Mário Jorge Rodrigues da Silva Nunes Salgueiro

HUMAN-ROBOT TEAMWORK: A KNOWLEDGE-BASED SOLUTION

Lisboa
2008

UNIVERSIDADE NOVA DE LISBOA

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica e de Computadores

HUMAN-ROBOT TEAMWORK: A KNOWLEDGE-BASED SOLUTION

Mário Jorge Rodrigues da Silva Nunes Salgueiro

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores.

Orientador: Prof. José Barata

Lisboa
2008

NEW UNIVERSITY OF LISBON

Faculty of Sciences and Technology

Electrical and Computer Engineering Department

HUMAN-ROBOT TEAMWORK:

A KNOWLEDGE-BASED SOLUTION

Mário Jorge Rodrigues da Silva Nunes Salgueiro

Dissertation presented at the Faculty of Sciences and Technology of the New University of Lisbon to obtain the Master degree in Electrical and Computer Engineering.

Supervisor: Prof. José Barata

Lisboa

2008

Acknowledgements

This dissertation has been supported by the Portuguese SME company IntRoSys, S.A [1].

First, I would like to express my gratitude to my supervisor, Prof. José Barata, for his many suggestions and constant support and motivation during this work. My sincere appreciation to my colleagues at IntRoSys for all the relevant comments and friendship; namely, Carlos Cândido, Luis Almeida, Paulo Santos, Vasco Santos and mainly Pedro Santana who I would like to give a very special thanks for all his support and motivation.

I would like to underline how important it was for me all the love and support given by my girlfriend Ana, who I much care and love, without her support, it would have been too much difficult to have successfully achieved this endeavour. I would also to thank my lovely mother and friend Domitilia, for all her love and support, my father and adviser Anibal, my best friend and brother Luis and his adorable wife Neuza. I would also like to thank my girlfriend's parents, Maria and Álvaro, for all their support during this important moment of my life. To finish all the acknowledgements, I want to express how great is to be friend of João Leandro, and how his friendship helped me during this dissertation.

[1] www.introsys-online.com

Sumário

Equipas compostas por humanos e robôs impõem novos desafios ao conceito de trabalho em equipa. Isto provém do facto de robôs e seres humanos terem capacidades sensorimotoras significativamente diferentes. Esta dissertação contribui para a solução deste problema propondo um sistema multi-agentes baseado em conhecimento, para suportar o desenho e execução de trabalho em equipa estereotipado, (isto é, recorrente). O formalismo de *fluxo de trabalho cooperativo* é utilizado para especificar planos de equipa, e adaptado para permitir a partilha de informação entre actividades em execução. Esta nova funcionalidade permite interacções fortemente acopladas entre os membros da equipa, algo essencial quando entidades incorporadas precisam de interagir.

Desta forma, ao invés de se focar no (re) planeamento automático do trabalho em equipa, esta dissertação propõe uma solução baseada em conhecimento, complementar e intuitiva, para rápida distribuição e adaptação para equipas de humanos e robôs de pequena escala.

Através de uma ferramenta gráfica, o coordenador da equipa tem a possibilidade de visualizar o estado global da missão, e.g. que actividade cada membro da equipa está a executar. Um assistente pessoal, i.e. um agente computacional, informa os operadores humanos das actividades em que se encontram, assim como das suas obrigações uns para com os outros.

Resultados experimentais obtidos em ambientes de simulação, assim como com uma equipa composta por um robô real e dois operadores, demonstraram as capacidades do sistema em trabalhos estereotipados em equipas compostas por humanos e robôs.

Abstract

Teams of humans and robots pose new challenges to the teamwork field. This stems from the fact that robots and humans have significantly different perceptual, reasoning, communication and actuation capabilities. This dissertation contributes to solving this problem by proposing a knowledge-based multi-agent system to support design and execution of stereotyped (i.e. recurring) human-robot teamwork. The cooperative workflow formalism has been selected to specify team plans, and adapted to allow activities to share structured data, even during their execution. This novel functionality enables tightly coupled interactions among team members.

Rather than focusing on automatic teamwork planning, this dissertation proposes a complementary and intuitive knowledge-based solution for fast deployment and adaptation of small scale human-robot teams.

In addition, the system has been designed in order to improve task awareness of each mission participant, and of the human overall mission awareness.

A set of empirical results obtained from simulated and real missions proved the concept and the reusability of such a system. Practical results showed that this approach used is an effective solution for small scale teams in stereotyped human-robot teamwork.

Acronyms

Acronym	Description
ACL	Agent Communication Language
API	Application Programming Interface
BDI	Beliefs, Desires and Intentions
BPEL	Business Process Execution Language
FA	Flexible Awareness
GRATE	Generic Rules and Agent model Testbed Environment extended version
GUI	Graphical User Interfaces
HRI	Human-Robot Interaction
JADE	Java Agent DEvelopment framework
JaWE	Java Workflow Editor
JDK	Java Development Kit
KB	Knowledge Base
KBS_HRT	Knowledge-Based System for Human-Robot Teamwork
MAS-IM	Multi-Agent System Interaction Mechanism
ME	Mission Execution
OWL	Web Ontology Language
RDF	Resource Description Framework

Acronym	Description
SA	Stereotyped Awareness
STEAM	a Shell for TEAMwork
TO	Tele-Operation
TWE	Together Workflow Editor community edition
TWE*	Together Workflow Editor community edition extended
UML	Unified Modeling Language
WFDM	WorkFlow Design and Monitoring
WfMC	Workflow Management Coalition
XML	eXtensible Markup Language
XPDL	XML Process Definition Language

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The motivation underlying this dissertation is the development of a system to enable human-robot teamwork in stereotypes tasks, such as: humanitarian demining, surveillance and search & rescue missions.

After several years of intensive research, robots are finally emerging out of the semi-structured office and shop-floor environments. Robots can now help humans in the execution of life threatening tasks, such as surveillance, search & rescue, and demining, which are unstructured environments by nature. In these demanding scenarios, where human operators are overwhelmed with information and under a considerable amount of stress, robots can be used to reduce the need of human presence in these dangerous applications.

However, the challenges found by the researchers when they try to include robots as teammates are so big that the usually assumed simple issue of deploying a tele-operated robot for the accomplishment of a mission is a daunting task. This idea is supported by the study developed by [Carlson and Murphy, 2005], where it was shown that one can not expect current state-of-the-art field robots to operate without failures between 6 and 20 hours in the search & rescue domain. A failure is the inability of the robot or the equipment used with the robot to function in accordance with the plan. Because of that a simple method to discover robots which are damaged or can not perform his task could be a good way to improve teamwork.

Research on teamwork has been carried out in order to minimize these problems [Tambe, 1997,

Scerri et al., 2003, Sierhuis et al., 2005, Nourbakhsh et al., 2005, Sycara and Sukthankar, 2006]. Most of the solutions grew from work on multi-robot and multi-agent systems properly adapted to include humans. This dissertation proposes to see the problem also from another perspective, i.e. to include robots and software agents as participants on human-centred operational procedures, supported by knowledge management concepts usually employed in human organisations. Both views are complementary rather than mutually exclusive.

These human-centred operational procedures are normally knowledge intensive tasks, which can be approximately represented by a set of templates. Knowledge engineering methodologies [Sierhuis et al., 2005] can be used to grasp and formalise domain experts knowledge into the form of templates. These templates, which in the case of teamwork may also be called team-plans, need to be adapted according to the situation. In this case, much of the work on automatic teamwork (re)planning could be employed [Sycara and Sukthankar, 2006].

High granularity script based solutions (i.e. scripts very specific in a certain domain) are better to solve this problem [Murphy et al., 2001] because they are excellent when optimal solutions are pursued. But the end user is most certainly an operator with little knowledge on robotics, hindering the use of complex scripting languages for inter-agent interactions specification.

Other work on embodied human-robot teams, such as described in [Long et al., 2005] and [Sukhatme et al., 2001], do not report any sort of intuitive workflow-based solution which operators can configure, monitor, and alter the system in real-time over a multi-agent system platform.

A workflow which is a sequence of operations, performed by one or more persons, a simple or complex mechanism or machines. A workflow is a model to represent real work for further assessment (e.g. for describing a reliably repeatable sequence of operations). Workflows are used in computer programming to capture and develop human-machine interaction. It is very similar to flow charts but without the need to understand computers or programming, because all information is visually available.

So it is necessary to allow humans to perform such adaptation, leading to the need of a

visual interface. It can never be expected that graphical-based solutions optimally represent the interests of a plan designer. However, the goal is to have a general purpose system which evolves with operation utilisation. Since some mission templates become stable and requiring higher optimality, system designers may consider to code them and make them available as a new black-box for the operator.

It is important to note that this work does not provide an automated planning mechanism like is usually used in the theoretical teamwork field. But for future improvements is essential the knowledge learned from that field of investigation. For example, adjustable autonomy which is a fundamental aspect for a proper adaptation of the system in runtime. Adjustable autonomy can be based on the principle of a transfer-of-control strategy [Scerri et al., 2003]. The transfer-of-control strategy makes possible, firstly transfer of the decision-making control (e.g. from an agent to a user or vice-versa). Secondly the change of an agent's pre-specified coordination constraints with team members, rearranging activities as needed (e.g. reordering tasks to save time to make the decision). This strategy attempts to provide the highest quality decision, while avoiding coordination failures.

In [Scerri et al., 2003, Sycara and Sukthankar, 2006] it is mentioned the difficulties of pre-defined plans to properly address real problems as they only allow to cope with expectable events. Another difficulty is that autonomous task allocation and scheduling are also faulty, in particular in complex dynamic environments. Model-free solutions, such as those found in swarm intelligence models [Bonabeau et al., 1999] are of particular interest when information is mainly of local nature, but their outcome is hardly predictable. Model-based solutions, as those found in the theoretical teamwork field [Sycara and Sukthankar, 2006] typically require a shared mental state among all teammates, therefore global information, but the outcome can be optimal and predictable. You may consider for instance that most of the clues to detect and solve exceptions to the plan in complex situations are simply not observable without complex tacit human knowledge, which is continuously evolving while the mission unfolds. Bearing this in mind, the workflow formalism has been adopted as a direct mapping of the plan and its visual representation. In addition, workflows are a common formalism to represent activities

in human organisations (e.g. [Camarinha-Matos, 2003]), thus providing easier integration of software agents and mobile robots in knowledge intensive tasks, where human participation is paramount.

Another benefit of a knowledge-based component is that actual human-robot teams encompass few elements, meaning that the cost of having a human taking care of major strategic decisions for the entire team is possibly less than having the system taking uninformed strategic decisions. This ideological stance is relevant to concretely define the role of humans, robots, and software agents in the decision making.

Amongst the several questions puzzling researchers when teaming humans and robots, it is the problem of human-robot interaction awareness which is becoming more and more popular. The fact that, in average, 30% of operators time is spent (re-)acquiring awareness [Yanco and Drury, 2004], shows its relevance to human-robot teamwork.

Moreover, our daily experience is a rich source of information about the way humans react as a team to complex situations. Hence, its analysis can only be fruitful for the problem being addressed. When joining a team, humans usually follow procedures, i.e. successful stereotyped courses of action, allowing them to collaborate with minimal communication and negotiation requirements. This allows humans to act promptly to situations even in the absence of relevant information. Thus, providing awareness information to team members is done more efficiently when an underlying procedure, i.e. mission plan, is considered.

This knowledge-based component provide the human with visual description of the mission status, which is essential to improve its *Humans' overall mission awareness*, which is a particularly interesting topic of the broader concept of *Human-Robot Interaction (HRI) awareness* [Drury et al., 2003]. Its relevance stems from the fact that it is not solvable by typical map-centric and video-centric interfaces [Drury et al., 2007]. Due to the human-centred nature of life threatening domains, such as surveillance, search & rescue, or humanitarian demining, efficient mechanisms to foster human's overall mission awareness is paramount.

However, since in dynamic environments, stereotyped plans eventually fail in unpredictable ways. Team members must be provided with mechanisms to deal with the situation in a flexible

way, i.e. without the constraints of an underlying plan. For the particular case of a human team member, an easily deployable toolbox that dynamically links to other teammates in order to fetch state information is fundamental.

In this dissertation the team execution plan is seen as a specification of a knowledge intensive task that should be obtained directly from expert users via knowledge acquisition processes.

1.1 Dissertation Outline

This dissertation is organised as follows:

Chapter 1 the remaining of this chapter introduces the reader to the problem of human-robot teamwork and awareness. A complementary and intuitive knowledge-based solution for fast deployment and adaptation of small scale teams operating in life threatening tasks is presented.

Chapter 2 presents the state-of-the-art on human, agent, human-agent, human-robot teamwork.

Chapter 3 presents the knowledge-based component for the Human-Robot teamwork problem.

Chapter 4 presents the case studies using the knowledge-based component in a humanitarian demining and surveillance tasks. Experimental results and a briefly discussion about them are also presented here.

Chapter 5 discusses the results of the knowledge-based component and the main contributions of this dissertation. Further research opportunities on the subject are also described.

1.2 Problem Statement: Human-Robot Teamwork in Stereotyped Tasks

As stated, the goal of this dissertation is to develop a system to enable human-robot teamwork in stereotyped tasks, such as, humanitarian demining, surveillance and search & rescue missions.

The major questions related to this topic are:

1. How to include robots as team members on human-centred operational procedures? Solving this problem, the mission performance can be improved and the number of human operators in dangerous applications, such as the ones described above, can be reduced.

2. Can a mission be specified in a simple way? That can be done with an intuitive solution in which an user can design the mission plan, even with little knowledge on programming languages.

3. How a team member knows what is his role in the mission? Team member plan has the activities to be done and what are the interactions among team members, without the needing of each one knows all about all the others team members.

4. How to adapt a mission? Once more, with an intuitive solution, which allows fast mission adaptation, leading the mission plan to evolve from the experience acquired on mission field.

5. How to enhance the operators' overall awareness of the mission? A monitoring tool that gives the current state of the mission, trying to fill the gap not solved by typical map-centric and video-centric interfaces which do not provide overall mission awareness.

1.3 Proposed Solution

This dissertation presents an architecture for teamwork in stereotyped tasks, the Knowledge-Based System for Human-Robot Teamwork (KBS_HRT) (see Figure 1.1). KBS_HRT core is

domain knowledge acquired from domain experts and applied in the coordination of human-robot teams to augment their *awareness*.

This *domain knowledge* must be acquired, formalised, adapted, and employed in the actual coordination of the team members performing a mission. For now, it is sufficient to know that such knowledge is mainly in the form of *mission templates*, which implement a generic team plan for a given task. Roughly, a mission template is specified in terms of a workflow, in which several teammates, i.e. participants, execute their activities while exchanging messages asynchronously. For example, a robot may be performing a *goto-xy* activity while being guided by a human operator involved in a *guide-robot* activity.

The proposed knowledge-based teamwork approach is composed of four not necessarily sequential steps, namely:

Mission template specification. Mission templates are knowledge intensive tasks specifications, i.e. domain knowledge maintained in a knowledge base supported by a well specified ontology. Mission templates are specified by domain experts with workflows through a tool for workflow design. This step refers to the typical knowledge extraction and formalisation stages considered in the knowledge engineering discipline.

