



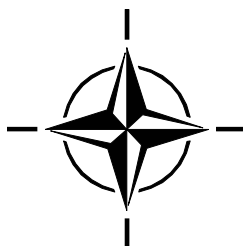
RTO TECHNICAL REPORT

TR-IST-032

Multi-Robot Systems in Military Domains

(Les systèmes multi-robots dans
les domaines militaires)

Final Report of Task Group IST-032/RTG-014.



Published December 2008





RTO TECHNICAL REPORT

TR-IST-032

Multi-Robot Systems in Military Domains

(Les systèmes multi-robots dans
les domaines militaires)

Final Report of Task Group IST-032/RTG-014.

The Research and Technology Organisation (RTO) of NATO

RTO is the single focus in NATO for Defence Research and Technology activities. Its mission is to conduct and promote co-operative research and information exchange. The objective is to support the development and effective use of national defence research and technology and to meet the military needs of the Alliance, to maintain a technological lead, and to provide advice to NATO and national decision makers. The RTO performs its mission with the support of an extensive network of national experts. It also ensures effective co-ordination with other NATO bodies involved in R&T activities.

RTO reports both to the Military Committee of NATO and to the Conference of National Armament Directors. It comprises a Research and Technology Board (RTB) as the highest level of national representation and the Research and Technology Agency (RTA), a dedicated staff with its headquarters in Neuilly, near Paris, France. In order to facilitate contacts with the military users and other NATO activities, a small part of the RTA staff is located in NATO Headquarters in Brussels. The Brussels staff also co-ordinates RTO's co-operation with nations in Middle and Eastern Europe, to which RTO attaches particular importance especially as working together in the field of research is one of the more promising areas of co-operation.

The total spectrum of R&T activities is covered by the following 7 bodies:

- AVT Applied Vehicle Technology Panel
- HFM Human Factors and Medicine Panel
- IST Information Systems Technology Panel
- NMSG NATO Modelling and Simulation Group
- SAS System Analysis and Studies Panel
- SCI Systems Concepts and Integration Panel
- SET Sensors and Electronics Technology Panel

These bodies are made up of national representatives as well as generally recognised 'world class' scientists. They also provide a communication link to military users and other NATO bodies. RTO's scientific and technological work is carried out by Technical Teams, created for specific activities and with a specific duration. Such Technical Teams can organise workshops, symposia, field trials, lecture series and training courses. An important function of these Technical Teams is to ensure the continuity of the expert networks.

RTO builds upon earlier co-operation in defence research and technology as set-up under the Advisory Group for Aerospace Research and Development (AGARD) and the Defence Research Group (DRG). AGARD and the DRG share common roots in that they were both established at the initiative of Dr Theodore von Kármán, a leading aerospace scientist, who early on recognised the importance of scientific support for the Allied Armed Forces. RTO is capitalising on these common roots in order to provide the Alliance and the NATO nations with a strong scientific and technological basis that will guarantee a solid base for the future.

The content of this publication has been reproduced directly from material supplied by RTO or the authors.

Published December 2008

Copyright © RTO/NATO 2008
All Rights Reserved

ISBN 978-92-837-0043-2

Single copies of this publication or of a part of it may be made for individual use only. The approval of the RTA Information Management Systems Branch is required for more than one copy to be made or an extract included in another publication. Requests to do so should be sent to the address on the back cover.

Table of Contents

	Page
Membership Roster	viii
Executive Summary and Synthèse	ES-1
Chapter 1 – Introduction	1-1
1.1 Background	1-1
1.2 Military Benefit	1-1
1.3 Theme	1-1
1.4 Justification	1-2
Chapter 2 – General Information	2-1
2.1 Dates and Places of Meetings	2-1
2.2 General Useful Information	2-1
2.3 Status of the Technical Team: (LPTT or not)	2-1
2.4 Status of Activities	2-1
2.4.1 Netherlands	2-1
2.4.2 Canada	2-2
2.4.3 USA	2-2
2.4.4 Germany	2-2
2.4.5 Belgium	2-3
2.4.6 France	2-3
2.5 Expectations of Future Progress	2-3
2.6 Liaison with Other Bodies	2-3
Annex A – Publication from the Core Group	A-1
Bridging the Gap in Military Robotics – A NATO Workshop on Short-Term Realizable Military Robots	A-2
Military Tasks and Requirements	A-2
Task 1: Reconnaissance and Surveillance for Tactical Support for the Forces on the Ground Including NBC	A-2
Task 2: De-Mining – Tactical and Post-Conflict – Clearing Roads and Fields from AP and AT Mines	A-3
Task 3: Convoying – Transport of Goods	A-3
Task 4: Checking Vehicles and People for Explosives and Weapons at Checkpoints	A-3
Task 5: Carry Equipment for Dismounted Soldier	A-3
Technological Gaps and the Way to Close Them	A-4
Communication	A-4
Robot Platforms	A-5
Sensing and World Modelling	A-6

Navigation and Mission Planning	A-7
Human-Robot Interaction	A-7
Multi-Robot Systems	A-9
Follow-up: The Core Group	A-9
Authors	A-10

Annex B – White Paper from the Core Group **B-1**

Annex C – Multi-Robot Systems Workshop 2002 **C-1**

Multi-Robot Systems: From Swarms to Intelligent Automata	C-1
Preface	C-1
Index	C-2
Part I – Localization, Mapping and Navigation	C-2
Part II – Distributed Surveillance	C-2
Part III – Manipulation	C-2
Part IV – Coordination and Formations	C-2
Part V – Sensor and Hardware Issues	C-3
Part VI – Design and Learning	C-3
Part VII – Human/Robot Interaction	C-3

Annex D – Multi-Robot Systems Workshop 2003 **D-1**

Multi-Robot Systems: From Swarms to Intelligent Automata Volume II	D-1
Preface	D-1
Index	D-2
Part I – Planning and Task Allocation	D-2
Part II – Coordination and Control	D-2
Part III – Sensor and Information Fusion	D-2
Part IV – Learning	D-2
Part V – Self-Reconfigurable Robots	D-3
Part VI – Large-Scale Robot Teams	D-3
Part VII – Human-Robot Teams	D-3
Part VIII – Communication Constraint and Networks	D-3
Part IX – Poster Abstracts	D-3

Annex E – Multi-Robot Systems Workshop 2005 **E-1**

Multi-Robot Systems: From Swarms to Intelligent Automata Volume III	E-1
Preface	E-1
Index	E-2
Part I – Task Allocation	E-2
Part II – Coordination in Dynamic Environments	E-2
Part III – Information / Sensor Sharing and Fusion	E-2
Part IV – Distributed Mapping and Coverage	E-2
Part V – Motion Planning and Control	E-3
Part VI – Human-Robot Interaction	E-3

Part VII – Applications	E-3
Part VIII – Poster Short Papers	E-3

Annex F – Survey of Robotic Control and Teleoperation Architectures **F-1**

Technical Report	F-1
Contents	F-1
Introduction	F-1
Requirements	F-2
Control Architectures for Autonomous Robots	F-4
CLARAty: Coupled Layer Architecture for Robotic Autonomy	F-5
CAMPOUT	F-7
Architectural Components	F-7
DCA	F-9
KAMARA	F-9
OAA and Saphira	F-9
TeamBots	F-9
MissionLab	F-10
MOBILITY (Commercial Product)	F-11
ControlShell (Commercial Product)	F-12
OROCOS (EU Funded Project)	F-13
Teleoperation Applications	F-13
CGAT	F-13
Control System Structure	F-14
WITS	F-15
Telematics Applications	F-16
Comparison Criteria	F-16
Comparison of the Different Applications	F-18
Conclusion	F-19
Related Technologies	F-19
RoboML	F-19
ACE-TAO	F-19
Conclusion	F-20
References	F-20

Annex G – Survey of Software Frameworks for Robotics Applications **G-1**

Technical Report	G-1
Introduction	G-1
Requirements	G-2
Distributed Computing Systems	G-3
Evolution of Programming Languages	G-3
Component-Based Development	G-4
COM (Component Object Model)	G-4
JavaBeans	G-4
Distributed Computing	G-6
Sockets through Native Software API	G-6

Middleware	G-6
Object Request Brokers (ORBs)	G-8
DCOM	G-10
CORBA	G-11
Java RMI	G-14
Java RMI-IIOP	G-16
Jini	G-16
Comparison of the Different Technologies	G-18
Multi-Agents Systems	G-19
Introduction – Definition	G-19
Agent Architecture	G-19
Modular Horizontal Architecture	G-20
Blackboard-Based Architectures	G-20
Subsumption Architecture	G-20
Competitive Tasks	G-20
Production Systems	G-21
Classifier-Based Systems	G-21
Fuzzy Logic Architecture	G-21
Connectionist	G-21
Agent Perception	G-22
Agent Action	G-22
Action as Response to Influences	G-22
Action as Computing Processes	G-23
Action as Local Modification	G-23
Action as Physical Displacement	G-23
Action as Command	G-23
Knowledge Representation	G-23
Multi-Agent Systems Organisation	G-24
Agent Interactions and Cooperation	G-24
Agent Communication	G-25
Collaboration and Distribution of Tasks	G-25
Centralized Allocation	G-26
Distributed Allocation	G-27
Coordination of Actions	G-27
Coordination by Synchronization	G-28
Coordination by Planning	G-28
Reactive Coordination	G-29
Coordination by Regulation	G-29
Eco-Problem Solving	G-29
Taxonomies of Agents	G-29
MAS Frameworks	G-30
Comparison between Distributed Computer Systems and Multi-Agent Systems	G-30
Telerobotics Applications	G-31
TA 2001	G-31
Other Frameworks	G-32
Conclusion	G-32

Annex H – Multi-Robot System Taxonomy	H-1
Annex I – Results of the Preliminary Questionnaire Regarding Multi-Robot Systems	I-1
Annex J – Measures of Effectiveness in the Field of Multi-Robot Systems	J-1

Membership Roster

List of Technical Team members (including address, telephone, fax, and e-mail and indicating the **Principal National Representative** with an asterisk).

Leader: Frank E. SCHNEIDER
Nation: GE
Job Title: Mr.
Company: FGAN/FKIE
Address: Research Establishment for Applied Science (FGAN)
Neuenahrer Str. 20
D-53343 Wachtberg
GERMANY
Phone: +49 (228) 943.54.81
Fax: +49 (228) 943.55.08
E-mail: Frank.Schneider@fgan.de

Expert: Eric COLON
Nation: BEL
Job Title: Cdt d'Avi Ir.
Company: Royal Military Academy
Address: Department of Applied Mechanics
30, avenue de la Renaissance
B-1000 Brussels
BELGIUM
Phone: +32 (2) 737.65.52
Fax: +32 (2) 737.62.12
E-mail: eric.colon@mapp.rma.ac.be

Expert: Bruce DIGNEY
Nation: CAN
Job Title: Dr.
Company: DRDC
Address: Autonomous Land Systems
Defence R&D Canada Suffield
Box 4000
Medicine Hat, Alberta T1A 8K6
CANADA
Phone: +1 (403) 544-4123
Fax: +1 (403) 544-4704
E-mail: Bruce.Digney@drdc-rddc.gc.ca

Expert: Charles FREYHEIT
Nation: FRA
Company: DGA/DSA/SPART
Address: 10, Place Clémenceau
BP 19
92211 St. Cloud
FRANCE
Phone: +33 (0)1 47 71 45 86
Fax: +33 (0)1 47 71 43 70
E-mail: charles.freyheit@dga.defense.gouv.fr

Expert: E.J.A. Van ZIJDERVELD
Nation: NLD
Job Title: Dr. Ir.
Company: TNO
Address: TNO Physics and Electronics
Laboratory
Oude Waalsdorperweg, 63
P.O Box 96864
2509 JG The Hague
NETHERLANDS
Phone: +31 (70) 374.01.29
Fax: +31 (70) 374.06.42
E-mail: vanzijderveld@fel.tno.nl

Expert: Krzysztof KOZLOWSKI
Nation: POL
Job Title: Prof. Dr.
Company: Poznan University of Technology
Address: Institute of Control and Systems
Engineering
ul. Piotrowo 3a
60-965 Poznan
POLAND
Phone: +48 (61) 665 21 97
Fax: +48 (61) 665 21 99
E-mail: krzysztof.kozlowski@put.poznan.pl

Expert: Henrik CHRISTENSEN
Nation: SWE
Job Title: Prof. Dr.
Company: Royal Institute of Technology /
EURON
Address: NADA, CVAP/CAS
Kungl Tekniska Högskolan
SE-100 44 Stockholm
SWEDEN
Phone: + 46 8 790 6792
Fax: + 46 8 723 0302
E-mail: hic@nada.kth.se

Expert: Kenneth PINK
Nation: GBR
Company: QinetiQ
Address: Cody Technology Park
Farnborough GU14 0LX
UNITED KINGDOM
Phone: +44 (0) 1252-393320
Fax: +44 (0) 1252-393091
E-mail: kpink1@QinetiQ.com

Expert: Alan SCHULTZ
Nation: USA
Job Title: Mr.
Company: Naval Research Laboratory
Address: Naval Research Laboratory
Code 5515, Building 1, Room 102
4555 Overlook Avenue, SW
Washington, DC 20375-5337
USA
Phone: +1 (202) 767-2684
Fax: +1 (202) 767-2166
E-mail: schultz@aic.nrl.navy.mil



Multi-Robot Systems in Military Domains

(RTO-TR-IST-032)

Executive Summary

IST-032/RTG-014 aim was to consider the potential of modern multi-robot systems (MRS) for the use in military domains. There has been substantial interest in the use of MRS for a variety of military purposes. A better understanding of human computer and/or robot interaction requirements and development of techniques for evaluating the effectiveness of MRS to meet military purposes will provide military users with a basis for determining their requirements for this technology. The group identified possible application areas for Human-Multi-Robot Systems in military domains:

- Reconnaissance, surveillance and target acquisition (RSTA).
- Ordnance disposal, mine clearing operation, de-mining and NBC decontamination.
- Security, defence and sniper discovery.
- Tactical considerations like decoy, deception and precision strike.
- Transport, convoying and rescue.
- Sensor and communication networks.

The outcomes of the IST-032/RTG-014 are:

- 1) Final report (01 June 2005).
- 2) 3 x International Multi-Robot-Systems Workshops at the NRL in Washington; 2002 – 2003 – 2005 (cont. every 2y); Results published in Springer books.
- 3) Surveys on:
 - Robotic control and Teleoperation architectures; and
 - Software frameworks for robotics applications.
- 4) Multi robot system taxonomy.
- 5) Preparation of the “1st European Tactical Robotic Trials”; planned 05/2006; lead Germany.

Les systèmes multi-robots dans les domaines militaires (RTO-TR-IST-032)

Synthèse

L'objectif de l'IST-032/RTG-014 était d'étudier le potentiel des systèmes multi-robots (MRS) modernes dans l'optique de leur utilisation dans des domaines militaires. Le recours à des MRS pour accomplir diverses tâches militaires suscite un vif intérêt. Une meilleure compréhension des fondamentaux de l'interaction homme-ordinateur et/ou homme-robot et le développement de techniques d'évaluation de l'efficacité des MRS en vue de répondre aux besoins militaires fournira aux utilisateurs militaires une base pour déterminer leurs exigences envers cette technologie. Le groupe a identifié des domaines d'application éventuels pour les systèmes multi-robots-humanoïdes :

- Reconnaissance, surveillance et acquisition d'objectif (RSTA).
- Neutralisation des explosifs et munitions, opérations de relevage des champs de mines, déminage et décontamination nucléaire, biologique et chimique (NBC).
- Sécurité, défense et localisation des snipers.
- Considérations tactiques telles que l'utilisation de leurres, la déception et les frappes de précision.
- Transport, opérations de convoyage et de secours.
- Réseaux de capteurs et de communication.

Le travail de l'IST-032/RTG-014 s'est conclu par :

- 1) Un rapport final (en date du 1^{er} juin 2005).
- 2) 3 ateliers internationaux sur les systèmes multi-robots, qui se sont tenus au Laboratoire de recherche navale (NRL) de Washington en 2002 – 2003 – 2005 (tous les 2 ans) ; Résultats publiés dans les livres Springer.
- 3) Des enquêtes sur :
 - Les architectures de contrôle robotique et de téléopération ; et
 - Les environnements logiciels pour les applications robotiques.
- 4) La taxinomie des systèmes multi-robots.
- 5) La préparation des « 1^{ers} essais robotiques tactiques européens » ; programmés en mai 2006 ; menés en Allemagne.

Chapter 1 – INTRODUCTION

This report describes the goals, set-up and achievements of NATO RTO working group IST-032/RTG-014 “Multi-robot systems in military domains”.

1.1 BACKGROUND

In the mid 1990’s the Multi-Robot Systems (MRS) technology became a new subject for Research and Development in several NATO nations. The main objective was the substitution of personnel in hazardous environments and during fatigue activities. So far few applications of co-operative robotics have been reported and supporting theory is still in a formative stage.

The study of multiple robot systems naturally extends research on single robot systems, but is also a discipline itself: multiple robot systems can accomplish tasks that no single robot can accomplish since ultimately a single robot, no matter how capable, is spatially limited. Multi-Robot Systems are also different from other distributed systems because of their implicit “real world” environment.

There is a multitude of efforts in MRS across NATO members and PfP nations. Currently, there is no significant communications channel among these disparate efforts. In order to prevent duplication of efforts and to facilitate exchange and co-operation, it is necessary to bring together all the parties in this area to discuss their efforts, findings, concerns. These meetings will forge co-operation and joint efforts.

1.2 MILITARY BENEFIT

There has been substantial interest in the use of MRS for a variety of military purposes. A better understanding of human computer and/or robot interaction requirements and development of techniques for evaluating the effectiveness of MRS to meet military purposes will provide military users with a basis for determining their requirements for this technology. Possible application areas for Human-Multi-Robot Systems in military domains are:

- Reconnaissance, surveillance and target acquisition (RSTA);
- Ordnance disposal, mine clearing operation, de-mining and NBC decontamination;
- Security, defence and sniper discovery;
- Tactical considerations like decoy, deception and precision strike;
- Transport, convoying and rescue; and
- Sensor and communication networks.

Currently, there are a number of disparate efforts across NATO and PfP countries. The potential benefits to the military user community should be exchange of information and forging communications and joint efforts of all concerned nations. The nations of France, Germany, Belgium, Netherlands, the United Kingdom, Italy, Canada and the United States are involved in this proposed activity. Additional nations should be recruited to this effort.

1.3 THEME

The aim of the exploratory group on “multi robot systems in military domains” is to consider the potential of modern multi-robot systems for the use in military domains and to draft TOR for a project.

1.4 JUSTIFICATION

Substitution of personnel in hazardous environments and during fatigue activities is an important military requirement. Many soldiers are injured or killed during the handling and manipulation of dangerous material, especially if the work has to be repeated over a longer period of time. The present generation of uninhabited vehicles are individual special-purpose systems, e.g. for de-mining or bomb disposal, with quite limited capabilities. Most of these systems have, because of the complex and non-co-operative environment, just a small degree of autonomy. Autonomy is increasing constantly but in the near future only systems with adjustable autonomy allow efficient and effective mission-operation. In addition, successful task accomplishment necessitates the inset of multiple robot systems in most of the relevant domains. Current research is concerned with the topic of how single autonomous robots can operate as a robot-team. The goal is to attain the highest possible level of autonomy for single robots as well as for groups of robots. These systems also have to be supervised and guided by a human operator with the aid of an assistance-system. Such special C2I systems already exist for single robot systems. However, supervising and controlling more than one active robot is a highly complicated and demanding task, which has not been addressed yet. However, the requirement for cuts in the defence budgets leads to the reduction of personnel in all NATO-partner countries, while at the same time casualties in out-of-area missions become ever more difficult to be politically justified. This significantly increases the demand of personnel substitution from potentially hazardous missions.

Chapter 2 – GENERAL INFORMATION

2.1 DATES AND PLACES OF MEETINGS

- 0th meeting: 22-23 January 2001; FGAN-FKIE, Wachtberg, Germany
- 1st meeting: 27-28 November 2001; Royal Military Academy, Brussels, Belgium
- 2nd meeting: 18-21 March 2002; Naval Research Laboratory, Washington; USA
- 3rd meeting: 05-07 September 2002; Instituto Superior Técnico; Lisbon, Portugal
- 4th meeting: 19-20 March 2003; Naval Research Laboratory, Washington; USA
- 5th meeting: 15-16 September 2003; University of Catania, Catania, Italy
- 6th meeting: 20-22 September 2004; FGAN-FKIE, Wachtberg, Germany

2.2 GENERAL USEFUL INFORMATION

The RTG has a regular information exchange with the EURON Special Interest Group “Co-operative robotics” as well as with the 6thFP of the EU and the CEPA 15.

2.3 STATUS OF THE TECHNICAL TEAM: (LPTT OR NOT)

Active Members of the RTG are: ITA, FRA, GBR, USA, SWE, POL, NLD, CAN, BEL, and DEU.

2.4 STATUS OF ACTIVITIES

2.4.1 Netherlands

The Netherlands were in charge of designing and maintaining an unclassified NATO website for the RTG. The website is online since 2001. It includes:

- Basics
- Publications
- Conferences
- Links
- Literature
- Dates
- Members
- Abbreviations

Please see URL: <http://www.rta.nato.int/ISTET015/Main.htm> for further details.

The NL representative is also involved in the work package “Measures of effectiveness for multi robot systems”.

The NL representative also co-organised the NATO workshop on short-term realizable (multi-)robot system in military domains.

2.4.2 Canada

The Canadians, with support from the Netherlands, are in charge of compiling a survey about measures of effectiveness. The CA representative presented new updated approach of a MOE matrix. Please see Annex J for further details.

The workshop showed that such measures of effectiveness for multi robot system are needed very desperately. Not only to categorise the different systems but also some kind of “hard” metric/number/measure is needed to compare the systems. The approach of comparing such systems through competitions like robot soccer seems to be not a very systematic scientific one.

Due to some personnel changes at the Canadian side the work package is hibernating since 2003 but might be continued in the future.

2.4.3 USA

The RTG would like to thank Alan C. Schulz from the NRL (the US representative) for his tremendous work in organising and hosting the “International Workshop on Multi Robot Systems” (see annexes and <http://www.pao.nrl.navy.mil/robots/> for details). Alan Schultz, with support from the other RTG members, agreed to hold future workshops as a regular event, every second year.

The US is also in charge of compiling a keyword tree and a literature database. The US representative presented an improved keyword tree as well as a quite large literature database. Some of the new entries of the database are from the book “Robot Teams” by Lynne Parker and Tucker Balch. All members are asked to send any keywords or literature regarding MRS directly to the US representative. The work should be continued.

Please see Annex H for the keyword tree and the web site for the up-to-date literature database. The database is too large to include it in the minutes.

2.4.4 Germany

Germany was in charge of the activity to design a questionnaire regarding MRS. The questionnaire was sent to 255 different research facilities, companies, universities and other institutions concerned with MRS.

Please see Annex I for details.

The GE representative also organised the NATO workshop on short-term realizable (multi-)robot system in military domains.

In military robotics, industry often has a leading role in defining the military use of robotics and more or less standard robots are introduced into the military operations without very thoroughly defined military functional requirements. These robots are then tested on how they function in a military operational environment. According to the NATO working group IST-032/RTG-014, this undesired situation needs improvement.

To try to solve this situation for the first time in military robotics, an integral analysis was made in the form of a workshop to bring together military users, industry and researchers and bridge the gap on robotics between these three groups by generating a roadmap, indicating what gaps between military user requirements for robotics and industrial robotic capabilities are to be expected in the year 2008, what actions should be taken to close them in time and who should take action.

The workshop participants were 70, divided amongst military, industry, research and government, from sixteen mainly European countries.

Please see Annex A and Annex B for details.

2.4.5 Belgium

The Belgians were in charge of compiling two surveys:

- 1) Review of robotic control and teleoperation architectures.
- 2) Review of software frameworks for robotics applications.

Both surveys were finished. The RTG would like to express its compliments to Eric Colon for his work. The surveys can be found in Annex F and Annex G.

2.4.6 France

Since France did not frequently participate in the group, the work package about Multi Agent Systems is discontinued.

2.5 EXPECTATIONS OF FUTURE PROGRESS

The RTG will concentrate on multi robot systems that are:

- 1) Short term realisable;
- 2) Field deployable; and
- 3) Allow a combination of UGV, UAV and UUV.

We hope that the “International Workshop on Multi Robot Systems” will continue since it is a unique high quality forum for information exchange in the field of multi-robot systems.

One of the major milestones is the workshop on “Short term realisable (multi-)robot systems in military domains”. The RTG members agreed on the fact that the resulting road map and generic project plan, developed by experts in that area will be an excellent booster for the activities in the field of MRS. It might actually be the source for a joint R&D project in the MRS field. (See Annex A and Annex B for details on results out of this workshop).

2.6 LIAISON WITH OTHER BODIES

Yet, there is no liaison with other bodies. But it is planned to invite experts from other RTG to the next workshop on “Short term realisable (multi-)robot systems in military domains” to give there expertise.



Annex A – PUBLICATION FROM THE CORE GROUP

After the September 2004 workshop in Bonn on roadmaps for short-term realizable military robots, a so-called Core Group was formed to continue the work on this topic. One of the results is a publication on the results of the workshop that will be published in various magazines and is presented in this annex. Another result of the Core Group, a full-blown white paper on requirements and gaps in short-term military robotics is presented in Annex B.

Now follows the publication.

Bridging the Gap in Military Robotics

A NATO Workshop on Short-Term Realizable Military Robots

In military robotics, industry has a rather leading role in defining the military use of robotics. Often more or less standard robots are introduced into the military operations without very thoroughly defined military functional requirements. These standard robots are then tested on how they function in a military operational environment. To the opinion of the NATO working group IST-032/RTG-014¹, this is an undesired situation that needs improvement.

Therefore, the working group organized a workshop in Bonn, Germany to bring together military users, industry and researchers and bridge the gap on robotics between military users, industry and researchers. A final outcome of the workshop would be a roadmap, indicating what gaps between military user requirements for robotics and industrial robotic capabilities are to be expected in the year 2008, what actions should be taken to close those gaps in time, and who should take action. It was recognized during the workshop that this was the first time such an integral analysis was made.

The workshop consisted of several meetings during three days and was attended by about 70 people from the military, industry, research and government, from sixteen mainly European countries.

MILITARY TASKS AND REQUIREMENTS

Starting point in the workshop was defining the tasks for which the military would most like to have robotic support by the year 2008, including the functional requirements. This would put user requirements upfront the process of bridging the gap between military users, industry and research.

The five most relevant tasks identified are explained in the following sections.

Task 1: Reconnaissance and Surveillance for Tactical Support for the Forces on the Ground Including NBC

In this task, the users envisage to use a single UGV (Unmanned Ground Vehicle) for zone, area, route and point reconnaissance. By combining these various activities in a single UGV, users try to explicitly indicate that they would prefer a UGV that can be used in multiple settings.

Users want the UGV to give them information about location and movement of persons and vehicles (civil and military), including as much as possible an Identification Friend or Foe (IFF) indication. The UGV should also provide information on NBC contaminated locations and map information on routes. It should warn the operator when a threat for an area of responsibility is detected.

Users also made requirements about potential use under all weather conditions, in all climates and in all sorts of terrain. This sort of UGV is likely to merge information from various sensors and vehicles and should give the operator only information when relevant. The UGV should be able to communicate with other manned or unmanned vehicles to point targets and to identify persons and vehicles with high reliability, and it should have sufficient capabilities to protect itself.

