts>
            <name>Task allocation</name>
            <operatingPower>1.0</operatingPower>
            <required>
                <comment>Tasks should be sent via net</comment>
                <refComponent>network-wifi</refComponent>
            </required>
        </component>
    </components>
    <config>
        <agentInstanceLimit>10</agentInstanceLimit>
        <crossoverRate>0.35</crossoverRate>
        <doubletteChromosomesAllowed>false</doubletteChromosomesAllowed>
        <generationsLimit>15000</generationsLimit>
        <generationsStep>20</generationsStep>
        <keepPopulationSizeConstant>true</keepPopulationSizeConstant>
        <minimumPopSizePercent>0</minimumPopSizePercent>
        <mutationRate>15</mutationRate>
        <nearInfinity>1.0E12</nearInfinity>
        <nearZero>1.0E-12</nearZero>
        <populationSize>20</populationSize>
        <selectFromPrevGen>0.95</selectFromPrevGen>
        <selectorOriginalRate>0.9</selectorOriginalRate>
    </config>
    <costModel>
        <assembly>
            <b0>10.0</b0>
            <b1>5.0</b1>
            <b2>0.02</b2>
            <k>3.0</k>
        </assembly>
        <design>
            <b0>40.0</b0>
            <b1>10.0</b1>
            <b2>0.5</b2>
            <k>2.0</k>
        </design>
        <energyLoss>
            <b0>0.0</b0>
            <b1>1.0</b1>
            <b2>0.01</b2>
            <k>2.0</k>
        </energyLoss>
```