Mission template instantiation. In the mission template nothing is said about which individual (e.g. robot, human) will play the role of a given participant in the team. After selecting and adapting the mission template, the human operator recruits individuals to be part of the team. This process is supported by a multi-agent system. Each individual is represented by *mission execution proxy agent*, which is registered in a yellow pages service. A middle agent is used to lookup for individuals able to fulfil the role of a given participant in a specific mission. Then each participant receives the part of the workflow necessary to know how to accomplish his role in the mission, making then which one ready to execute that.

Mission execution. After receiving the request to start the mission, each *mission execution proxy agent* follows the part of the workflow corresponding to its participant. As it was

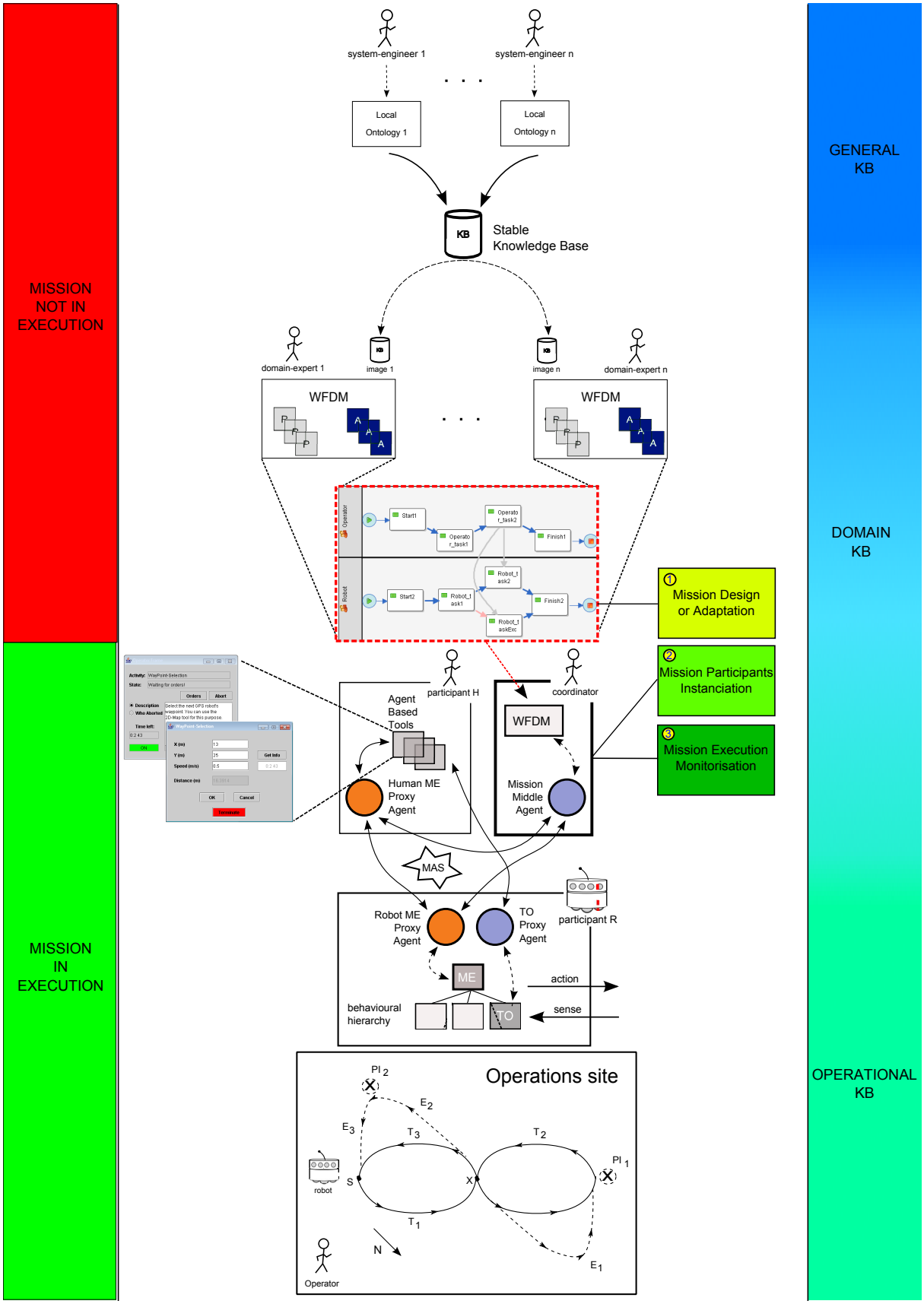

Figure 1.1: Knowledge-Based System for Human-Robot Teamwork Overview

introduced above, in this process software agents exchange messages so that participants can act in a coordinated way. This is a fully distributed execution in a sense that each participant has its own *mission execution proxy agent* concerned with a limited part of the whole mission template, and it only exchanges messages with participants where is an explicit link in the mission template.

Mission template adaptation. On field, human operators select mission templates according to the context of the operations. Adaptations to the mission template (i.e. workflow) are expected and performed by the operator through the same tool used by domain experts to specify the mission template. In this process it is expected that some adapted mission templates are added to the mission templates library for further reuse, allowing the knowledge base to grow as the field experience evolves.

1.4 Further Readings

Some of the concepts covered in this dissertation have been published in [Santana et al., 2008c]. Others belong to a technical report [Santana et al., 2008b], mainly the HRI awareness concepts. As it was said before, this dissertation is part of a project supported by the company IntRoSys, S.A.. To know more about this project which aims the development of affordable service robots for humanitarian demining, see also these recent publications [Santana et al., 2007] and [Santana et al., 2008a].

Chapter 2

State-of-the-Art

This chapter surveys the state-of-the-art on Human Teamwork, Agent Teamwork, Human-Agent Teamwork and Human-Robot Teamwork with some emphasis on Human-Robot Interactions Awareness.

Understanding Human Teamwork (c.f. Section 2.1) fosters the comprehension of what is a team and teamwork. Agent and Human-Agent Teamwork (c.f. Sections 2.2 and 2.3) help understanding how agents can be useful to enhance and support the human-robot teamwork. In Section 2.4 some of Human-Robot Teamwork works is addressed, with some emphasis in the Human-Robot Interactions Awareness.

2.1 Human Teamwork

The aim of this section is to answer the following questions:

- What is a team?
- What is teamwork?

Since the goal is to have robots belonging to human teams, and not to include humans in robot teams, these questions must be answered taking into account human teamwork characteristics (refer to [Paris et al., 2000] for a thorough review on the field).

2.1.1 What is a Team?

Teams are more than collections of individuals. One cannot simply label a group of individuals a team and expect that they will perform as a team [Bass, 1980].

In [Salas et al., 1992], a team is conceived as a "distinguishable set of two or more people who interact dynamically, interdependently, and adaptively toward a common and valued goal/objective/mission, who have each been assigned specific roles or functions to perform, and who have a limited life-span membership".

In [Paris et al., 2000] some of the characteristics that make a team to be more than a collection of individuals were described. In this dissertation that characteristics are specified and explained, and they are the following: *task interdependencies*; *team members coordination*; *specialised members roles and responsibilities*; *task relevant knowledge*; *existence of shared goal*; *communication among team members*; *adaptive strategies to respond to change*; *information redundancy*.

Task interdependencies means what information each task has to provide and/or to receive from the others tasks (e.g. some input parameters of a task to be updated by others tasks output parameters). This characteristic is very related with *team members coordination*, i.e. synchronisation of tasks and the information that each team members need to share to increase the team performance in the mission. *Specialised members roles and responsibilities* relate to the ability of a team member to perform specific role, i.e. a task can only be allocate to a member which can do it, and to each role a responsibility can be associated (e.g. if a member with high responsibility fails, the mission is aborted). Strictly connected to this last characteristic, *task relevant knowledge* is the knowledge about the task which the team member needs to perform that task, i.e. only a specialised member knows what information needs to perform a specific task. *Existence of shared goals*, i.e. teammates need to share a state of affairs which they will going to achieve or obtain, and these goals can be parts of a full mission. *Communication among team members* requires a common language (e.g. a person from Japan and other from Portugal need a common language to understand each other). This feature improves the team performance because without that almost all the others team characteristics could not be realised, at least

with efficacy. These communications can be very intensive when used to tightly coupled interactions among team members. *Adaptive strategies to respond to change* relates to the allocation of new team members and associate tasks to them or reallocate new tasks to team members or even both, in a mission. *Information redundancy* relate to availability of multiple sources of information which can lead the teammate to filter that information. This characteristic increases robustness in the task accomplishment, but at the same time, it means more communication between teammates and more memory capacity to keep that information.

In analogy, human-robot teams require common goals which can be part of a major goal, and to accomplish that, team members will have specific roles (e.g. a human can be a supervisor or an operator, and a robot can be surveillance-robot or demining-robot) to perform, which only they know how to do. To perform their roles they need a common language (e.g. a language defined in an ontology) to communicate, allowing information exchange about activities (their parameters and state) between teammates, enabling different types of interactions. The relevant knowledge to complete a task needs to be specified somewhere (e.g. ontology). In a robot, it is the minimum information which is needed to perform their behaviours. Information redundancy will allow more robustness in human-robot teams, as in human teams. For that robots require more memory and processor speed. Adaptive strategies to respond to a mission change and shared commons goals, lead us to a share mental state among humans and robots. But the fact that robots and humans have significantly different perceptual, reasoning, communication and actuation capabilities, can lead us to see the problem from a different point of view. A human can be as a team manager (e.g. as a football manager), defining strategies and thinking about humans and robots goals, along the mission. This human is a domain expert, i.e. a human which knows very well a specific mission domain. This domain expert knows where humans and robots will act and their capabilities, join them in a better way. Robots and humans probably can learn from the decisions made by these domain experts, in order to allow automatic mission planning mechanisms, in the future.

2.1.2 What is Teamwork?

Teamwork is more than aggregate the individual behaviours of team members. Teamwork is the cooperative effort between the members of a group or team to achieve a common goal. It is how these groups function to produce effective synchronized output.

The most representative theories about what is teamwork in human teams have been studied by psychologists since 1950's. Teamwork is the seamless integration of specific cognitive, behavioural and affective competencies that allow team members to adapt and optimize their performance. These three dimensions are resume below [Cannon-Bowers et al., 1995]:

1. Cognition or knowledge category includes information about the task such as as team mission, objectives, norms, problem models, and resources;

2. Teamwork skills include behaviours such as adaptability, performance monitoring, leadership, communication patterns, and interpersonal coordination;

3. Attitudes measure the participants' feelings about the team: team cohesion, mutual trust, and importance of teamwork.

These are the three dimensions which humans and robots should have. Then to understand how humans and robots can interact as a team, it is important to describe some the most significant theories in agent teamwork and human-agent teamwork. After that will be easier to understand how software agents can support human-robot teamwork.

2.2 Agent Teamwork

Two widely accepted formalisms are used in Agent Teamwork, Joint Intention theory
[Cohen and Levesque, 1991] and SharedPlans theory [Grosz and Kraus, 1998]
[Grosz and Kraus, 1996].

Joint intentions can be viewed as a joint commitment to perform a collective action to achieve a certain joint goal. There exists a notion of joint mental attitude which is based on

the concept of joint persistent goal. This theory offers a framework for studying numerous teamwork issues and a guide for implementing multi-agent systems. An example of agent teamwork which use the notion of joint intentions at its core was GRATE* (Generic Rules and Agent model Testbed Environment extended version) [Jennings, 1995].

In SharedPlans, the concepts of individual and shared plans are formalised. Here it is explored how a shared plan for a team action evolves from partial, possibly with partial recipe, to a final complete form with complete recipes for the team action and all the subsidiary actions at different levels of abstraction. A toolkit which is based in this formalism is Collagen [Rich and Sidner, 1997]. In the evolution of SharedPlans, team members may only possess different partial views of the tree, although the common recipe tree becomes complete from external. In pursuing their common goals, it is the shared plans that ensure team members to cooperate smoothly rather than prohibiting each other's behaviour, which may occur otherwise due to the partial views of the common recipe tree.

STEAM [Tambe, 1997] is an example of a hybrid teamwork model using both formalisms, trying to achieve a better teamwork performance.

From theoretical work on agent teamwork, team behaviour has the following features:

1. Agents need to share the goals and an overall plan, which they must follow in a coordinated way. In some cases, partial knowledge of the environment must also be shared to enhance "situation awareness".

2. Agents need to share the intention to execute the plan to reach the common goal.

3. Team members must be aware of their capabilities and how they can fulfil roles required by the team high level plan.

4. Team members should be able to monitor their own progress toward the team goal and monitor team mates activities and team joint intentions.

These teamwork features were used by many systems and have been successfully implemented. For example, it was used to support human collaboration [Chen and Sycara, 1998] or

for disaster response [Nair et al., 2003].

All the features described above lead for the need of a share mental model.

Mental models are knowledge structures, cognitive representations or mechanisms which are use to organize new information, to describe, explain and predict events, as well as to guide team members interactions. A team mental model would reflect the dependencies or interrelationships between team objectives, team mechanisms, temporal patterns of activity, individual roles, individual functions, and relationships among individuals.

Shared mental models allow team members to implicitly and more effectively coordinate their behaviours, i.e. they allow a better recognition of individual responsibilities and information needed by teammates, activities monitor, diagnose deficiencies, and provide support, guidance, and information as needed.

However, avoiding the use of a share mental state and using a predetermined plan has the advantage that each team member knows *exactly* what is its role and what is the information that it needs to share among others team members, during the mission. This will reduce mainly, the data transfer and time of communication between teammates. Another advantage is the avoidance of inconsistent share mental state, due to a wrong or different mental state of the environment by teammates, which can lead to bad decisions by team members.

There are other examples of architectures using software agents for teamwork, as [Pechoucek et al., 2000], [Barata and Camarinha-Matos, 2004] and [Barata, 2005], and these works are about coalitions. In these works, the cooperation among software agents is strong, and these agents may represent machines, persons or entities.

2.3 Human-Agent Teamwork

Although both human teamwork and software agent teamwork have been thoroughly studied, there is little research on hybrid teamwork, i.e. human-agent teamwork. The different roles that agents can play in hybrid human-agent teams are the following [Sycara and Lewis, 2002]:

1. Supporting individual human team members in completion of their own tasks. These

agents often function as personal assistant agents and are assigned to specific team members (e.g. [Chalupsky et al., 2001]). Task-specific agents utilized by multiple team members (e.g. [Chen and Sycara, 1998]) also belong in this category. In both cases the agents help humans to complete their tasks.

2. Supporting the team as a whole, rather than focusing on task-completion activities, these agents directly facilitate teamwork by aiding communication, coordination among human agents, and focus of attention.

3. Assuming the role of ”virtual human” (i.e. a software agent capable to play the role of a human) within the organization, (e.g. [Traum et al., 2003]).

Because it is difficult to create a software agent that is as effective as a human at task performance and teamwork skills, acting as a ”virtual human” is the hardest role in hybrid teamwork. Experimental results in [Sycara and Lewis, 2002], suggest that agents supporting the team as a whole might be the most effective role for agents in hybrid teams.