¹ IST-032/RTG-014 is called “Multi-robot systems in military domains”. Although this group’s primary focus is on multi-robot systems, in the September 2004 workshop focus was put on single robots instead. This is caused by the quite short time horizon of the workshop, looking into robots being available in the year 2008, while multi-robots are not expected to be close to being functional by then.

Task 2: De-Mining – Tactical and Post-Conflict – Clearing Roads and Fields from AP and AT Mines

In the de-mining UGV, users envisage to combine different types of de-mining that are currently distinct disciplines. Especially tactical and post-conflict de-mining have quite different requirements in de-mining speed and de-mining accuracy.

This UGV should be capable of detecting all types of anti-personnel (AP) and anti-tank (AT) mines and optionally mark the mines. Wherever possible, the UGV should either remove the mine or disarm it in place. When a mine is found, the UGV should warn the operator and also provide the mine's location and type, including the estimated time and success rate to clear the mine. Optionally, the UGV should inform the operator when the mine is marked or disarmed. It should be possible to operate the UGV both in a tele-operated and in an autonomous mode when clearing the assigned area or route. The UGV should be usable on all kinds of roads and fields as well as in urban terrain. It should operate under all climatic and weather conditions. The UGV should be able to receive and use airborne information on possible mine locations.

Task 3: Convoying – Transport of Goods

In this task, the military envisage to use UGVs in a convoy for transport of goods both on roads and in heavy terrain to reach non-compound troops. This type of UGV should be able to transport goods in an adequate time, meaning in a time comparable to that of a human driver.

The UGV should either be capable of loading and unloading itself or be capable to be handled by one or more specialized UGVs in the convoy. The location of the convoy should be known at all times. The UGV must be able to follow a non-physically linked, man driven vehicle and it must also be able to move autonomously, whereby the operator should be able to take over control at any time. To the users, an acceptable solution would be a leader-follower vehicle concept. Highly important is the capability of this type of UGV to mix in normal traffic, including all legal issues involved.

Task 4: Checking Vehicles and People for Explosives and Weapons at Checkpoints

This type of UGV should approach a suspected vehicle or person and search for weapons and explosives.

When found, the UGV should alert the operator and keep the vehicle and person from moving away. The UGV should be able to shield off an explosion for own forces or buildings. It should also be able to identify the type of explosive and alert all persons in the vicinity of the explosive. The typical working environment for this UGV is at checkpoints, meaning that it is to be operated on a road. For optimal usability, the UGV should memorize cars and persons spotted at the checkpoint for analysis later on.

Task 5: Carry Equipment for Dismounted Soldier

This UGV is meant to carry the equipment and supplies of a small number of soldiers. Users indicated that it should also be usable to transport wounded soldiers.

The UGV should be able to follow soldiers nearly anywhere they go and therefore it should be usable in all terrains. It should be possible to operate the UGV from a distance, for instance to make the purposely left behind UGV drive up to the soldiers as soon as the operational situation is safe enough. To be useful, the UGV must operate for at least 48 hours on a stretch, and preferably should weigh less than 100 kilograms. Users indicated that the UGV should be able to provide power to the squad equipment and should also be a communication up-link. The UGV should have self defence against thieves, and should memorize who touched it.

TECHNOLOGICAL GAPS AND THE WAY TO CLOSE THEM

During the workshop, industry and researchers defined the current status of technology and the level of technology that is expected to be achieved by the year 2008 at the current speed of technological development. For this purpose, six technological groups were organised along the lines of six different fields of technological interest.

Together with military users, these technological groups also identified the gaps they foresee for the year 2008 between user requirements and the then expected technology status. After that, roadmaps were constructed that identify which actions should be taken in order to achieve the required level of technology by 2008. It was also identified who should take action and how this should be organized.

The outcome of this process is described below for each of these six technological groups.

Communication

Communication is essential for the use of all kinds of robot systems. In most cases, especially when using multi-robot systems where several robots deliberately co-operate in an autonomous manner, there is a demand for wireless communication to achieve high flexibility. In single robot systems, the communication system is usually used to control the robot and to get information from the system sensors (e.g. vision, radar, et cetera). For example, in a reconnaissance and surveillance scenario the task is to gather information about the surrounding area of the robot system. Multi-robot systems combine the functionality of single robot systems to achieve a higher efficiency and to cope with scenarios that are more complex; in the surveillance scenario example an object could be observed by a multi-robot system from different positions and with different sensors. Through the results of a sensor data fusion process, it would be possible to get a more complete and exhaustive situation awareness than achievable with only one robot or sensor. Because of co-operating robot systems, there is a high diversity of demands for the communication system.

The following demands on the communication system were identified:

- Mobile and wireless ad-hoc communication;
- High ranges;
- High data rates;
- Multipoint communication;
- Adjustment to the varying availability of the network;
- Compliance with Quality of Service (QoS) demands;
- Prioritization of data;
- Secure communication; and
- Power awareness.

A generic robot communication system should meet these requirements but today's technology does not support all of these requirements at the same time. Satellite communication for example may allow high data rates over a long distance but there is no solution for a moving robot system. Additionally, most satellite antennas are too big for up to medium robot systems. Other technologies like HF, UHF, and VHF do support high ranges but lack high data rates. GSM, GPRS and UMTS do support medium and high data rates but are in need of an existing infrastructure that either may be under foreign control or does not even exist in the area of operation. Communication technologies like Wireless LAN, Laser or Infrared support ad-hoc communication and high data rates, but have their own weaknesses.

Overall, the currently available communication technologies cannot support all of the above requirements and it is unlikely that a single communication technology may support all of the requirements in 2008. Thus, a new communication approach for robot systems may have to combine different communication technologies. Possibly the integration of the Software Defined Radio (SDR) concept will be of advantage. Another promising approach would be the development of special mobile ad-hoc network (MANET) protocols that focus on covering all the requirements mentioned above. As such, a protocol usually works as a routing protocol on the IP layer it could benefit from improvements on lower layers, for example the integration of the SDR concepts.

Robot Platforms

The robotic platform is the glue that holds together all the other aspects of a fieldable tactical military unmanned ground vehicle. Unless the platform exhibits a high degree of mobility and outstanding ruggedness it will fail to achieve its target location. If the UGV can not deploy its sensors at the correct location then the mission is rendered useless.

The platform should merge the drive system, a power supply system sufficient for the required mission period, an advanced communication system capable of returning real-time information to the user and a user friendly Human Machine Interface (HMI) that allows long term, stress free operations. The platform must have a very high immunity to electro-magnetic interference, and logistically any tactical UGV must not impose a heavy load on available systems or manpower.

As all these functions rely on the chassis or platform to hold the system together, any new tactical platform must be modular in concept. Like the aircraft and scientific industries who already have standards on shape and hole mounting patterns we should strive to arrive at a common standard such that any type of sensor pack could be incorporated into a UGV of a given size. Mounting the sensor is but a part of the integration problem; all equipment requires power, which must be supplied via a connector. Standardisation on power supplies and connectors joining platforms and equipment together would be an advantage.

For near term tactical robotic platforms following developments are needed:

- 1) Develop and integrate the latest power cell technology into the UGV;
- 2) Adopt the latest very high efficiency motor drives and power train systems to give very high mobility even when damaged;
- 3) Refine wheeled and track transmission and suspensions systems (the author thinks that in the near term fieldable walking remote control vehicles will not be possible, however these are a long term aspiration);
- 4) Use of new materials and building methods to reduce mass yet retain performance, including ballistic protection;
- 5) Development of HMI system to ensure tactical robots do not give a high work load on user; and
- 6) Develop an EMC hardening program to ensure C3 systems operate in real world battle space.

In short develop modular platforms that can continue to support the array of sensors and other equipment necessary to prosecute a mission. Be able to adapt to new technologies, be fully integrated into the tactical ISTAR scheme and operate in the real world.

While the near term looks at developing and extending current systems the long term should be looking into achieving what has been identified as a tactical need today but can not yet be realised:

- 1) The users identified that an ideal tactical UGV platform should be capable of going wherever a man can go. Currently no fieldable UGV can jump across a deep ditch or leap through a window. While laboratory grade walking humanoid robots exist they are far from fieldable in a tactical environment.

- 2) Development of highly dextrous humanoid robotics similar to those proposed for the International Space Station would allow any vehicle or plant currently driven by a human to be operated under remote control without the need to modify the vehicle. While laboratory experiments and small scale field trials are currently taking place these will need time and funding to mature into fieldable robotics.
- 3) Research facilities have demonstrated autonomous and semiautonomous operations of UGVs under laboratory conditions as well as swarming and co-operative operations between groups of UGVs or UAVs. These concepts show promise for future operations but is not thought mature enough yet to be fieldable in a real world scenario, for these to mature they too will need funding.

Sensing and World Modelling

The mission success of any robot highly depends on its sensors and world model. The quality of sensor gathered information is important for tele-operated robots that pass this information directly to an operator, but also and even more for autonomous robots that use their sensor information for autonomous navigation and all sorts of autonomous mission dependent information collection subtasks. A good world model is a ‘must have’ for autonomous robots as this is the robot’s total view on the ‘outside world’ and thus the robot’s basis for coherent navigation and execution of mission tasks.

Good sensors and world model are essential for basic information on the robot’s own location and movements, but also for tasks like region observation, route planning or automated detection and recognition of typical targets. The right sensors to use depend on the actual tasks to perform. Examples of current sensors are infrared (imaging) sensors, (HD)TV/CCD-sensors, laser and acoustic sensors and even radar antennas or arrays including mini-SAR (depending on the size of the robot).

Because of the variety of conditions that robots will be operated in, most sensors should be usable under all weather and environmental conditions and in all sorts of terrain. For many tasks the information should be processed on-board of the UGV, and should be effective and efficient by means of information compression and filtering of relevant information.

Concerning sensors and world modelling for military robots, following gaps were identified:

- 1) Obstacle avoidance and negotiation, terrain modelling and classification, and transport in normal traffic, including unstructured terrain. This gap is considered vital as it determines the vehicle’s basic capabilities for navigation, and can only be partly solved by 2008.
- 2) Mine detection, de-mining, chemical and biological sensing (nuclear detection and identification at contact range is considered solved). This gap is considered not vital but important.
- 3) Environmental mapping, sensor fusion at limited visibility, situational awareness as well as human and vehicle detection and recognition. This gap is considered to be solved in research forums for civil applications, and knowledge to be transferred to military applications. In that sense, this gap is not of relevance for a military robot road map.

The greatest challenge will be in multi-sensor suites including sensor fusion, meaning that information from diverse sensors (for instance video and radar) on the UGV is analyzed and then merged into a more complete and robust view on the robot’s ‘outside world’ than can be achieved by any single sensor. Also technological solutions for all weather and environmental conditions and variations of (complex structured) terrain are a challenge. Sensor fusion within reconnaissance vehicles including navigation and mission planning is seen as matured in 2008.

Navigation and Mission Planning

This section includes three different aspects that should be differentiated. Starting from the top level we have the Mission Planning, the Path planning and the Navigation. Mission Planning is mission specific and considerably changes according to the scenario.

Starting with the data provided by the Mission Planning, the Path Planning produces paths and waypoints taking into account the kinematics and dynamic capabilities of the robots involved in the mission. The Navigation consists in avoiding those obstacles while following the initial paths. In order to avoid or pass obstacles and to recover from small path changes (obstacles avoidance and passing) the robots must be equipped with distance and position sensors. Control architectures for Navigation are generally deliberative, meaning that there is a strong coupling between the sensors data and the motion commands sent to the robot actuators. These architectures are based on simple behaviours that are combined using Behaviour Coordination Mechanisms (BCM). In order to implement co-ordination, the system must at least provide the following capabilities: sensor information distribution and distributed behaviour communication and coordination mechanisms (for example through standard Message Passing Protocols like JAUS).

When analyzing the five selected military tasks, several navigation aspects were found vital for military purposes but not yet foreseen to be solved by the year 2008 under the current development speed. Therefore following vital gaps were found in the field of navigation and mission planning:

- Autonomous road following;
- Autonomous driving in mixed traffic (max speed of 50 km/h);
- Moving in all terrains with tactical behaviour in (nearly) all weather conditions; and
- Following leader (manned or autonomous), any type of vehicle.

To achieve the roadmap, following has to be done in the coming years:

- Concept and agree on real target scenarios, prioritise different driving conditions;
- Decide on and develop experimental systems;
- Organise trials (from lab to field prototype to fully operational);
- Define performance measures;
- Improve navigation technology through experimental systems;
- Manage a navigation technology group;
- Develop coordination and interaction with other technology groups; and
- Find funds.

Human-Robot Interaction

While mastering the operation of a fully manually controlled UGV usually is a simple task and operators can be trained effectively using existing equipment, for a realistic combat environment fully manual control was found to be insufficient. This is caused by the emerging operator fatigue and lack of awareness of combat activities at his site. In order to keep the soldier operator aware of the battle development and to allow him to react quickly to improve his personal safety, it is mandatory to shorten the current very long delay needed for the human operator to adapt himself from the remotely monitored environment back to the local real world. Fully manual control does work fine for training of non-experts and for maintaining the capabilities of already well trained operators.

ANNEX A – PUBLICATION FROM THE CORE GROUP

For adequate remote operation in a military operation environment, it is mandatory to upgrade from a continuous manually remote controlled system into a supervised autonomous one. This approach is used by so-called telerobots.

For training systems using Agent Based Systems, the current technology level is at a relatively preliminary state for enabling training of non-experts in very short periods of time, as well as the feasibility to develop such equipment in less than three years. The opinion in the workshop was that the technology is well understood and similar systems are operational in other relevant areas. We assumed that training non-experts for operating relatively sophisticated equipment, like the telerobots, would take one month and that operation is feasible to be done in the next years.

The group considered different aspects of human-robot-interaction including:

- Workload/occupation level for operator performing basic UGV control in simple/ difficult terrain.
- Substituting/supporting UGV operator training/instructing using interactive simulations.
- Evaluating the performance of the human-robot team.
- Defining measures of effectiveness for the human-robot team.
- Consistent interface design for different UGVs for common UGV functions (on/off, manoeuvring, parking etc.), consisting of:
 - Standardized controls (e.g. manoeuvring);
 - Standardized symbolic representation (e.g. ISO, DIN, MIL based symbols); and
 - Standardized layout or sub-layouts for interface components.
- Providing robot execution plan to operator ahead of manoeuvre.
- Scaling operator to robot ratio on demand (adapting to unexpected workload peaks).
- No limitations on interaction caused by UGV losing line of sight (LOS) contact with operator is seen by the users as an important feature.
- No degradation of performance (e.g. speed, accuracy) for basic UGV control when operator is wearing protective gloves/ vest/ full ABC protection.

Following technical issues were found vital to solve before 2008:

- Workload/occupation level less than 50% for operator performing basic UGV control in simple terrain / difficult terrain (75%);
- Providing robot execution plan to operator ahead of manoeuvre;
- Development of appropriate wearable user interface; and
- Evaluating the performance and measures of effectiveness of the human-robot team.

Some other issues were considered not vital, but nevertheless important to be solved:

- Important issue: scaling operator to robot ratio on demand (adapting to unexpected workload peaks);
- Important issue: consistent interface design for different UGVs for common UGV functions (standardized symbolic representation, standardized layout.). Also scaling operator to robot ratio on demand (adapting to unexpected workload peaks); and
- Important issue: Non-degradation of performance because of use of any protective equipment (gloves, vest, NBC gear).

Multi-Robot Systems

A multi-robot system is a system, comprising more than one robot, in which the tasks and activities of the multi-robot system are shared between the robots. The nature of the task sharing could range from simple workload sharing, or distributed sensing, to more complex co-operative or collaborative behaviours. The architecture of the multi-robot system could be fully distributed or it could have a hierarchical command and control structure.

Several multi-robot systems research projects have been performed, such as projects funded by the US, DARPA, which are all aimed at fulfilling reconnaissance or surveillance scenarios. A famous example is Cognitive Colonies, a project that uses robots for reconnaissance tasks (see picture). In Europe, many projects, e.g. Martha and Corom, tried to develop obstacle avoidance and anti collision between units or co-ordination and adaptivity. All projects are still at an exploratory stage, meaning that there are no current operational examples of multi-robot systems in the strategic domain. Therefore, it is important to note that multi-robot systems relate to a potential rather than an actual technology.

Using several initial assumptions – particularly concerning the level of autonomy – and focusing on scenarios relevant for multi-robot systems “Reconnaissance and surveillance for tactical support”, “De-mining – tactical and post-conflict” and “Convoying, transport of goods”, the following vital technology gaps were identified in relation to multi-robot systems:

- 1) To interact with other robots performing different, specialised tasks.
- 2) To perform a task with multiple, collaborative robots.
- 3) To autonomously divide a task, specified by the operator, between several robots.
- 4) Co-operative Perception: to collectively recognise objects of interest.
- 5) The ability to autonomously manage and prioritise events.
- 6) Co-operative Perception: the ability to share data from multiple sources.
- 7) To interact with other robots performing exactly the same task.

Work on most of these issues should start immediately and a new research program - in which the research and industry communities work together - should be established on multi-robot interaction. In fact, they already do work together, but a lack of funding prevents desirable progress. A problem is that military users are interested in multi-robot systems but still have to consolidate funding. Another problem is that current multi-robot system research is ad hoc, as there is no real ‘user pull’.

A solution to this might be a European version of the US DARPA, being a private/public co-operation. This organisation should lobby for funding and define research demands. The initiative for such a private/public co-operation should come from military users in the various countries, in mutual co-operation. This could be part of WEAG or OCCAR.

FOLLOW-UP: THE CORE GROUP

In order to keep the integrating outcomes of the workshop alive, it was decided at the workshop to establish a so-called Core Group, consisting of users, industry and researchers. Although the users initially consist of only military, other users that have related tasks are also invited to participate.

The Core Group is partially a NATO activity and partially a EURON activity, with the NATO part focussing on supporting military-like tasks by robots while the EURON activity focuses on stimulating research to achieve goals relevant to the users and therefore the industry.

ANNEX A – PUBLICATION FROM THE CORE GROUP

One of the main activities of this Core Group is the organization of a capability show, to be held in the second quarter of 2006, aiming at a closer mutual understanding on the needs and possibilities of robotics between military and industry. This capability show is to be followed by a research contest to be held in 2007 that aims to support closing the gap between research and industry. More information on the Core Group and its current activities can be found on the website www.european-robotics.org.

AUTHORS

This publication is a joint effort of following participants.

Group	Name	Role	E-mail
Communication	F.E. Schneider	Chair and Section coordinator	Frank.Schneider@fgan.de
Human-robot interaction	R. Granot	Section coordinator	rgranot@inter.net.il
Human-robot interaction	J. Roning	Co-author	Juha.Roning@oulu.fi
Multi-robot systems	A. Winfield	Section coordinator	Alan.Winfield@uwe.ac.uk
Multi-robot systems	E.J.A. van Zijderveld	Co-author	Erik.vanZijderveld@tno.nl
Multi-robot systems	W. Laurent	Co-author	laurent.walle@cybernetix.fr
Navigation and mission planning	E. Colon	Section coordinator	Eric.Colon@rma.ac.be
Robot platforms	K. Pink	Section coordinator	kpink1@QinetiQ.com
Robot platforms	R. Castelli	Co-author	castelli@macroswiss.com
Sensing and world modelling	D. Krogmann	Section coordinator	Dirk.Krogmann@diehl-bgt-defence.de

Annex B – WHITE PAPER FROM THE CORE GROUP

After the September 2004 workshop in Bonn on roadmaps for short-term realizable military robots, a so-called Core Group was formed to continue the work on this topic. One of the results is a full-blown white paper on the requirements and gaps in short-term military robotics as found in the workshop, which has been published as a separate RTO Technical Report, RTO-TR-IST-052 – links to both the original white paper and RTO-TR-IST-052 are provided below.

Another result of the Core Group, a shorter publication on the same topic, is presented in Annex A.

[Click here to access original white paper.](#)

[Click here to access RTO-TR-IST-052.](#)



Annex C – MULTI-ROBOT SYSTEMS WORKSHOP 2002

Multi-Robot Systems: From Swarms to Intelligent Automata

Proceedings from the 2002 NRL Workshop on Multi-Robot Systems

Edited by Alan C. Schultz and Lynne E. Parker
Kluwer Academic Publishers (ISBN 1-4020-0679-9)

This proceedings volume documents recent cutting-edge developments in multi-robot Systems research, and is the result of a workshop on Multi-Robot Systems that was held in March 2002 at the Naval Research Laboratory in Washington, D.C. The workshop was held as part of the NATO working group IST-032/RTG-014 on Multi-Robot Systems and preceded this group's formal meeting. It brought together top researchers working in areas relevant to designing teams of autonomous vehicles, including robots and unmanned ground, air, surface, and undersea vehicles. The workshop focused on the challenging issues of team architectures, vehicle learning and adaptation, heterogeneous group control and cooperation, task selection, dynamic autonomy, mixed initiative, and human and robot team interaction. A broad range of applications of this technology is presented in this volume, including UCAVs (Unmanned Combat Air vehicles), micro-air vehicles, UUVs (Unmanned Underwater Vehicles), UGVs (Unmanned Ground vehicles), planetary exploration, assembly in space, clean up, and urban search and rescue. This proceedings volume represents the contributions of the top researchers in this field and serves as a valuable tool for professionals in this interdisciplinary field.

PREFACE

In March 2002, the Naval Research Laboratory brought together leading researchers and government sponsors for a three-day workshop in Washington, D.C. on Multi-Robot Systems.

The workshop began with presentations by various government program managers describing application areas and programs with an interest in multi-robot systems. Government representatives were on hand from the Office of Naval Research, the Air Force, the Army Research Lab, the National Aeronautics and Space Administration, and the Defence Advanced Research Projects Agency.

Top researchers then presented their current activities in the areas of multi-robot systems and human-robot interaction. The first two days of the workshop concentrated on multi-robot control issues, including the topics of localization, mapping, and navigation; distributed surveillance; manipulation; coordination and formations; and sensors and hardware. The third day was focused on human interactions with multi-robot teams. All presentations were given in a single-track workshop format. This proceedings document the work presented by these researchers at the workshop.

The invited presentations were followed by panel discussions, in which all participants interacted to highlight the challenges of this field and to develop possible solutions. In addition to the invited research talks, students were given an opportunity to present their work at poster sessions.

This workshop was held in advance of the formal meeting of the NATO working group IST-032/RTG-014 on Multi-Robot Systems in Military Domains. The workshop itself was held, in part, as a way to let the NATO working group members learn more about current efforts within the United States.

We would like to thank the Naval Research Laboratory for sponsoring this workshop and providing the facilities for these meetings to take place, and to the Office of Naval Research for their generous student travel grants.

We are extremely grateful to Magdalena Bugajska and Mitchell A. Potter for their vital help (and long hours) in editing these proceedings. Michelle Caccivio provided the administrative support to the workshop.

INDEX

Part I – Localization, Mapping and Navigation

On the Positional Uncertainty of Multi-Robot Cooperative Localization
Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios

A Multi-Agent System for Multi-Robot Mapping and Exploration
Kurt Konolige, Didier Guzzoni, and Keith Nicewarner

Distributed Heterogeneous Sensing for Outdoor Multi-Robot Localization, Mapping, and Path Planning
Lynne E. Parker, Kingsley Fregcne, Yi Guo, and Raj Madhavan

Mission-Relevant Collaborative Observation and Localization
Ashley W. Stroupe, and Tucker Batch

Deployment and Localization for Mobile Robot Teams
Andrew Howard and Maja J. Mataric

Multiple Autonomous Robots for UXO Clearance, the Basic UXO Gathering System (BUGS) Project
Tuan N. Nguyen, Christopher O'Donnell, and Tuan B. Nguyen

Part II – Distributed Surveillance

Programming and Controlling the Operations of a Team of Miniature Robots
Paul E. Rybski, Sascha A. Stoeter, Maria Gini, and Nikolaos Papanikolopoulos

Autonomous Flying Vehicle Research at the University of Southern California
Srikanth Saripalli, David J. Naffin, and Gaurav S. Sukhatme

Part III – Manipulation

Distributed Manipulation of Multiple Objects Using Ropes
Bruce Donald, Larry Gariepy, and Daniela Rus

A Distributed Multi-Robot System for Cooperative Manipulation
Aveek Das, John Spletzer, Vijay Kumar, and Camilla Taylor

Part IV – Coordination and Formations

A Layered Architecture for Coordination of Mobile Robots
Reid Simmons, Trey Smith, M. Bernardine Dias, Dani Goldberg, David Hershberger, Anthony Stentz, and Robert Zlot

Stability Analysis of Decentralized Cooperative Controls
John T. Feddema and David A. Schoenwald

Snow White and the 700 Dwarves
Brian H. Wilcox

Part V – Sensor and Hardware Issues

GOATS: Multi-platform Sonar Concept for Coastal Mine Countermeasures
Henrik Schmidt and Joseph R. Edwards

Design of the UMN Multi-Robot System
Andrew Drenner, Jan Burt, Brian Chapeau, Tom Dahlin, Bradley Kratochvil, Colin McMillen, Brad Nelson, Nikolaos Papanikolopoulos, Paul E. Rybski, Kristen Stubbs, David Waletzko, and Kemal Berk Yesin

Simulating Self-Organization with the Digital Hormone Model
Wei-Min Shen and Cheng-Ming Chuong

Part VI – Design and Learning

Architecting a Simulation and Development Environment for Multi-Robot Teams
Stephen Balakirsky, Elena Messina, and James Albus

RobotSoccer: A Multi-Robot Challenge
Manuela M. Veloso

Part VII – Human/Robot Interaction

Human-Robot Interactions: Creating Synergistic Cyber Forces
Jean C. Scholtz

Communicating with Teams of Cooperative Robots
D. Perzanowski, A.C. Schultz, W. Adams, M. Skubic, M. Abramson, M. Bugajska, E. Marsh, J.G. Trafton, and D. Brock

Robot as Partner: Vehicle Teleoperation with Collaborative Control
Terrence Fong and Charles Thorpe, Charles Baur

Adaptive Multi-Robot, Multi-Operator Work Systems
Aaron C. Morris, Charles K. Smart, and Scott M. Thayer

User Interaction with Multi-Robot Systems
David Kortenkamp, Debra Schreckenghost, and Cheryl Martin

Human-Robot Interactions in Robot-Assisted Urban Search and Rescue
Robin Murphy and Jenn Casper

Usability Issues for Designing Multi-Robot Missions
Ronald C. Arkin

Perception-Based Navigation for Mobile Robots
K. Kawamura, D. M. Wilkes, A.B. Koku, and T. Keskinpala



Annex D – MULTI-ROBOT SYSTEMS WORKSHOP 2003

Multi-Robot Systems: From Swarms to Intelligent Automata Volume II

Proceedings from the 2003 International Workshop on Multi-Robot Systems

Edited by Alan C. Schultz, Lynne E. Parker and Frank E. Schneider
Kluwer Academic Publishers (ISBN 1-4020-1185-7)

This proceedings volume documents recent cutting-edge developments, in multi-robot systems research and is the result of the Second International Workshop on Multi-Robot Systems that was held in March 2003 at the Naval Research Laboratory in Washington, D.C. This Workshop brought together top researchers working in areas relevant to designing teams of autonomous vehicles, including robots and unmanned ground, air, surface, and undersea vehicles. The workshop focused on the challenging issues of team architectures, vehicle learning and adaptation, heterogeneous group control and cooperation, task selection, dynamic autonomy, mixed initiative, and human and robot team interaction. . A broad range of applications of this technology are presented in this volume, including UCAVs (Unmanned Combat Air Vehicles), micro-air vehicles, UUVs (Unmanned Underwater Vehicles), UGVs (Unmanned Ground Vehicles), planetary exploration, assembly in space, clean-up, and urban search and rescue. This proceedings volume represents the contributions of the top researchers in this field and serves as a valuable tool for professionals in this interdisciplinary field.