```
            <sysDesign>
                <b0>280.0</b0>
                <b1>20.0</b1>
                <b2>2.0</b2>
                <k>2.0</k>
            </sysDesign>
            <sysMaint>
                <b0>8.0</b0>
                <b1>2.0</b1>
                <b2>0.1</b2>
                <k>2.0</k>
            </sysMaint>
            <systemReplRate>0.005</systemReplRate>
        </costModel>
        <missions>
            <areaCoverageMission>
                <areaSizeX>120.0</areaSizeX>
                <areaSizeY>150.0</areaSizeY>
                <mobileBase>mobile-base</mobileBase>
                <mobileBaseSpeed>2.0</mobileBaseSpeed>
                <workDensity>0.9</workDensity>
                <workDevice>mowing-machine</workDevice>
                <workDeviceWidth>1.2</workDeviceWidth>
            </areaCoverageMission>
            <transportationMission>
                <areaSizeX>120.0</areaSizeX>
                <areaSizeY>150.0</areaSizeY>
                <mobileBase>mobile-base</mobileBase>
                <mobileBaseSpeed>8.0</mobileBaseSpeed>
                <workDensity>0.04</workDensity>
                <loader>loader</loader>
                <targetOffsetX>20.0</targetOffsetX>
                <targetOffsetY>10.0</targetOffsetY>
            </transportationMission>
        </missions>
        <name>Grass trimming project</name>
    </project>
```

Annex 2 – Tests of various parameters of the genetic algorithm

The author used the same mission definition as for complex experiments of genetic algorithm (see 5.4.1) and changed genetic algorithm parameters provided to the processing module using XML configuration file. The list of changes includes such positions as mutation, crossover and natural selection rates.

The author focused on impact analysis of three parameters of genetic algorithm: mutation and crossover rates, natural selection threshold. For one of the trials author have changed these parameters as follows:

- ✓ mutation rate was increased to 17% of genes in average up from 7%;
- ✓ crossover rate increased to 65% of chromosomes up from 35%;
- ✓ natural selection transferred only 55% of individuals to next generation down from 95%.

The changes are made with the aim to increase variations in the population during the evolution. However author found that these changes negatively affect the overall performance of the genetic algorithm and do not produce expected increase in heuristic search speed. Figure A show two processes with original parameters (red and blue) and other two identical processes with changed parameters (green and yellow).

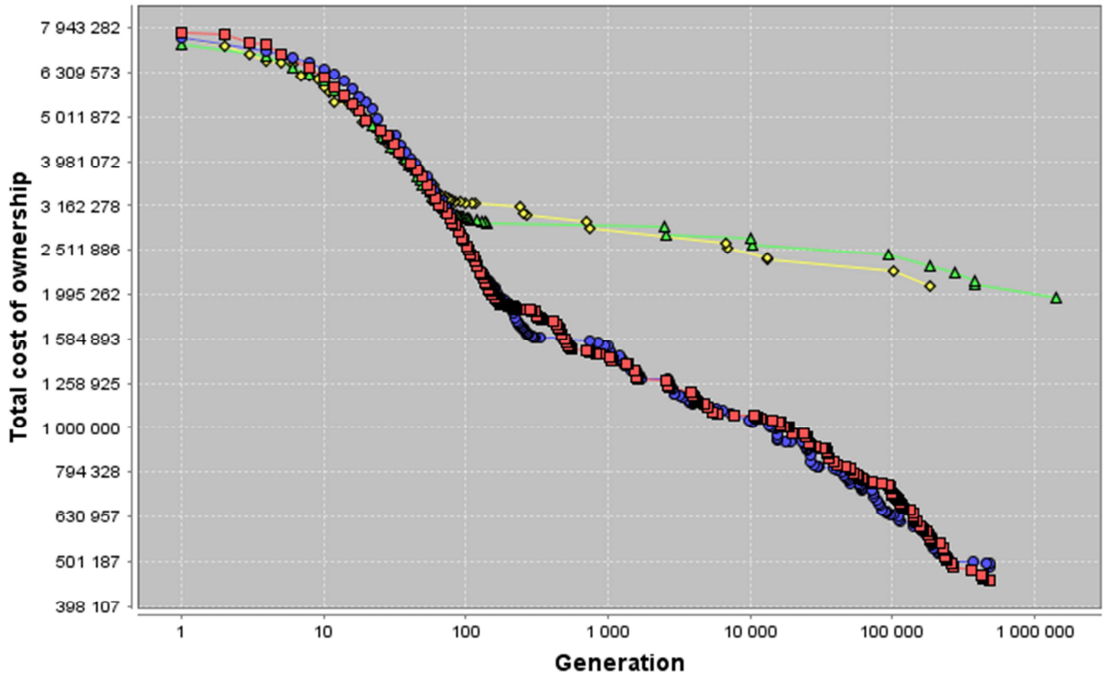


Figure A. **Evaluation processes with different parameters of genetic algorithm**
Source: *GAMBot-Eva* software

As it is seen from the chart modified parameters lead to stagnation of the evolution. After about first 100 generations the changes become rare and the best found

solution remains unchanged. This could be explained by low selection rate of fittest chromosomes for the next generation. According to the parameters almost half of the population is initialized randomly on every generation, and another half of population is produced from previous generation.

Also there are notable clusters of change cases through the evolution. It could be explained by high impact of random factor on the evolution. Eventually new best solution is found and it is updated right away in the next generations because of guidance of fitness function. Then again stagnation takes place until next eventual finding of best solution.

In addition the changes in parameters of genetic algorithm had negative impact on processing time. The time required to process 500 000 generations for genetic algorithm with modified parameters was equal to 2 weeks in comparison with 2 days for original parameters on the same hardware. This could be explained by high utilization of population randomization which minimizes the possibility to apply caching mechanisms.

Annex 3 – *Stage* simulation software configuration file

```
resolution 0.02
interval_sim 100
threads 4

window (
  size [ 862 683 ]
  scale 9.495

  center [ -5.535  1.801 ]
  rotate [ 0.000 0.000 ]

  show_data 1
  show_flags 1
  show_footprints 1
  show_blocks 1
)

#=======================
# lawn and walls
#=======================
define wall_V model (
  color "grey30"
  size [ 0.250 82.000 2.000 ]
  gui_move 0
)

define wall_H model (
  color "grey30"
  size [ 80.000 0.250 2.000 ]
  gui_move 0
)

wall_V ( pose [ -45.0   0.0 0.0 0.0] )
wall_V ( pose [  36.0   0.0 0.0 0.0] )
wall_H ( pose [  -5.0  41.0 0.0 0.0] )
wall_H ( pose [  -5.0 -41.0 0.0 0.0] )