Other research work investigated *trust* concepts as the fundamental building block for effective human-agent teamwork [Lenox et al., 2000] and the types of shared knowledge that promote mutual understanding between cooperating humans and agents [Okamoto et al., 2006].

However, many of the facets of human-agent teamwork models, such as communication protocols for forming mutual intelligibility, performing team monitoring to assess progress, forming joint goals, addressing task interdependencies have received little attention.

2.4 Human-Robot Teamwork

Robots are becoming more and more useful helping humans in a larger number of tasks. In demanding scenarios, such as surveillance, search & rescue, and demining, where human operators are overwhelmed with information and under a considerable amount of stress, robots can be used to reduce the need for human presence in these dangerous applications.

However, even the simple task of operating a tele-operated robot to accomplish a mission, is by itself already a great challenge. This is supported by the study [Carlson and Murphy, 2005], where it is stated that current state-of-the-art of field robots in the search & rescue domain, cannot operate without failures between 6 and 20 hours.

Most of the approaches for teamwork exist in result from work on multi-robot and multi-agent systems properly adapted trying to include humans. Several distributed behaviour-based architectures (e.g. [Parker, 1998] and [Murphy et al., 2001]), presented good solutions in multi-robots systems. But when we try to include humans in a system like this, robots cannot explain their actions and their role as a team member because normally they do not have an explicit teamwork model and then a teamwork plan cannot be easily defined for human understand.

On the other hand, some distributed agent-based architectures, as [Sierhuis et al., 2005] and [Nourbakhsh et al., 2005], were implemented and achieved good solutions for human-robot teamwork. The software agents presented in these architectures play some of the roles mentioned in section 2.3.

[Sierhuis et al., 2005] is a distributed agent-based architecture for human-robot teamwork which integrates diverse mobile entities, in lunar and planetary surface operations. The software agents are implemented using Brahms which is a BDI(Beliefs, Desires and Intentions) multi-agent modelling and simulation environment that facilitates understanding and configuring the cooperation between people and systems. However, Brahms use an activity-based approach. Brahams agents are both deliberative and reactive. Each Brahms model has three types of software agents, Personal, Task and Communication agents. *Personal agents* assist an autonomous external entity (person or robot) in performing their activities. *Task agents* assist *Personal agents* in executing low-level tasks associated with a particular activity. *Communication agents* agentify external systems, the others agents can communicate with an external system as if communicating with another agent. [Nourbakhsh et al., 2005] is an agent-based architecture, too, which uses four types of agents. *Interface agents* which facilitate user interaction. *Task agents* seek to accomplish user goals. *Middle agents* provide infrastructure for dynamic runtime discovery of robots and agents that can perform a given task. *Information*

agents can access various external information sources. As it was said in the introduction of this section, field robots still have high failure rates, during a mission. Because of that, it is very important to find an easy way to find out robots that are damaged or can not perform its task.

However, none of these architectures report any sort of intuitive solution in which humans can configure, monitor, and alter the mission in real-time.

The possibility of designing a mission in real-time is very important, because most of the clues to detect and solve exceptions to the plan in complex situations are simply not observable without complex tacit human knowledge, which is continuously evolving while the mission unfolds. Then, the mechanism for mission configuration and adaptation has to be intuitive and fast, in order to support the end user which is most certainly an operator with little knowledge on robotics.

Mission monitoring is also very important to improve the understanding of the progress of the mission. This can be used, for example, to know when and why exceptions occurred during a mission. This feature leads us to a specific area in Human-Robot Teamwork field, which is the Awareness in Human-Robot Interactions. More specifically, to one of the five components of HRI awareness, which is the Humans' overall mission awareness.

In the field of HRI, when n humans and m robots work together on a synchronous task, HRI awareness consists in the next five components [Drury et al., 2003]:

1. Human-robot: the understanding that the humans have of the locations, identities, activities, status and surroundings of the robots. In addition, the understanding of the certainty with which humans know the aforementioned information, must also be taken into account.

2. Human-human: the understanding that humans have of the locations, identities and activities of their fellow human collaborators.

3. Robot-human: the robots' knowledge of the humans' commands needed to a specific activity and any human-delineated constraints that may require command non-compliance or a modified course of action.

4. Robot-robot: the knowledge that the robots have of the commands given to them, if any, by other robots, the tactical plans of the other robots, and the robot-to-robot coordination necessary to dynamically reallocate tasks among robots if necessary.

5. Humans' overall mission awareness: the humans' understanding of the overall goals of the joint human-robot activities and the measurement of the moment-by-moment progress obtained against the goals.

From these HRI types of awareness, the Humans' overall mission awareness is the one requiring further work. According to [Drury et al., 2007], that component of HRI awareness is not solvable by typical map-centric and video-centric interfaces, which are the typically used. This component is seen as an important aspect of HRI awareness for the mission performance improvement, specifically in stereotyped missions, whereby the global or overall perspective can be very helpful to supervise the mission execution and to do its configuration.

Despite the importance of the previous point all the other awareness components are also relevant since in average, 30% of the operator time is spent (re-)acquiring awareness [Yanco and Drury, 2004], showing its relevance in human-robot teamwork. Because HRI awareness is so relevant in human-robot teamwork, methods and solutions to solve this problem became of paramount importance.

Chapter 3

Knowledge-Based System for Human-Robot Teamwork

This chapter presents the main contribution of this dissertation, the Knowledge-Based System for Human-Robot Teamwork (KBS_HRT). In short, the KBS_HRT core is the *domain knowledge*, which must be acquired, formalised, adapted, and employed in the *coordination* of human-robot teams augmenting their *awareness* in order to improve the mission performance. This *domain knowledge* is represented by mission templates. A mission template is specified using a workflow, whereby several teammates, i.e. participants, execute activities while exchanging messages asynchronously. The mission template or workflow implements a generic team plan for a given mission. It is a generic team plan because the real mission performers are not assigned yet. Section 3.2 and 3.4 has a detailed explanation about how team plans can be specified and executed, respectively.

Figure 3.1 presents the KBS_HRT architecture, which is composed of four not necessarily sequential steps and they are the following:

- Mission template specification;
- Mission template instantiation;
- Mission execution;

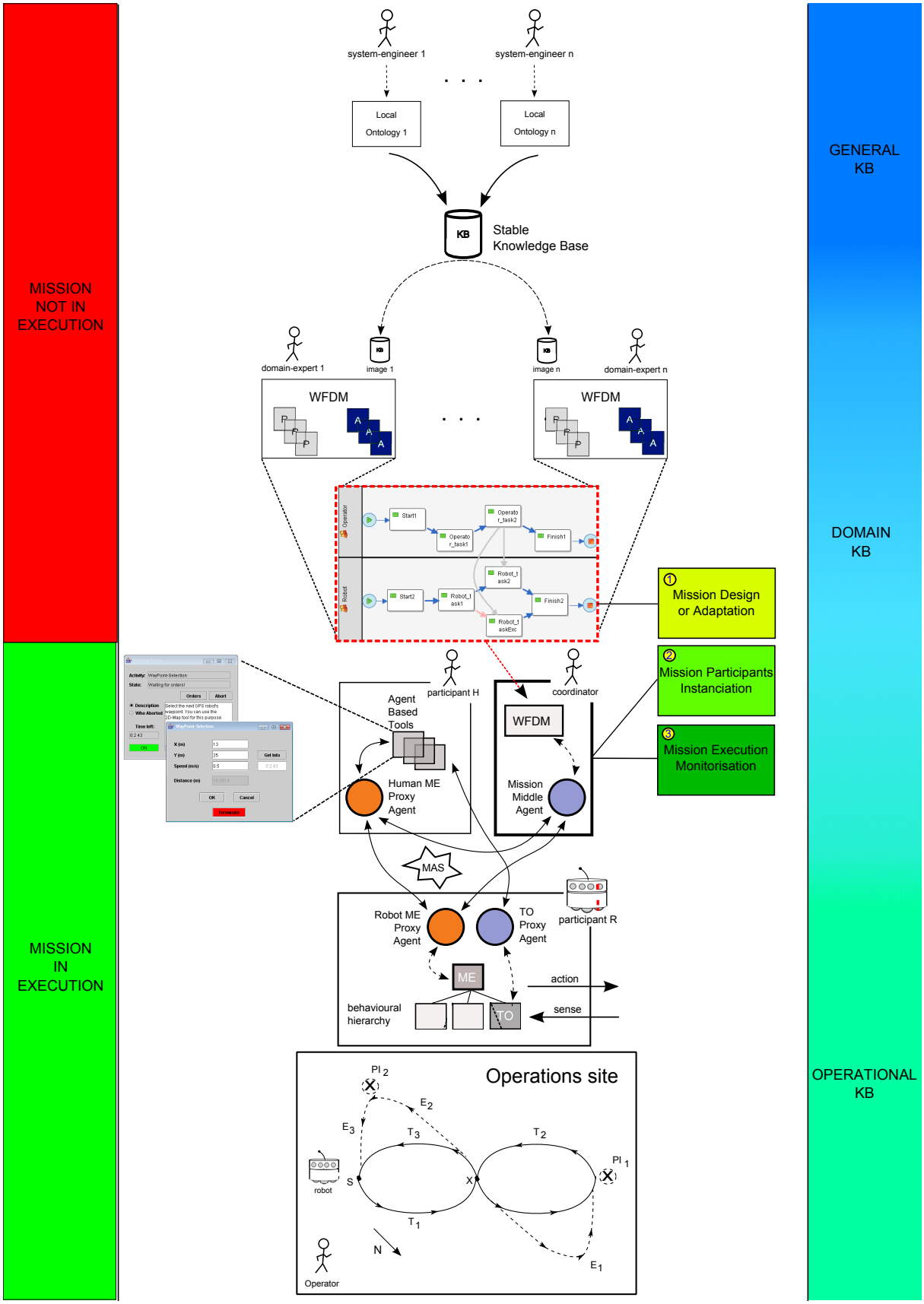

Figure 3.1: KBS_HRT Architecture Overview

- Mission template adaptation.

Mission templates are workflows specified by domain experts using a workflow design tool, normally applied to human organisations, which was properly adapted for the human-robot teamwork case. These mission templates are knowledge intensive tasks specifications, i.e. domain knowledge, maintained in a knowledge base supported by a well specified ontology (see Section 3.1).

After the specification of the mission through the tool for workflow design, the instantiation of the physical entities to the participants in the plan (workflow) was to be made. In the mission template nothing is said about which physical entity (e.g. human, robot) will play the role of a given participant in the team. Each physical entity is represented by one *mission execution proxy agent* (see Section 3.3.2), which is registered in a yellow pages service. A *mission middle agent* is used to lookup for physical entities able to fulfil the role of a given participant in a specific mission, and then presents that information to the *coordinator*. The coordinator is an entity representative of human operators/experts responsible for formalising, adapting, instantiating, and monitoring a mission. Then after selecting and adapting the mission template, the coordinator recruits physical entities to be a team member. Each participant receives the part of the workflow necessary to know how to accomplish his role in the mission, making it ready to execute. Figure 3.2 presents the mission template instantiation done by the coordinator and supported by the multi-agent system.

After receiving the request to start the mission, each *mission execution proxy agent* executes the part of the workflow corresponding to its participant (cf. Section 3.4). During the mission execution agents exchange messages so that participants can act in a coordinated way, at the beginning, during and at the end of their activities. This is a fully distributed execution, because each participant has its own *mission execution proxy agent* concerned with a limited part of the whole mission template, i.e. participant plan and not the mission plan, and it only exchanges messages with participants with whom it has an explicit message link in that mission template.

The coordinator selects mission templates according to the context of the operations. But on the field, some adaptations to the mission template are expected and they are performed by

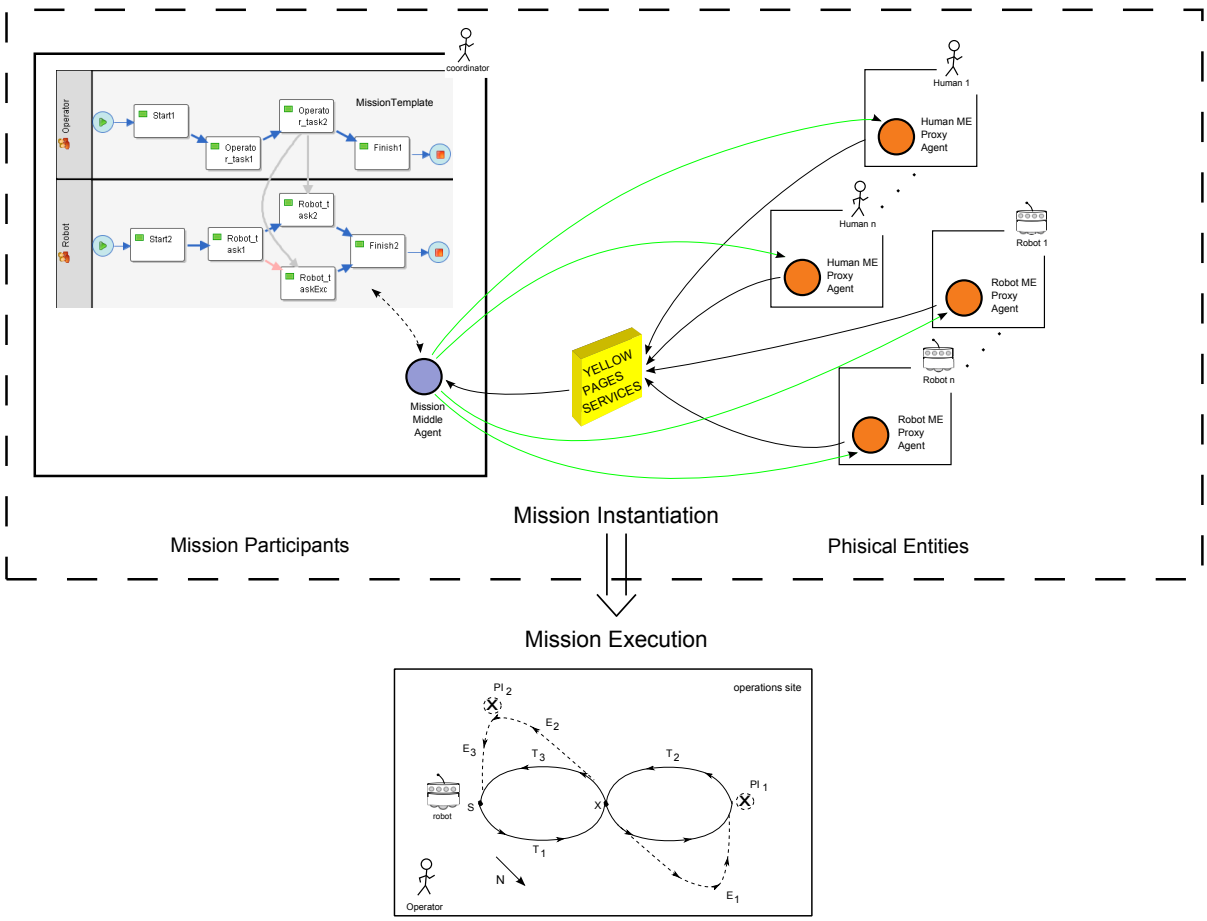

Figure 3.2: Mission Instantiation

the coordinator through the same tool used to specify the mission template. In this process it is expected that some adapted mission templates are added to the mission templates library for further reuse, allowing the knowledge base to evolve with the experience in the field.