PREFACE

The Second International Workshop on Multi-Robot Systems was held in March 2003 at the Naval Research Laboratory in Washington DC, USA. Bringing together leading researchers and government sponsors for three days of technical interchange on multi-robot systems, the workshop follows last year's highly successful gathering, but with a wider international participation.

Like last year, the workshop began with presentations by various government program managers describing application areas and programs with an interest in multi-robot systems. U.S. Government representatives were on hand from the Office of Naval Research, the Air Force, the Army Research Lab, the National Aeronautics and Space Administration, and the Defense Advanced Research Projects Agency.

Top researchers from across the globe then presented their current activities in the areas of multi-robot systems and human-robot interaction. Research was presented in a wide range of areas including planning and task allocation, coordination and control, communication constraints and networking, sensor and information fusion, learning, self-reconfigurable robots, large-scale robot teams, as well as issues in human interaction with multi-robot teams. All presentations were given in a single-track workshop format. This proceedings documents the work presented at the workshop.

The invited presentations were followed by panel discussions, in which all participants interacted to highlight the challenges of this field and to develop possible solutions. In addition to the invited research talks, researchers and students were given an opportunity to present their work at poster sessions.

This workshop was held in advance of the formal meeting of the NATO working group IST-032/RTG-014 on Multi-Robot Systems in Military Domains. The workshop itself was held, in part, as a way to let the NATO working group members learn more about current efforts within the United States.

We would like to thank the Naval Research Laboratory for sponsoring this workshop and providing the facilities for these meetings to take place, and to the Office of Naval Research and DARPA for their generous student travel grants.

We are extremely grateful to Mitchell A. Potter and Magdalena Bugajska for their vital help (and long hours) in editing these proceedings. Michelle Caccivio provided the administrative support to the workshop.

INDEX

Part I – Planning and Task Allocation

On Architectural and Decisional Issues for Multi-Robot Cooperation
Rachid Alami

A Framework for Studying Multi-Robot Task Allocation
Brian P. Gerkey, Maja J. Mataric

Market-Based Multi-Robot Planning in a Distributed Layered Architecture
Dani Goldberg, Vincent Cicirello, M. Bernardine Dias Reid Simmons, Stephen Smith, Anthony Stentz

Collaborative Tasking of Tightly Constrained Multi-Robot Missions
Douglas C. MacKenzie

Part II – Coordination and Control

Multi-Objective Navigation and Control Using Interval Programming
Michael R. Benjamin

Cooperative Relative Localization for Mobile Robot Teams: An Ego-Centric Approach
Andrew Howard, Maja J Mataric and Gaurav S. Sukhatme

Fractional Bandwidth Reacquisition Algorithms for VSW-MCM
Benjamin Cook, Daniel Marthaler, Chad Topaz, Andrea Bertozzi, Mathieu Kemp

Dynamic Multi-Robot Coordination
Douglas Vail, Manuela Veloso

Part III – Sensor and Information Fusion

Information Sharing in Teams of Self-Aware Entities
Jerry Franke, Brian Satterfield, Steve Jameson

Realizing Virtual Sensors by Distributed Multi-Level Sensor Fusion
Edgar Nett, Stefan Schemmer

Information Fusion and Control for Multiple UAVs
Salah Sukkarieh, Eric Nettleton, Ben Gwcholsky and Hugh Durrant-Whyte

Part IV – Learning

Multistrategy Learning Methods for Multirobot Systems
R.C. Arkin, Y. Endo, B. Lee, D. MacKenzie, E. Martinson

Part V – Self-Reconfigurable Robots

Distributed Locomotion Algorithms for Self-Reconfigurable Robots Operating on Rough Terrain
Zack Butler, Daniela Rus

Self-Assembly in Space via Self-Reconfigurable Robots
Wei-Min Shen, Peter Will, Berok Khoshnevis

Part VI – Large-Scale Robot Teams

Optimization of Swarm Robotic Systems via Macroscopic Models
Alcherio Martinoli, Kjerstin Eastern

CentiBOTS: Large Scale Robot Teams
Kurt Konolige, Charles Ortiz, Regis Vincent, Andrew Agno, Michael Eriksen, Benson Limketkai, Mark Lewis, Linda Briesemeister, Enrique Ruspini, Dieter Fox, Jonathan Ko, Benjamin Stewart, Leonidas Guibas

The Effect of Heterogeneity in Teams of 100+ Mobile Robots
Lynne E. Parker

Part VII – Human-Robot Teams

Collaborative Tools for Mixed Teams of Humans and Robots
David J. Bruemmer, Miles C. Walton

Mixed-Initiative Control of Large Human-Robot Teams
Christopher L. Johnson

Experiments in Human-Robot Teams
Curtis W. Nielsen, Michael A. Goodrich, Jacob W. Crandall

Impact of Autonomy in Multirobot Systems on Teleoperation Performance
Boris Trouvain, Hans L. Wolf, Frank E. Schneider

Part VIII – Communication Constraint and Networks

Decentralized Motion Planning for Multiple Robots Subject to Sensing and Communication Constraints
Guilherme A.S. Pereira, Aweek K. Das, Vijay Kumar, Mario F.M. Campos

Exploiting the Interactions Between Robotic Autonomy and Networks
Jason Redi and Joshua Bers

Part IX – Poster Abstracts

Observing Multiple Targets with Multiple Mobile Robots Using Probabilistic Behavior-Based Techniques
Tucker Balch, Ashley Stroupe

Dynamic Coverage via Multi-robot Cooperation
Maxim A. Batalin, Gaurav S. Sukhatme

Emergent Control Surveillance Networks
R.R. Brooks, J. Lamb, M. Pirretti, M. Zhu, S.S. Iyengar

On Dynamic Reconfiguration of Multi-Robot Formations

Rafael Fierro, Aveek K. Das

Distributed Multi-Robot Mapping

Dieter Fox, Jonathan Ko, Benjamin Stewart, Kurt Konolige, Benson Limetkai

An Endogeneous Architecture for Cooperative Sensor Teams

Ben Grocholsky, Alexei Makarenko and Hugh Durrant-Whyte

Efficient Inference Grounding for Distributed Multi-Robot Teams

Aaron Khoo, Ian Douglas Horswill

Annex E – MULTI-ROBOT SYSTEMS WORKSHOP 2005

Multi-Robot Systems: From Swarms to Intelligent Automata Volume III

Proceedings from the 2005 International Workshop on Multi-Robot Systems

Edited by Lynne E. Parker, Frank E. Schneider and Alan C. Schultz
Kluwer Academic Publishers (ISBN 1-4020-3388-5)

This proceedings volume documents recent cutting-edge developments in multi-robot Systems research. This volume is the result of the Third International workshop on Multi-Robot Systems that was held in March 2005 at the Naval Research Laboratory in Washington, D.C. This workshop brought together top researchers working in areas relevant to designing teams of autonomous vehicles, including robots and unmanned ground, air, surface, and undersea vehicles. The workshop focused on the challenging issues of team architectures, vehicle learning and adaptation, heterogeneous group control and cooperation, task selection, dynamic autonomy, mixed initiative, and human and robot team interaction. A broad range of applications of this technology is presented in this volume, including UCAVs (Unmanned Combat Air Vehicles), micro-air vehicles, UUVs (Unmanned Underwater Vehicles), UGVs (Unmanned Ground Vehicles), planetary exploration, assembly in space, clean-up, and urban search and rescue. This proceedings volume represents the contributions of the top researchers in this field and serves as a valuable tool for professionals in this interdisciplinary field.

PREFACE

The Third International Workshop on Multi-Robot Systems was held in March 2005 at the Naval Research Laboratory in Washington, D.C., USA. Bringing together leading researchers and government sponsors for three days of technical interchange on multi-robot systems, the workshop follows two previous highly successful gatherings in 2002 and 2003. Like the previous two workshops, the meeting began with presentations by various government program managers describing application areas and programs with an interest in multi-robot systems. U.S. Government representatives were on hand from the Office of Naval Research and several other governmental offices. Top researchers in the field then presented their current activities in many areas of multi-robot systems. Presentations spanned a wide range of topics, including task allocation, coordination in dynamic environments, information/sensor sharing and fusion, distributed mapping and coverage, motion planning and control, human-robot interaction, and applications of multi-robot systems. All presentations were given in a single-track workshop format. This proceedings document the work presented at the workshop. The research presentations were followed by panel discussions, in which all participants interacted to highlight the challenges of this field and to develop possible solutions. In addition to the invited research talks, researchers and students were given an opportunity to present their work at poster sessions. We would like to thank the Naval Research Laboratory for sponsoring this workshop and providing the facilities for these meetings to take place. We are extremely grateful to Magdalena Bugajska, Paul Wiegand, and Mitchell A. Potter, for their vital help (and long hours) in editing these proceedings and to Michelle Caccivio for providing the administrative support to the workshop.

This workshop was held in advance of the formal meeting of the NATO working group IST-032/RTG-014 on Multi-Robot Systems in Military Domains. The workshop itself was held, in part, as a way to let the NATO working group members learn more about current efforts within the United States.

We would like to thank the Naval Research Laboratory for sponsoring this workshop and providing the facilities for these meetings to take place, and to the Office of Naval Research and DARPA for their generous student travel grants.

INDEX

Part I – Task Allocation

The Generation of Bidding Rules for Auction-Based Robot Coordination

Craig Tovey, Michail G. Lagoudakis, Sonal Jain, and Sven Koenig

Issues in Multi-Robot Coalition Formation

Lovekesh Vig and Julie A. Adams

Sensor Network-Mediated Multi-Robot Task Allocation

Maxim A. Batalin and Gaurav S. Sukhatme

Part II – Coordination in Dynamic Environments

Multi-Objective Cooperative Control of Dynamical Systems

Zhihua Qu, Jing Wang, and Richard A. Hull

Levels of Multi-Robot Coordination for Dynamic Environments

Colin P. McMillen, Paul E. Rybski, and Manuela M. Veloso

Parallel Stochastic Hill-Climbing with Small Teams

Brian P. Gerkey, Sebastian Thrun, Geoff Gordon

Toward Versatility of Multi-Robot Systems

Colin Cherry and Hong Zhang

Part III – Information / Sensor Sharing and Fusion

Decentralized Communication Strategies for Coordinated Multi-Agent Policies

Maayan Roth, Reid Simmons, and Manuela Veloso

Improving Multirobot Multitarget Tracking by Communicating Negative Information

Matthew Powers, Ramprasad Ravichandran, Frank Dellaert, and Tucker Balch

Enabling Autonomous Sensor-Sharing for Tightly-Coupled

Cooperative Tasks Lynne E. Parker, Maureen Chandra, and Fang Tang

Part IV – Distributed Mapping and Coverage

Merging Partial Maps without Using Odometry

Francesco Amigoni, Simone Gasparini, and Maria Gini

Distributed Coverage of Unknown/Unstructured Environments by Mobile Sensor Networks

Ioannis Rekleitis, Ai Peng New, and Howie Choset

Part V – Motion Planning and Control

Real-Time Multi-Robot Motion Planning with Safe Dynamics
James Bruce and Manuela Veloso

A Multi-Robot Testbed for Biologically-Inspired Cooperative Control
Rafael Fierro, Justin Clark, Dean Hougen, and Sesh Commuri

Part VI – Human-Robot Interaction

Task Switching and Multi-Robot Teams
Michael A. Goodrich, Morgan Quigley, and Keryl Cosenzo

User Modelling for Principled Sliding Autonomy in Human-Robot Teams
Brennan Sellner, Reid Simmons, and Sanjiv Singh

Part VII – Applications

Multi-Robot Chemical Plume Tracing
Diana Spears, Dimitri Zarzhitsky, and David Thayer

Deploying Air-Ground Multi-Robot Teams in Urban Environments
L. Chaimowicz, A. Cowley, D. Gomez-Ibanez, B. Grocholsky, M.A. Hsieh, H. Hsu, J.F. Keller, V. Kumar, R. Swaminathan, and C.J. Taylor

Precision Manipulation with Cooperative Robots
Ashley Stroupe, Terry Huntsberger, Avi Okon, and Hrand Aghazarian

Part VIII – Poster Short Papers

A Robust Monte-Carlo Algorithm for Multi-Robot Localization
Vazha Amiranashvili and Gerhard Lakemeyer

A Dialogue-Based Approach to Multi-Robot Team Control
Nathanael Chambers, James Alien, Lucian Galescu, and Hyuckchul Jung

Hybrid Free-Space Optics/Radio Frequency (FSO/RF) Networks for Mobile Robot Teams
Jason Derenick, Christopher Thorne, and John Spletzer

Swarming UAVS Behavior Hierarchy
Kuo-ChiLin

The GNATs – Low-Cost Embedded Networks for Supporting Mobile Robots
Keith J. O'Hara, Daniel B. Walker, and Tucker R. Balch

Role Based Operations
Brian Sattereld, Heeten Choxi, and Drew Housten

Ergodic Dynamics by Design: A Route to Predictable Multi-Robot Systems
Dylan A. Shell, Chris V. Jones, and Maja J. Mataric



Annex F – SURVEY OF ROBOTIC CONTROL AND TELEOPERATION ARCHITECTURES

Technical Report

Eric Colon

Robotics Laboratory
Royal Military Academy

March 2002

CONTENTS

In this document, we summarise and analyse existing robotic control and teleoperation architectures included their software implementation. After the introduction where a layered structure is proposed and discussed, we present the requirements expressed by the Clwar community. The architecture overview has been divided in two sections: the first one contains a review of existing control architectures for autonomous robots while the second one is devoted to teleoperation architectures. A comprehensive comparison of the latter ones is presented at the end of the section.

INTRODUCTION

A control architecture defines the abstract design of a class of agents: the set of components in which perception, reasoning, and action occur; the specific functionality and interface of each component, and the interconnection topology between components. This definition identifies a number of architectural issues (such as perception action components, interfaces and topology between components etc.) that are useful in specifying and describing a particular architecture.

In order to ensure the flexibility and modularity of the architecture, it is necessary to clearly distinguish between the structure of the control (processes, communication, event handling,...) and the functionality (control algorithms, sensor processing, ...). The framework should consist of structural components and application components.

We consider the following top down decomposition:

- System architecture (The way the components are linked together).
- Application components (Control, navigation, path-planning, vision, ...).
- Infrastructure components (Mediator, logging, configuration, ...).
- Components architecture (Internal organisation).
- Basic services (events, messages, threads, synchronisation, timer).
- Language (C,C++, Java, ...) and programming model (OO, binary model-COM, CORBA, DCOM).

At the higher level, robot control architectures can be broadly characterised as deliberative (based on planning), reactive (tight coupling between sensing and actuation), or a hybrid blend of the two. Deliberative and reactive approaches have advantages and drawbacks and it seems advantageous to combine both approaches to profit from their advantages while trying to cancel their drawbacks.

Hierarchical decomposition is the classical approach for developing motion control for autonomous or semi-autonomous robots (we are principally dealing here with mobile robots but the analyse also applies to manipulators).

Typical robot and autonomy architectures are comprised of three levels – Planning, Executive, and Functional (or Control). These levels are usually organised by the level of abstraction in which they operate.

The top Planning level constructs high-level plans utilising AI planning search techniques. In the past, these algorithms have typically been computationally intensive and required a significant amount of time to respond to new updates or changes. Domain knowledge for this level is encoded in a declarative model, where it can easily be utilised by different search techniques.

The Executive level is responsible for execution of plans produced by the planning level. The executive level typically performs further expansion of planned activities based on current execution context. This level is also responsible for monitoring activities as execution proceeds and for handling exceptions as they arise. This level must quickly react to changes, so its behaviour is usually more responsive than that of the planning level. Domain knowledge at the Executive level is typically represented using procedural representations such as looping constructs, conditionals, etc.

The Functional level is responsible for low-level control of the robot. This level typically consists of real-time control loops that directly command the rover hardware and which tightly couple sensors to actuators.

REQUIREMENTS

The following table lists the different requirements for control architecture of a Clawar machine. These criteria have been defined in the task 18 of the Clawar network [MUSC].

We have classified the requirements proposed in this report in different categories. However sometimes a criterion has implications in different categories. The meaning of the categories follows the table. We also review how these criteria have been considered in the design of CoRoDe (our future framework implementation).

Some aspects have not been considered in this study, namely:

- Software portability (OS and hardware).
- Simultaneous control of multiple robots (co-operation).

N°	Category	Criteria
1	OS / H	O/S allows Soft/Hard RT tasks
2	OS / API	Process monitoring
3	H / API	Flexible communication medium
4	H / API	Networking capabilities
5	H / API	Safety
6	API	Connectivity to other software package
7	F / H	Sensor access
8	F	Flexible control station

N°	Category	Criteria
9	F	Hierarchical graphical representation of modules
10	F	Easy exchange between simulation and real system
11	F	Different level of security/access
12	F	Data logging
13	F / A	On-line sensor selection/monitoring
14	F / A	On-line control parameters modification
15	F / A	Free input/output mapping
16	A	Variable levels of autonomy
17	T	Automatic documentation
18	T	Functionality description

OS: Operating System

H: Hardware

API: Application Programming Interface and Programming Language

F: Framework (basic functionality provided by the implementation)

A: Application (functionality provided by application specific modules)

T: Tools (External applications):

- 1) The goal of CoRoDe is to link together and to integrate different robots, sensors and processing modules in a high level network. RT control of robots should be done locally on RT OS or micro controllers.
- 2) It should be possible to add wrappers to such functionality in order to make them available remotely. However, it is not directly addressed by CoRoDe.
- 3) The main communication medium will be Ethernet links (wire or wireless) Serial links should also be considered between high level services and legacy systems.
- 4) This is the one of the basic requirement for CoRoDe.
- 5) Safety: CRC, Timeouts, permissions are considered in CoRoDe.
- 6) The use of open standards and the existence of API should allow the interaction between different software packages.
- 7) Sensors will be accessible if it benefits the application.
- 8) The implementation of adaptive GUI is one of the basic requirements for CoRoDe.
- 9) This will be considered in the development of CoRoDe.
- 10) The use of wrappers around real systems and the definition of standards interface will allow the transparent switch between real and simulated systems.
- 11) This is one of the basic services of the framework.
- 12) idem.
- 13) This will be part of the sensor service.

- 14) idem.
- 15) The definition of interface and data processing modules as filters should allow this functionality.
- 16) This will be implemented as application modules for the framework.
- 17) There exists some software for doing this. (Javadoc, doc++).
- 18) The utilisation of object oriented, component-based software and services should allow fulfilling this requirement.

CONTROL ARCHITECTURES FOR AUTONOMOUS ROBOTS

It is difficult to really compare existing systems because depending on the applications and primary goals and motivation for the development; the focus is put on different aspects of the control architecture. We do not claim that this review is exhaustive but it gives a good idea of what are the research directions followed by the robotic community.

The first part compares some layered architectures with special attention accorded on response times. The second part describes existing control architectures that have been successfully applied to autonomous robots.

The hierarchical structure of the motion control system proposed by [VBRU] for the indoors mobile robot EMIR-1 consists of five levels:

Layer	Function	Response Time
L1	Task specification	
L2	Trajectory generation	
L3	Path following controller	20 ms
L4	Servo control	1 ms
L5	Servo drivers and actuators	

We are here interested by the first three layers, the two bottom-layers being related to automatic control and electronics.

Another hierarchical approach used for the control of a mine detection vehicle at TNO [VDHE] consists of five layers (we permuted the order of the layers name to keep consistency with the previous description):

Layer	Function	Response Time
L1	Task	5 s
L2	Planner (route planning, path points generation, ...)	500 ms
L3	Services: Path control, terrain modelling	50 ms
L4	Kernel: Real Time Processing	5 ms
L5/L6	BIOS I/O, Hardware I/O	

There is a good correspondence between both approaches. However, the L1 in the second approach is at a higher level as it uses a symbolic description and natural language. The L2 contains the L1 and L2 of the first system.

It is impossible to build exact models of real systems and to predict all possible events that can occur during the use of the robot. The behaviour-based architecture was proposed by [Brooks] to cope with those problems. However, it seems that purely reactive systems cannot totally substitute for planned behaviour in solving complex tasks. Many searchers have consequently proposed mixed architecture.

The architecture developed by the DGA (DCE/ITA) is a MAS that combines reactive and deliberative (planning) levels [LUZE]. Both levels having access to a common model database. The deliberative level is linked to the communication level for user interaction.

We consider another example: the USU-ODV robot from the Utah State University [MOOR]. It is a six-wheeled omni directional autonomous robot. This system exhibits a three-level, hierarchical control scheme.

Layer	Function	Response Time
L1	Mission and Path planner	10s
L2	Path tracking controller	200 ms
L3	Low level	100 ms

The system has three distinct modes of control operation: manual closed loop and exception. The exception control block contains logic; it controls and monitors the mission. It can invoke a re-plan event, call emergency procedure or execute an emergency shutdown. The net effect is to make the vehicle a hybrid system.

Saffiotti in [SAFF] proposed a multi-valued logic approach to integrate planning and control.

The control scheme proposed for the Nomad200 in [COLO] fuses the L1 and L2 layers. An environmental map is incrementally built at run time. The user indicates goals directly on the map and a path from the current point is generated based on partial knowledge of the environment. At each control loop iteration, a new path is computed based on the updated map and motion commands for following this path are also computed and sent to the servo controllers. This hybrid architecture combines deliberative (path planning) and reactive (obstacle avoidance) properties into a single algorithm. Motion commands are provided by a fuzzy logic controller. This system is not only elegant but has also proven to be robust.

We give now an overview of complete control architectures.

CLARAty: Coupled Layer Architecture for Robotic Autonomy

(<http://robotics.jpl.nasa.gov/tasks/claraty/>)

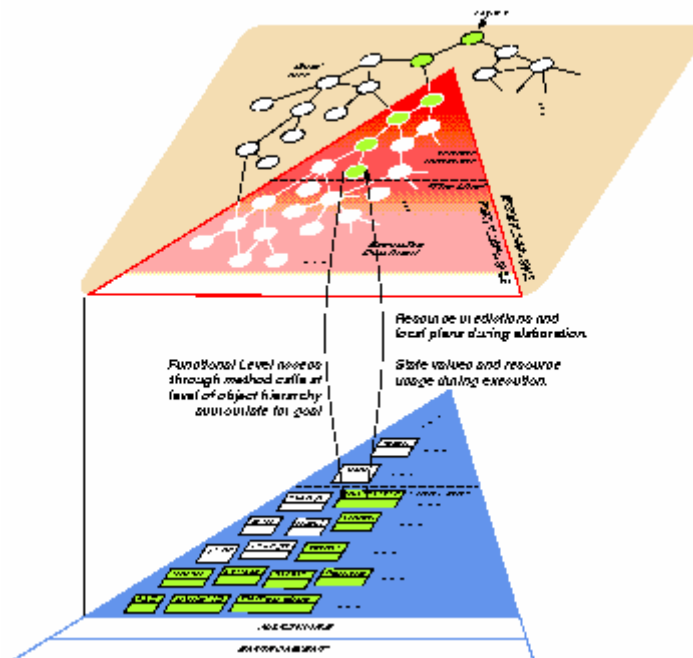
Initially, CLARAty [VOLP] is being structured to encompass the existing functionality of existing NASA robotics software infrastructure. This is enabling immediate utilisation for the implementation of Artificial Intelligence techniques for autonomy and surface operations re-sequencing, and providing a growth path by which existing systems may move into this new architecture. However, extensions of operations to more difficult scenarios, including multiple rover co-ordination, will require utilisation of the full-scale architecture, and may require reformulation of the individual system control implementation as specified by it. Therefore, in future years, this task will demonstrate the multi-asset scenario, as a working proof of the utility of the newly developed formulation.

CLARAty is a two-layered architecture (JPL) where the top Decision Layer contains techniques for autonomously creating a plan of robot commands and the bottom Functional Layer provides standard robot capabilities that interface to system hardware. The decision layer is the result of the fusion of the planning and executive layer. One difference between the CLARAty architecture and the conventional three-level architectures is the explicit distinction between levels of granularity and levels of intelligence. The system decomposition allows for intelligent behaviour at very low levels while still maintaining the structure of the different abstraction levels. This is similar in concept to hybrid reactive and deliberative systems.

Functional layer: An OO system decomposition provides several abstractions for the components of the system. There are three main types of classes: data structure, generic classes and specialised classes. The later are derived from generic ones to add some specific functionality.

Generic and specialised classes can be divided in physical and functional classes. A physical class defines the structure and behaviour of a physical object (motor, joint, wheel, arm, camera,...) while a functional class describes the interface and functionality of a generic algorithm (TrajectoryGen, ObjectFinder, VisualNavigator, StereoVision, ...). Generic functional components may sometimes use generic physical components in their implementation.

Decision layer: the two-tiered architecture enables each tier to operate at all levels of granularity and blends declarative and procedural techniques for decision making. The first instantiation of CLARAty is utilising the CLEAr planning and execution system. CLEAr is a hybrid controller system that is built on top of the CASPER continuous planner and TDL executive system. CASPER provides a soft-real-time capability for performing plan generation, execution, monitoring and re-planning. The planner and executive operate on the same set of activities and timelines and all capabilities are allowed on both near- and far-term activities. The Decision Layer queries the Functional Layer for state and resource information (current and prediction).



CAMPOUT

(<http://prl.jpl.nasa.gov/projects/ate/technology/campout.html>)

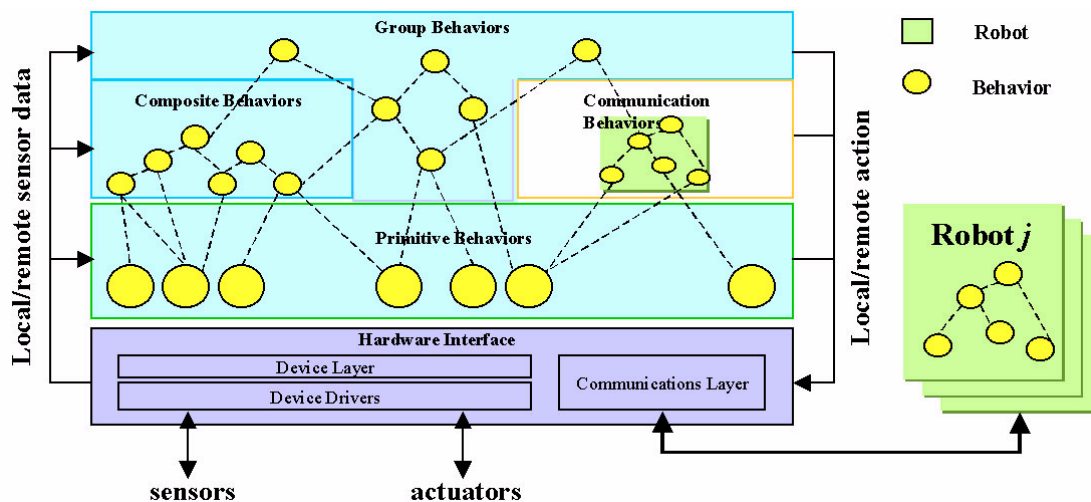
CAMPOUT is a distributed control architecture based on a multi-agent or behaviour-based methodology, wherein higher-level functionality is composed by co-ordination of more basic behaviours under the downward task decomposition of a multi-agent planner.

Objectives:

- Develop a scalable control architecture for the autonomous intelligent co-ordination of multiple, flight-relevant robots, enabling canonical grasp, lift, force, move operations in sloped and rocky natural terrain (acquire-transport-deploy).
- Provide a methodology for analysis, design, specification, and verification of specific implementations of the architecture.
- Develop infrastructure and tools to support implementation and evaluation.