#=======================
# house
#=======================
define roof model (
  #Dark Red
  color_rgba [ 0.65 0.05 0.05 1 ]
  gui_move 0
)

define house model (
  #Light Sky Blue
  color_rgba [ 0.53 0.81 0.98 1 ]
  gui_move 1
  obstacle_return 1
)

house (
  size [ 12.000 15.000 4.000 ]
  pose [ 27.500 -34.000 0.000 90.000]
  name "house1"
  roof(
    size [ 12.000 15.000 1.000 ]
    roof(
      size [ 6.000 15.000 1.000 ]
      roof(
        size [ 1.000 15.000 1.000 ]
      )
    )
  )
)

house (
  size [ 6.000 18.000 3.000 ]
  pose [ -10.000 15.000 0.000 0.000 ]
  name "house2"
  roof(
    size [ 6.000 18.000 1.000 ]
    roof(
      size [ 3.000 18.000 1.000 ]
      roof(
        size [ 1.000 18.000 1.000 ]
      )
    )
  )
)

#=======================
# grass
#=======================
include "grass.def"
include "grass_simple.inc"

#=======================
# straw
#=======================
include "straw.def"
include "straw_pool.inc"

#=======================
# dumpster
#=======================
include "dumpster.def"

dumpster (
  pose [ -40.000 -35.000 0.000 0.000 ]
)

#=======================
# robots
#=======================
include "robot.def"

robot_mower(
  name "rob_mower1"
  pose [ -40.000 2.000 0.000 0.000]
)
robot_mower(
  name "rob_mower2"
  pose [ -40.000 4.000 0.000 0.000]
)
robot_mower(
  name "rob_mower3"
  pose [ -40.000 6.000 0.000 0.000]
)
robot_transp(
  name "rob_transp1"
  pose [ -40.000 8.000 0.000 0.000]
)
robot_transp(
  name "rob_transp2"
  pose [ -40.000 10.000 0.000 0.000]
)
robot_transp(
  name "rob_transp3"
  pose [ -40.000 12.000 0.000 0.000]
)
robot_univ(
  name "rob_univ1"
  pose [ -40.000 14.000 0.000 0.000]
)
robot_univ(
  name "rob_univ2"
  pose [ -40.000 16.000 0.000 0.000]
)
robot_univ(
  name "rob_univ3"
  pose [ -40.000 18.000 0.000 0.000]
)
```

Annex 4 – *Player* software configuration file

```
driver (
  name "stage"
  provides [ "6650:simulation:0" ]
  plugin "stageplugin"
  worldfile "mrs_grass_simple.world"
  usegui 1
)

driver (
  name "stage"
  provides [ "6661:position2d:0" "6661:ranger:0" "6661:blobfinder:0" "6661:fiducial:0" ]
  model "rob_mower1"
)

driver (
  name "stage"
  provides [ "6662:position2d:0" "6662:ranger:0" "6662:blobfinder:0" "6662:fiducial:0" ]
  model "rob_mower2"
)

driver (
  name "stage"
  provides [ "6663:position2d:0" "6663:ranger:0" "6663:blobfinder:0" "6663:fiducial:0" ]
  model "rob_mower3"
)

driver (
  name "stage"
  provides [ "6671:position2d:0" "6671:ranger:0" "6671:blobfinder:0" "6671:fiducial:0" ]
  model "rob_transp1"
)

driver (
  name "stage"
  provides [ "6672:position2d:0" "6672:ranger:0" "6672:blobfinder:0" "6672:fiducial:0" ]
  model "rob_transp2"
)

driver (
  name "stage"
  provides [ "6673:position2d:0" "6673:ranger:0" "6673:blobfinder:0" "6673:fiducial:0" ]
  model "rob_transp3"
)

driver (
  name "stage"
  provides [ "6681:position2d:0" "6681:ranger:0" "6681:blobfinder:0" "6681:fiducial:0" ]
]
  model "rob_univ1"
)

driver (
  name "stage"
  provides [ "6682:position2d:0" "6682:ranger:0" "6682:blobfinder:0" "6682:fiducial:0" ]
  model "rob_univ2"
)

driver (
  name "stage"
  provides [ "6683:position2d:0" "6683:ranger:0" "6683:blobfinder:0" "6683:fiducial:0" ]
  model "rob_univ3"
)
```