Mission specification, adaptation, instantiation and monitoring are performed in the Work-Flow Design and Monitor (WFDM) (see Section 4.1.3 to see how it is implemented). The next sections will explain how KBS_HRT is implemented.

3.1 General, Domain and Operational Knowledge Base

The knowledge plays a key role in the proposed architecture. The knowledge in the system is not of a single kind, meaning that it has different persistence and abstraction levels. As it widely known in the knowledge engineering field, a proper management of knowledge is

a fundamental aspect in an organisation. A layered knowledge base has been considered in order to encompass disparate knowledge formalisms, each one adapted to a specific purpose. Concretely, the knowledge base is divided into three main layers (see Fig. 3.3):

General knowledge base includes frame-based concepts like time and space, and all these concepts will be spread all over the system, including the others two knowledge base layers. It is done in an ontology which is defined and maintained by the system engineer/s (see Section 4.1.2 to know about its current implementation).

Domain knowledge base is defined by domain experts using knowledge acquisition sessions. This knowledge base mainly gathers knowledge from intensive task specifications in the form of workflows (mission specification templates). Section 3.2 describes a set of adaptations made to allow the input parameters of the activities to be changed, even when they are running.

Operational knowledge base includes all specialisations of the mission templates developed for particular purposes, such as a particular application domain. This knowledge is defined as refinements performed by the end-user (e.g. operator) over a mission template. Both operational and domain knowledge are specified in a tool for workflow design, allowing to integrate concepts of human organisations management to human-robot teamwork.

The ontology is composed of five top classes (see Figure 3.4): Participants, Activities, Termination Conditions, Parameters and Units classes. These classes establish relationships among them in order to augment the system expressiveness. The following relations were identified:

1. Participants and Activities (R_{PA})- Several activities can be specified in the ontology (e.g. *Goto_XY*, *Path_Following* and *Tele-Operation*). Each of these activities can only be executed by a given subset of participant types. This association between activities and participant types resembles the notion of skill. For example, a *Goto_XY* can be performed by a set of different mobile robots and that is explicit in the ontology. This relation will

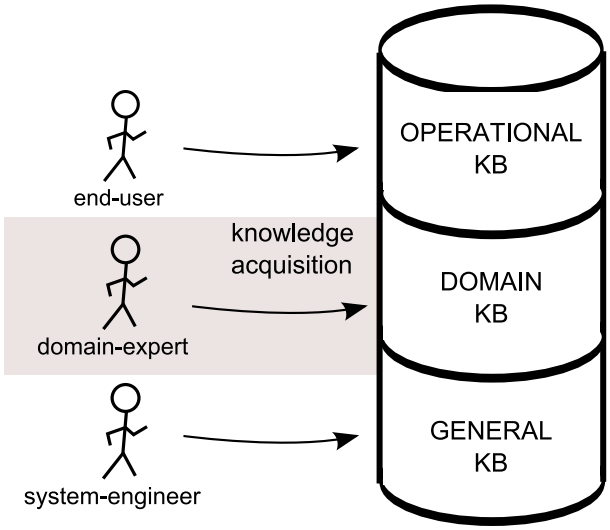

Figure 3.3: Knowledge Base

help the domain experts to know what are the activities that each participant can perform in a mission.

2. Activities, Parameters and Units (R_{APU})- Each activity has input and/or output parameters and these parameters have units (e.g. metric units). It is possible to associate units to parameters according to the activity's type. In addition, each input and/or output parameter specific to each activity can be defined as optional. This helps domain experts, because in the workflow design tool the activities' parameters will appear as being optional or to be filled in.

3. Activities and termination Conditions (R_{AC})- Each activity has termination conditions. All the activities have default and possibly extended termination conditions. Default conditions are *not-ok-aborted*, *not-ok-time-out*, or *ok*. Extended conditions enable the specification of new termination conditions to a determined activity. For example, termination conditions *1* and *2* in a Track Selection activity. Termination condition *1* will occur if the operator selects the first track and *2* if he choose the second track.

The general knowledge base is defined and updated by system engineers in run time, without affecting the mission templates (see Figure 3.5). Updating the activities and participants on WFDM tool (represented by the blue and grey squares, respectively) can only be done when

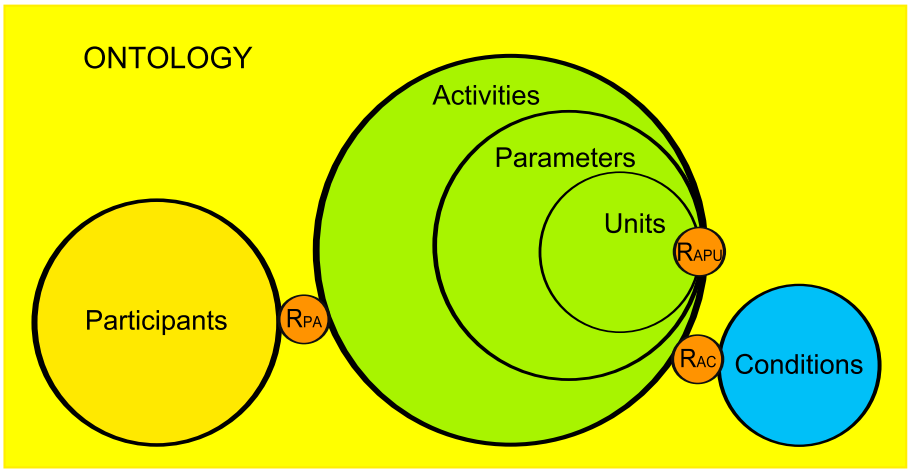

Figure 3.4: Ontology classes and their relation

a stable version of the general knowledge base exists (in this case it will be a XML document with that information).

It is important to state that this process of creating a stable version is not implemented in the way represented in Figure 3.5. In fact, with the current implementation, only one ontology is used, but it is important to know how a several engineers can collaborate in the creation, modification and use of an ontology. For that, some problems have to be solved, like problems of information integration, knowledge-level interoperation and knowledge base development. A solution is to use a tool for collaborative ontology construction which already has that problems solved. A good solution is Ontolingua [Farquhar et al., 1997] software which provides already a distributed collaborative environment to browse, create, edit, modify and use ontologies. This was not implemented because it is not the focus of this dissertation, the focus is collaboration among teammates during a mission and the way how the ontology is built it is indifferent, with one or more system engineers, to prove the concept which is proposed in this dissertation.

The ontology contains information about the type of team members, the activities they can perform, and how these activities are parameterised. This ontology supports the WFDM tool which is generic in the sense that does not require reprogramming if new activity types are added to the system.

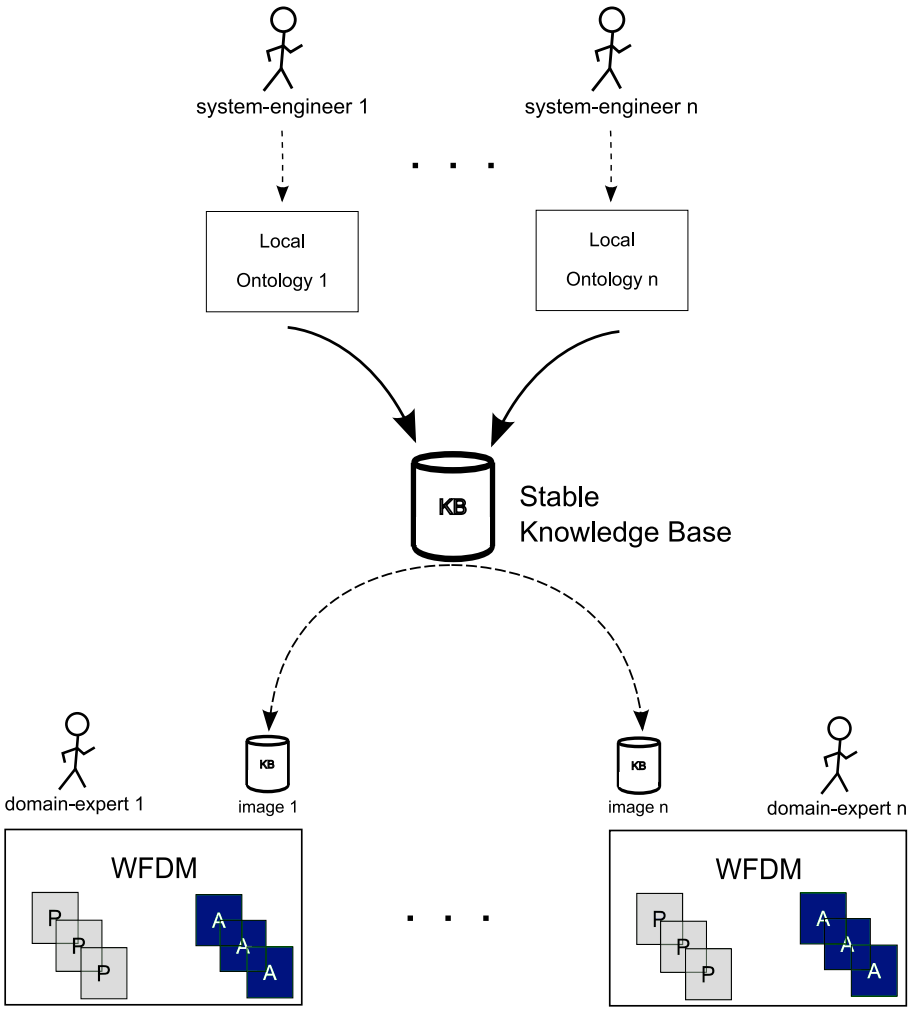

Figure 3.5: Knowledge update and distribution

3.2 Team Plans Specification

3.2.1 Introduction

Most of the solutions in human-robot teamwork are inspired from work on multi-robot and multi-agent systems properly adapted to include humans. In this dissertation the problem is seen from another perspective, i.e. to include robots and software agents as participants on human-centred operational procedures, supported by knowledge management concepts usually employed in human organisations. These human-centred operational procedures can be approximately represented by a set of templates. These templates, in this case of teamwork may also be called team plans and are defined as workflows.

A workflow is defined in XPDL, the XML Process Definition Language [1] [WfMC, 2001] which is a format standardized by the Workflow Management Coalition (WfMC) [2] to interchange Business Process definitions between different workflow products like modelling tools and workflow engines. Workflow Management Coalition is a consortium, formed to define standards for the interoperability of workflow management systems. It was founded in May of 1993 as an offshoot of the Black Forest Group with original members including IBM, Hewlett-Packard, Fujitsu, ICL, Staffware and approximately 300 software and service firms in the business software sector. Since its foundation, the use of XML has become more widespread and today its focus is principally around process definition file interchange, using the standard XPDL.

XPDL defines a XML schema for specifying the declarative part of workflow, in contrast with BPEL (Business Process Execution Language), XPDL is not a compiled executable programming language, but a process design format for storing the visual diagram and process syntax of workflow business process models, as well as extended product attributes.

Here some extensions were made in XPDL format, mainly to enable interactions between activities even when they are running, i.e. *Data-Flows* (see 3.2.2). Typically, workflows describe sequences of activities, which only can exchange data at transition time.

[1] http://www.wfmc.org/standards/xpdl.htm
[2] http://www.wfmc.org/

Using a visual and intuitive solution based on workflow, human operators can design and adapts a team plan even with little knowledge on robotics, avoiding the use of complex scripting languages.

3.2.2 Mission Template Specification

This subsection describes each component used to specify a workflow in the WFDM. A package may contain one or more workflows. Each workflow associates participants, such as a robot or a human, to a set of linked activities as can be seen in the Figure 3.6. In that example, the plan is separated in two rows, each one corresponding to a participant and the rectangles filled by blue and white colours are active and not active activities, respectively. Activities can have deadlines, to be terminated or to be turn on. When a deadline occurs, the participant can leave the mission, force all the participants to abort, or directly jump to another activity. Activities are linked by transitions and optionally by data flow links. That is represented in the Figure 3.6 by arrows linking the activities which are blue and pink, the conditional and exceptional transitions, respectively, and data-flows arrows are grey. Transitions are associated to conditions defined in terms of activities termination codes. In accordance with the activity termination code, one or more transitions leading to activities in any team member, will be activated. According to the specification, the output parameters of the terminated activity, can then be passed to the input parameters of the outgoing activities. The user can also select if a transition is sufficient or necessary to activate an activity to which it is connected. Both participants may have independent execution plans, which are not necessarily synchronised. Data-flow links enable active activities in different participants to communicate during their execution. Although much of the tasks are expected to be performed autonomously by each team member, many are situations where this is not true. A team member may be performing a task while being guided by any other team member. Thus it is essential for the plan specification to account for these modulatory signals, which are not simple input parameters considered at the start/end of the task.

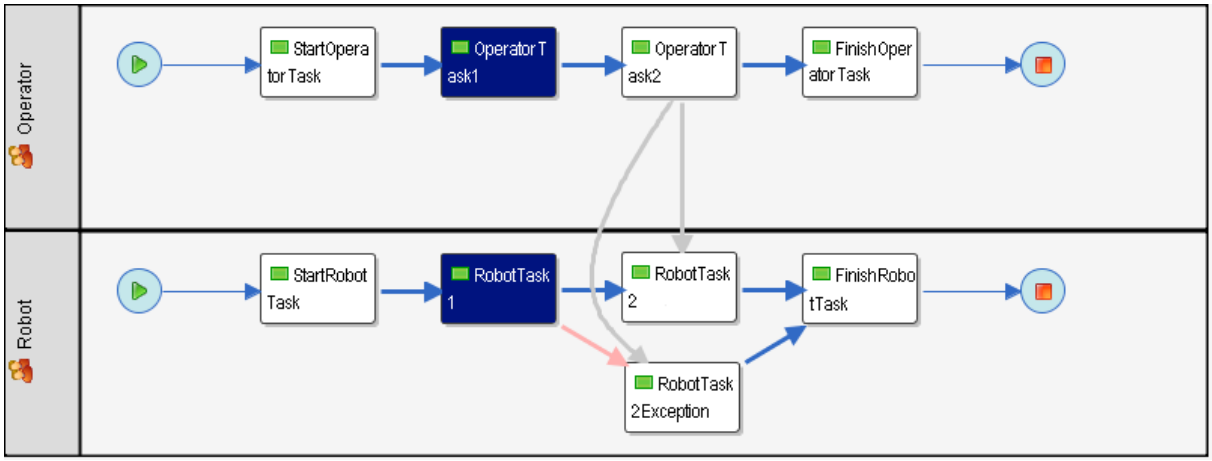

Figure 3.6: Generic mission plan example using WFDM

- The plan is separated in two rows, each one corresponding to a participant.
- The rectangles filled by blue and white colours are active and not active activities, respectively.
- The arrows linking the activities are transitions and data-flows. Blue and pink colours are conditional and exceptional transitions, respectively, and data-flows arrows are grey.