Approach:

- Leverage existing SRR/FIDO code at device driver level.
- Leverage existing behaviour fusion code.
- Finite State Automata.
- Develop communication behaviour library for general co-ordination.
- Design templates for primitive and group behaviour development.
- Design and specification of behaviour hierarchy for co-ordinated transport.



Architectural Components

Behaviour Representation

In our architectural methodology we formalise a behaviour, b , as a mapping, $b: P^* \times X \rightarrow [0; 1]$, that relates each percept sequence $p \in P^*$ and action $x \in X$ pair, (p, x) , to a preference value that reflects the action's desirability. The percept describes possible (processed or raw) sensory input and the N -dimensional action space is defined to be a finite set of alternative actions. The described mapping assigns to each action $x \in X$ a

preference, where the most desired actions are assigned 1 and undesired actions are assigned 0, from that behaviours point of view. Note that this definition of behaviour does not dictate how the mapping is to be implemented but provides a general recipe for behaviour with a well-defined interface (useful when composing behaviours regardless of their roles in a behaviour hierarchy). This representation does not exclude implementation using a look-up-table, a finite state machine, a neural network, an expert system, control laws (such as PID etc.), or any other approach for that matter. Note also that this representation does not restrict us to reactive behaviours since it could have internal state. In that sense, each behaviour can be implemented using whichever approach is appropriate. Finally, traditional, single-valued behaviours fall within this representation because, $b: P^* \rightarrow X$ can be represented by a multi-valued output where all x are associated with 0 but the single x which is selected by the behaviour. In CAMPOUT, this representation is implemented using an N-dimensional array, which will contain the desirability values recommended by behaviour.

Behaviour Composition

Behaviour composition refers to the mechanisms used for building higher-level behaviours by combining lower-level ones. A major issue in the design of behaviour-based control systems is the formulation of effective mechanisms for co-ordination of the behaviours' activities into strategies for rational and coherent behaviour. In a behaviour-based architecture, a complex control problem is divided into a set of simpler control problems (implemented using behaviours) that collectively solve the original complex control problem. To do this, it is necessary to address the problem of the co-ordination of the activities of the behaviours using behaviour co-ordination mechanisms (BCMs) so to satisfy the initial complex system's control objectives. This is achieved through the coordination of the activities of lower-level behaviours within the context of a high-level behaviour's task and objective.

An explicit design goal of CAMPOUT has been to support not one but an arbitrary number of Behaviour Co-ordination Mechanisms (BCMs). BCMs can be divided into two main classes: arbitration and command. For a detailed overview, discussion, and comparison of behaviour co-ordination mechanisms see. Since different BCMs often require different behaviour representations, CAMPOUT uses a multi-valued behaviour representation, which is general enough for a large class of applications.

Behaviour Co-ordination Mechanisms

If behaviours are viewed as operands, then BCMs are the operators used to combine behaviours into higher-level behaviours. In this section, we describe the BCMs that are readily available in CAMPOUT for behaviour composition. BCMs can be divided into two main classes: arbitration and command fusion, which are complementary.

Arbitration mechanisms select one behaviour, from a group of competing ones, and give it ultimate control of the system (the robot) until the next selection cycle. This approach is suitable for arbitrating between the set of active behaviours in accord with the system's changing objectives and requirements under varying conditions. It can focus the use of scarce system resources (sensory, computational, etc.) on tasks that are considered relevant. CAMPOUT implements the following arbitration mechanisms:

- Priority-based arbitration: which is a subsumptive-style, priority-based arbitration mechanism, where behaviours with higher priorities are allowed to suppress the output of behaviours with lower priorities.
- State-based arbitration: which is based on the Discrete Event Systems (DES) formalism, and is suitable for behaviour sequencing.

Command fusion mechanisms combine recommendations from multiple behaviours to form a control action that represents their consensus. Thus, this approach provides for a co-ordination scheme that allows all behaviours to simultaneously contribute to the control of the system in a co-operative rather than a

competitive manner, which makes them suitable for tightly-coupled tasks that require spatio-temporal co-ordination of activities. CAMPOUT provides a number of complementary mechanisms for fusion:

- Voting techniques.
- Fuzzy command fusion mechanisms.
- Multiple objective behaviour fusion.

DCA

(<http://www.nada.kth.se>)

DCA is distributed control architecture for robotics, which supports communications and modularity [PETE]. The DCA system was designed with mobile robot performing manipulation in mind. The MSDE has been developed to provide portable communications and process management for real-time systems. The components are GeneralComms, NameServer, TimeServer, Executer and DBManager. The DCA consists of instances of two functionally different parts: a supervisor that organises the execution of controller modules. Different processes can be grouped and treated as a single process from an external viewer. The adoption of process algebra provides the potential of direct task-level specification in a manner, which is human friendly as well as suitable for automatic planners. This architecture has been used for controlling a door opening system.

KAMARA

(<http://www.ipr.ira.uka.de>)

KAMARA [LAEN] is a MAS for controlling multi-robotic systems. The robot system consists of several subcomponents like two manipulators, hand-eye-cameras, one overhead-camera and a mobile platform. Extensions to the distributed control architecture KAMARA (KAMROs Multi Agent Robot Architecture) are developed to overcome co-ordination problems, for example caused by the independent task execution of both manipulator systems. The decentralised world representation can be used for parallel task execution. The described intelligent control architecture replaces the former control architecture of the autonomous robot KAMRO.

OAA and Saphira

(<http://www.ai.sri.com/~konolige/saphira/>)

Saphira is the software development environment delivered with the Pioneer mobile robot family. It is an integrated sensing and control system for robotics applications. The software runs a reactive planning system with a fuzzy controller, behaviour sequencer, and deliberative planner. There are integrated routines for sonar sensor interpretation, map building, and navigation. Saphira applications are structured as client/server and is written in C. Extension to other robotics systems is very hard.

Saphira has also been integrated as an agent within the Open Agent Architecture (OAA) [GUZZ].

TeamBots

(<http://www.teambots.org/>)

TeamBots is a Java-based collection of application programs and Java packages for multi-agent mobile robotics research. The TeamBots distribution is a full source-code release. The simulation environment is entirely written in Java. Execution on mobile robots sometimes requires low-level libraries in C, but Java

is used for all higher-level functions. At present, TeamBots will run on the Nomadic Technologies' Nomad 150 robot and (very soon) on Personal Robotics' Cye robot.

TeamBots supports prototyping, simulation and execution of multi-robot control systems. Robot control systems developed in TeamBots can run in simulation using the TBSim simulation application, and on mobile robots using the TBHard robot execution environment.

The TeamBots simulation environment is extremely flexible. It supports multiple heterogeneous robot hardware running heterogeneous control systems. Complex (or simple) experimental environments can be designed with walls, roads, opponent robots and circular obstacles. All of these objects may be included in a simulation by editing an easily understandable human-readable description file.

Because the software is written in Java, it is extremely portable. TeamBots runs under Windows9x, Linux, MacOS and any other operating environment supporting Java 1.2 or later.

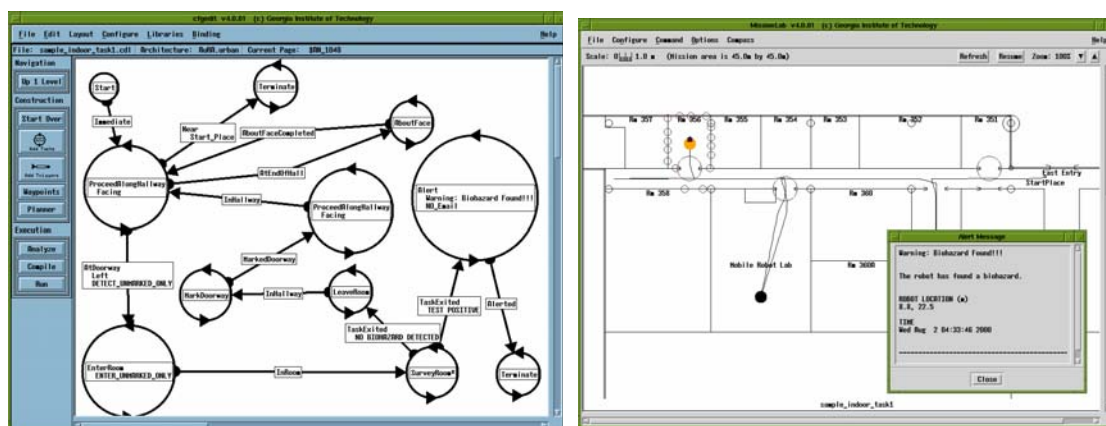
Concerns about Java: some researchers are concerned that Java is too slow to support real time robot control. Our experience contradicts this conclusion. As an example, in simulation (without graphics) our Java coded behaviour-based control systems run at up to 30 kHz rates on conventional 200 MHz Pentium machines. The primary bottleneck to runtime efficiency on real robots is sensor and control I/O. On Nomad 150 robots for instance, we are limited to 10Hz control rates because this is the maximum rate control commands can be transmitted to the robot (control programs written in C cannot run any faster). In our experience, the benefits of Java (correctness, ease of use, rapid development) far outweigh the negligible runtime overhead.

Rem. A native implementation is available for The Nomad150 (with sources).

MissionLab

(<http://www.cc.gatech.edu/ai/robot-lab/research/MissionLab/>)

MissionLab [MKEN] is developed by the Mobile Robot Laboratory at Georgia Tech. MissionLab takes high-level military-style plans and executes them with teams of real or simulated robotic vehicles. MissionLab supports execution of multiple robots in both simulation and actual robotics platforms, including device drivers for controlling ActivMedia Pioneer AT, RWI Urban Robot, and Nomadics Technologies Nomad 150 & 200. Each vehicle executes its portion of the mission using reactive control techniques developed at Georgia Tech.



Key components:

- mlab – console-like program from which a user monitors the progress of experimental runs of the robot executables.
- CfgEdit – graphical tool for building robot behaviours – the designer can build complex control structures with the point and click of a mouse.
- cdl – compiler that translates the configuration description language of the robot missions.
- cnl – compiler that compiles the configuration network language to generate a C++ code.
- HServer – hardware server that directly controls all the robot hardware and provides a standard interface for all the robots and sensors.

MissionLab runs exclusively on Linux.

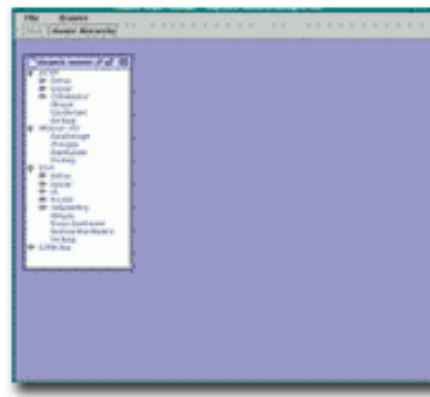
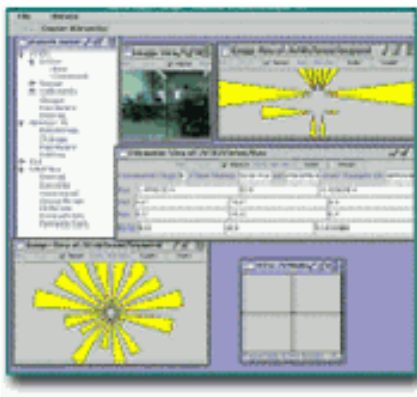
MOBILITY (Commercial Product)

(<http://www.rwi.com/>)

Introduced in October 1998, the MOBILITY™ Robot Software Development Environment Package is revolutionary new object-oriented, CORBA-based robotics control architecture. This breakthrough product offers robotics researchers a set of flexible, powerful, extensible “building blocks” and easy-to-use tools for construction of virtually any kind of robot control system. The MOBILITY™ architecture facilitates quick start-up and an unprecedented level of code re-use and transportability, allowing roboticists to get their research underway more easily and swiftly than ever before, on an extraordinarily robust and capable foundation. Easy-to-follow example programs get even novice researchers quickly up to speed.

MOBILITY™ is designed from the ground up to provide the best, most capable robotics development environment available. MOBILITY™ offers these unparalleled features:

- Extensibility over time.
- Full support for multiple robot systems.
- Easy integration among teams of researchers, even in different locations.
- Unprecedented level of code reusability.
- Parallel and distributed processing for robot control.
- Unprecedented level of robot independence.



With the Mobility Object Manager, or “MOM,” Mobility’s Java-based graphical user interface, you can easily observe, tune, configure and debug your entire Mobility™ robot control environment, as your programs are actually running. Launch programs, create objects, edit object properties, connect and configure objects and control which objects are active. Teleoperate your robot. Launch a variety of object viewers that give you a second-by-second visualisation of your robot’s actuators, sensors, algorithm outputs and debugging information - all from an easy-to use, central management point.

MOBILITY™ is specifically designed to address the most critical issue in robotics research: system integration. MOBILITY™ provides the growing robotics research and development community with the tools to swiftly and easily integrates their research ideas into working robot systems.

MOBILITY™ represents a powerful new paradigm for fast, reliable, efficient robotics development efforts. With MOBILITY™, iRobot underlines its continuing commitment to develop and provide low-cost, highly capable hardware and software building blocks and a common set of development tools and interfaces in support of advanced robotics research.

ControlShell (Commercial Product)

(<http://www.rti.com>)

ControlShell is a software-development framework designed for creating real-time intelligent-control applications developed by Real Time Innovations Inc. *ControlShell* is a *graphical* programming environment with a *component-based* approach that enables you to build complex real-time systems quickly. Perhaps even more importantly, *ControlShell* promotes *sharing* and *reuse* of code objects using integrated repository-management.

The following is a list of the *ControlShell* features:

- Object modelling Use graphical diagrams to describe both behavioural and functional system aspects for physical systems.
- Graphical editors build discrete event (behavioural) and sampled data (functional) systems.
- Hierarchical graphical composition Order objects from primitive and composite components.
- Interface centric design methodology Focus on identifying and managing the interfaces between objects.
- Repository management system Design with pre-built component repositories.
- Automatic C++ code generation Generate primitive components and types quickly.
- Runtime engines Execute periodic sampled data and discrete event driven processes.
- Runtime shell with scripting capability Interact with executing applications through an interpretative, menu driven shell.
- Live debugging link Connect graphical diagrams to the running application.
- Integration with RTI’s StethoScope Integrate *ControlShell* with a data-monitoring environment.

ControlShell is designed as a *programming system*. As the community develops more and richer sets of component repositories, you will be able to build complex systems without any custom coding.

Several goals have driven *ControlShell* design:

- Practical focus on real-time, intelligent-control systems.
- Rapid iterative-development cycle.

- Reusable software modules that do not require modification.
- Intuitive understanding of the final system.

ControlShell run-time executive provides the following facilities:

- Dynamic building of your application at initialisation time by directly loading your graphical diagrams.
- Sorting of execution order for components in the sampled-data portion of the application based on input-output (data-flow) dependencies.
- Creation of tasks for each sample-rate specified by the application.
- Creation of tasks for the finite-state machines specified by the application.
- Full command-line menus to start, stop, and view your application as it runs.
- A lookup service to allow you or your code to locate any component or signal in your application.

OROCOS (EU Funded Project)

(<http://www.orocos.net>)

Orocos is a modular, distributed and configurable soft and hard real-time software framework for advanced motion control. This framework separates the structure of the control (e.g., subdivision into threads, IPC between threads, event handling, distribution over networked computing nodes, required RTOS functionality, etc.) from its functionality (e.g., motion interpolation, control algorithms, sensor processing, etc.). Hence, it provides architecture-neutral components from which one can build motion control applications. The framework consists of: the functional components of planning, sensing, control and modelling, and the infrastructure components of event handling, name serving, time keeping, configuration, data logging and access control. The presented framework design does not cover just one single system architecture, nor does it provide one single software component. The framework is rather an application generating code base, i.e., a set of components from which the skeleton of any motion controls application can be constructed.

TELEOPERATION APPLICATIONS

CGAT

(<http://www.cse.dmu.ac.uk/~arg/tmmi/technology.html>)

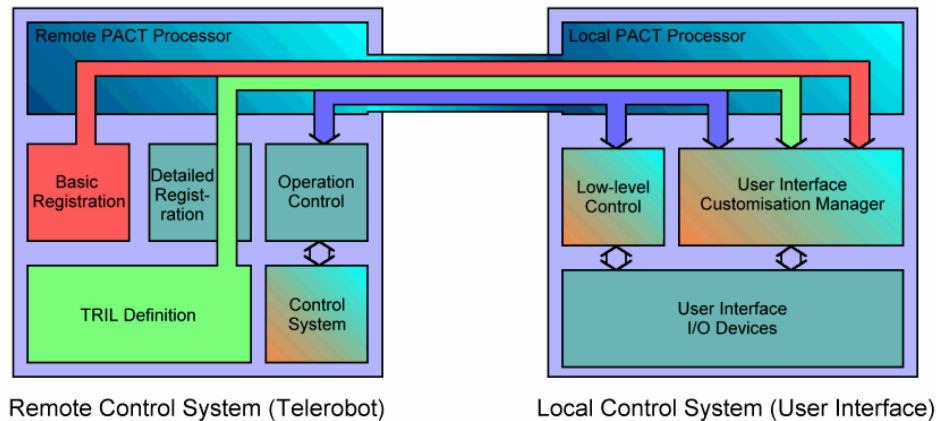
The aim of the Generic Control Architecture for Telerobotics (GCAT) [GRAV] is to realise the concept of *plug-and-play* telerobotics; the ability of a number of different types and configurations of telerobots to be controlled from a single adaptable operator interface.

GCAT consists of three main components:

- A standard model of the notion of a telerobot, which provides a common language and constraints for the architecture.
- Protocol for Abstract Control of Telerobots (PACT) – a protocol defining dynamic message exchange between a telerobot and operator interface at an abstract level (i.e. based upon the standard model, and not any specific robot type). This includes transmission of fundamental information about the structure of the robot.
- TeleRobot Interface Language (TRIL) – an interface definition language permitting a telerobot to provide very detailed and specific information about itself to an interface before operation begins.

The use of an expressive language such as this allows more sophisticated and interactive operator interfaces to be realised than with PACT messages alone.

Overview of the Generic Control Architecture for Telerobotics



Using the information supplied by the telerobot, the operator interface adapts itself to suit characteristics such as the robot’s physical size and shape, sensor data available, the structure of manipulators, etc.

Control System Structure

The control system is based around the principles of a popular AI approach known as the Subsumption Architecture (SA) as developed by Brooks.

This design approach has been widely adopted in the field of *autonomous* mobile robotics. However, in order to adapt the SA to telerobotic systems, a technique used by Arkin in his telerobotic systems was taken and adapted to suit the SA. This allowed a number of control modes from classical teleoperation theory to be implemented in a high performance semi-autonomous control system.

The control system is built around layers consisting of individual behavioural modules. Each module is interfaced with a number of sensors and actuators and performs a single simple task within the controller. The complete controller consists of the combination of a number of such modules within a priority network where some modules are allowed to suppress the output of others (e.g. collision avoidance can override movement behaviours). Sets of modules, which act together to perform identifiable competencies within the system, are grouped together into layers. A simplified version of such a network consisting of 8 behaviours grouped into 6 layers is shown below. The largest layer (made up of three modules) is the collision avoidance layer.

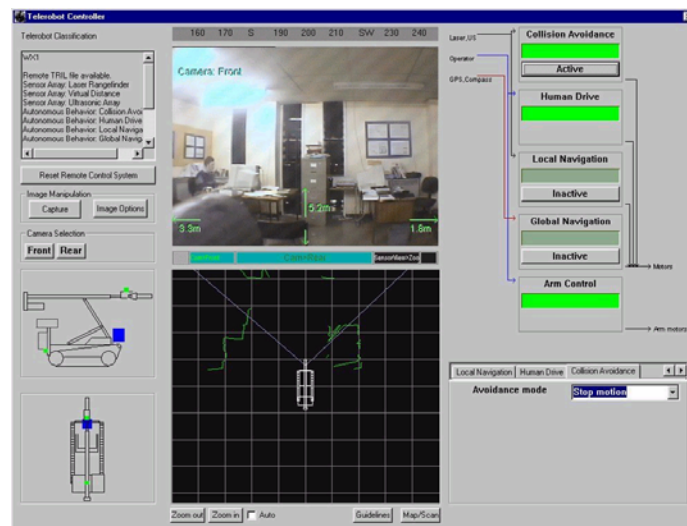
Operator control is represented in the network in two ways:

- High-level control consists of the operator’s ability to activate and deactivate any particular layer of the network at will (not shown explicitly on the diagram).
- Tightly integrated control is achieved by representing human commands within the system as modules themselves (“Operator Arm Control” and “Operator Drive Control”), such that their signals can be merged with those of autonomous modules.

By selecting different combinations of active layers, this control system allows the telerobot to be operated in a number of different modes:

- *Direct Control* – The most basic mode of operation allows the operator to deactivate all levels of autonomy and control the system without any intervention.
- *Collision Prevention* – The system can act in a safe mode by overriding any operator actions, which would cause a collision with an obstacle in the environment as detected by sensors.
- *Collision Avoidance* – In this mode the system is capable of making small course changes to prevent collisions or align the robot with walls whilst under operator guidance.
- *Local Navigation* – The system is capable of taking navigational instructions in relative terms (e.g. move 10 meters to the left) and guides itself based on information from the laser scanner and electronic compass.
- *Global Navigation* – The system can also take absolute navigational instructions and use the GPS sensor for guidance. As can be seen in the diagram (above) Local and Global navigation can be overridden by collision avoidance (“Stop” and “Steer” modules) when necessary.

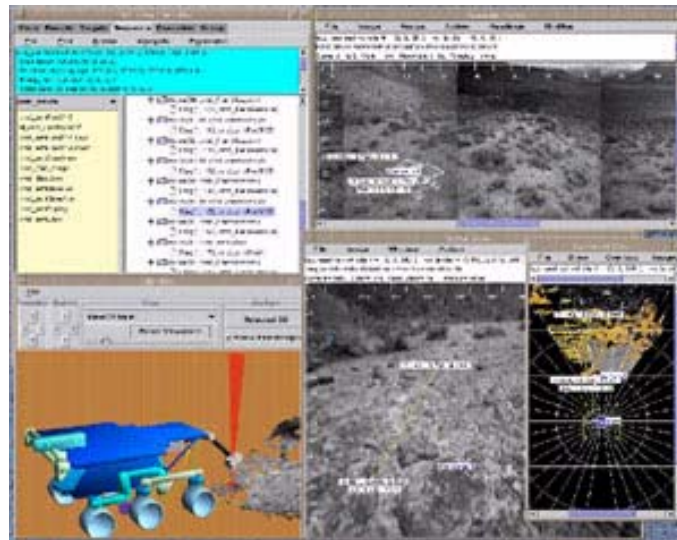
The operator interface to the system (also known as the Local Control System) is implemented on a Windows NT workstation fitted with a video capture card for input of the analogue video signals fed back from the telerobot.



WITS

(<http://robotics.jpl.nasa.gov/tasks/wits>)

WITS [BACK] has been developed to provide Internet-based distributed ground operations for planetary lander and rover missions. WITS provides an integrated environment for operations at a central location and collaboration with geographically distributed scientists.



The first version of WITS was using CGI scripts for client server communication and a VRML plug-in for 3D visualisation. The new version is entirely based on Java 2 tools like RMI, Java3D and Java Cryptography extensions. The client is run either as a Java applet using a web browser or an applet viewer, or as a Java application.

Telematics Applications

We present now an analysis of the relevant papers presented at the Telematics Applications in Automation and Robotics conference (TA2001).

List of considered papers (among more than 100 presented at the conference):

1	A system to facilitate telematic implementation	J. Ogness	117
2	Miro : Middleware for autonomous mobile robots	St. Enderle	149
3	A virtual operating systems for internet applications	P. Pavon	183
4	Migrating a corba-based tele-testing architecture to Internet	J. Garbajosa	189
5	REAL : A virtual laboratory for mobile robots experiments	E. Guimaraes	209
6	A Java based system for navigation and teleoperation of a mobile robot	P. Skrypczynski	215
7	A user interface for telecontrol of a robot over the internet	R. Behnke	221
8	A remote laboratory for teaching mobile robots	F. Rodriguez	307
9	Reality-driven visualisation of automation systems via the Internet based on Java and XML	G. Nusser	407
10	Multi-user environment for a teleoperated laboratory	H. Hoyer	437

Comparison Criteria

Robot : **Single** – **Many**. The first letter refers to the number of different robots the system can manage simultaneously while the second letter indicates how many different robots can be used with the system.

# Robots	Simultaneously	Sequentially
7	0	3

It means that on 7 applications, only 3 can cope with different robots (generally limited to 2 or 3 available robots). The possibility to extend the application to new robots and more particularly the UI is inherent to only one system (1).

Control : **Teleoperation – Algorithms**. Indicates if the system offers the possibility to choose or develop control algorithms or if it is restricted to pure teleoperation.

Method of control	Teleoperation	Algorithm
7	6	3

We observe that almost all applications allow the teleoperation of a single robot. One application just let the user visualise the robot. Three applications have navigation algorithm possibilities. One project allows the user to upload his own written navigation algorithms (5).

User Interface: Means **Proprietary** or **Browser** user interface. The M indicates if several users can connect to the system simultaneously.

UI	Web Browser	Proprietary	Multi-User (share)
10	8	2	all – 1

All applications (?) seem to let different users having access to the application simultaneously but generally, only one user can interact with the robot while the others has to wait and can only visualise what happens. Only (2) allows concurrent access to the robot.

Video: indicates the number of video images displayed in the interface.

Video	#
7	5

Only 5 applications offer video images.

3D: indicates the type of standard used.

3D	VRML	Java3D
8	1	1

3D models are used in one visualisation application (Java3D); one non-robotic application uses VRML. One manipulator teleoperation system proposes Augmented Reality video images.

C/S Architecture: refers to the communication architecture.

Client/Server Architecture	CORBA	RMI	Sockets	HTTP/CGI
9	5	1	3	–

We see clearly the Corba have the favour of the current developers.

No project uses the Microsoft DCOM.

JINI as an emerging technology is also absent.

Language: lists the programming languages used in the application.

Language	Java	C++	XML	Other
10	10	2	2	1

All applications make use of the famous Java language, some in combination with other languages. The recent XML is also present in 2 systems. Off course, all applications using a browser as main interface uses Java as its main language.

Comparison of the Different Applications

(CGAT and Wits have been added to this table – n°11 and 12 respectively)

#	# Rob/Sim	Control	UI	Video	3D	C/S Archi	Language
1	S – M	T	P – M	no	no	Sockets	Java
2	S – M	T – A	P – M	1	no	Corba (TAO)	C++ (Java)
3	–	–	B – M	–	–	Corba	Java
4	–	–	B – M	–	–	Corba	Java
5	S	T - A	B – M	2	no	Corba (Mico)	Java, C, C++, XML
6	S	T -A	B – M	1	no	Socket	Java
7	S (manip)	T	B – M	4	AR	?	Java
8	S	T	B – ?	no	no	Corba	Java
9	S – M	no (Vis)	B – ?	1	Java3D	RMI	Java, XML
10	–	–	B – M	–	VRML	Sockets	Java
11	S – M	T – A	P – 1	1	No (#2D)	Sockets	C++ (MS)
12	S – M	T – A	B – P	still imag.	J3D	RMI	Java

Conclusion

The general tendency for recent, current and future applications is oriented towards **Java** as programming language, **Corba** as the software middleware and **Web Browsers** as container for user interfaces. No generic tools are available for developing teleoperation applications.

RELATED TECHNOLOGIES

RoboML

Robotic Markup Language, or RoboML, is an XML-based language for data representation and interchange in robotic applications [MAKA]. Combined with a set of communication protocols, it is aimed at providing a common interface for hardware and software robotic agents communicating via Internet networks.

A complex robotic application utilises various languages independently designed for its diverse subsystems: robot programming, inter-agent communication, knowledge representation inside the embedded system and user interface components. In many areas of robotics, including home, entertainment, and industrial applications, it is beneficial to use Internet technologies for communication between hardware and software robotic agents. However, incompatibility between various inner representations as well as between protocols for agent-specific data interchange in robotics diminishes the advantages brought by using well-established standards of Internet networks. The need for a common language for data representation and interchange between various subsystems (agents) of a robotic application is the motivation for development of RobML.

ACE-TAO

(<http://www.cs.wustl.edu/~schmidt/ACE.html>)

The ADAPTIVE Communication Environment (ACE) is a freely available, open-source object-oriented (OO) framework that implements many core patterns for concurrent communication software [SCHM]. ACE provides a rich set of reusable C++ wrapper facades and framework components that perform common communication software tasks across a range of OS platforms. The communication software tasks provided by ACE include event de-multiplexing and event handler dispatching, signal handling, service initialisation, interprocess communication, shared memory management, message routing, dynamic (re) configuration of distributed services, concurrent execution and synchronisation.

ACE is targeted for developers of high-performance and real-time communication services and applications. It simplifies the development of OO network applications and services that utilise interprocess communication, event de-multiplexing, explicit dynamic linking, and concurrency. In addition, ACE automates system configuration and reconfiguration by dynamically linking services into applications at run-time and executing these services in one or more processes or threads.

TAO is a real-time implementation of CORBA built using the framework components and patterns provided by ACE. TAO contains the network interface, OS, communication protocol, and CORBA middleware components and features. TAO is based on the standard OMG CORBA reference model, with the enhancements designed to overcome the shortcomings of conventional ORBs for high-performance and real-time applications. TAO, like ACE, is freely available, open source software.

The Miro (see Telematics Application section) relies on ACE-TAO for its basic communication and synchronisation needs.

CONCLUSION

This report is far from being exhaustive but gives a good overview of current development in autonomous and teleoperation robotic control. It appears that the research community has accomplished much valuable work but as in many research projects, there is a lack of exploitation and availability. Sometimes organisations do not want to share their results or do not have the resources to support past developments.

From the review, only one framework is publicly available, Teambots from CMU. Teambots is completely written in Java.

The second available framework is ACE, which is used for developing high-performance distributed applications. This architecture has been used to develop many applications.

REFERENCES

- [MUSC] Control and Software Requirements for CLAWAR machines, G. Muscatto & et al., Clawar Year 4 Report Technical Task 18, 2002.
- [VBRU] Computer Controlled Motion and Robotics, Appendix A: Motion Control of Free-Navigation AGVs, H. Van Brussel, J. De Schutter & K.T. Song, Lecture notes of the integrated european course in mechatronics, KUL, 1991.
- [VDHE] UGV Testbed for Autonomous Navigation in Unstructured Terrain, J. Van Den Heuvel, UVS Tech 2001, RMA-Brussels.
- [LUZE] Autonomy and Its Control for Military UGVs, D. Luzeaux, A. Dalgarrondo, D. Dufour, UVS Tech 2001, RMA-Brussels.
- [MOOR] A six-wheeled omnidirectional autonomous mobile robot, K. Moore & N. Flann, IEEE Control Systems Magazine, December 2000.
- [SAFF] A multivalued logic approach to integrating planning and control, A. Saffiotti, K. Konolige & E. Ruspini, Artificial Intelligence n° 76 – 1995.
- [COLO] Development and evaluation of distributed control algorithms for the mobile robot Nomad200, E. Colon & Y. Baudoin, Proc. of Mobile robot and automated vehicle control systems, SPIE's Photonics East -1996, Boston.
- [VOLP] The CLARAty Architecture for Robotic Autonomy, R. Volpe, I.A.D. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das Proceedings of the 2001 IEEE Aerospace Conference, Big Sky Montana, March 10-17 2001.
- [MKEN] Multi Agent Specification and Execution, D. MacKenzie, R.C. Arkin, Autonomous robots, 1-25, Kluwer Academic Publisher, Boston.
- [LAEN] A Distributed Control Architecture for Autonomous Mobile Robots: Implementation of the Karlsruhe Multi-Agent Robot Architecture KAMARA, Thomas Laengle, Tim C. Lueth, Ulrich Rembold, Heinz Woern, Advanced Robotics, International Journal of the Robotics Society of Japan, Vol. 4, No. 12.
- [GRAV] A Generic Control Architecture for Telerobotics, Graves, A.R., & Czarnecki, C.A., in Proceedings of Towards Intelligent Mobile Robots '99, U Nehmzow and C Melhuish (eds.),

Bristol, UK, 26th March 1999, University of Manchester, Department of Computer Science, Technical Report Series, Report No. UMCS-99-3-1.

- [GUZZ] Many Robots Make Short Work, Guzzoni, D., Cheyer, A., Julia, L., and Konolige, K., in *AI Magazine*, **18**(1), pp. 55-64 (Spring 1997).
- [PETE] DCA: A distributed Control Architecture for Robotics, L. Peterson, D. Austin, H. Christensen, IROS 2001.
- [BACK] Internet-based operations for the Mars polar lander mission, P. G. Back & all, Proc. of the 2000 IEEE ICRA, San Francisco, California, April 2000.
- [MAKA] Computational Issues on Design and Implementation of an Autonomous Guided Vehicle, Maxim Makatchev, Ph.D. Thesis, City University of Hong Kong.
- [SCHM] The ADAPTIVE Communication Environment: Object-Oriented Network Programming Components for Developing Client/Server Applications, D.C. Schmidt, 11th and 12th Sun Users Group Conference, December 1993 and June 1994.

Application Name	Lab	Contact	Address
CLARAty	NASA JPL	Issa Nesnas	http://robotics.jpl.nasa.gov/tasks/claraty/
CAMPOUT	NASA JPL	Paul Schenker	http://prl.jpl.nasa.gov/projects/ate/technology/campout.html
DCA	RIT Stockholm	L. Peterson	http://www.nada.kth.se
KAMARA	IPR Karlsruhe	T. Laengle	http://www.ipr.ira.uka.de
OAA and Saphira	SRI	D. Guzzoni	http://www.ai.sri.com/~konolige/saphira/
TeamBots	Carnegie Mellon	Tucker Balch	http://www.teambots.org/
MissionLab	Georgia Tech	Ronald Arkin	http://www.cc.gatech.edu/ai/robot-lab/research/MissionLab/
MOBILITY	iRobot		http://www.rwi.com/
ControlShell	RIT		http://www.rti.com
Orocos	Euron	H. Bruyninck	http://www.orocos.net
WITS	NASA JPL	Paul Backes	http://robotics.jpl.nasa.gov/tasks/wits
CGAT	DeMontfort	Alan Graves	http://www.cse.dmu.ac.uk/~arg/tmmi/technology.html



Annex G – SURVEY OF SOFTWARE FRAMEWORKS FOR ROBOTICS APPLICATIONS

Technical Report

Eric Colon

Robotics Laboratory
Royal Military Academy

January 2002

INTRODUCTION

Robotic systems are inherently multi-disciplinary and for such applications software aspects are of prime importance. Real projects necessarily involve several robots from the same or different model. Moreover, robotics applications imply the use of external hardware and sensors each having their control system, what generally lead us to a distributed architecture.

It means that somewhere, we need some kind of communication between the different pieces. Since the beginning of robotic research all efforts have been directed to well focus aspects of robotic (navigation and path planning, perception, etc). Algorithms have been written in different languages, on different hardware and operating systems, using different communication protocols. Because robotic was in its infancy, most of the researchers only focused on their problems and the implementation of the algorithm generally targeted a given robotic system (mobile platform with given hardware, operating system and library). For example when writing distributed applications, many researchers rewrote client/server scheme defining their own protocol for negotiation and data exchange.

We can state that we now have at our disposal many excellent algorithms but that we lack some kind of standard to easily reuse those bricks into new applications. Existing programs have to be modified, translated, ported, ... or simply (!) completely rewritten from scratch. If we look what happen in software engineering, we observe that the last years have seen the emergence of new software techniques. Object-oriented languages and components have greatly improved the software reusability. Everything being naturally pushed by the excessive extension of the Internet. What is needed is a software architecture that enables agile, flexible, dynamic composition of resources and permits their use in a variety styles to match present and changing computing needs.

Since a couple of years, some researchers have begun to work in this way. The actual tendency is to use Internet standard to implement the communication aspects of robotic applications.

As other labs, we are facing the challenge described above. We have developed many interesting applications involving mobile robots, vision system, tracking and location system, 3D modelling that cannot be integrated. One could state that it is due to a lack of organisation or vision. Those applications have been developed with the techniques and the knowledge available at this time. Ten years ago, no standard was available for distributed application development. There was no universal language as Java, the Internet was only known by a couple of specialists and Linux had less than 1 year.

The list of the different computer elements we have used and that we would like to integrate are:

- Different OS: Unix, Linux, Windows;
- Different programming languages: C, C++, Assembler, Java; and

- Hardware and device drivers and API: some are only available under Windows, API are generally written in C or C++.

The different components of these applications run on different computers or micro-controllers and communicate through different media: Ethernet (wire or radio), Serial (RS232, ...), CAN, ...

The question is how to link and integrate such different parts into a unified framework?

We pretend that it is greatly time now to pay attention to this aspect. Robotics is a fascinating discipline, and it is sometime an art, but if we want to keep our credibility, we must make use of all the available tools to improve our systems and to reduce their development time.

What do we need?

We need a general framework, some kind of standard, to let different systems communicate and collaborate. We do not think we need to re-invent the wheel once again and to propose “the architecture” that will solve all the problems but that will be adopted by nobody. No, we think we must chose existing techniques, proven or promising ones, and to build our future on it.

REQUIREMENTS

It is obvious that it will not be possible to have a world wide agreement on the kind of hardware, OS and language that should be use for building robotic systems. Every lab wants to keep its know-how, material and freedom. Therefore, what we need is a kind of glue to link the different bricks together. We do not either want to throw away the existing (and expensive) systems we have in our labs. Nevertheless, everybody agrees that it should be convenient to pick some navigation algorithm from University X and combine it with the perception one from Lab Y and to integrate it within minutes in an existing robotic system.

We do not argue to promote a universal system. We target those applications that require the communication and the collaboration between homogenous or heterogeneous systems. It should allow the easy deployment of network robotics application. It should facilitate the integration of different components allowing the developer to pay attention to the benefit.

Which kind of applications are we targeting?

Command and control applications including tele monitoring, tele-operation (including shared, traded and supervised control), collaboration (between users), for single and multi robots (of the same or different models).

The framework chosen should allow the following possibilities:

- Computer Hardware independence;
- OS independence;
- Language portability;
- Link to legacy systems;
- Provide an universal viewer;
- Open to all robots;
- Allow the remote user to use its own algorithms; and
- Easy reuse of software components.

The User Interface should allow the display of

- Video,
- Audio,
- Text,
- 2D vector graphics,
- Bitmap graphics,
- 3D

and to be connectable to haptic interface.

The GUI should be adaptive, it should adapt automatically to the system capabilities. For instance, if we add a camera component to the application, the GUI must adapt itself to display the images.

To build such a framework, different conceptual implementation can be considered: from a direct hard-coded network to a true open Multi-Agent System (MAS). In the next sections, we successively review the SOTA in the following disciplines: Distributed computing, Artificial Intelligence (Multi-Agent Systems) and finally we present some results from the on-going tele-robotics and tele-learning developments.

DISTRIBUTED COMPUTING SYSTEMS

We begin this section with a short summary of the programming languages evolution. We present afterwards the parallel evolution of software technologies used for writing network applications.

Evolution of Programming Languages

Over the last 50 years, programming languages have evolved from binary machine code to powerful tools that create complex abstractions. They have become more and more powerful and easier for programmers to use and learn. While several languages have enjoyed their days in the spotlight, history has proven that no language can stay on top forever.

First, programming languages become increasingly like spoken language. This makes learning the syntax of the language easier and improves comprehension.

Second, reuse of code becomes easier. Modular programming in any language reduces duplicate code. With, object-oriented languages, reuse is gained by inheritance and extreme modularity. With component-based languages, reuse is accomplished through the acquisition of prewritten components.

Next, because code reuse is easier, programming languages generally include a larger code base as they evolve from one language to the next. The code base of the language allows programmer to do common programming tasks, using existing code. Programming libraries, such as the Microsoft Foundation Classes for C++, or Java's Abstract Windowing Toolkit, make frequent coding tasks easier and faster.

The fourth trend in the progression of programming languages is a reduction in the number of lines of code required to accomplish the same task. If you reduce the number of statements or lines of code that are required to accomplish a particular task, programmers become more efficient. Reducing the number of lines of code is accomplished by code reuse, code libraries, syntax simplification.

New programming languages have fewer rules for programmer to remember and extend those rules to larger and larger concepts. For example, the move from C++ to Java took memory allocation and de-allocation out

of the hand of the programmer. This allows the programmer to focus more on the business need than of the reclamation of memory. In addition, the syntax includes fewer “reserved words”, as we evolve from one language to the next.

Finally, the evolution of languages has noted the performance increases of the computer itself, and consequently worried less and less about the performance of the language. Newer languages have heavier code bases, and produce larger footprints, but run just as fast, because of innovations in computing hardware.

[M Kamath www.goodstart.com]

Component-Based Development

Component-based development (CBD) is the building of software systems out of pre-packaged generic elements. The current popularity of CBD results from the convergence of four phenomena originating from quite different backgrounds:

- The progress of modern software engineering that emphasizes reuse;
- The widespread success of useful techniques for building GUI, databases, ... out of components (VBX, ActiveX, OCX, ...);
- The push for interconnection technologies: CORBA, DCOM, Enterprise JavaBeans; and
- The generalization of objects technology in the software world at large.

Recent discussion has focussed on binary components made possible by COM and Corba, but there is no reason to take a restrictive view of components. Object-oriented libraries and frameworks provide components in their own right. Furthermore, object technology provides the only serious known technical basis to build binary components.

[Computer Magazine July 1999]

COM (Component Object Model)

COM refers to both a specification and implementation developed by Microsoft Corporation, which provides a framework for integrating components. This framework supports *interoperability* and *reusability* of distributed objects by allowing developers to build systems by assembling reusable components from different vendors, which communicate, via COM. By applying COM to build systems of pre-existing components, developers hope to reap benefits of *maintainability* and *adaptability*.

COM defines an application-programming interface (API) to allow for the creation of components for use in integrating custom applications or to allow diverse components to interact. However, in order to interact, components must adhere to a binary structure specified by Microsoft. As long as components adhere to this binary structure, components written in different languages can inter-operate.

More information is provided in the section devoted to Distributed Computing where DCOM, which is the distributed extension of COM, is described.

JavaBeans

The JavaBeans component architecture is the platform-neutral architecture for the Java application environment. It is the ideal choice for developing or assembling network-aware solutions for heterogeneous hardware and operating system environments, within the enterprise or across the Internet. In fact, it is the only component architecture one should consider when developing for the Java platform.

The JavaBeans component architecture extends “Write Once, Run Anywhere” capability to reusable component development. In fact, the JavaBeans architecture takes interoperability a major step forward: your code runs on every OS and within any application environment. A beans developer secures a future in the emerging network software market without losing customers that use proprietary platforms, because JavaBeans components interoperate with ActiveX. JavaBeans architecture connects via bridges into other component models such as ActiveX. Software components that use JavaBeans APIs are thus portable to containers including Internet Explorer, Visual Basic, Microsoft Word, Lotus Notes, and others.

The JavaBeans specification defines a set of standard component software APIs for the Java platform. The specification was developed by Sun with a number of leading industry partners and was then refined based on broad general input from developers, customers, and end-users during a public review period.

The JavaBeans API makes it possible to write component software in the Java programming language. Components are self-contained, reusable software units that can be visually composed into composite components, applets, applications, and servlets using visual application builder tools. JavaBean components are known as *Beans*.

Components expose their features (for example, public methods and events) to builder tools for visual manipulation. A Bean’s features are exposed because feature names adhere to specific *design patterns*. A “JavaBeans-enabled” builder tool can then examine the Bean’s patterns, discern its features, and expose those features for visual manipulation. A builder tool maintains Beans in a palette or toolbox. You can select a Bean from the toolbox, drop it into a form, modify its appearance and behaviour, define its interaction with other Beans, and compose it and other Beans into an applet, application, or new Bean. All this can be done without writing a line of code. The following list briefly describes key Bean concepts:

- Builder tools discover a Bean’s features (that is, its properties, methods, and events) by a process known as *introspection*.
- *Properties* are a Bean’s appearance and behaviour characteristics that can be changed at design time. Builder tools introspect on a Bean to discover its properties, and expose those properties for manipulation.
- Beans expose properties so they can be *customized* at design time. Customisation is supported in two ways: By using property editors, or by using more sophisticated Bean customizers.
- Beans use *events* to communicate with other Beans. A Bean that wants to receive events (a listener Bean) registers its interest with the Bean that fires the event (a source Bean). Builder tools can examine a Bean and determine which events that Bean can fire (send) and which it can handle (receive).
- *Persistence* enables Beans to save and restore their state. Once you have changed Bean’s properties, you can save the state of the Bean and restore that Bean later, property changes intact. JavaBeans uses Java Object Serialization to support persistence.
- A Bean’s *methods* are no different from Java methods, and can be called from other Beans or a scripting environment. By default, all public methods are exported.

Enterprise JavaBeans (EJB) technology defines a model for the development and deployment of reusable Java server components. The EJB architecture logically extends the JavaBeans component model to support server components. Server components are application components that run in an application server. EJB technology is part of Sun’s Enterprise Java platform, a robust Java technology environment that can support the rigorous demands of large-scale, distributed, mission-critical application systems. EJB technology supports application development based on a multitier, distributed object architecture in which most of an application’s logic is moved from the client to the server. The application logic is partitioned into one or more business objects that are deployed in an application server.

The Enterprise JavaBeans architecture defines a standard model for Java application servers to support “Write Once, Run Anywhere” (WORA) portability. WORA is one of the primary tenets of Java technology. The Java virtual machine (JVM) allows a Java application to run on any operating system. However, server components require additional services that are not supplied directly by the JVM. These services are supplied either by an application server or by a distributed object infrastructure, such as CORBA or DCOM. Traditionally, each application server supplied a set of proprietary programming interfaces to access these services, and server components have not been portable from one application server to another. For example, a server component designed to run in BEA Tuxedo could not execute in IBM TXSeries without significant modification. The EJB server component model defines a set of standard vendor-independent interfaces for all Java application servers.

Enterprise JavaBeans technology takes the WORA concept to a new level. Not only can these components run on any platform, but also they are also completely portable across any vendor’s EJB-compliant application server. The EJB environment automatically maps the component to the underlying vendor-specific infrastructure services.

[see <http://java.sun.com>]

Distributed Computing

Distributed computing models permit components to be spread across multiple computers to yield the benefits of increased scalability, better performance and varying degrees of component reuse. Two areas of distinction among these models, which highlight their adaptability and extensibility, are whether interactions among components are pre-configured (hard-wired) and where control for using the component or service lies (e.g., requester/client, provider/server, mediator).

Distributed applications rely on network communications. Different techniques can be use to deploy such an application:

- Sockets;
- Middleware (DCOM, Corba, RPC, Java-RMI); and
- Intelligent networks (Jini,...).

Sockets through Native Software API

Native API for writing communication software is available on all platforms. Standard communication protocols can be sued depending on the application purposes. Those API are written in C or C++ and offer different abstraction levels in function of the API. (Java, Tel, Perl,...).

The main disadvantage is that the software implementation is different. Different names are used for functions with different parameters, initialisation and names. This obliges the programmer to learn different libraries in order to write hybrid network applications. This is acceptable for full-time programmers but not for roboticists or trainees who do not have that much time to devote to the writing of such applications.

This option naturally offers the big advantage that the programmer can tailor the code and write very efficient and small footprint code. Nevertheless, in our opinion it is preferable to write intelligent high-level applications that perform real tasks than to spend most of the time for writing low-level code.

Middleware

Middleware is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middleware is essential to

migrating mainframe applications to client/server applications and to providing for communication across heterogeneous platforms. This technology has evolved during the 1990s to provide for interoperability in support of the move to client/server architectures. The most widely publicized middleware initiatives are the Open Software Foundation's Distributed Computing Environment (DCE), Object Management Group's Common Object Request Broker Architecture (CORBA), and Microsoft's COM/DCOM.

As outlined in Figure 1, middleware services are sets of distributed software that exist between the application and the operating system and network services on a system node in the network.

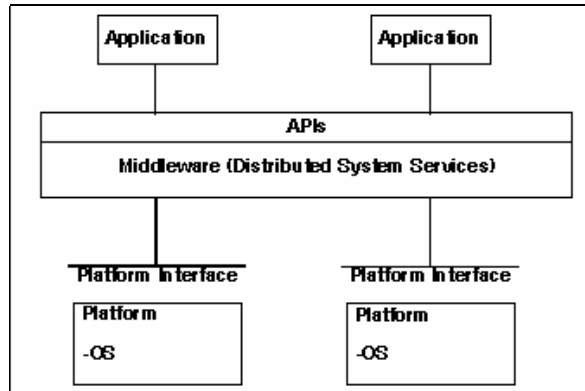


Figure 1: Use of Middleware.

Middleware services provide a more functional set of Application Programming Interfaces (API) than the operating system and network services to allow an application to:

- Locate transparently across the network, providing interaction with another application or service.
- Be independent from network services.
- Be reliable and available.
- Scale up in capacity without losing function.

Middleware can take on the following different forms:

- Transaction processing (TP) monitors that provide tools and an environment for developing and deploying distributed applications.
- Remote Procedure Call (RPCs), which enable the logic of an application to be distributed across the network. Program logic on remote systems can be executed as simply as calling a local routine.
- Message-Oriented Middleware (MOM), which provides program-to-program data exchange, enabling the creation of distributed applications. MOM is analogous to email in the sense it is asynchronous and requires the recipients of messages to interpret their meaning and to take appropriate action.
- Object Request Brokers (ORBs), which enable the objects that comprise an application to be distributed and shared across heterogeneous networks.

Usage Considerations

The main purpose of middleware services is to help solve many application connectivity and interoperability problems. However, middleware services are not a panacea:

- There is a gap between principles and practice. Many popular middleware services use proprietary implementations (making applications dependent on a single vendor's product).
- The sheer number of middleware services is a barrier to using them. To keep their computing environment manageably simple, developers have to select a small number of services that meet their needs for functionality and platform coverage.
- While middleware services raise the level of abstraction of programming distributed applications, they still leave the application developer with hard design choices. For example, the developer must still decide what functionality to put on the client and server sides of a distributed application.

The key to overcoming these three problems is to fully understand both the application problem and the value of middleware services that can enable the distributed application. To determine the types of middleware services required, the developer must identify the functions required, which fall into one of three classes:

- Distributed system services, which include critical communications, program-to-program, and data management services. This type of service includes RPCs, MOMs and ORBs.
- Application enabling services, which give applications access to distributed services and the underlying network. This type of services includes transaction and database services such as Structured Query Language (SQL).
- Middleware management services, which enable applications and system functions to be continuously monitored to ensure optimum performance of the distributed environment.

In this text, we will limit our review to ORBs.

Object Request Brokers (ORBs)

Among all possibilities presented above, ORBs are the most popular middleware. CORBA, DCOM and Java-RMI are all based on this principle.

An object request broker (ORB) is a middleware technology that manages communication and data exchange between objects. ORBs promote *interoperability* of distributed object systems because they enable users to build systems by piecing together objects from different vendors that communicate with each other via the ORB. The implementation details of the ORB are generally not important to developers building distributed systems. They are only concerned with the object interface details. This form of information hiding enhances system *maintainability* since the object communication details are hidden from the developers and isolated in the ORB.

Technical Detail

ORB technology promotes the goal of object communication across machine, software, and vendor boundaries. The relevant functions of an ORB technology are:

- Interface definition;
- Location and possible activation of remote objects; and
- Communication between clients and object.

An object request broker acts as a kind of telephone exchange. It provides a directory of services and helps establish connections between clients and these services. Figure 2 illustrates some of the key ideas.

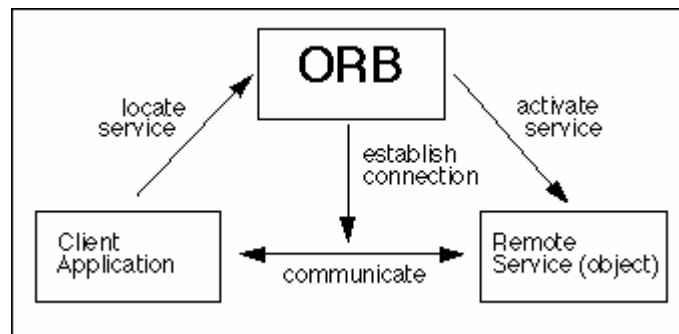


Figure 2: Object Request Broker Functions.

The ORB must support many functions in order to operate consistently and effectively, but many of these functions are hidden from the user of the ORB. It is the responsibility of the ORB to provide the illusion of locality, in other words, to make it appear as if the object is local to the client, while in reality it may reside in a different process or machine. Thus the ORB provides a framework for cross-system communication between objects. This is the first technical step toward interoperability of object systems.

The next technical step toward object system interoperability is the communication of objects across platforms. An ORB allows objects to hide their implementation details from clients. This can include programming language, operating system, host hardware, and object location. Each of these can be thought of as a “transparency,” and different ORB technologies may choose to support different transparencies, thus extending the benefits of object orientation across platforms and communication channels.

There are many ways of implementing the basic ORB concept; for example, ORB functions can be compiled into clients, can be separate processes, or can be part of an operating system kernel. These basic design decisions might be fixed in a single product; or there might be a range of choices left to the ORB implementer.

There are two major ORB technologies:

- The Object Management Group’s (OMG) Common Object Request Broker Architecture (CORBA) specification.
- Microsoft’s Component Object Model.

An additional, newly emerging ORB model is Remote Method Invocation (RMI); this is specified as part of the Java language/virtual machine. RMI allows Java objects to be executed remotely. This provides ORB-like capabilities as a native extension of Java. Another emerging middleware is Jini, which is based on Java, and RMI but provides additional facilities for implementing distributed applications.

Usage Considerations

Successful adoption of ORB technology requires a careful analysis of the current and future software architectural needs of the target application and analysis of how a particular ORB will satisfy those needs. Among the many things to consider are platform availability, support for various programming languages, as well as implementation choices and product performance parameters. After performing this analysis, developers can make informed decisions in choosing the ORB best suited for their application’s needs.

Remark

Microsoft DCOM and Java-RMI are APIs while CORBA is only a specification. Various implementations of the CORBA are available as free or commercial products (Examples include ORBIX by IONA Technology,

NEO by SunSoft, VisiBroker by VisiGenic, PowerBroker by Expersoft, SmallTalkBroker by DNS Technologies, Object Director by Fujitsu, DSOM by IBM, DAIS by ICL, SORBET by Siemens Nixdorf, NonStop DOM by Tandem, Java Idl and RMI-IIOP).

DCOM

Distributed COM is an extension to COM that allows network-based component interaction. While COM processes can run on the same machine but in different address spaces, the DCOM extension allows processes to be spread across a network. With DCOM, components operating on a variety of platforms can interact, as long as DCOM is available within the environment.