Now with more detail, each component used to specify a workflow in the WFDM will be explained.

Package

A *package* contains *workflow processes*, which in turn associate *participants*, such as a robot or a human, to a set of linked *activities*. The *package* acts as a container for grouping together a number of individual process definitions. Table 3.1 shows the package attributes.

Name	*M/O*	*Description*
Id	M (mandatory)	Used to identify the package.
Name	O (optional)	Name of the package.

Table 3.1: Package attributes

Workflow Process

Workflow process is the process definition entity which provides contextual information that applies to other entities within the process, i.e. the *workflow process* defines the elements that composed a workflow. It describes and contains the process itself. Table 3.2 shows the workflow process attributes.

$Name$	M/O	$Description$
Id	M	Used to identify the workflow process.
Name	O	Name of the model, used to identify the workflow process.
Description	O	Textual description of a workflow process.

Table 3.2: Workflow Process attributes

Workflow Participant

All referenced *workflow participants* have to be defined in the scope where they are used. The *workflow participant* is defined by a id and a $type$. The $type$ represents a physical entity defined in the ontology (e.g an operator or a robot).

This definition is an abstraction level between the *workflow participant* and the real performer of an activity. During run time these abstract definitions are evaluated and assigned to a concrete human, robot or software agent.

Each *workflow participant* is only allowed to run a single activity at a time. The reason behind this constraint is the fact that activities represent a partial of mission, and not a partial behaviour. If such parallelism is required, then the participant in question must be decomposed in two or more participants. See for instance the case of a robot equipped with a tele-operation camera. Both physical entities can have orthogonal behaviours and therefore only hardly can be represented by the same participant. Then these entities may have to be decomposed in robot and tele-operation camera *workflow participants*. Table 3.4 shows the workflow participant attributes.

Name	*M/O*	*Description*
Id	M	Used to identify the workflow participant definition.
Name	M	Used to identify the participant type.
Description	O	Short textual description of the workflow participant.

Table 3.3: Workflow Participant attributes

Workflow Activity

Generally, any process is composed of a number of steps, which lead to the overall goal. *Workflow process* consists of a number of *workflow activities*. The *workflow activity* is a piece of work that will be done by a single *workflow participant* which can interact with others *workflow participants*, at run time and/or at the start and/or termination of that *workflow activity*.

Each activity maps its *input parameters* to its *output parameters* according to any given policy. Input parameters can be defined by default or given through one link (i.e. a transition or a data-flow) or more, with output parameters belong to others activities. These links are only enabled if the input and output parameters belong to the same class in the ontology.

All the input and output parameters are supported by XPDL *Extended Attribute* feature. *Extended Attributes* are the primary method to support such extensions. These attributes are filled by the user, when necessary.

In addition, activities can have *deadlines*. When a deadline is reached the participant can:

- Leave the mission, warning the others to this fact;
- Request all the others to abort their missions;
- Terminate its activity due to an activity timeout, predefined in the mission;
- Jump to another activity without met the preconditions of a previous activity because the time specified in the mission to begin that has expired.

Table 3.4 shows the workflow activity attributes.

Name	*M/O*	*Description*
Id	M	Used to identify the workflow activity definition.
Name	M	Text used to identify the workflow activity.
Performer	M	Workflow participant id who performs the workflow activity.
Deadline	O	Specification of deadlines and actions to be taken if they are reached.
Description	O	Textual description of the activity.
Extended Attributes	O	Input parameters belong to the activity.

Table 3.4: Activity attributes

Transition

Links between two *workflow activities* are established by *transitions* and optionally by *data flow links*. Circular transitions can be made too, i.e. a transition from an activity to itself.

Transitions are more than just links between activities. They are conditional or exceptional (e.g. used when occurs a deadline in an activity), in particular their activation is dependent of the *workflow activity's* termination code, allowing many possible courses of action.

Transitions can also be used to convey data from the output parameters of an activity to the input parameters of another activity. Linked parameters must be of the same class.

Each outgoing *transition* from a specific *workflow activity* has several termination codes and these codes may be different from activity to activity. Each *workflow activity* can have one or more outgoing *transitions*, but only one of these can be active at the same time inside of a *workflow participant*, i.e. transitions linking activities belong to the same participant. For *transitions* which link activities belong to different participants, the above rule is not applied.

The only way to be able to navigate in the participant workflow is if a termination code occurs, activating an outgoing transition of the participant's current activity. Otherwise, the participant in cause ends his mission and goes to the $finish$ activity.

A transition can be necessary, sufficient or without influence in the activation of a determined activity. An activity becomes activated only when the outgoing transition that links the previous activity and this activity in a participant workflow is active. In addition, if exist nec-

essary and/or sufficient transitions, all *necessary* and at least one *sufficient* transition have to be active also. These synchronisation options (necessary and sufficient transitions) only have sense between different participants, because it is always necessary one intra-participant transition to navigate in the participant plan. The transitions which are not necessary nor sufficient, are only used to exchange data from output to input parameters of an activity, when these are already active before the activity has been active. Table 3.5 shows the transition attributes.

$Name$	M/O	$Description$
Id	M	Used to identify the transition.
Name	O	The name of the transition.
Condition	M	Termination condition and synchronisation associated to the transition.
From	M	Determines the $FROM$ activity of a transition.
To	M	Determines the TO activity of a transition.
Description	O	Textual description of the transition.
Extended Attributes	O	Output parameters associated to $FROM$ activity.

Table 3.5: Transition attributes

Figure 3.7 shows four distinct examples using, necessary (Nec), sufficient (Suf) and/or none of these synchronisation transition options, to activate the activity "TASK" of the participant B, namely:

1. All necessary transitions have to be active (Fig. 3.7 top-left);
2. Only one sufficient transition active is enough (Fig. 3.7 top-right) ;
3. All necessary and at least (b) or (d) sufficient transition active (Fig. 3.7 bottom-left);
4. Necessary transition (c), and at least (b) or (d) sufficient transition active, and (a) and (e) can be active or not (Fig. 3.7 bottom-right).

Data-Flow Links

Normally workflow models [Hollingsworth, 1995] are mainly concerned with the automation of business processes. All these processes can be well defined with activities which are only

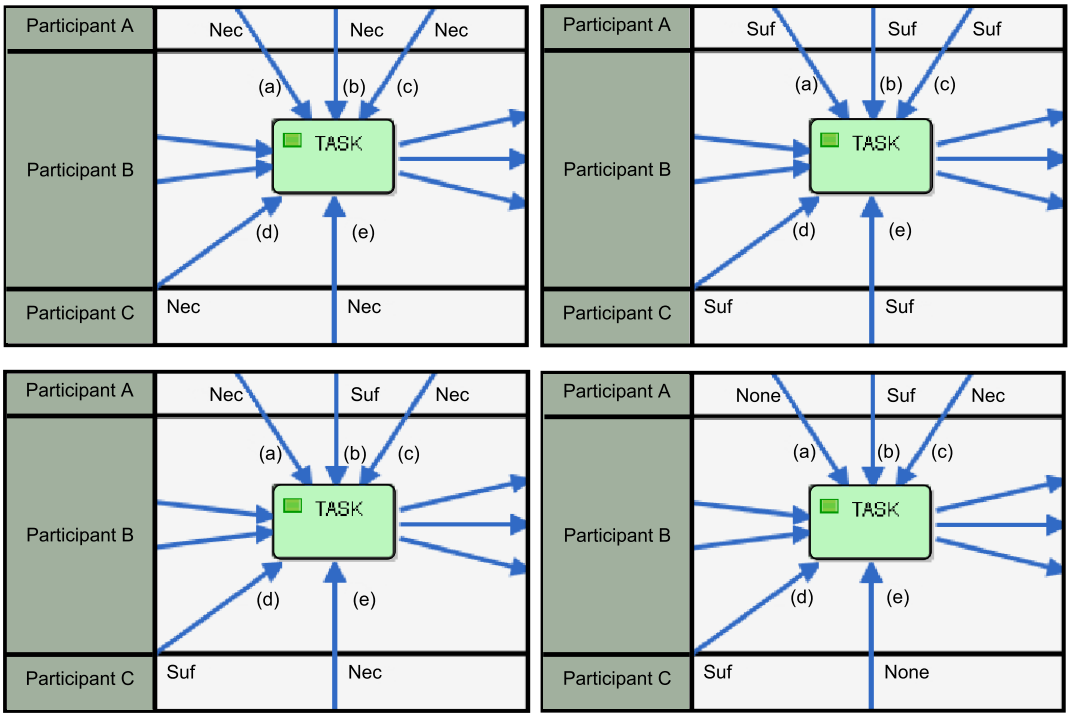

Figure 3.7: Four distinct examples using the synchronisation transition options

synchronised at start and finish phases, i.e. they can execute isolated.

These workflow models were applied efficiently in many applications but when these applications have activities that need to exchange information during its execution, these models have limitations.

In order to support that kind of exchange of data in the workflow model presented in this dissertation, the *data-flow* entity was created to represent these interactions, complementing the transition entity which only allow flow of data between activities when they start and finish. Data-flow is inspired on the idea of cooperative workflow used in business management as described in [Godart et al., 2000]. This work states that ”A cooperative workflow is a workflow where some activities executing in parallel can share some intermediate results during execution.”. Without data-flows the activities are like black-boxes, with inputs and outputs, but with no intermediate visible results.

The motivation for the development of the data-flow entity is as follows. Much of the tasks are expected to be performed autonomously by each team member and there are many situations in which participants must interact in a tightly coupled way. For instance, in a tele-operation

situation, a team member (e.g. the robot) can be performing an activity while being guided in real time by another team member (e.g. human operator). Thus, it is essential for the plan specification to account for these modulatory signals, which are not simple input parameters considered at the start of the activity. These modulatory signals are represented by data-flow links. Task-dependent interactions, at system level, (e.g. team member (typically human) must be able to terminate another team member's (typically robotic) activity) using data-flow.

As transitions, data-flow links connect the output parameters from an activity, to the input parameters of another activity, but at this case, these activities must be running, to allow this data exchange. This data exchange only occurs between different participants. As for transitions, linked input and output parameters must be of the same class. Table 3.6 shows the data-flow attributes.

Name	*M/O*	*Description*
Id	M	Used to identify the data-flow.
Name	O	The name of the data-flow.
From	M	Determines the $FROM$ activity of a data-flow.
To	M	Determines the TO activity of a data-flow.
Description	O	Textual description of the data-flow.
Extended Attributes	O	Outputs parameters associated to $FROM$ activity.

Table 3.6: Data-Flow attributes

In Figure 3.8, a UML class diagram shows the classes of the system, their interrelationships and attributes.

3.3 Multi-Agent System for Teamwork

A workflow, as specified before, is a static definition of a team plan. It is a resource rather than a mechanism. This section describes how this static information is instantiated, and finally executed.

Typically, workflow engines are centralised service oriented entities, meaning that there is a central mechanism that invokes remote services in a specified order. It also means that rarely

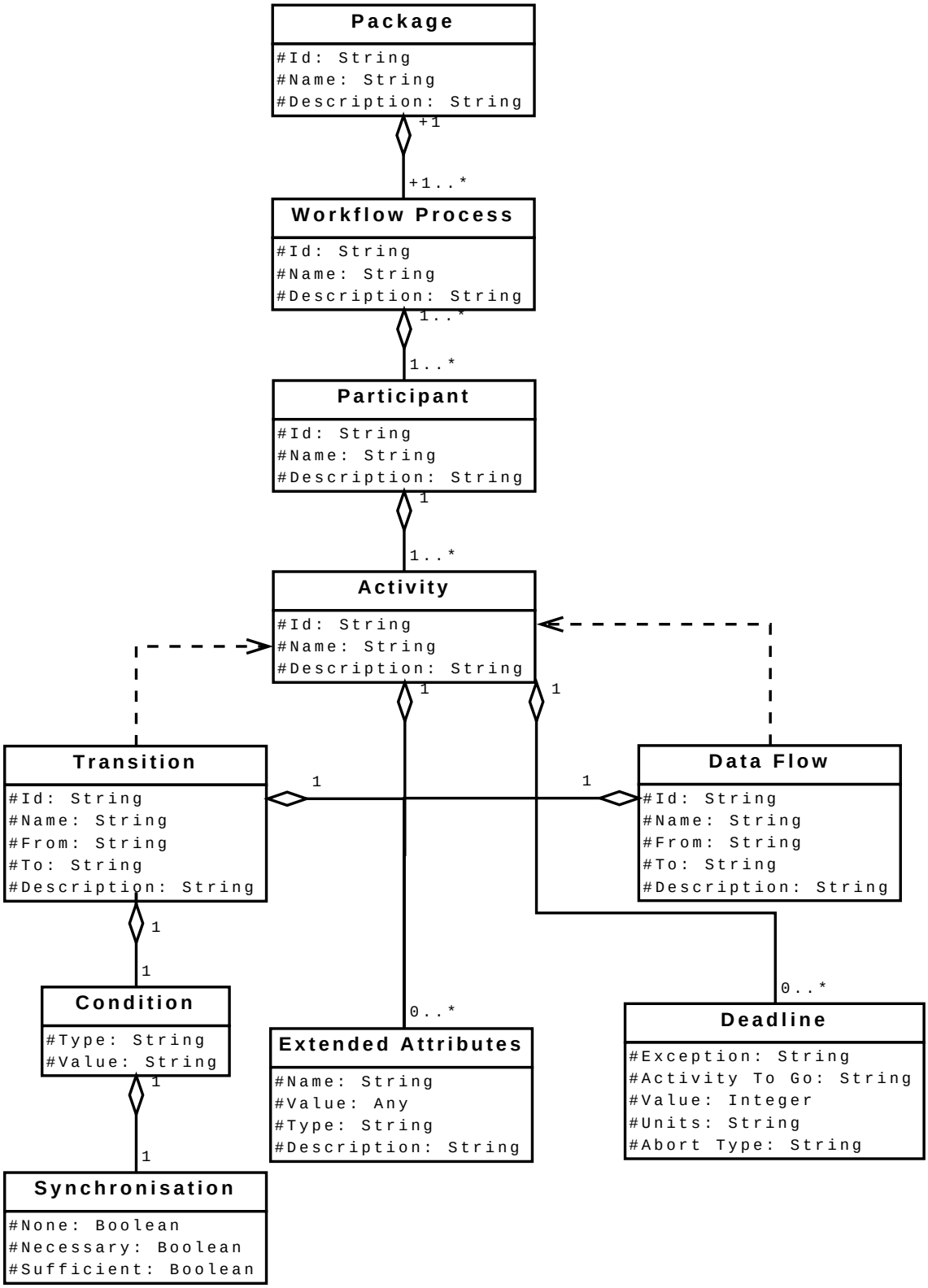

Figure 3.8: Mission specification components UML Class Diagram

these services can interact in a dynamic way rather than just by means of input and output parameters updated at the start or end of the activities. This limits their application in domains where activities/services have intricate run-time interactions, such as those found in mobile robotics. Instead, specific messages flow between agents in order to allow them to update activities input parameters at any time, i.e. even after the activity has begun. These parameters operate as pointers to remote data structures, whose contents are dynamic.