It is best to consider COM and DCOM as a single technology that provides a range of services for component interaction, from services promoting component integration on a single platform, to component interaction across heterogeneous networks. In fact, COM and its DCOM extensions are merged into a single runtime. This single runtime provides both local and remote access.

While COM and DCOM represent “low-level” technology that allows components to interact, OLE (Object Linking and Embedding), ActiveX and MTS (Microsoft Transaction Services) represent higher-level application services that are built on top of COM and DCOM. OLE builds on COM to provide services such as object “linking” and “embedding” that are used in the creation of compound documents (documents generated from multiple tool sources). ActiveX extends the basic capabilities to allow components to be embedded in Web sites. MTS expands COM capabilities with enterprise services such as transaction and security to allow Enterprise Information Systems (EIS) to be built using COM components.

COM+ is the evolution of COM that integrates MTS services and message queuing into COM, and makes COM programming easier through a closer integration with Microsoft languages as Visual Basic, Visual C++, and J++. COM+ will not only add MTS-like quality of service into every COM+ object, but it will hide some of the complexities in COM coding.

The distinctions among various Microsoft technologies and products are sometimes blurred. Thus, one might read about “OLE technologies” which encompass COM, or “Active Platform” as a full web solution.

The most important usage considerations are:

- *Platform support.* COM and DCOM are best supported on Windows platforms. However, Microsoft has released a version of COM/DCOM for MacOS that supports OLE-style compound documents and the creation of ActiveX controls. Software AG, a Microsoft partner, has released DCOM for some UNIX operating. However, DCOM over non-Windows platforms has few supporters. Until DCOM for alternate platforms has solidified, the technology is best applied in environments that are primarily Windows-based.
- *Platform specificity of COM/DCOM components.* Because COM and DCOM are based on a native binary format, components written to these specifications are not platform independent. Thus, either they must be recompiled for a specific platform, or an interpreter for the binary format must become available. Depending on your perspective, the use of a binary format may be either an advantage (faster execution, better use of native platform capabilities) or a disadvantage (ActiveX controls, unlike Java applets, are NOT machine independent).
- *Security.* Because COM/DCOM components have access to a version of the Microsoft Windows API, “bad actors” can potentially damage the user’s computing environment. In order to address this problem, Microsoft employs “Authenticode” which uses public key encryption to digitally sign components. Independent certification authorities such as VeriSign issue digital certificates to verify the identity of the source of the component. However, even certified code can contain instructions that accidentally, or even maliciously, compromise the user’s environment.

- *Support for distributed objects.* COM/DCOM provides basic support for distributed objects. There is currently no support for situations requiring real time processing, high reliability, or other such specialized component interaction.

More technical information can be found at [<http://www.sei.cmu.edu/str/descriptions/com.html>]

CORBA

The Common Object Request Broker Architecture (CORBA), is the Object Management Group's answer to the need for interoperability among the rapidly proliferating number of hardware and software products available today. Simply stated, CORBA allows applications to communicate with one another no matter where they are located or who has designed them. CORBA 1.1 was introduced in 1991 by Object Management Group (OMG) and defined the Interface Definition Language (IDL) and the Application Programming Interfaces (API) that enable client/server object interaction within a specific implementation of an Object Request Broker (ORB). CORBA 2.0, adopted in December of 1994, defines true interoperability by specifying how ORBs from different vendors can interoperate.

The (ORB) is the middleware that establishes the client-server relationships between objects. Using an ORB, a client can transparently invoke a method on a server object, which can be on the same machine or across a network. The ORB intercepts the call and is responsible for finding an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of an object's interface. In so doing, the ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

In fielding typical client/server applications, developers use their own design or a recognized standard to define the protocol to be used between the devices. Protocol definition depends on the implementation language, network transport and a dozen other factors. ORBs simplify this process. With an ORB, the protocol is defined through the application interfaces via a single implementation language-independent specification, the IDL. In addition, ORBs provide flexibility. They let programmers choose the most appropriate operating system, execution environment and even programming language to use for each component of a system under construction. More importantly, they allow the integration of existing components. In an ORB-based solution, developers simply model the legacy component using the same IDL they use for creating new objects, then write "wrapper" code that translates between the standardized bus and the legacy interfaces.

CORBA is a single step on the road to object-oriented standardization and interoperability. With CORBA, users gain access to information transparently, without them having to know what software or hardware platform, it resides on or where it is located on an enterprises' network. The communications heart of object-oriented systems, CORBA brings true interoperability to today's computing environment.

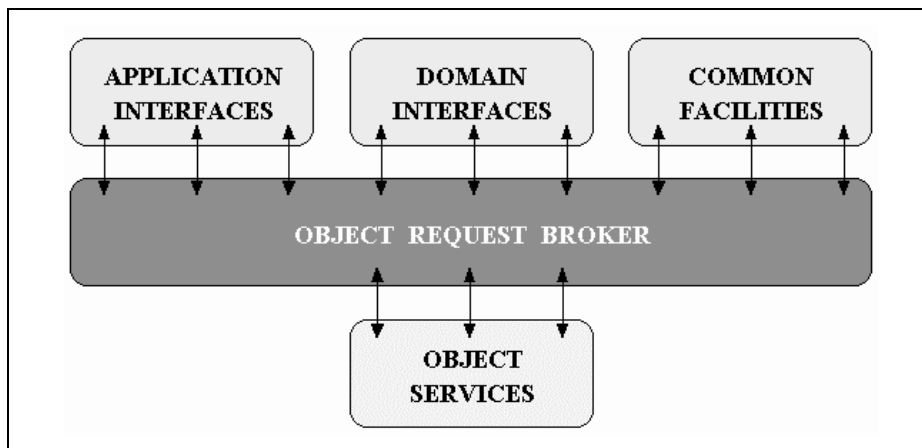


Figure 3: OMG Reference Model Architecture.

The following figure illustrates the primary components in the CORBA ORB architecture. Descriptions of these components are available below the figure.

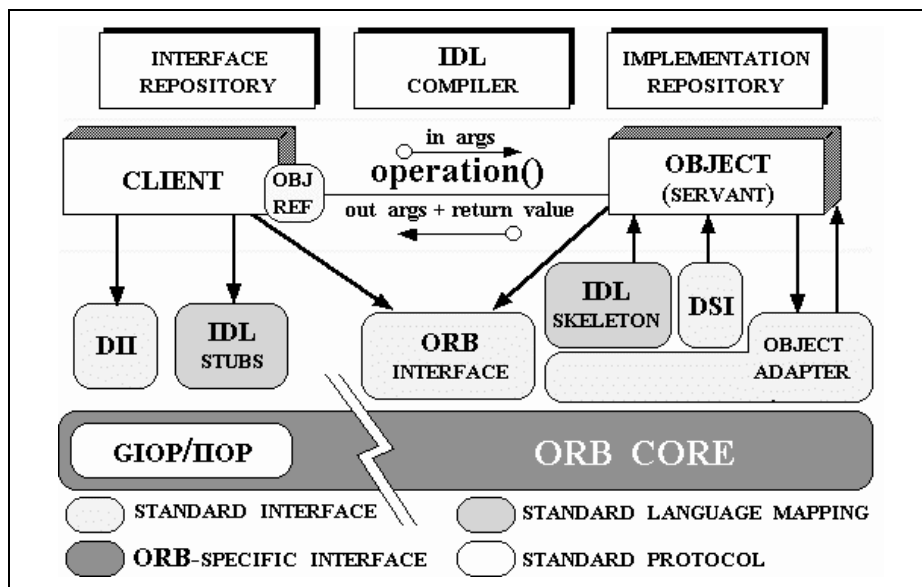


Figure 4: CORBA ORB Architecture.

- **Object** – This CORBA programming entity that consists of an *identity*, an *interface*, and an *implementation*, which is known as a *Servant*.
- **Servant** – This implementation programming language entity defines the operations that support a CORBA IDL interface. Servants can be written in a variety of languages, including C, C++, Java, Smalltalk, and Ada.
- **Client** – This program entity invokes an operation on an object implementation. Accessing the services of a remote object should be transparent to the caller. Ideally, it should be as simple as calling a method on an object, i.e., `obj->op(args)`. The remaining components in Figure 2 help to support this level of transparency.
- **Object Request Broker (ORB)** – The ORB provides a mechanism for transparently communicating client requests to target object implementations. The ORB simplifies distributed

programming by decoupling the client from the details of the method invocations. This makes client requests appear to be local procedure calls. When a client invokes an operation, the ORB is responsible for finding the object implementation, transparently activating it if necessary, delivering the request to the object, and returning any response to the caller.

- **ORB Interface** – An ORB is a logical entity that may be implemented in various ways (such as one or more processes or a set of libraries). To decouple applications from implementation details, the CORBA specification defines an abstract interface for an ORB. This interface provides various helper functions such as converting object references to strings and vice versa, and creating argument lists for requests made through the dynamic invocation interface described below.
- **CORBA IDL stubs and skeletons** – CORBA IDL stubs and skeletons serve as the “glue” between the client and server applications, respectively, and the ORB. A CORBA IDL compiler automates the transformation between CORBA IDL definitions and the target programming language. The use of a compiler reduces the potential for inconsistencies between client stubs and server skeletons and increases opportunities for automated compiler optimisations.
- **Dynamic Invocation Interface (DII)** – This interface allows a client to directly access the underlying request mechanisms provided by an ORB. Applications use the DII to dynamically issue requests to objects without requiring IDL interface-specific stubs to be linked in. Unlike IDL stubs (which only allow RPC-style requests), the DII also allows clients to make non-blocking *deferred synchronous* (separate send and receive operations) and *one-way* (send-only) calls.
- **Dynamic Skeleton Interface (DSI)** – This is the server side’s analogue to the client side’s DII. The DSI allows an ORB to deliver requests to an object implementation that does not have compile-time knowledge of the type of the object it is implementing. The client making the request has no idea whether the implementation is using the type-specific IDL skeletons or is using the dynamic skeletons.
- **Object Adapter** – This assists the ORB with delivering requests to the object and with activating the object. More importantly, an object adapter associates object implementations with the ORB. Object adapters can be specialized to provide support for certain object implementation styles (such as OODB object adapters for persistence and library object adapters for non-remote objects).

Java Implementation of CORBA: Java IDL

Java™ IDL is a technology for distributed objects, that is, objects interacting on different platforms across a network. Java IDL is similar to RMI (Remote Method Invocation), which supports distributed objects written entirely in the Java programming language. However, Java IDL enables objects to interact regardless of whether they are written in the Java programming language or another language such as C, C++, COBOL, or others.

This is possible because Java IDL is based on the Common Object Request Brokerage Architecture (CORBA), an industry-standard distributed object model. A key feature of CORBA is IDL, a language-neutral Interface Definition Language. Each language that supports CORBA has its own IDL mapping and, as its name implies, Java IDL supports the mapping for Java. To support interaction between objects in separate programs, Java IDL provides an Object Request Broker, or ORB. The ORB is a class library that enables low-level communication between Java IDL applications and other CORBA-compliant applications.

CORBA technology is an integral part of the Java 2 platform. It consists of an Object Request Broker (ORB) written in Java (with a small bit of native code), APIs for the RMI programming model, and APIs for the IDL programming model. The IDL programming model, known as Java IDL, consists of both the Java CORBA ORB and the `idlj` compiler that maps the OMG IDL to Java bindings that use the Java CORBA ORB.

TAO-ACE: A Real-Time CORBA Implementation

TAO is a real-time implementation of CORBA built using the framework components and patterns provided by ACE. TAO contains the network interface, OS, communication protocol, and CORBA middleware components and features. TAO is based on the standard OMG CORBA reference model, with the enhancements designed to overcome the shortcomings of conventional ORBs for high-performance and real-time applications. TAO, like ACE, is freely available, open source software.

The ADAPTIVE Communication Environment (ACE) is a freely available, open-source object-oriented (OO) framework that implements many core patterns for concurrent communication software. ACE provides a rich set of reusable C++ wrapper facades and framework components that perform common communication software tasks across a range of OS platforms. The communication software tasks provided by ACE include event de-multiplexing and event handler dispatching, signal handling, service initialisation, interprocess communication, shared memory management, message routing, dynamic (re)configuration of distributed services, concurrent execution and synchronization.

ACE is targeted for developers of high-performance and real-time communication services and applications. It simplifies the development of OO network applications and services that utilize interprocess communication, event de-multiplexing, explicit dynamic linking, and concurrency. In addition, ACE automates system configuration and reconfiguration by dynamically linking services into applications at run-time and executing these services in one or more processes or threads.

Java RMI

Distributed systems require that computations running in different address spaces, potentially on different hosts, be able to communicate. For a basic communication mechanism, the Java programming language supports sockets, which are flexible and sufficient for general communication. However, sockets require the client and server to engage in applications-level protocols to encode and decode messages for exchange, and the design of such protocols is cumbersome and can be error-prone.

An alternative to sockets is Remote Procedure Call (RPC), which abstracts the communication interface to the level of a procedure call. Instead of working directly with sockets, the programmer has the illusion of calling a local procedure, when in fact the arguments of the call are packaged up and shipped off to the remote target of the call. RPC systems encode arguments and return values using an external data representation, such as XDR. RPC, however, does not translate well into distributed object systems, where communication between program-level *objects* residing in different address spaces is needed. In order to match the semantics of object invocation, distributed object systems require *remote method invocation* or RMI. In such systems, a local surrogate (stub) object manages the invocation on a remote object.

The Java platform's remote method invocation system has been specifically designed to operate in the Java application environment. The Java programming language's RMI system assumes the homogeneous environment of the Java virtual machine (JVM), and the system can therefore take advantage of the Java platform's object model whenever possible.

RMI applications are often comprised of two separate programs: a server and a client. A typical server application creates some remote objects, refers to them accessible, and waits for clients to invoke methods on these remote objects. A typical client application gets a remote reference to one or more remote objects in the server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth. Such an application is sometimes referred to as a distributed object application.

Distributed object applications need to:

- Locate remote objects: applications can use one of two mechanisms to obtain references to remote objects. An application can register its remote objects with RMI's simple naming facility, the `rmiregistry`, or the application can pass and return remote object references as part of its normal operation.
- Communicate with remote objects: details of communication between remote objects are handled by RMI; to the programmer, remote communication looks like a standard Java method invocation.
- Load class bytecodes for objects that are passed around: because RMI allows a caller to pass objects to remote objects, RMI provides the necessary mechanisms for loading an object's code, as well as for transmitting its data.

The following illustration depicts an RMI distributed application that uses the registry to obtain a reference to a remote object. The server calls the registry to associate (or bind) a name with a remote object. The client looks up the remote object by its name in the server's registry and then invokes a method on it. The illustration also shows that the RMI system uses an existing Web server to load class bytecodes, from server to client and from client to server, for objects when needed.

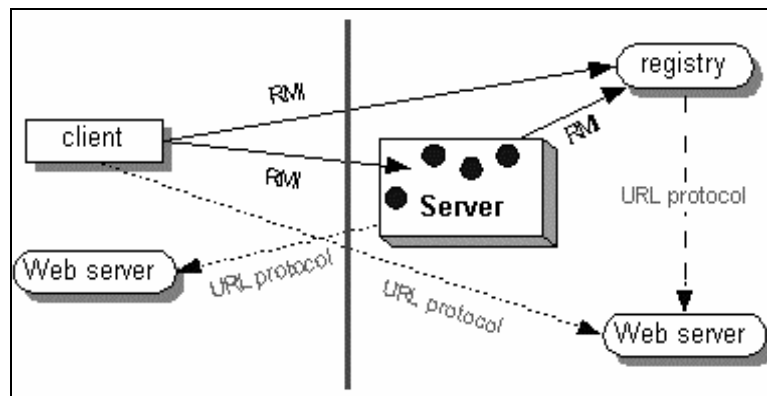


Figure 5: Java RMI Communication Scheme.

One of the central and unique features of RMI is its ability to download the bytecodes (or simply code) of an object's class if the class is not defined in the receiver's virtual machine. The types and the behaviour of an object, previously available only in a single virtual machine, can be transmitted to another, possibly remote, virtual machine. RMI passes objects by their true type, so the behaviour of those objects is not changed when they are sent to another virtual machine. This allows new types to be introduced into a remote virtual machine, thus extending the behaviour of an application dynamically.

An application, which has the ability to download code dynamically, is often called a based-based application. Such applications usually require full agent-enabled infrastructures. With RMI such applications are part of the basic mechanisms for distributed computing on the Java platform.

Despite of all its advantages, RMI lacks interoperability with other languages, and, because it uses a non-standard communication protocol, cannot communicate with CORBA objects. Previously Java programmers had to choose between RMI and CORBA/IIOP (Java IDL) for distributed programming solutions. Now, by adhering to a few restrictions, RMI objects can use the IIOP protocol, and communicate with CORBA objects. This solution is known as RMI-IIOP, which combines RMI-style ease of use with CORBA cross-language interoperability. It is presented in the next section.

Java RMI-IIOP

RMI over IIOP is part of the Java 2 Platform, Standard Edition, v1.3. RMI over IIOP provides the ability to write CORBA applications for the Java platform without learning CORBA Interface Definition Language (IDL). To work with CORBA applications in other languages, IDL can be generated from Java programming language interfaces using the `rmic` compiler with the `-idl` option. Use the `rmic` compiler with the `-iiop` option to generate IIOP stub and tie classes, rather than Java Remote Messaging Protocol (JRMP) stub and skeleton classes.

RMI over IIOP includes the full functionality of a CORBA Object Request Broker (ORB). The good news for developers is that the Java Standard Edition, v1.2 and higher includes a fully functional ORB that is available in every deployment of the Java 2 Platform, Standard and Enterprise Editions. RMI-IIOP combines the best features of Java Remote Method Invocation (RMI) with the best features of Common Object Request Broker Architecture (CORBA). It provides a powerful environment in which to do distributed programming using the Java platform.

Like RMI, RMI-IIOP speeds distributed application development by allowing developers to work completely in the Java programming language. When using RMI-IIOP to produce Java technology-based distributed applications, there is no separate Interface Definition Language (IDL) or mapping to learn. Like RMI, RMI-IIOP provides flexibility by allowing developers to pass any serializable Java object (Objects By Value) between application components. Like CORBA, RMI-IIOP is based on open standards defined with the participation of hundreds of vendors and users in the Object Management Group. Like CORBA, RMI-IIOP uses Internet Inter-ORB Protocol (IIOP) as its communication protocol. IIOP eases legacy application and platform integration by allowing application components written in C++, Smalltalk, and other CORBA supported languages to communicate with components running on the Java platform.

With RMI-IIOP, developers can write remote interfaces in the Java programming language and implement them simply using Java technology and the Java RMI APIs. These interfaces can be implemented in any other language that is supported by an OMG mapping and a vendor supplied ORB for that language. Similarly, clients can be written in other languages using IDL derived from the remote Java technology-based interfaces. Using RMI-IIOP, objects can be passed both by reference and by value over IIOP.

The RMI Programming Model

Several scenarios will define how you will want to create distributed applications. Here are some of them:

- If you have been developing CORBA applications using IDL for some time, you will probably want to stay in this environment. Create the interfaces using IDL, and define the client and server applications using the Java programming language to take advantage of its “Write Once, Run Anywhere” portability, its highly productive implementation environment, and its very robust platform.
- If all of your applications are written in the Java programming language, you will probably want to use Java RMI to enable communication between Java objects on different virtual machines and different physical machines. Using Java RMI without its IIOP option leverages its strengths of code portability, security, and garbage collection.
- If you are writing most of your new applications using the Java programming language, but need to maintain legacy applications written in other programming languages as well, you will probably want to use Java RMI with its IIOP compiler option.

Jini

Jini is the name for a distributed computing environment that can offer “network plug and play”. A device or a software service can be connected to a network and announce its presence, and clients that

wish to use such a service can then locate it and call it to perform tasks. Jini can be used for mobile computing tasks where a service may only be connected to a network for a short time, but it can more generally be used in any network where there is some degree of change. There are a large number of scenarios where this would be useful.

A Jini system or *federation* is a collection of clients and services all communicating by the Jini protocols. Often this will consist of applications written in Java, communicating using the Java Remote Method Invocation mechanism. Although Jini is written in pure Java, neither clients nor services are constrained to be in pure Java. They may include native code methods, act as wrappers around non-Java objects, or even be written in some other language altogether. Jini supplies a “middleware” layer to link services and clients from a variety of sources.

When you download a copy of Jini⁷, you get a mixture of things. Firstly, Jini is a specification of a set of middleware components. This includes an API (Application Programmer’s Interface) so that a programmer can write services and components that make use of this middleware. Secondly, it includes an implementation (in pure Java) of the middleware, as a set of Java packages. By including these in the classpath of a client or service, we can invoke the Jini middleware protocols to join in a Jini djinn. Finally, Jini requires a number of “standard” services, and Sun gives basic implementations of each of these. These implementations are not an official part of Jini⁷, but are included to get the developer going. In practice, most users are finding these sufficient to do substantial work with Jini.

Jini is just one of a large number of distributed systems architectures, including industry-pervasive systems such as CORBA and DCOM. It is distinguished by being based on Java, and deriving many features purely from this Java basis.

There are other Java frameworks from Sun, which would appear to overlap Jini, such as Enterprise Java Beans (EJBs). EJB’s make it easier to build business logic servers, while Jini could be used to distribute these services in a “network plug and play” manner.

In a running Jini system, there are three main players. There is a *service*, such as a printer, a toaster, a marriage agency, etc. There is a *client*, which would like to make use of this service. Thirdly, there is a *lookup service* (service locator), which acts as a broker/trader/locator between services and clients. There is an additional component, and that is a *network* connecting all three of these, and this network will generally be running TCP/IP. (The Jini *specification* is independent of network protocol, but the only current *implementation* is on TCP/IP).

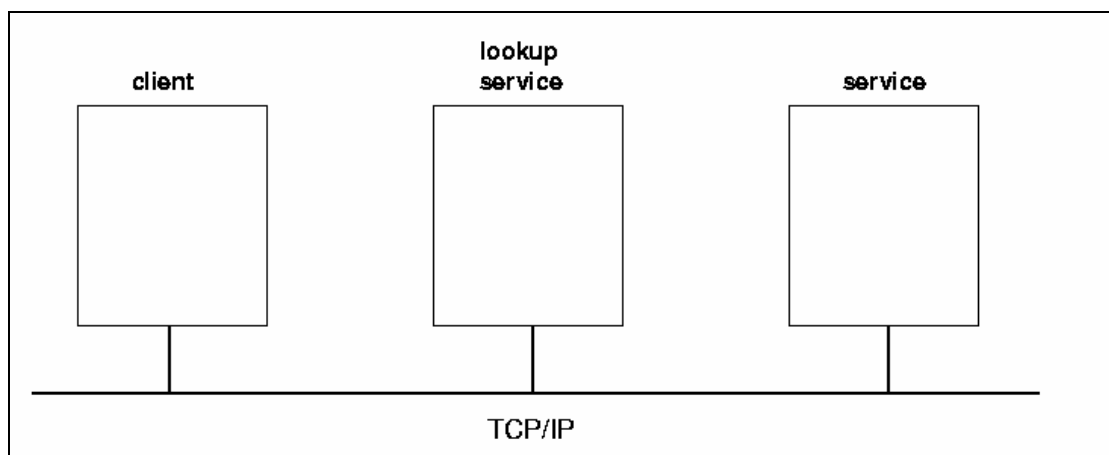


Figure 6: Jini Main Actors.

Code will be moved around between these three pieces, and marshalling the objects does this. This involves serializing the objects in such a way that they can be moved around the network, stored in this “freeze-dried” form, and later reconstituted by using included information about the class files as well as instance data. This is done using Java’s socket support to send and receive objects.

In addition, objects in one JVM (Java Virtual Machine) may need to invoke methods on an object in another JVM. Often this will be done using RMI (Remote Method Invocation), although the Jini specification does not require this and there are many other possibilities.

Particular implementations must make use of support services such as RMI daemons and HTTP (or other) servers. The particular support services required depend on implementation details, and so may vary from one Jini component to another.

Comparison of the Different Technologies

The following table summarises some characteristics of the technologies reviewed in the preceding sections:

Technology	Language	Platform	Required
DCOM	C++, Java (MS)	Windows, Unix	
CORBA : Java-IDL Java-IIOP Other	Java (link to native libraries possible) C,C++	All JVM varies	Corba ORB
Java-RMI	Java (link to native libraries possible)	All JVM	RMI registry
Jini	Java (link to native libraries possible)	All JVM	RMI registry HTTP server Lookup services

The architecture of CORBA, DCOM and Java/RMI provide mechanisms for transparent invocation and accessing of remote distributed objects. Though the mechanisms that they employ to achieve remoting may be different, the approach each of them take is more or less similar.

The three technologies can run on any platform as long as an implementation exists for this platform. Obviously, DCOM is better supported on windows platform while RMI requires a Java Platform. CORBA relies on the availability of ORBs and libraries on a given platform.

DCOM components can be implemented with different languages (C++, Java, Delphi) as the specification is at the binary level. CORBA relies on the availability of idl compilers and vendors’ implemented libraries but offers the widest choice of programming language. RMI relies clearly on Java and needs a JVM to run (native libraries can be used through the Java Native Interface).

Jini cannot be compared to RMI and CORBA. RMI and CORBA can be seen as middleware technologies that enable components and objects to communicate over a network. Jini, on the other hand, provides an interaction model and the infrastructure for distributed objects to cooperate with each other, or to work in a coherent, robust, and scalable way.

Jini specifies functionalities that allow setting up a network of objects that dynamically link together and perform useful work. Jini does not specify how the server and client objects communicate. This kind of work is the specialty of the other two technologies. RMI and CORBA do not compete against each other but are complementary and can be mixed to build up a complex network.

How to choose between the different technologies?

Jini is clearly at a higher level than other technologies; it provides facilities for objects management and interaction. Communication between objects can make use of middleware technologies (If we have to connect to legacy CORBA applications we can make use of IDL or IIOP) or even raw socket implementation. We see that choosing Jini does not restrict our self to the Java world. This solution is flexible but requires more support services (like HTTP servers).

To conclude this section we can state that Jini offers a new and novel distributed computing model and provides an infrastructure for objects to find one-another, join together to get work done, to disconnect and reconnect, and to recover from network or software failure. However, Jini is not sufficient for building intelligent applications. Jini provides basic networking facilities and is a kind of glue that can be used to facilitate network management and communication between different services.

In the next section we will explore the fascinating world of Artificial Intelligence, more precisely the domain of Multi-agents systems.

MULTI-AGENTS SYSTEMS

Introduction – Definition

Many people to mean many different things have used the term “agent”. Even within the Agent Research Community, there are at least the following variants on the term *agent*: mobile agent, learning agents, autonomous agents, planning agents, simulation agents, distributed agents,...

Without entering into philosophical discussion (we can find as many definitions as researchers), we propose there the following definition from Stan Franklin, which seems appropriate to us because of its generality:

[ref: Is it an Agent, or just a Program? Taxonomy for Autonomous Agents]

An (autonomous) agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and to affect what it senses in the future.

This definition is very broad and does not distinguish pure software agents from agents having a physical reality like robots.

Agent Architecture

Agents constitute a special case of organisation. The term generally employed to describe the internal organisation of an agent is that of architecture. We present here the main architecture used in building agents. [Ferber]

Modular Horizontal Architecture

Modular Horizontal Architectures (MHA) are conceived as being an assembly of modules, each carrying out a specific horizontal function. In this type of architecture all links are fixed, that is, the designer predefines the data circulation mode; this can be seen as the main advantage and drawback of this architecture. Advantage because the processing scheme is predictable and drawback because of its lack of flexibility.

There are two phases: an ascending one and a descending one. In the ascending phase, signals coming from the outside through sensors or letterboxes are filtered in such a way as to obtain information of an increasingly abstract nature. The highest function is carried out by the decision-making module, which decides to act based on the data it receives and in accordance with its own objectives. We then move into the descending phase, corresponding to the implementation of the decisions. The planning module determines the actions that need to be carried out to attain the selected objective. These are then transmitted to the execution module.

Blackboard-Based Architectures

The Blackboard (BB) model is based on a division into independent modules which do not communicate any data directly but which interact indirectly by sharing data. These modules, called *Knowledge Sources (KS)* work on a space, which includes all the elements, required for solving a problem. The architecture of a BB based system comprises three subsystems:

- KS;
- The *shared base*, which encompasses all the data exchanged between the KS; and
- A control device, which manages conflicts of access between the KS. This controller principally determines how good the BB system will perform.

BB architectures have numerous advantages, including a remarkable flexibility in describing modules and articulating their functioning. Its interest lies in the fact that it is both opportunist and centralized. The main drawback of BB architecture is its relative inefficiency, owing to the highly expressive character of its control. In any case, the advantage of this architecture is the possibility to implement any agent structure in terms of boards elements and KS, corresponding to a sort of “meta-architecture”.

Subsumption Architecture

Subsumption architecture (SA) breaks an agent down into vertical modules, each of them being responsible for a very limited type of behaviour (proposed by Brooks).

The interactions between the modules are fixed, and are effected through the intermediary of a dominance relationship, defined in the design stage. The modules carry out their tasks in parallel, but if the dominant produces a result, then only this one will be taken into consideration. It is therefore possible to construct relatively complex subsystems using such architecture. However, once again the designer defines the assembly of modules and the dominance relationships that exist between them. This technique has been mostly used to describe reactive agents¹.

Competitive Tasks

In the Competitive Tasks (CT) architecture, an agent is made up of an assembly of tasks, only one of which can be active at any time. These tasks are thus in competition for selection by a decision mechanism which takes account of various parameters: the weight accorded to the task at given moment, the application context, data coming from the outside, etc.

¹ Reactive agents do not use internal models of the environment and do not plan actions. Perceptions are directly used to generate behaviors. They are opposed to cognitive agents.

The selection module receives the readings for the various task parameters from each of the tasks, and selects the one for which the evaluation function is the highest. Once selected, the task is activated, and the previous current task is deactivated.

The main problem resides in the design of the evaluation function to perform selection.

Production Systems

A Production System (PS) is defined as the combination of a database, a production rule base and an interpreter, the inference engine. They are generally given in the following form:

If <list of conditions> then <list of actions>

where <list of conditions> is associated with the elements in the database and <list of actions> comprises elementary actions such as adding or deleting elements.

Some actions can also activate execution commands for the agent directly. When a rule can validate each of the conditions on its list, it executes the corresponding actions. If several rules can be activated, we say that they are in conflict, and the inference engine control system activates the rule with the highest priority (internal parameters or time criteria).

PS present two major drawbacks:

- The action part of the rules generally includes action deletion mechanisms; the engine is said to be non-monotonous. The result obtained depends on the order the rules are applied.
- Rules cannot be combined.

PS can be applied to small systems with limited number of rules.

Classifier-Based Systems

Systems based on classifiers are special cases of production rules systems intended to design evolutionary autonomous systems. Two mechanisms modify the production system scheme: a credit attribution system (the weights of rules which succeeded are increased) and a system for reproducing rules by genetic algorithms. These classifier systems are essentially used to create adaptive behaviours.

Fuzzy Logic Architecture

Fuzzy logic (FL) extends the classic Boolean logic by using continuous values ranging between 0 and 1. FL allows one to express knowledge in a rule format that is close to natural language expression.

Instead of a match and conflict resolution phase where we select a triggered rule to fire, in fuzzy systems, all rules are evaluated, because every fuzzy rule can be true to some degree. Rules are then combined using extended AND/OR operations to produce at the end (defuzzification) a crisp numerical value.

Connectionist

Connectionist architectures (CA) are made up of identical elements called neurons. Each neuron determines its output function based on the input values it receives from other neurons. The classic NN model is composed of three layers of neurons. The first layer is directly connected to the sensors and the last one to the actuators. The intermediate layer, called internal or hidden layer, serve to memorize internal states.

The weights of the connections between neurons can be defined in different ways:

- Predefined at design time;
- Learned by using a back-propagation algorithm (Static); and
- Adapted on-line using fuzzy inference networks or genetic algorithms.

The main drawback of CA is the lack of transparency of network internal functioning.

Agent Perception

In order for an agent to take some intelligent action, it first has to be able to perceive what is going around it, to have some idea of the state of the world. The information comes in through its sensors, which may or may not be grounded in the physical world. Just like people, agents must be able to distinguish the normal events from the significant events. Being able to notice or recognize information hidden in data is not easy. It requires intelligence and domain knowledge. To be useful personal assistants, agents must be perceptive.

If perception is the ability to recognize patterns, and our agent receive its inputs as streams of data, then we are entering the realm of machine learning and pattern recognition. Learning is not only useful for adapting behaviour; it is also quite handy when we want to recognize changes in our environment. Neural networks and decision trees can be used to automatically map between world states and actions.

Agent Action

Once our agent has perceptively recognized that a significant event has occurred, the next step is to take some action. This action could be to realize that there is no action to take, or it could be to send a message to another agent to take an action on our behalf. It could also be a real action like a manoeuvre to avoid an obstacle. Like people, agents take action through *effectors*.

There are many ways of modelling actions and their consequences. Here are some of the existing formalisms:

- Action as transformation of global state.

The functional transformation of states model is certainly the most classical model, all AI planning theory was developed within this framework and most theories on multi-agent planning refers to it. The STRIPS-like planners are the famous representatives in this category. This model is limited because:

- It considers a static world and all possible modifications are the results of agent's actions (staticity postulate);
- It produces sequential plans difficult to translate to simultaneous actions (sequentially postulate); and
- Actions are described by their results and not by the actions performed by the agents (universality postulate).

Action as Response to Influences

This model extends the preceding one in order to be able to take into account of the consequences of the simultaneous actions of agents and model phenomena stemming from interactions between agents. This model also makes it possible to describe actions considered as displacements in a physical space. It dissociates what is produced by the agents (the gesture) from what is actually happening, by application of the "laws of the universe". The latter are the responses of the environment, which acts by combining all

the influences it receives from agents, and then deducing from them a new global state in conformity with its own particular nature.

Action as Computing Processes

The Universe can be seen as a set of activities, which are carried on in parallel and called processes. These actions give rise to the production of observable events. We are then interested in the interactions, which take place between these processes and the possibility of linking them together so that they can be executed in parallel. The most common ways of representing processes are finite-state automata, register automata and Petri nets.

Finite-state automata, like register automata, can represent only sequential processes and consequently when problems become more complex, we need more complex formalisms like Petri nets. Petri nets can be considered as essentially asynchronous process-modelling tools based on a representation, which is both graphic and mathematical. They are used in a number of fields where it is necessary to represent mechanisms, which are implemented in parallel.

Action as Local Modification

We assume here that the world is made up of a network whose nodes represent ‘real’ entities and whose arcs correspond to the links between these entities. Each node acts in an independent manner and modifies its own state in response to its perception. Cellular automata are a simplified example of such a model. They can be considered as ‘degenerate’ MAS in which the agents have become fixed.

Action as Physical Displacement

The concept of potential fields is predominantly used in reactive systems to determine the behaviour of agents. The general principle is to assume that the objects in the environment are emitting signals, whose intensity is proportional to the distance to the goal. Obstacles have potential field values increasing when one approaches them. The agent tries to find a trajectory offering the best compromise between the attractions of the goal on the one hand and the repulsion of the obstacles on the other.

The physical approach to action proves very fruitful in two cases:

- When robots are moving physically in their environment.
- When actions depend on time and it is possible to use concepts of fields to describe the agents’ tendencies to do something.

However, these methods are not naturally integrated into any logical conception of intention and action and it is almost impossible to predict the behaviour of a system in the presence of such autonomy.

Action as Command

The problem of action consists in causing variations in a certain number of inputs in a physical system to obtain specific output variable values. Therefore, action is no longer merely a transformation of states of the world but a complex activity entirely directed towards a goal, which takes account of the reactions of the environment and the corrections to be made to previous actions. This view cannot express the interactions that can take place between agents nor the distribution of tasks between agents.

Knowledge Representation

Languages for representing knowledge are used by cognitive agents to describe internal models of the world in which they move and to allow them to reason and to make predictions about the future based on

the data available to them. These languages are most frequently associated with logical systems, that is, with formal systems which have a syntax and semantics rigorously defined to make their inferences explicit. They serve not only to express the mental states of the agents and, in particular, to represent the content of their goals and their beliefs, but also to describe agents' behaviour. When the agents are simple and reactive, we may perhaps be able to do without this layer of language. However, for cognitive agents, or for agents who are even slightly complex, the use of languages of this type is almost imperative. In this family, we essentially find AI languages, such as rules-based or blackboard-based languages, and languages for the structured representation of knowledge such as semantics nets or frames.

The Knowledge Interchange Format is a language that was expressly designed for the interchange of knowledge between agents. KIF is based on predicate logic and can bridge the gap between different internal knowledge representations.

Multi-Agent Systems Organisation

When considering MAS, the two most important aspects are:

- Interaction between agents;
- Communication between agents; and
- Collaboration between agents.

We can imagine applications made of single agents, but it is obvious that agents become interesting when they interact with each other or with humans. We then speak about Multi-Agents Systems and add the word distributed when agents reside on different machine spread over a network.

We can formally organize agents' organization or let the organization spontaneously emerge from the agency.

Agent Interactions and Cooperation

An interaction occurs when two or more agents are brought into a dynamic relationship through a set of reciprocal actions. The agents interact through a series of events, during which they are in contact with each other in some way, whether this contact is direct or takes place through another agent or through the environment.

The main interaction situations can be classified in relation to three criteria: the objectives or intentions of the agents, the relationships of these agents to the resources they have available, and the means (or skills) available to them to achieve their ends. These three main elements can be used to create a typology of interaction situations: independence, simple collaboration, and obstruction,...

Cooperation can present two different aspects: we can consider that cooperation is the attitude of agents that decide to work together or from an observer point of view, we can interpret behaviours and qualify them as cooperative or not by using indicators such as the interdependence of actions or the amount of communications.

We can classify the different methods of cooperation as follows: grouping and multiplication, communication, specialization, collaboration by sharing task and resources, coordination of actions, conflict resolution by arbitration and negotiation.

By favouring collective performances from the agent, cooperation offers a certain number of advantages, which are expressed, either as individual improvements, such as productivity increase or an increase in the probability of survival for individuals, or by collective improvements, such as the accomplishment of tasks which would have been impossible otherwise.

Agent Communication

In multi-agent systems, communication is the basis for interactions and social organization. It is because agents communicate that they can cooperate and coordinate their actions. When they talk to each other, they can do this in a variety of ways. They can talk directly to each other or they can talk through an interpreter or *facilitator*. The minimum requirement is that they speak the same language or different languages that are translated by the interpreter.

We can distinguish two communication levels: the basic language -the syntax and format of the messages- and a higher-level -the meaning or semantics. While the first level is quite evident, the second one is less obvious. The communicators need to have a shared vocabulary of words and their meaning. This shared vocabulary is called ontology.

Each specific domain can have its own ontology. When agents negotiate about a plan or a task in a given domain, they use a common ontology to ensure, there are no doubts about the semantics of the message terms.

In the US, a research project funded by the DARPA has the goal of developing a high-level communication standard based on speech acts, to allow cognitive agents to cooperate. KQML focuses on message formats and message-handling protocols between running agents. Only the message protocols are specified, not the language used to represent the content of the message.

KQML messages are called *performatives*, which is a term from speech-act theory. There are various types of speech acts, including *directives*, which are commands or requests; *representatives*, which state facts or beliefs; and *commissives*, such as promises or threats. Each message is intended to implicitly perform some specified action. There are a large number of performatives defined in KQML, and most agent-based systems support only a small subset. The performatives, or message types, are reserved words in KQML. Using performatives, agents can ask other agents for information, tell other agents facts, subscribe to the services of agents, and offer their own services.

KQML uses ontology's (explicit specifications of the meaning, concepts, and relationships applicable to some specific domain) to ensure that two agents communicating in the same language can correctly interpret statements in that language. For example, in English, when we say "Java," are we referring to coffee, an island, or a programming language? To eliminate this ambiguity, every KQML message explicitly states what ontology is being used.

Two agents who want to communicate using KQML require the services of a KQML facilitator or matchmaker. The agents can register themselves as providers of services or information by using the advertise performative. Agents can also ask the matchmaker to recommend other agents using the recommend recruit and broker performatives. The matchmaker provides a centralized meeting-place for agents, and establishes a community where agents can interact. The matchmaker plays an important role in a multi-agent system; the alternative would be for every agent to query other agent whenever it needed to collaborate.

Although KQML has been used in many MAS applications, it does not mean that every KQML agents can talk to each other. In almost all cases, the implementations of KQML are highly dependent on the performatives used and the exact interaction protocols that are defined. KQML has been extended and modified for use as the basic agent communication language (ACL) in the FIPA² standards.

Collaboration and Distribution of Tasks

Task allocation (or distribution) involves the definition of the organizational mechanisms through which agents can combine their skills to perform collective work. So we need to describe how tasks are allocated,

² Foundation for Intelligent Physical Agents.

knowing that the capacities of an agent depend as much on its intrinsic aptitudes and the power resources available to it, as on external resources and environmental constraints.

When breaking down tasks, it is necessary to make them as independent of one another as possible, in such a way as to reduce the coordination required. In particular, we must try to minimize the quantity of data that tasks have to send to each other, and to make sure that tasks can make use of local resources, in order to diminish conflicts in connection with resources.

An automatic task allocation system should be capable of connecting agents needing information, or task realization, the clients, and agents capable of supplying a service, the suppliers or servers. The same agents are frequently clients and servers at the same time, their status as clients or suppliers being generally determined in a dynamic fashion during the execution of a MAS. We then say that an agent is taking the role of a supplier, without this being one of its intrinsic characteristics. Other roles can be defined, such as that of a broker or trader, which brings servers and clients into contact.

The breakdown of tasks can be managed by centralising the allocation process or by distributing it among all the agents concerned.

Centralized Allocation

In a centralised allocation mode, two cases may arise: a hierarchical or an egalitarian structure.

If the subordination structure is hierarchical, it will be the superior agent that will order a subordinate to carry out the task. We can speak of rigid or defined allocation. This allocation mode is similar to procedure call and is will not be further considered here.

In contrast, if the structure is egalitarian, distribution then involves the definition of special agents -traders or brokers- which manage all the allocation procedures by centralizing the requests from the clients and the bids for service from the servers, in order to match up these two categories of agents. This makes it possible to apply centralized modes to variables organizations.

There are ways to optimise the allocation by priority selection of the most skilled agents, or those that minimize a cost function. We can envisage two cases:

- If the trader knows the evaluation functions set by the suppliers, it can sort out the skilled agents based on the results of the evaluation and carry out its request in this order.
- If the trader does not know the evaluation functions, it should first ask each agent to formulate its proposition before actually selecting the one that is to be considered the best. This technique resembles a contract net mechanism (see below).

The interesting point about centralized allocation is that we can update the set of agents and their respective skills, and therefore favour the coherence of the system. In addition, the requirements for optimisation are more easily satisfied, the trader knowing the set of agents available, and it is easier for it to select the “best” of the agents in relation to a given task request.

Nevertheless, the major drawback of this type is that it creates a bottleneck, and so considerably reduces the performance levels of the system as soon as the number of agents and requests increases. In fact, the number of messages the trader has to manage grows as the square of the number of agents, which indicates that this method is not recommended for a large number of agents.

Finally, a centralized system is very sensitive to failures. In effect, if the trader breaks down, the entire system falls to pieces. It is possible to improve this situation by using several traders, but it raises other

problems like maintaining coherence between them. When it is necessary for the system to distribute its allocation mechanisms, it is preferable to go to a true distributed task allocation mode.

Distributed Allocation

In a distributed allocation mode, each agent individually tries to obtain the services it needs from the suppliers. These modes apply only to variable organizations, that is, to those which can assume that the relationship between clients and suppliers can evolve over time without calling the general structure of the organization into question. We can distinguish two distributed allocation mechanisms:

- The *acquaintance network* mode of allocation assumes that clients have a representation of other agents and of their capabilities (or at least to be able to connect to other agents for querying capabilities).

There are essentially two modes of acquaintance network allocation, depending on whether agents are authorized to delegate their requests to other agents or not. We speak about direct allocation or allocation by delegation:

- In the direct mode, an agent can have a task carried out only by an agent that it knows directly. The agent that wants to have a task carried out applies in turn to each of the agents it knows, which has the desired skill until one of them accepts.
- The technique of allocation by delegation makes it possible to link together clients and suppliers that do not know each other directly. This allocation mode allows a supplier, which is asked to carry out a task to send it to another agent if it is not capable of doing it. The search is generally performed in parallel and the agents must therefore be able to deal with multiple acceptances.
- *The request for bids* mode of allocation, which is better known under the name of *contract net*. It has the advantage to be very dynamic and easy to implement.

The contract net is a task allocation mechanism based on a market-like protocol. The relationship between the client, or the manager, and the suppliers, or bidders, is channelled through a request for bids and an evaluation of the proposals submitted by the bidders. The request for bids has four stages:

- The first stage is devoted to the request for bids itself. The manager sends a description of the task to perform to the agents of the system.
- Based on this description, the bidders draw up proposals that they submit to the manager.
- The manager receives and evaluates proposals and awards the contract to the best bidder.
- The bidder which has been awarded the contract and which therefore becomes the contractor sends a message to the manager to confirm or infirm (which triggers a re-evaluation of the bids) its commitment.

The contract net system generates a large number of messages and can therefore be applied only to small-size nets carrying out tasks broken down into very small elements.

It is obviously possible either to create variations of one of these approaches or to merge them to try to combine their advantages.

Coordination of Actions

When several agents are working together, it is necessary to manage a certain number of supplementary tasks that are not directly productive but serve to improve the way in which those activities are carried out.

The coordination of actions is one of the main methods of ensuring cooperation between autonomous agents (see ‘Agent Interactions and Cooperations’). Actions have to be coordinated for four main reasons:

- The agents need information and results produced by other agents.
- Resources are limited.
- We want to optimise costs.
- We want to allow agents having separated but interdependent objectives to meet their objectives while profiting from this inter-dependence.

The problem of the coordination raises several questions:

- With what should actions be coordinated?
- What is the mutual dependence between actions (space and time)?
- What are the relationships between actions (negative, neutral, positive)?

Several parameters are used to characterize the various forms of coordination of action. Ferber proposes eleven characteristics grouped into five categories:

- **Temporal:** Rapidity, adaptability and predictiveness characterize the relationship of the coordination system over time.
- **Organizational:** it relates to the manner in which the coordination of actions is organized and to its centralization, centralization/distribution, the communication mode, and the freedom of actions.
- **Quality and efficiency:** optimality, avoidance of conflicts and the number of agents being coordinated are measures for these characteristics.
- **Realization:** it relates to the means required for the realization of a coordination system: the degree and quantity of data, the degree of mutual representation, the difficulty of implementation.
- **Generalization:** it indicates to what extent a coordination method is general, by authorizing some heterogeneity in the agent or by applying to different domains.

On the basis of above criteria, we can distinguish four main forms of coordination of actions:

- Coordination by synchronization;
- Coordination by planning;
- Reactive coordination; and
- Coordination by regulation.

Coordination by Synchronization

To synchronize several actions it is necessary to define the manner in which actions are time-related, in order to time them in the right order and carry them out just at the right moment. Synchronization constitutes the lowest level of the coordination of actions. Petri nets are generally used to describe and solve the problems of synchronization.

Coordination by Planning

Planning actions in multi-agent universes can be broken down into three distinct stages: making plans, synchronizing/coordinating plans and executing plans. One or several agents can be involved in these operations and consequently, the three main classic modes of organisation in multi-agent planning are:

- Centralized planning for multiple agents;
- Centralized coordination for partial plans; and
- Distributed planning.

Reactive Coordination

In contrast to the previous approaches, reactive coordination considers that it is often simpler to act directly, without planning what one wishes to do in advance. All information relating to their behaviour is located in the environment, and their reactions depend solely on the perception they may have of it.

When the agents have dependent goals and the actions of some can improve those of others, the general principle consists of using the capacities of reactive agents to react to modifications of the environment. Almost all techniques come down to the following ones:

- Use of potential fields (or vector fields).
- Use of marks to coordinate the action of several agents.

Coordination by Regulation

The principle is to set rules of behaviour that aim to eliminate possible conflicts. This technique is inspired, by all regulations used to define what is good behaviour, to avoid conflicts as far as possible.

Eco-Problem Solving

Eco-problem solving is a method, which takes the opposite course to the classic techniques. Rather than formalizing a problem in a global manner and then defining solution methods which apply directly to its definition, we assume that it is preferable to reformulate the problem in such a way that it is defined as a set of agents in interaction, which are trying to achieve their own goals individually.

Taxonomies of Agents

The table that follows lists several of the properties an agent could enjoy.

These properties may help us further classify agents in useful ways.

Property	Other Names	Meaning
Reactive	(Sensing and acting)	Responds in a timely fashion to changes in the environment
Autonomous		Exercises control over its own actions
Goal-oriented	Pro-active purposeful	Does not simply act in response to the environment
Temporally continuous		Is a continuously running process
Communicative	Socially able	Communicates with other agents, perhaps including people
Learning	Adaptive	Changes its behaviour based on its previous experience

Property	Other Names	Meaning
Mobile		Able to transport itself from one machine to another
Flexible		Actions are not scripted
Character		Believable “personality” and emotional state

Agents may be usefully classified according to the subset of these properties that they enjoy. Every agent, by the above definition, satisfies the first four properties. Adding other properties produces potentially useful classes of agents, for example, mobile, learning agents. Thus, a hierarchical classification based on set inclusion occurs naturally. Mobile, learning agents are then a subclass of mobile agents.

There are, of course, other possible classifying schemes. For example, we might classify software agents according to the tasks they perform, for example, information gathering agents or email filtering agents. Alternatively, we might classify them according to their control architecture. Agents may also be classified by the range and sensitivity of their senses, or by the range and effectiveness of their actions, or by how much internal state they possess.

Yet, another possible classification scheme might involve the environment in which the agent finds it, for example software agents as opposed to artificial life agents. In addition, there must be many, many more such possibilities.

MAS Frameworks

When we search on the Internet for the words agent or multi-agent, we receive thousands of links. Some repositories have links to main agent development tools: the site Agent Builder has almost 40 links to research projects and more than 25 links to commercial products.

We consider in this section only general-purpose frameworks. It is quite difficult to classify these developments because their description is not standardized. The well-known development systems are:

- ABLE (IBM)
- AgentBuilder (Reticular Systems Inc)
- Aglets (IBM)
- FIPA-OS
- Gossip (Tryllian, Inc)
- Jade (CSELT Spa)
- JATLite (Standford)
- Jess (Sandia National Lab)
- Voyager (ObjectSpace, Inc)
- Zeus (BT)

COMPARISON BETWEEN DISTRIBUTED COMPUTER SYSTEMS AND MULTI-AGENT SYSTEMS

Is it an agent or a program? What is the difference between a distributed application and a distributed multi-agent system?

Distributed Agent technology (the “multi” qualifier can be omitted) can be thought of as the next step in the evolution of programming methodologies. **In the beginning**, there were machine and assembly languages. These evolved into higher level programming languages able to break apart programming steps into **subroutines**. A next generalization allowed programmers to group collections of subroutines into **libraries** or **modules**. A subsequent innovation added the notion of object orientation: data and routines could be grouped into a single **object**, which further encapsulated the internals of the routines and increased modularity and reuse. Distributed Object technologies, such as CORBA or DCOM, then broke the rule that every object must reside on the local machine; now object libraries could post services through a broker, and the objects themselves could even be written in different programming languages, as long as they used the same Interface Definition Language.

So, what can **Distributed Agents** possibly add to the Distributed Object paradigm? With distributed objects, even though objects may run on different platforms, applications generally form a single monolithic entity of tightly bound objects, with hand-coded calls to known methods of pre-existing objects.

In a distributed **agent** framework, we conceptualise a dynamic community of agents, where multiple agents contribute services to the community. When a given agent, instead of calling a known subroutine or asking a specific agent to perform a task, requires external services or information the agent submits a high-level expression describing the needs and attributes of the request to a specialized **Facilitator agent**. The Facilitator agent will make decisions about which agents are available and capable of handling sub-parts of the request, and will manage all agent interactions required to handle the complex query. **The advantage?** Such distributed agent architecture allows the construction of systems that are more flexible and adaptable than distributed object frameworks. Individual agents can be dynamically added to the community, extending the functionality that the agent community can provide as a whole. The agent system is also able to adapt to available resources in a way that hardcoded distributed objects systems cannot.

TELEROBOTICS APPLICATIONS

TA 2001

This is a summary of the reviewing and analyse of relevant papers presented at the Telematics Applications in Automation and Robotics conference (TA2001). The general tendency for recent, current and future applications is oriented towards **Java** as programming language, **Corba** as the software middleware and **Web Browsers** as container for user interfaces.

Some additional interesting points come out from this analyse:

- No one has mentioned the use of Software Components (like Javabeans, or COM).
- Most development focus on a well-defined applications which consequently are difficult to extend or generalize.
- No applications present all properties a general development framework should possess. But if we consider all applications we can extract all the characteristics we target:
 - Distributed
 - Portable
 - Extensible
 - Reusable
 - Customable user-interface (with 3d and video)
 - Allowing teleoperation and support algorithms

- Multi-robot (simultaneously)
- ...

Other Frameworks

See the technical report “Review of robotics control and teleoperation architectures” for more information.

CONCLUSION

The future development in the domain of the control and robotics will have to take into account current and emerging computer engineering technologies. Object oriented programming, components, middleware, distributed computing and Multi-agent systems need to be seriously considered when implement new control systems. All these technologies will obviously increase reusability and improve development time what should consequently decrease development costs.

For the moment, developers are facing dilemma when choosing programming technologies. There is no panacea and all solutions present advantages and drawbacks. However, most of the technologies offer bridging possibilities to, a priori incompatible, competing solutions.

Annex H – MULTI-ROBOT SYSTEM TAXONOMY

The following hierarchy is a first approach to structure the activities and areas of the multi-robot system in a kind of keyword tree. The tree will change and improve as the work of the RTG continues.