Agents are most suited to applications that require communications between components, sensing or monitoring of the environment, or autonomous operation. Agent technology is use in applications that reason about the messages or objects received over a network. Multi-agent systems are also suited for applications that require distributed, concurrent processing capabilities.

In this system workflows, are instantiated (1) , and executed (2) using a multi-agent platform (see Section 4.1.1 to know about its actual implementation), which provides two main functionalities, namely: (1) a yellow pages service for participant registration and lookup, (2) plus an inter-agent messaging infrastructure based on Agent Communication Language (ACL), built upon the concepts stored in the frame-based knowledge base.

Figure 3.9 presents an use case, in which three entities have been considered. The *coordinator* is an entity representative of human operators/experts responsible for formalising, adapting, instantiating, and monitoring a mission. *Robotic participant* and *human participant* are entities representatives of robots and humans, respectively, which are involved in the execution of a mission.

3.3.1 Mission Coordinator

By using the WFDM tool, the coordinator design (or adapts), instantiates, and monitors the execution of the mission. The WFDM tool interacts with the *mission middle agent* in order to provide the coordinator with the list of available physical entities able to play the role of each mission participant. The coordinator is responsible for the final selection. Afterwards, the part of the plan corresponding to each participant is sent to its *Mission Execution (ME) proxy agent*.

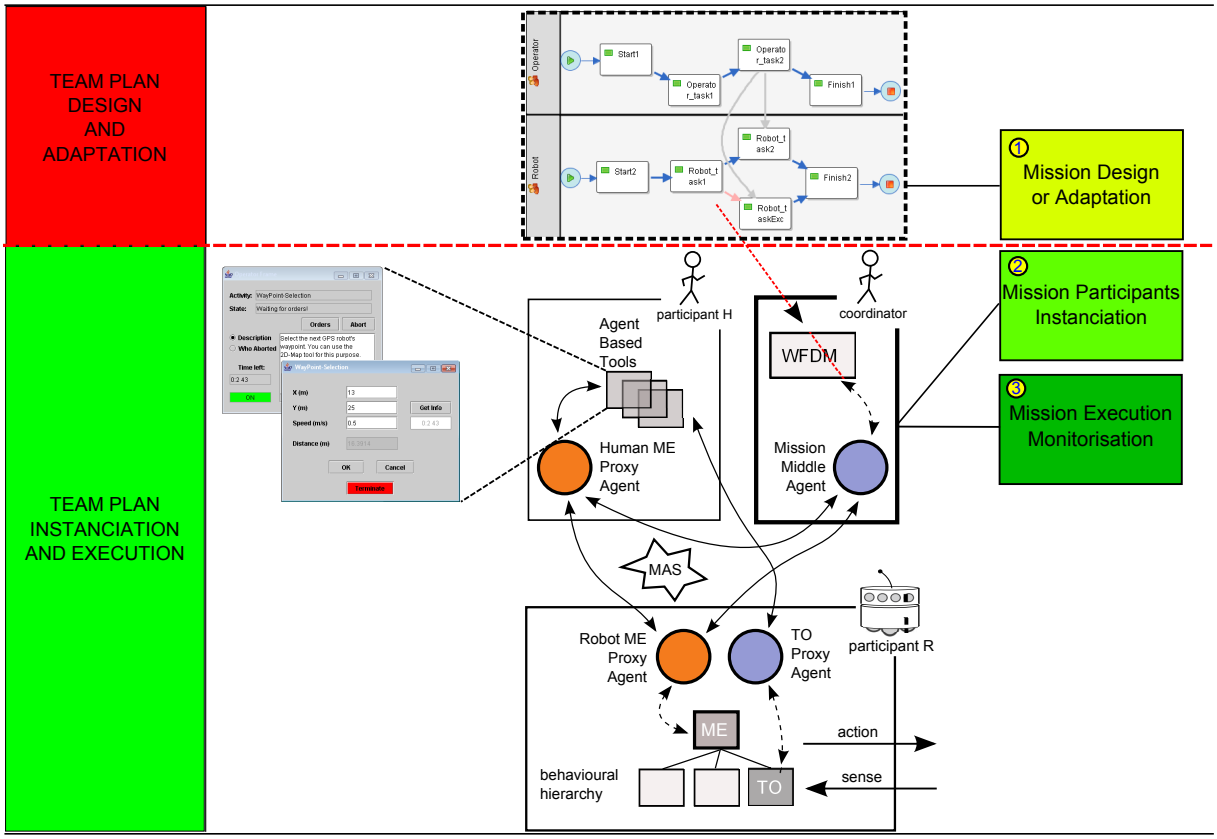

Figure 3.9: Multi-agent system for teamwork use case.

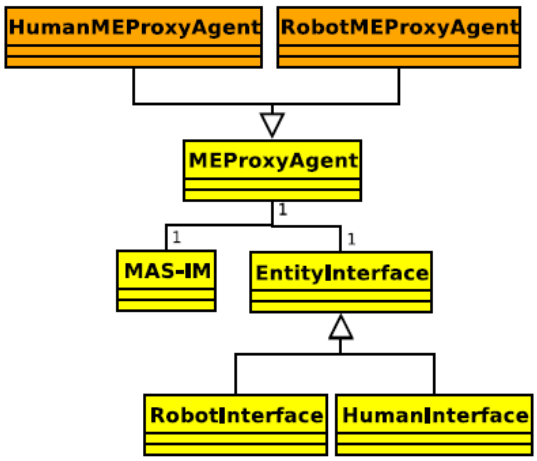

Figure 3.10: ME proxy agents UML class diagram.

Finally, the team initialisation is done by the *mission middle agent*, informing all *ME proxy agents* of the event.

During mission execution, each *ME proxy agent* is allowed to inform the *mission middle agent* of its execution state (e.g. which activity is currently being executed). This information is then presented through the WFDM tool to the coordinator, reflecting the status of the mission. This symbolic information augments the coordinator's *mission awareness*.

3.3.2 Mission Participants

During a mission execution, each physical entity participating in the team plan is represented by a *ME proxy agent* (see Fig. 3.10).

A *ME proxy agent* aggregates two main components: (1) the *Multi-Agent System Interaction Mechanism* (MAS-IM), and the (2) *Entity Interface*. The MAS-IM is the component that enables the agent to interact with other agents in the multi-agent community, as well as to exploit its middleware services (e.g. yellow pages service). The *mission middle agent* also aggregates an instance of the MAS-IM class for the same purposes. The *Entity Interface* class is responsible for the direct interaction with the physical entity, through messages to and from its control system (in the robotic case) and updates to the graphical interface (in the human case).

In short, the *ME proxy agent* interacts with others alike (through the MAS-IM module) to enable the unfolding of the team plan. It interacts with its represented physical entity, through the *Entity Interface*, to request the actual execution of a given activity and to fetch the current values of activity's output parameters.

The way *ME proxy agents* interact with their respective physical entities (i.e. robotic and human), as well as, how these agents interact among themselves during the unfolding of the mission is discussed in the next subsections.

Robotic Participant

In the case of the robotic entity, the *robot ME proxy agent* extends the base class *ME proxy agent* by considering the specific *Robot Interface* class. This interface interacts with the executive layer, here represented by a behavioural architecture (based on [Correia and Garção, 1995]) implemented in C/C++, for actually controlling the robot (see Fig. 3.9). The behavioural paradigm [Arkin, 1998] was selected due to its ability to map high level activities onto situated perception-action loops.

Both *Robot Interface* class and *Mission Execution (ME) behaviour* (see Fig. 3.9), i.e. the interface to the behavioural architecture, interact using plain sockets. Messages reaching the *ME behaviour* contain the activity to be executed in addition to the values of its input parameters. Even if the activity does not change, each time the value of an input parameter arrives, a new message is generated. Then, the *ME behaviour* activates a chain of subordinated behaviours (e.g able to perform *goto-xy* while *avoid-obstacles*), feeding them with the current values of the input parameters.

According to post-condition rules, subordinated behaviours terminate, generating an output termination flag and a condition that lead to termination. This information is propagated upwards eventually reaching the *ME behaviour*, which will then inform the *ME proxy agent* about the occurrence. This way, the *ME proxy agent* knows if the current activity has been terminated and how. In parallel to this loop, the *ME behaviour* also periodically informs the *ME proxy agent* about the currently active behaviour's output parameters.

Hence, messages flow between *ME proxy agent* and *ME behaviour* at a pace not necessarily dependent on activity transitions. Input and output parameters values are exchanged whenever a change occurs, and a incoming and outgoing data-flow link are associated to such activity, respectively.

Human Participant

Such as in the robotic case, the *human ME proxy agent* also extends the *ME proxy agent*. *Human ME proxy agent* will play the role of a *personal assistant* that dynamically changes the graphical interface in the *Human Interface* class, while the mission unfolds.

The graphical interface (see Fig. 3.11) provides the human with static information about the current activity (e.g. a human readable description and a suggestion how to use auxiliary tools, such as the tele-operation) along with dynamic information, such as the current values of its input parameters. The frame in the rear illustrates the main front-end with activity's description. The frame in the front is dynamically adapted, according to the current activity.

The activity and its input parameters are provided by the *ME proxy agent* associated to the human entity. Typically, the human operator is required to provide input, which will be the output parameters of the current activity. The human operator can also request to terminate its activity, in addition to abort its participation in the mission. All these informations are then passed to the *ME proxy agent*.

Some authors (e.g. [Nourbakhsh et al., 2005]) use plain socket connections for telemetry and tele-operation data and structured agent communication messages for strategic and tactic purposes. This however breaks down the modularity and abstraction of the system modules. To avoid this loss, the *TO proxy agent* uses a compressed form of the general ontology for faster communications. This compressed version of the ontology is still related to the expanded version through adhesion relations like *compressed-version-of*. An example is given. A *Location* concept is composed of *Coordinates*, having a *Value* and a *Unit*. A compressed version of a *Coordinate* concept, i.e.*Coordinate_Compressed*, only has a *Value*, as it assumes a given unit.

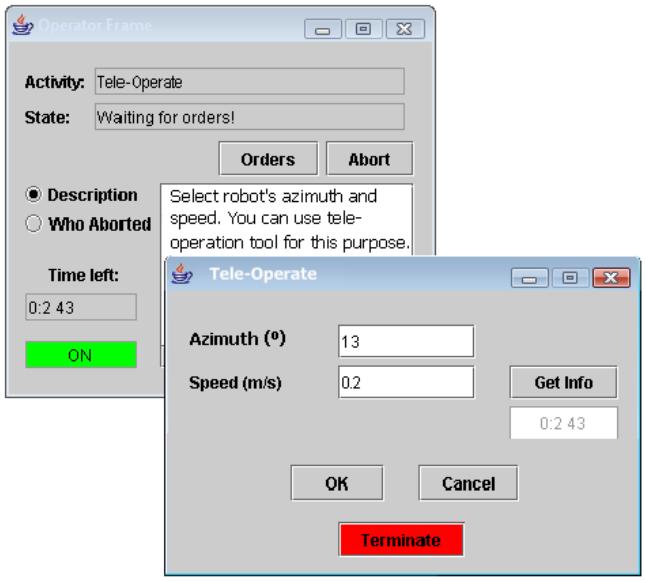

Figure 3.11: Personal assistant.

3.4 Mission Execution

Let us start with a motivating example to explain the mission execution. At a given moment, the *human ME proxy agent* knows that its current activity is *teleoperate-robot*. In this situation, its *Human Interface* class adapts the graphical interface (see Fig. 3.11) so that the human (e.g. operator) can fill in the next azimuth to be followed by the robot. Each time that the operator updates this field, its *ME proxy agent* sends an inter-agent message to the *robot ME proxy agent*, currently executing the activity *being-teleoperated*. Then, through its *Robot Interface*, the *robot ME proxy agent* updates the robot's behavioural architecture according to the incoming message, consequently modulating the *Tele-Operation (TO) behaviour*. Both proxy agents are expecting the mentioned messages exchange, based on data-flow links specified in the team plan.

In addition, the human is also provided with a message that suggests the use of the tele-operation tool [Santos et al., 2007], as can be seen in Figure 3.11. Using this tool, the human can tele-operate the robot with a joystick, while inspecting its telemetry data. This tool sends tele-operation messages to the *Tele-Operation (TO) proxy agent*, which in turn interacts with the *TO behaviour*. The existence of the *TO proxy agent* is justified by the need of customised

messages for high bandwidth requirements. This example also highlights the role of the *personal assistant*: to provide mission awareness, to ask for direct input, or to guide the operator when using other tools.

As referred, each *ME proxy agent* receives from the *mission middle agent* the part of the team plan corresponding to the participant that it is representing. Then, it executes its part of the mission according to the following algorithm:

1. Obtain participant's *start-activity*.

2. While the current activity is not terminated, it updates its input parameters with the contents of incoming, from other *ME proxy agents*, data-flow messages. In addition, *ME proxy agents* send data-flow messages to others alike, whose contents are the current activity's output parameters values. These messages go through data-flow links connecting the current activity with other participants' active activities.

   The aforementioned process of updating the activity's input parameters is done by sending a message to the participant's *Entity Interface* class, which is able to interface directly with the entity's execution layer (e.g. behavioural architecture). In turn, the execution layer provides the *ME proxy agent* with the current values of the activity's output parameters, through the *Entity Interface* class.

   The activity's termination event, along with its code (e.g. *not-ok-aborted*, *not-ok-time-out*, or *ok*), is delivered to the *ME proxy agent* through the *Entity Interface*.

3. When the current activity C terminates, the last obtained values of some of its output parameters are sent as messages to the *ME proxy agents* of those participants that have active incoming transitions from C. These transitions become active if their associated condition on the termination code of C (e.g. *not-ok-aborted*) are met. Output parameters considered in the construction of the message depend on how the mission has been specified. These messages are buffered in the receiving *ME proxy agents* allowing asynchronous subsequent consumption. These messages are *transition* messages.

4. Wait until one of the subsequent activities (i.e. those in the same participant linked to the terminated activity) becomes active. This activation will occur if all necessary and at least one sufficient of its transitions, if there are any, become active. This is assessed by verifying if any of the received *transition* messages refers to the necessary and sufficient transitions.

5. The parameters included in the received *transition* messages that enabled the current activity, are used to update its input parameters. If more than one message feeds the same input parameter (e.g. generated by activities in different participants), only one is selected according to a pre-specified – in the mission – arbitration policy. A default value also takes part of the competition. The actual update of activity's input parameters is carried out as in step 2, i.e. through the *Entity Interface*.

6. Return to step 2 until the current activity is *end-activity*.

To allow the coordinator to follow the mission unfolding, messages according to activities and transitions activation/deactivation events are sent to the *mission middle agent*, which in turn updates the WFDM graphical interface.

3.5 Human-Robot Interactions Awareness

Two dimensions were defined to represent HRI awareness: stereotyped and flexible awareness.

In short, according to a given mission plan, stereotyped awareness relates to the ability of informing participants (in particular humans) about their role in key moments of the mission. On the other hand, flexible awareness is needed, for dynamic environments, because stereotyped plans eventually fail in unpredictable ways and humans must have flexible ways of assessing the situation at hand.

To give stereotyped and flexible awareness to humans and robots, stereotyped and flexible awareness agents, were used, respectively. These agents are represented by Mission Execution proxy agents and mission middle agents, respectively, in this dissertation.

Based in these facts, this dissertation contributes to foster the HRI awareness in both ways, stereotyped and flexible. All the contributions are explained in the Sections 3.5.1 and 3.5.2. For further details about these two new awareness dimensions, refer to [Santana et al., 2008b].

3.5.1 Stereotyped Awareness

Stereotyped Awareness (SA): according to a stereotyped course of action, specified in terms of a team plan, the system pro-actively informs team members of their responsibilities to the team (i.e. their expected behaviour). Based on the same plan, team members can also exchange information for each of its activities.

Stereotyped awareness agents: make their represented physical entity aware of their role in the current state of the mission. Stereotyped awareness agents also interact among themselves in order to share relevant information for the progress of the mission.

Each stereotyped awareness agent is mostly responsible for maintaining its associated participant (e.g. human, robot) aware of its responsibilities to the team in key moments of the mission. To know how and when to invoke the participant for action, the agent possesses a partial team plan encompassing high-level descriptions of the participant's expected behaviour. The current information (e.g. current activity to be performed and its parameters) is passed to the agent, which then updates the control structure of the participant: a behavioural architecture in the case of a robot and a set of graphical user interfaces in the case of a human.

The Figure 3.12 represents the graphical human interface for stereotyped awareness. It self-configures to the activity currently active in the mission, which in this case is "Waypoint-Selection". The above frame illustrates the main front-end, where it is possible to depict the activity's description. The frame at the bottom is a pop-up to enter parameters that will generate the activity's output. The "Get-Info" labelled button allows to request other tools for certain parameters (such as a GPS point). Worth saying that these frames are automatically populated according to the specified activity's parameters on the ontology, and therefore having zero re-programming as new activities are added upon.

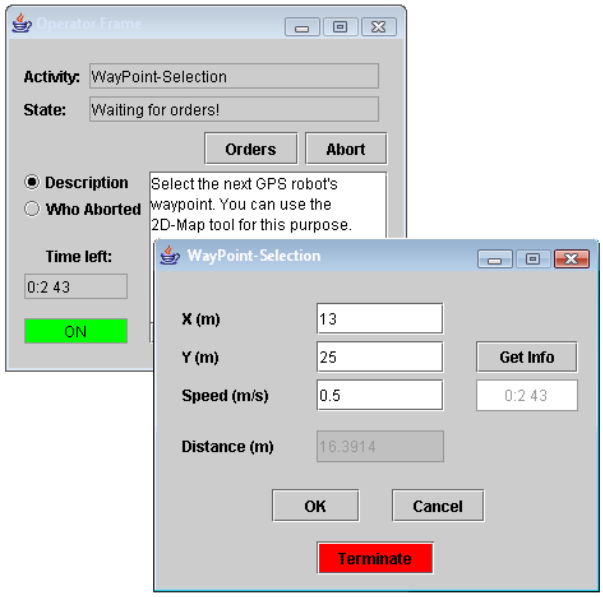

Figure 3.12: Graphical user interface for stereotyped awareness.

Representation and communication aspects of the architecture's stereotyped awareness layer are analysed in the following sections.

Representation

As it was referred in Section 3.2, the mission plan is implemented as a workflow specified with *activities* connected by *links*. The stereotyped awareness agent, in interaction with others alike, navigates in the team plan so as to know which information is key to make the participant aware of the mission.

Table 3.7 presents the key attributes of an activity, whereas table 3.8 exemplifies a possible activity's description.

The table 3.8 shows that the attributes of an activity are strong typed. For instance, the input attribute *point* is of type *GPSPoint*, which in turn is composed of a set of attributes, such as *longitude* of type *Longitude*. All this information is stored in the form of Frames in the system's ontology. Each concept has its formal description along with a human readable one. The latter is useful to the end user and the former is useful at the system level.

As it was mentioned in Section 3.1, inheritance among other semantic relationships are

Name	Human readable activity's unique name.
Description	Human readable description of the activity, i.e. the participant's expected behaviour.
Input	Information considered relevant, in the context of the activity, in order to enhance the mission awareness of the participant.
Output	Information relative to the activity's outcome considered by the participant as potentially relevant to enhance other participants' mission awareness.
Additional	Activity execution related parameters, such as timeout values.

Table 3.7: Description of activities' attributes.

Name	Goto-XY
Description	Move toward a given GPS point at a given speed.
Input	point::GPSPoint,speed::Speed,terminate::Terminate.
Output	distanceToTarget::Distance,done::Bool
Additional	timeout::Time

Table 3.8: Robot's activity example.

available in order to augment the systems expressiveness. For instance, it is possible to associate units to attributes according to the activity's type. It is also possible to associate activities to participant types, resembling the notion of skill. Thus, making a distance represented by a concept *Distance* rather than a simple *float*, provides much more information to the user, such as associated units, scale, and human readable information.

Table 3.9 describes another possible activity. Some of the output parameters of the previous activity can easily be linked to this activity's input parameters. Thus, both robot and human can interact in a seamless way. Text within straight parenthesis refers to information inherited from the super-activity, from which this one derives. Note that in this activity the operator is invited to use another tool to augment its awareness.

Being the stereotyped awareness layer concerned with low frequency message exchanging, it is possible to maintain these highly descriptive representations. Consequently, modularity, clarity, and scalability are enforced without incurring in efficiency loss.

In conclusion, the stereotyped awareness layer exploits the execution plan in order to make participants aware of their responsibilities. Making use of a ontology augments the systems

Name	Next-Inspection-Point-Selection [Waypoint-Selection]
Description	[Select the next GPS robot's waypoint. You can use the 2D-Map tool for this purpose.] The point refers to the next location to be inspected by the robot.
Input	[distanceToTarget::Distance,terminate::Terminate]
Output	[point::GPSPoint,speed::Speed,done::Bool],sensor::Sensor.

Table 3.9: Operator's activity example.

expressiveness and scalability. In addition, committing to a strong ontology enables the development of generic agents. New activities can be implemented at the control level, and then have their semantic description added to the ontology. Mission execution and stereotyped mission awareness can thus be maintained by generic agents adapted to the mission at hand by accessing to the ontology whenever needed.

Communications

Activities' input and output parameters can be bond together to allow participants to exchange relevant information. This connection can be done associated to transitions, which enable the sequencing of activities and the flow of data from output parameters of the incoming activity to the input parameters of the outgoing activity. This data transfer occurs at the exact transition time.

Although much of the tasks are expected to be performed autonomously by each team member, many are the situations where participants must interact in a tightly coupled way, i.e. data-flow, – see for instance the two activities presented in Tables 3.8 and 3.9, when the human can continuously adapt the waypoint being followed by the robot. In addition to the previous example, that configures a robot-human awareness scenario, human-robot interactions awareness is also enabled. Picture for instance the case of a human being continuously updated about the distance that the robot must travel before reaching the provided waypoint.

In short, for proper awareness, participants must be able to exchange messages during the execution of the activity. This can be implemented by having ACL messages flowing between stereotyped awareness agents. The multi-agent platform allied to the workflow engine abstracts

	Type	Human-Mission Awareness	Human-Robot Awareness
WFDM	Task-centered	detailed	insufficient
2D-Map	Map-centered	coarse	coarse
Teleop	Video-centered	absent	detailed

Table 3.10: Flexible awareness tools main characteristics.

all these problems from the agents location and communications.

3.5.2 Flexible Awareness

Flexible awareness (FA): stereotyped plans eventually fail, requiring the system to provide users with tools, typically map-centred and video-centred, to fetch awareness information in a customised way.

Flexible awareness agents allow humans and robots to perform unplanned information exchange for mutual awareness. Teleoperation is typically supported by these agents.

As aforementioned, it is not always possible to expect that the system is able to autonomously provide all necessary awareness information. Thus, in order to provide flexible access to awareness information, three complementary agent-based tools were developed in the project, whereby WFDM is one of them.

The table 3.10 summarises how each of the three tools contribute to HRI awareness.

The WFDM tool receives information, through its flexible agent, from the stereotyped awareness agents with their current status (e.g. current activity). Through this tool, it is possible to obtain overall mission awareness information. This is a kind of awareness which in [Drury et al., 2007] it is not solvable by typical map-centric and video-centric interfaces.

The follow two tools that will be explained are not part of this dissertation work. However, these three tools have direct or indirect interactions, in form to augment human-robot teams *awareness* and its explanation seems important to improve the comprehension of Table 3.10.

Teleop tool interacts, through its supporting flexible agent, fetching the mostly robot-centered telemetry data (e.g. wheel speed). In order to enhance human-robot awareness, this tool also

allows the operator to control a tele-operation camera. Several modes of tele-operation are available, such as direct motor control and reflexive operation. Of particular interest is the robot's ability to perform self-monitoring, i.e. when a collision occurs, the tele-operation camera automatically gazes towards the collision point. All these functionalities support human-robot awareness. As for the case of the Teleop tool, the 2D-Map tool also interacts with proxy agents to fetch telemetry able to be geo-referenced, such as localisation and obstacle distribution. It also interacts with stereotyped agents to obtain the current activity being executed by the participant in question. Geo-referenced information relative to the current activity is displayed (e.g. the target waypoint in a "Goto-XY" task). This map-centred human-robot and human task awareness information complement the workflow view of the WFDM tool with geo-referenced information.

Representation

The considerable amount of information being exchanged among flexible agents and stereotyped awareness agents deserves some considerations.

In general, the more abstract a concept is, the more information it conveys, and consequently less bandwidth is required for the same amount of information. However, the receptor must be provided with additional machinery to map the concept to the represented object. Therefore, in order to reduce the number of assumptions and consequently increase modularity, abstraction should be reduced as much as possible. As consequence, a clear trade-off between bandwidth economy and modularity holds.

Communications

Being based on polling, the communication protocols for flexible awareness interactions can be much more simpler than for the stereotyped case. In the flexible case information is polled in a periodic way, meaning that the absence of a reply is mitigated by the next request. Additional transactional protocols would result in unnecessary communication overhead. On the stereotyped case, however, the coordination among activities requires both sender and receiver

to agree on the reception of the messages. Otherwise the system may hang in deadlock situations. For example, failing to receive an activity's termination message means that the receptor will remain in that activity despite all other participants move on the mission as planned.

Chapter 4

Prototype Implementation

The previous chapters proposed a model of a knowledge-based component for teamwork, the KBS_HRT. In this chapter, the implementation is presented.

This work is part of a project aiming the development of affordable service robots [Santana et al., 2007, Santana et al., 2008a]. Missions have been tested in simulation and also with the all-terrain robot Ares (see a image of Ares in Figure 4.1).

The Player/Stage simulator [Gerkey et al., 2001, Gerkey et al., 2003] was used as a simulation environment. This is used to study how the system works without being dependent on the 'real' robot. This also allows the creation of situations that cannot be created in the real world because of cost, time, or the uniqueness of a resource. A simulator helps to do fast robot prototyping.

The Player Project [1] enables research in robot and sensor systems. The Player robot server is probably the most widely used robot control interface in the world. Its simulation back-end, Stage and Gazebo, are also very widely used. Released under the GNU General Public License, all code from the Player/Stage project is free to use, distribute and modify. Player is developed by an international team of robotics researchers and used at labs around the world.

[1] http://playerstage.sourceforge.net

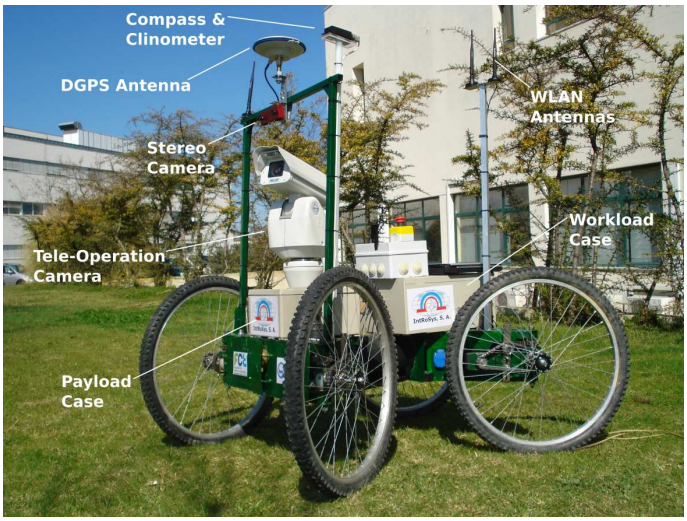

Figure 4.1: Ares Robot

4.1 Enabling Technologies

Three key technologies have been considered in the prototype implementation:

4.1.1 JADE - Supports the implementation of the multi-agent system for teamwork. JADE complements the workflow design which is only static definition of a team plan. The workflows are instantiated, and executed using this multi-agent platform.

4.1.2 Protégé - The general knowledge base is supported by this technology. It includes frame-based concepts like time and space, and it is spread all over the system. System engineers define and maintain the knowledge base using the Protégé tool.

4.1.3 TWE - It is a workflow design tool. It was extended to be able to support human-robot teamwork. All the static information used in this tool is collected from Protégé. This tool has a mission middle agent which is used to search for entities able to perform a role in a specific mission. This tool is also used to give to an operator the progress of the mission (i.e. mission status) at each moment.

4.1.1 JADE

JADE (Java Agent DEvelopment framework) [2] is a software framework fully implemented in Java language [Bellifemine et al., 1999, Bellifemine et al., 2003]. It supports the implementation of multi-agent systems through a middleware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. JADE implements a set of mechanisms used in this prototype implementation, such as: (1) Yellow pages service used by the *mission middle agent* when it is searching for entities able to provide a given service (e.g. searching for a human able to perform a robot-operator role) ; (2) Agent Communication Language (ACL) messages which are used in all the interactions between the software agents performing a specific mission (e.g. transition, data-flow messages);

4.1.2 Protégé

Protégé [3] is a free, open-source platform with a suite of tools to construct domain models and knowledge-based applications with ontologies [Gennari et al., 2003]. Protégé implements a rich set of knowledge-modelling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications. The Protégé platform supports two main ways of modelling ontologies via the Protégé-Frames and Protégé-OWL editors. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema.

An ontology describes the concepts and relationships that are important in a particular domain, providing a vocabulary for that domain as well as a computerized specification of the

[2] http://jade.tilab.com
[3] http://protege.stanford.edu/

meaning of terms used in the vocabulary. Ontologies range from taxonomies and classifications, database schemes, to fully axiomatized theories. In recent years, ontologies have been adopted in many business and scientific communities as a way to share, reuse and process domain knowledge. Ontologies are now central to many applications such as scientific knowledge portals, information management and integration systems, electronic commerce, and semantic web services.