APPLICATIONS

- Field Robotics
- Robot Soccer
- Military Application
- Service Robotics
- Urban Search and Rescue

ARCHITECTURES

- Behavioural
 - Subsumption*
 - Motor and Perceptual Schemas*
- Hierarchical
- Heterarchical
 - RCS*
- Hybrid
 - 3T*
- Multi-Agent Architectures
 - Alliance*
 - CAMPOUT*
- Cognitive Architectures (See Cognition)

ARTIFICIAL SOCIAL SYSTEMS AND SOCIOROBOTICS

- Emotional Models
- Ecological Models
- Economic Models
- Social Models

BEHAVIOURS

- Arbitration of Behaviours, Combining Behaviours
- Basis Functions

Coordination of Behaviours
Emergent Behaviours and Emergent Intelligence
Learning Behaviours
Learning Coordination of Behaviours

BIOLOGICAL MODELS AND MOTIVATIONS

Ant Colony
Swarm

COGNITION

Cognitive Modelling of Humans in Interface
Cognitive Models for Control of Robot
ACT-R
Soar
Emotional Models

COMMAND AND CONTROL

COMMUNICATIONS

Ad Hoc Mobile Networking

COOPERATION, COORDINATION, COLLABORATION

Agent Communications
Explicit Communications
Stigmergy
Learning Coordination

DEVELOPMENT TOOLS

DISTRIBUTED CONTROL

DISTRIBUTED KNOWLEDGE/DISTRIBUTED AI

DISTRIBUTED SENSING

DYNAMIC AUTONOMY

Mixed-initiative

EVOLUTIONARY ROBOTICS

Evolving Software/Behaviours/Coordination
Evolvable Hardware

HARDWARE / PLATFORMS

UAVs
UUVs
UGVs
Micro-Robots

HUMAN-ROBOT INTERACTION, USER INTERFACES AND MIXED-INITIATIVE SYSTEMS

Gaze Control
Multi-Modal Interfaces (Speech, Gestures, etc)
Immersive Environments
Teleoperation

IMPLEMENTATIONS

LEARNING AND ADAPTATION

Evolutionary Computation
Supervised Learning
Reinforcement Learning
Neural Networks
Unsupervised Learning
Off-Line Optimisation

MAP BUILDING

Metric Maps
Topological Maps

METRICS FOR INTELLIGENCE AND MULTI-ROBOT SYSTEMS

MODELLING AND SIMULATION

Environments
The Swarm System

MOBILITY

Concurrent Localization and Map Building
Localization / Position Estimation
Local Navigation and Collision Avoidance, Reactive Navigation
Map Building, Map Adaptation
Path Planning

PLANNING AND SCHEDULING

Mission Planning

PERCEPTION**REASONING METHODS**

Cognitive Methods
Spatial Reasoning
Temporal Reasoning

RECONFIGURABLE ROBOTS**REPRESENTATIONS**

Evidence Grids
Frames, Chunks
Hidden Markov Models
Neural Networks
Rule-Based Systems

SENSORS**SELF-MONITORING/SELF-DIAGNOSIS/SELF-AWARENESS****SELF-ORGANIZATION****SURVEYS AND OVERVIEWS****THEORY AND ANALYSIS**

Annex I – RESULTS OF THE PRELIMINARY QUESTIONNAIRE REGARDING MULTI-ROBOT SYSTEMS

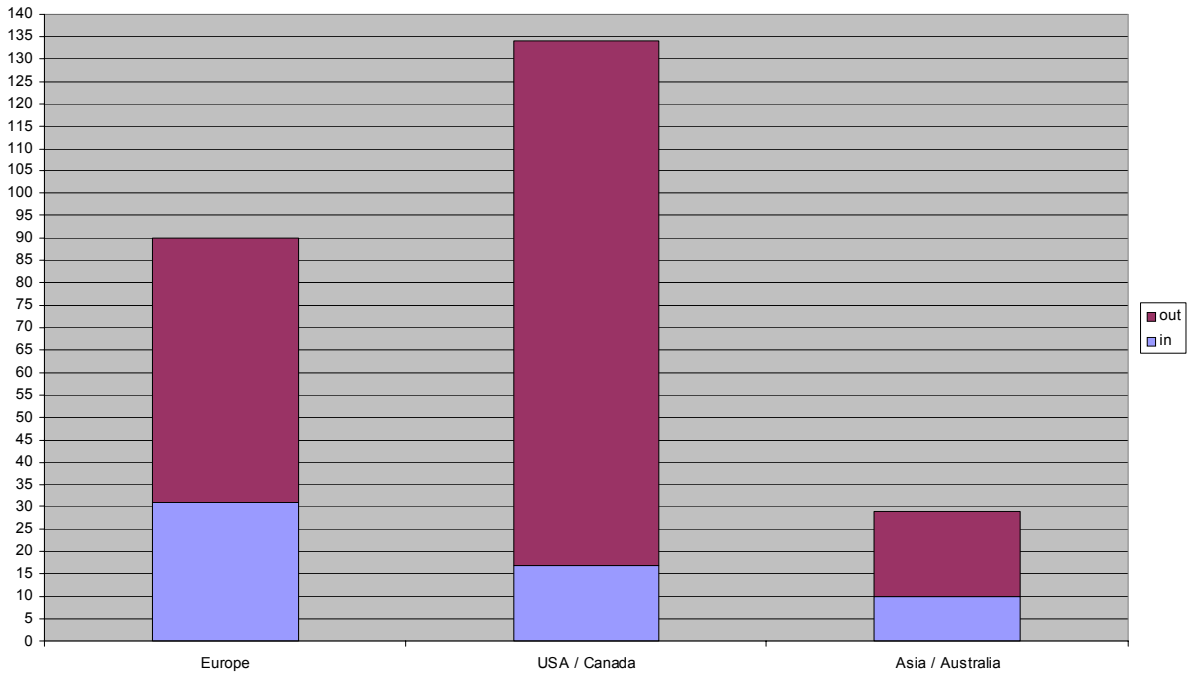
The questionnaire was sent to 255 research facilities, companies, universities and other institutions concerned with MRS.

Netherlands	18
Germany	18
Italy	13
Belgium	17
France	9
Great Britain	5
Portugal	3
Finland	1
Swiss	2
Spain	2
Sweden	2
USA	124
Japan	23
Australia	6
Canada	10

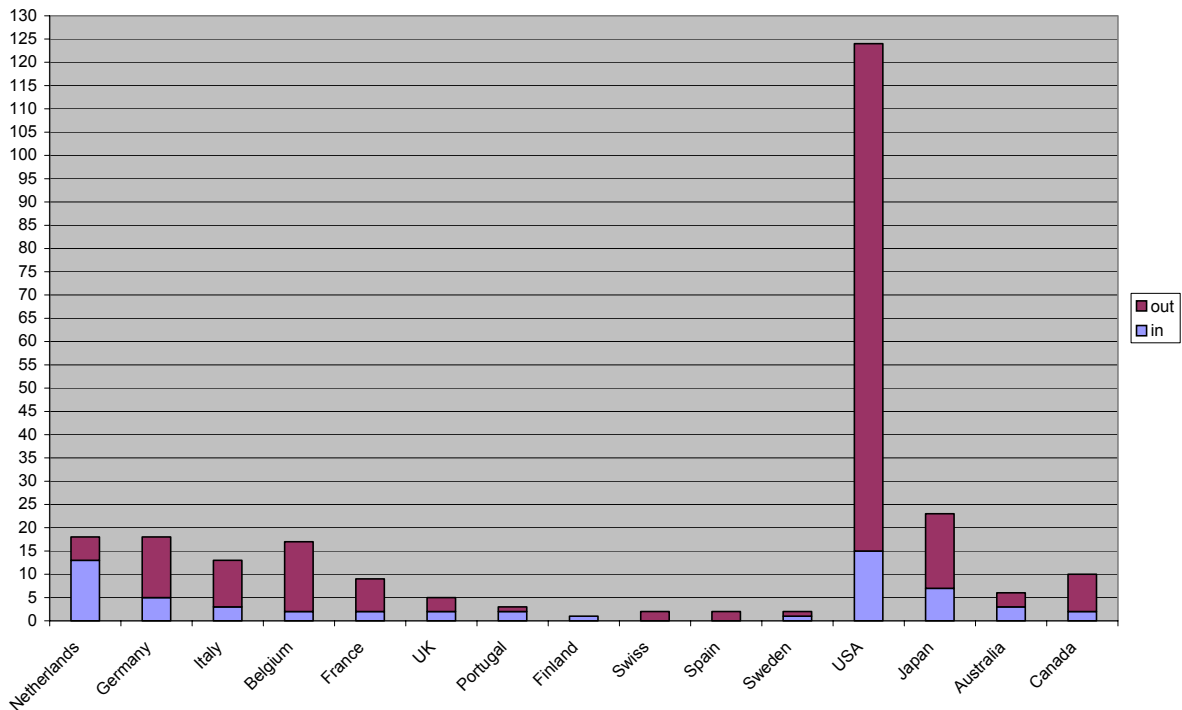
After three reminders, poor responses have been received so far from most nations. There is a total of 60 responses.

Europe	31
USA/Canada	19
Asia/Australia	10

ANNEX I – RESULTS OF THE PRELIMINARY QUESTIONNAIRE REGARDING MULTI-ROBOT SYSTEMS



Number of Questionnaire Sent Out (red) and Number of Received Answers (blue).



Same Data in a More Detailed View.

The question “Is your work more hardware or software related?” gave the following result:

<u>R&D in Hard-/Software?</u>	%
only Software	25
mostly Software	28
equal	35
mostly Hardware	12

The question “Do you own robots or do you use simulation only?” gave the following result:

<u>Simulation or real?</u>	%
only robots	45
robots and simulation	38
only Simulation	5
manipulators	5
sensors	5
other	3

The question “Name five keywords that describe your work/research area. gave the following result:

Keyword	#
Robot(ic)	32
Control	16
Agent	14
Learning	10
Architecture	9
Co-operation	9
Communication	8
Behaviour	8
Autonomous	7
Intelligent	5
Navigation	3
HCI/HRI	3

ANNEX I – RESULTS OF THE PRELIMINARY QUESTIONNAIRE REGARDING MULTI-ROBOT SYSTEMS



The RTG agreed on the fact that it would be very helpful to design an improved questionnaire and redistribute it at the next workshop. This would help to get more and direct impressions from the community.

Annex J – MEASURES OF EFFECTIVENESS IN THE FIELD OF MULTI-ROBOT SYSTEMS

The following slides show the first approach to a set of measures of effectiveness for multi-robot systems.

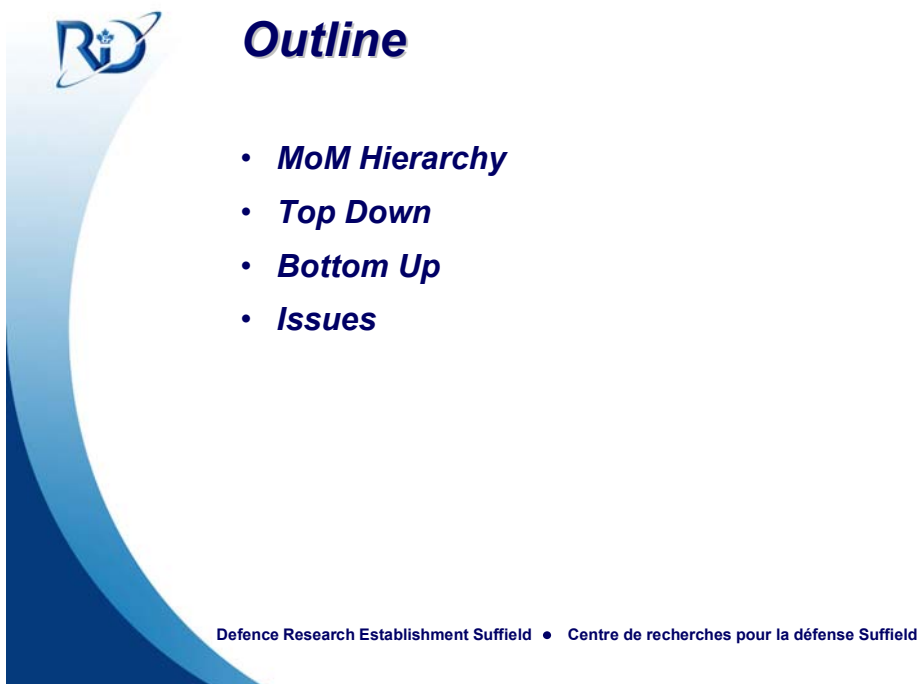



DEFENCE  DÉFENSE

Measures of Merit

DM Hanna, H/TVSS
T: 403-544-4833
C: 403-548-4587
Doug.Hanna@drdc-rddc.gc.ca

 Defence R&D Canada R et D pour la défense Canada 



 **Outline**

- **MoM Hierarchy**
- **Top Down**
- **Bottom Up**
- **Issues**

Defence Research Establishment Suffield • Centre de recherches pour la défense Suffield



Measures of Merit

- *MoPE – Measures of Policy Effectiveness*
- *MoFE – Measures of Force Effectiveness*
- *MoE – Measures of Effectiveness*
- *MoP – Measures of Performance*
- *DP – Dimensional Parameters*

Defence Research Establishment Suffield • Centre de recherches pour la défense Suffield



Analysis Goal

- *Purchase multi-robot system*
- *Select from several candidates*

Scenario Outline

- *Multi-ethnic society, civil strife*
- *Rugged terrain, poor comm's*
- *Population displaced*

Mandate

- *Destroy terrorist organization*

Defence Research Establishment Suffield • Centre de recherches pour la défense Suffield



Construct MoM's (↓)

<i>Tasks</i>	<i>MoM's</i>
Mandate <i>destroy al Queda</i>	MoPE <i>% of organization</i> <i>% assets destroyed</i>
Main Tasks <i>strike ops</i> <i>detain</i> <i>humanitarian assistance</i> <i>political stability</i>	MoFE <i># of sites</i> <i># of individuals</i> <i># of people helped/not elected government</i>

Defence Research Establishment Suffield • Centre de recherches pour la défense Suffield



Construct MoM's (↓)

<i>Tasks</i>	<i>MoM's</i>
Sub-tasks <i>establish presence</i> <i>locate target sites</i> <i>provide food</i>	MoE <i># of assets emplaced</i> <i># of sites</i> <i># of people fed/not fed</i> MoP <i>area searched by MRS</i> <i>tunnels cleared</i> DP <i>sensor sensitivity/range</i>

Defence Research Establishment Suffield • Centre de recherches pour la défense Suffield



Construct MoM's (↓)

Tasks	MoM's
Sub-tasks	DP
	<i>sensor sensitivity/range</i>
	MoP
	<i>area searched by MRS</i>
	<i>tunnels cleared</i>
Sub-tasks	MoE
	<i># of assets emplaced</i>
	<i># of sites</i>
	<i># of people fed/not fed</i>

Defence Research Establishment Suffield • Centre de recherches pour la défense Suffield



Measures of Merit

- Scenario based {
 - **MoPE – Measures of Policy Effectiveness**
 - **MoFE – Measures of Force Effectiveness**
- Capability based {
 - **MoE – Measures of Effectiveness**
 - **MoP – Measures of Performance**
 - **DP – Dimensional Parameters**

Defence Research Establishment Suffield • Centre de recherches pour la défense Suffield



Issues

- ***Conflicts between MoM's***
- ***Unclear relationships***
- ***Estimates required***
- ***Mission creep***
- ***Metrics / objective confusion***
- ***Standard MoM's ?***
- ***Scenario / capability based***

Defence Research Establishment Suffield • Centre de recherches pour la défense Suffield



REPORT DOCUMENTATION PAGE																					
1. Recipient's Reference	2. Originator's References	3. Further Reference	4. Security Classification of Document																		
	RTO-TR-IST-032 AC/323(IST-032)TP/191	ISBN 978-92-837-0043-2	UNCLASSIFIED/ UNLIMITED																		
5. Originator	Research and Technology Organisation North Atlantic Treaty Organisation BP 25, F-92201 Neuilly-sur-Seine Cedex, France																				
6. Title	Multi-Robot Systems in Military Domains																				
7. Presented at/Sponsored by	Final Report of Task Group IST-032/RTG-014.																				
8. Author(s)/Editor(s)	Multiple		9. Date December 2008																		
10. Author's/Editor's Address	Multiple		11. Pages 116																		
12. Distribution Statement	There are no restrictions on the distribution of this document. Information about the availability of this and other RTO unclassified publications is given on the back cover.																				
13. Keywords/Descriptors	<table style="width: 100%; border: none;"> <tr> <td style="width: 33%;">Adaptive systems</td> <td style="width: 33%;">Military operations</td> <td style="width: 33%;">Robotics</td> </tr> <tr> <td>Automation</td> <td>Mission profiles</td> <td>Robots</td> </tr> <tr> <td>Autonomous operation</td> <td>Multisensors</td> <td>Sensor characteristics</td> </tr> <tr> <td>Control equipment</td> <td>Remote control</td> <td>Systems engineering</td> </tr> <tr> <td>Design</td> <td>Remotely operated vehicles</td> <td>Tactical operations</td> </tr> <tr> <td>Land mine detection</td> <td>Reviewing</td> <td>Target acquisition</td> </tr> </table>			Adaptive systems	Military operations	Robotics	Automation	Mission profiles	Robots	Autonomous operation	Multisensors	Sensor characteristics	Control equipment	Remote control	Systems engineering	Design	Remotely operated vehicles	Tactical operations	Land mine detection	Reviewing	Target acquisition
Adaptive systems	Military operations	Robotics																			
Automation	Mission profiles	Robots																			
Autonomous operation	Multisensors	Sensor characteristics																			
Control equipment	Remote control	Systems engineering																			
Design	Remotely operated vehicles	Tactical operations																			
Land mine detection	Reviewing	Target acquisition																			
14. Abstract	<p>IST-032/RTG-014 aim was to consider the potential of modern multi-robot systems for the use in military domains. There has been substantial interest in the use of MRS for a variety of military purposes. A better understanding of human computer and/or robot interaction requirements and development of techniques for evaluating the effectiveness of MRS to meet military purposes will provide military users with a basis for determining their requirements for this technology. The group identified possible application areas for Human-Multi-Robot Systems in military domains:</p> <ul style="list-style-type: none"> • Reconnaissance, surveillance and target acquisition (RSTA); • Ordnance disposal, mine clearing operation, de-mining and NBC decontamination; • Security, defence and sniper discovery; • Tactical considerations like decoy, deception and precision strike; • Transport, convoying and rescue; and • Sensor and communication networks. 																				





BP 25

F-92201 NEUILLY-SUR-SEINE CEDEX • FRANCE
Télécopie 0(1)55.61.22.99 • E-mail mailbox@rta.nato.int



DIFFUSION DES PUBLICATIONS
RTO NON CLASSIFIEES

Les publications de l'AGARD et de la RTO peuvent parfois être obtenues auprès des centres nationaux de distribution indiqués ci-dessous. Si vous souhaitez recevoir toutes les publications de la RTO, ou simplement celles qui concernent certains Panels, vous pouvez demander d'être inclus soit à titre personnel, soit au nom de votre organisation, sur la liste d'envoi.

Les publications de la RTO et de l'AGARD sont également en vente auprès des agences de vente indiquées ci-dessous.

Les demandes de documents RTO ou AGARD doivent comporter la dénomination « RTO » ou « AGARD » selon le cas, suivi du numéro de série. Des informations analogues, telles que le titre et la date de publication sont souhaitables.

Si vous souhaitez recevoir une notification électronique de la disponibilité des rapports de la RTO au fur et à mesure de leur publication, vous pouvez consulter notre site Web (www.rto.nato.int) et vous abonner à ce service.

CENTRES DE DIFFUSION NATIONAUX

ALLEMAGNE

Streitkräfteamt / Abteilung III
Fachinformationszentrum der Bundeswehr (FIZBw)
Gorch-Fock-Straße 7, D-53229 Bonn

BELGIQUE

Royal High Institute for Defence – KHID/IRSD/RHID
Management of Scientific & Technological Research
for Defence, National RTO Coordinator
Royal Military Academy – Campus Renaissance
Renaissancelaan 30, 1000 Bruxelles

CANADA

DSIGRD2 – Bibliothécaire des ressources du savoir
R et D pour la défense Canada
Ministère de la Défense nationale
305, rue Rideau, 9^e étage
Ottawa, Ontario K1A 0K2

DANEMARK

Danish Acquisition and Logistics Organization (DALO)
Lautrupbjerg 1-5, 2750 Ballerup

ESPAGNE

SDG TECEN / DGAM
C/ Arturo Soria 289
Madrid 28033

ETATS-UNIS

NASA Center for AeroSpace Information (CASI)
7115 Standard Drive
Hanover, MD 21076-1320

FRANCE

O.N.E.R.A. (ISP)
29, Avenue de la Division Leclerc
BP 72, 92322 Châtillon Cedex

GRECE (Correspondant)

Defence Industry & Research General
Directorate, Research Directorate
Fakinos Base Camp, S.T.G. 1020
Holargos, Athens

HONGRIE

Department for Scientific Analysis
Institute of Military Technology
Ministry of Defence
P O Box 26
H-1525 Budapest

ITALIE

General Secretariat of Defence and
National Armaments Directorate
5th Department – Technological
Research
Via XX Settembre 123
00187 Roma

LUXEMBOURG

Voir Belgique

NORVEGE

Norwegian Defence Research
Establishment
Attn: Biblioteket
P.O. Box 25
NO-2007 Kjeller

PAYS-BAS

Royal Netherlands Military
Academy Library
P.O. Box 90.002
4800 PA Breda

POLOGNE

Centralny Ośrodek Naukowej
Informacji Wojskowej
Al. Jerozolimskie 97
00-909 Warszawa

PORTUGAL

Estado Maior da Força Aérea
SDFA – Centro de Documentação
Alfragide
P-2720 Amadora

REPUBLIQUE TCHEQUE

LOM PRAHA s. p.
o. z. VTÚLaPVO
Mladoboleslavská 944
PO Box 18
197 21 Praha 9

ROUMANIE

Romanian National Distribution
Centre
Armaments Department
9-11, Drumul Taberei Street
Sector 6
061353, Bucharest

ROYAUME-UNI

Dstl Knowledge and Information
Services
Building 247
Porton Down
Salisbury SP4 0JQ

SLOVENIE

Ministry of Defence
Central Registry for EU and
NATO
Vojkova 55
1000 Ljubljana

TURQUIE

Milli Savunma Bakanlığı (MSB)
ARGE ve Teknoloji Dairesi
Başkanlığı
06650 Bakanlıklar
Ankara

AGENCES DE VENTE

NASA Center for AeroSpace Information (CASI)

7115 Standard Drive
Hanover, MD 21076-1320
ETATS-UNIS

The British Library Document Supply Centre

Boston Spa, Wetherby
West Yorkshire LS23 7BQ
ROYAUME-UNI

Canada Institute for Scientific and Technical Information (CISTI)

National Research Council Acquisitions
Montreal Road, Building M-55
Ottawa K1A 0S2, CANADA

Les demandes de documents RTO ou AGARD doivent comporter la dénomination « RTO » ou « AGARD » selon le cas, suivie du numéro de série (par exemple AGARD-AG-315). Des informations analogues, telles que le titre et la date de publication sont souhaitables. Des références bibliographiques complètes ainsi que des résumés des publications RTO et AGARD figurent dans les journaux suivants :

Scientific and Technical Aerospace Reports (STAR)

STAR peut être consulté en ligne au localisateur de ressources
uniformes (URL) suivant: <http://www.sti.nasa.gov/Pubs/star/Star.html>
STAR est édité par CASI dans le cadre du programme
NASA d'information scientifique et technique (STI)
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
ETATS-UNIS

Government Reports Announcements & Index (GRA&I)

publié par le National Technical Information Service
Springfield
Virginia 2216
ETATS-UNIS
(accessible également en mode interactif dans la base de
données bibliographiques en ligne du NTIS, et sur CD-ROM)



BP 25

F-92201 NEUILLY-SUR-SEINE CEDEX • FRANCE
Télécopie 0(1)55.61.22.99 • E-mail mailbox@rta.nato.int



**DISTRIBUTION OF UNCLASSIFIED
RTO PUBLICATIONS**

AGARD & RTO publications are sometimes available from the National Distribution Centres listed below. If you wish to receive all RTO reports, or just those relating to one or more specific RTO Panels, they may be willing to include you (or your Organisation) in their distribution.

RTO and AGARD reports may also be purchased from the Sales Agencies listed below.

Requests for RTO or AGARD documents should include the word 'RTO' or 'AGARD', as appropriate, followed by the serial number. Collateral information such as title and publication date is desirable.

If you wish to receive electronic notification of RTO reports as they are published, please visit our website (www.rto.nato.int) from where you can register for this service.

NATIONAL DISTRIBUTION CENTRES

BELGIUM

Royal High Institute for Defence – KHID/IRSD/RHID
Management of Scientific & Technological Research
for Defence, National RTO Coordinator
Royal Military Academy – Campus Renaissance
Renaissancelaan 30
1000 Brussels

CANADA

DRDKIM2 – Knowledge Resources Librarian
Defence R&D Canada
Department of National Defence
305 Rideau Street, 9th Floor
Ottawa, Ontario K1A 0K2

CZECH REPUBLIC

LOM PRAHA s. p.
o. z. VTÚLaPVO
Mladoboleslavská 944
PO Box 18
197 21 Praha 9

DENMARK

Danish Acquisition and Logistics Organization (DALO)
Lautrupbjerg 1-5
2750 Ballerup

FRANCE

O.N.E.R.A. (ISP)
29, Avenue de la Division Leclerc
BP 72, 92322 Châtillon Cedex

GERMANY

Streitkräfteamt / Abteilung III
Fachinformationszentrum der Bundeswehr (FIZBw)
Gorch-Fock-Straße 7
D-53229 Bonn

GREECE (Point of Contact)

Defence Industry & Research General Directorate
Research Directorate, Fakinos Base Camp
S.T.G. 1020
Holargos, Athens

HUNGARY

Department for Scientific Analysis
Institute of Military Technology
Ministry of Defence
P O Box 26
H-1525 Budapest

ITALY

General Secretariat of Defence and
National Armaments Directorate
5th Department – Technological
Research
Via XX Settembre 123
00187 Roma

LUXEMBOURG

See Belgium

NETHERLANDS

Royal Netherlands Military
Academy Library
P.O. Box 90.002
4800 PA Breda

NORWAY

Norwegian Defence Research
Establishment
Attn: Biblioteket
P.O. Box 25
NO-2007 Kjeller

POLAND

Centralny Ośrodek Naukowej
Informacji Wojskowej
Al. Jerozolimskie 97
00-909 Warszawa

PORTUGAL

Estado Maior da Força Aérea
SDFA – Centro de Documentação
Alfragide
P-2720 Amadora

ROMANIA

Romanian National Distribution
Centre
Armaments Department
9-11, Drumul Taberei Street
Sector 6
061353, Bucharest

SLOVENIA

Ministry of Defence
Central Registry for EU and
NATO
Vojkova 55
1000 Ljubljana

SPAIN

SDG TECEN / DGAM
C/ Arturo Soria 289
Madrid 28033

TURKEY

Milli Savunma Bakanlığı (MSB)
ARGE ve Teknoloji Dairesi
Başkanlığı
06650 Bakanlıklar – Ankara

UNITED KINGDOM

Dstl Knowledge and Information
Services
Building 247
Porton Down
Salisbury SP4 0JQ

UNITED STATES

NASA Center for AeroSpace
Information (CASI)
7115 Standard Drive
Hanover, MD 21076-1320

SALES AGENCIES

**NASA Center for AeroSpace
Information (CASI)**

7115 Standard Drive
Hanover, MD 21076-1320
UNITED STATES

**The British Library Document
Supply Centre**

Boston Spa, Wetherby
West Yorkshire LS23 7BQ
UNITED KINGDOM

**Canada Institute for Scientific and
Technical Information (CISTI)**

National Research Council Acquisitions
Montreal Road, Building M-55
Ottawa K1A 0S2, CANADA

Requests for RTO or AGARD documents should include the word 'RTO' or 'AGARD', as appropriate, followed by the serial number (for example AGARD-AG-315). Collateral information such as title and publication date is desirable. Full bibliographical references and abstracts of RTO and AGARD publications are given in the following journals:

Scientific and Technical Aerospace Reports (STAR)

STAR is available on-line at the following uniform resource
locator: <http://www.sti.nasa.gov/Pubs/star/Star.html>
STAR is published by CASI for the NASA Scientific
and Technical Information (STI) Program
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
UNITED STATES

Government Reports Announcements & Index (GRA&I)

published by the National Technical Information Service
Springfield
Virginia 2216
UNITED STATES
(also available online in the NTIS Bibliographic Database
or on CD-ROM)