4.1.3 TWE

TWE (Together Workflow Editor community edition) is provided by Enhydra JaWE [4]. TWE[5] is the first graphical Java Workflow Editor fully implementing WfMC (Workflow Management Coalition) XPDL-Specifications (XML Process Definition Language). It can be used to edit and view every XPDL file which conforms to WfMC XPDL specifications. Advanced features like the consistency guided property editor dialogues for all XPDL objects make it really easy to create valid XPDL files. The representation of workflow participants as ”swimlanes” in the editor graphics gives a comprehensive overview of process responsibilities.

Here TWE was extended (TWE*) to implement the WFDM. (TWE*) provides all the previously presented interactions with the *mission middle agent*. In addition, the tool has been extended to include novel concepts too, such as the *data-flow link* one. The access to the system's ontology, stored in a Protégé knowledge base, has been integrated in the tool so that the set of possible activities, its input and output parameters, and other attributes could be dynamically accessed when designing a mission template. The outcome of this tool is strongly typed.

4.2 Case Studies

In order to assess the applicability of the proposed architecture, two case studies have been selected:

[4] http://jawe.enhydra.org
[5] http://www.together.at/together/prod/twe/index.html

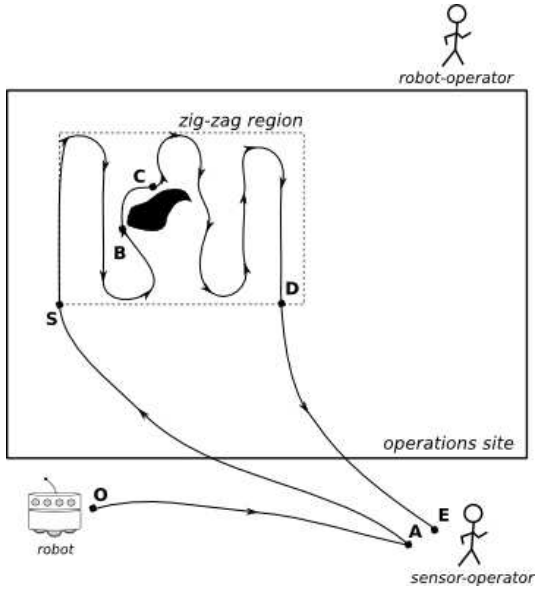

Figure 4.2: Demining case study illustration

1. Region scanning with a mobile robot carrying a scent sensor to detect minefields and two human operators helping the robot mission.

2. Surveillance of a terrain by a robot, travelling between two points of interest to be monitored, and two human operators helping the robot in its mission.

4.2.1 Demining Case Study

This case study consists in scanning a terrain with a mobile robot carrying a scent sensor to detect minefields. It is defined as a high-level task involving two humans, viz. one robot operator plus one sensor operator, and one robot. The goal is to determine if a given terrain is a minefield. When the mission starts the robot is equipped with a sensor able to determine the probability of a field to contain land mines. After analysing the terrain, the sensor is brought back to the sensor operator, which typically is located in a safe location away of the potential minefield. The robot operator, also remote to the operations site, helps the robot whenever needed.

Figure 4.2 illustrates the case study, whereas Figure 4.3 depicts how the mission workflow looks like in the GUI of the WFDM tool.

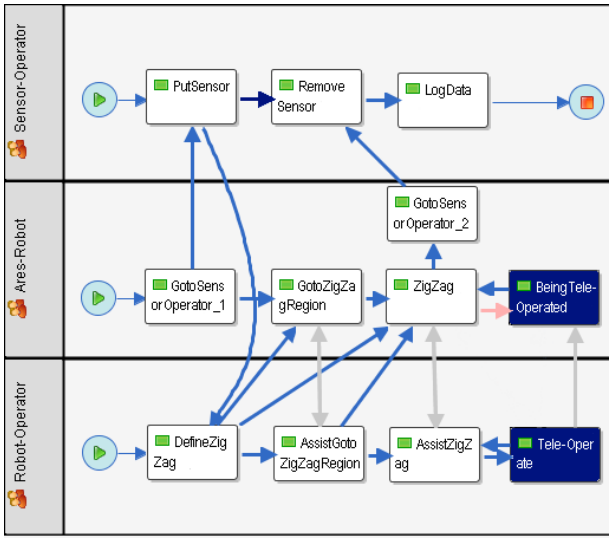

Figure 4.3: Demining case study workflow.

In this case study, the robot first needs to move to the operator that is handling the scent sensor in order to get the sensor (*GotoSensorOperator_1*). After reaching A, the activity *GotoSensorOperator_1* terminates and activates *PutSensor* activity, in which the sensor operator equips the robot with the sensor. Then the robot operator defines a zig-zag behaviour (i.e. a set of parallel lanes to be followed in a sequential manner) using the graphical interface of activity *DefineZigZag*. Afterwards the robot starts moving toward S in the direction of the defined zig-zag region (*GotoZigZagRegion*), while being modulated by the robot operator when necessary (*AssistGotoZigZagRegion)*. The zig-zag specifications are passed to the robot through a transition message as input parameters (e.g. S is provided as input parameters), whereas the *GotoZigZagRegion* modulating signal is passed through a data-flow link. As soon as the robot reaches S, the zig-zag behaviour is activated (*ZigZag*), which is also assisted by the robot operator (*AssistZigZag*). An example of assistance is "change to the next lane". If the robot departs too much from the lane being followed (e.g. as in B), for instance caused by the presence of a large obstacle, then *ZigZag* terminates with an exception. In response, the robot passes to tele-operation mode (*BeingTeleOperated*) and the current robot operator's activity is terminated (by data-flow).

Then, the robot operator is called to tele-operate the robot (*Tele-Operate*). In this case, the *Tele-Operate* provides *BeingTele-Operated* with tele-operation commands as data-flow. This

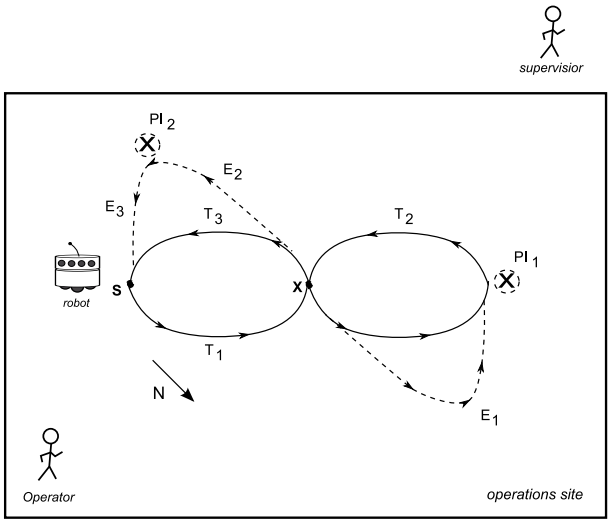

Figure 4.4: Surveillance case study illustration

is the current situation illustrated in Fig. 4.3. As soon as the operator considers the robot is again in a convenient position to resume its autonomous zig-zag behaviour (e.g. as in C), *Tele-Operate* terminates, which in turn requests *BeingTele-Operated* to terminate as well. Being again in autonomous zig-zag behaviour, the robot eventually reaches the end point (D) of the zig-zag region and *ZigZag* terminates. Then, the robot moves toward the sensor operator (*Goto-SensorOperator_2*), leaving the sensor there (*RemoveSensor*), whose data is logged (*LogData*). All *Goto** activities are of the same type *GotoXY*.

4.2.2 Surveillance Case Study

This case study consists in the cycle surveillance of a terrain, which has two points of interest to be monitored. It is defined as a high-level task involving two humans participants, viz. one supervisor plus one robot operator (they may be the same physical entity), and one robot, with a camera as payload. The goal is to watch the two points of interest in the region. The robot operator, also remote to the operations site, helps the robot if this leave the path defined. The supervisor is required to choose what is the next path when it is called for that purpose.

Figure 4.4 illustrates the case study, whereas Figure 4.5 depicts how the mission workflow looks like in the GUI of the WFDM tool.

In this case study, the robot starts from the point S and follows the track one (T_1) (*Track1*),

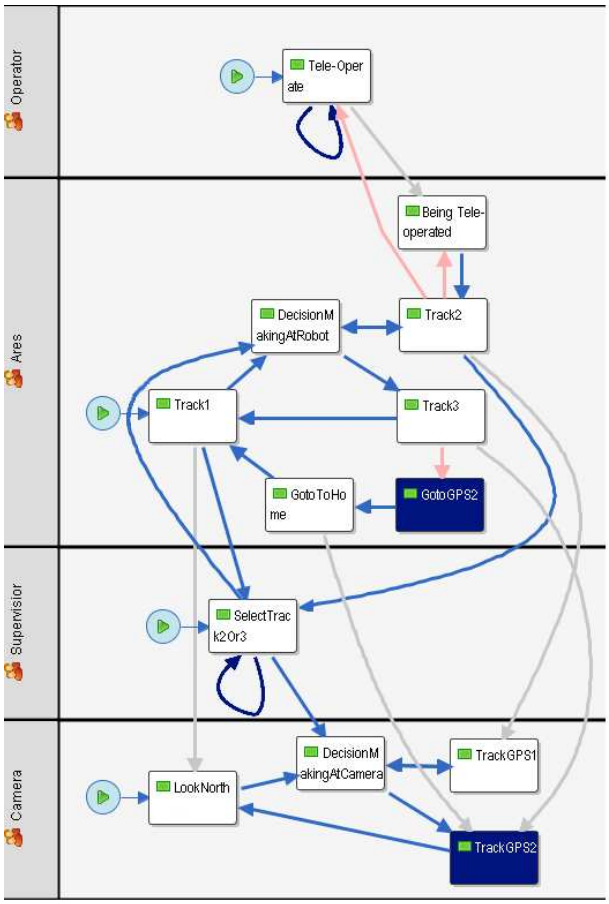

Figure 4.5: Surveillance case study workflow

looking with the camera to north (N) (*LookNorth*). After reaching X, the supervisor is required to choose the next track to be made (*SelectTrack2Or3*).

If the supervisor picks the track two, the robot starts to move, following the track two (T_2) (*Track2*), with the camera seeking the point of interest one (PI_1) (*TrackGPS1*). But for some reason the robot can leave the path T_2, then the robot operator is called to tele-operate the robot (*Tele-Operate*). The *Tele-Operate* provides *BeingTele-Operated* with tele-operation commands as data-flow. The operator drives (E_1) the robot back to the trajectory and the robot resume its autonomous "follow path" behaviour (*Track2*), terminating the activity *Tele-Operate*, which in turn requests *BeingTele-Operated* to terminate as well.

If the supervisor picks the track three, the robot starts to follow the track three (T_3) (*Track3*) and the camera seeks the point of interest two (PI_2) (*TrackGPS2*). Then if the robot leaves the path T_3, first it performs a *GotoXY* (E_2) to the point of interest two (PI_2) (*GotoGPS2*). This current situation is illustrated in Figure 4.5. Then the robot performs another *GotoXY* (E_3) to home(*GotoToHome*), which it is the S point.

The *DecisionMakingAtRobot* and *DecisionMakingAtCamera* are system activities to emulate the decision made by the supervisor at *SelectTrack2Or3*. This has been implemented due to the problem of parallelism within the same participant.

4.3 Experimental Results

The two experiments described above are intended to demonstrate the capabilities and weaknesses of the KBS_HRT.

Domain experts without any knowledge about workflows had some problems in the mission design, but with a little help of a workflow designer, explaining what is a participant, an activity, a transition and a data-flow and their relationships, everything became clarified. It is one of the reasons because workflows are used in human organisations, they are very intuitive to use.

The domain expert had no difficulty to allocate the physical entities to their roles at the beginning of the mission (e.g. see Fig. 4.6).

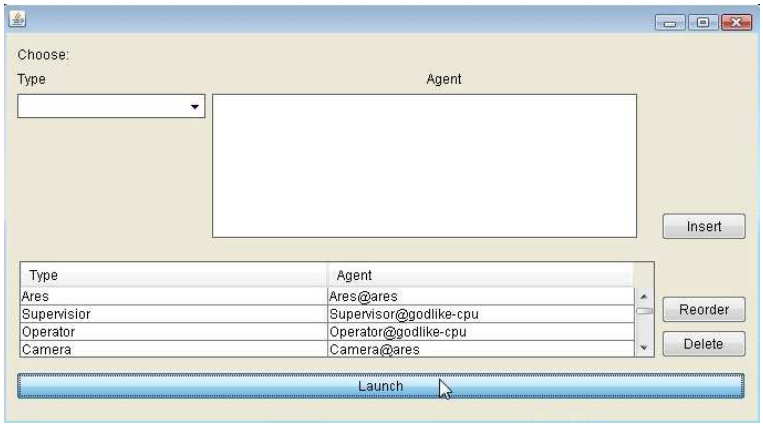

Figure 4.6: Physical entities allocation frame

From both experiences it became notorious that the mission design evolves from the experience in the mission field. Only viewing the field and the difficulty of the robot to execute a specific mission, the domain expert could draw some of the exceptions that in the initial design mission were not covered. An example, it is in demining case study, whereby the *tele-operate* and *being tele-operated* activities do not belonged in the first mission design. These activities were designed only when the domain expert saw that the robot departed too much from the lane being following, caused by a presence of a large obstacle (see Figure 4.3).

During the surveillance tests, a new input parameter on activity *goto XY* was added. It was the *path following name*. When in *goto XY* activity *path following name* parameter is defined, the path initial XY coordinates are used as the XY where the robot will go. That was very simple to add, to be used in a WFDM tool. Just had to add this parameter to activity to the ontology. After make the update of knowledge base by the WFDM tool, it is ready to be used, with no reprogramming in WFDM tool.

Normally an operator when uses for the first time the personal assistant GUI has difficulty to understand how to submit data-flow or a final activity order, having to be helped by someone more experienced. On the other hand, having overcome the difficulty up, the operator can very well understand what it must do to complete its work.

See Figures 4.7 and 4.8 from the tests field.

Figure 4.7: Ares being tele-operated (left) by a operator at the control centre (right)

Figure 4.8: Ares performing a *path following* activity while it looks to north with the surveillance camera

4.4 Discussion

From the two case studies, the following inferences were extracts:

(+) Mission design: A user with a limited knowledge on robotics, can define a mission intuitively using WFDM, almost only using the mouse, avoiding complex scripting languages. Keyboard is only needed if the user defines default values in activities' parameters;

(+) Physical entities allocation: Quick and easy allocation of physical entities to participants' role at the beginning of the mission. Mission middle agent can lookup for ME proxy

agents which represents a physical entity able to fulfil a role in a specific mission. If more than one physical entity can perform a specific role, the user only has to match to that the physical entity which he desires;

(+) Stereotyped mission awareness: The information provided to operators is sufficed for a proper decision making. A simple frame with activity's inputs to see and obligatory outputs to fill does not give much space for operator to mistake. Additional tools to use in a determinate activity to augment operator's task awareness are suggested in operator's frame also;

(+) Flexible mission awareness: WFDM tool showed to be a good option to display the status of the mission, i.e. to provide global mission awareness, when the user needs it. Namely, it happens when the user needs to know, who are the participants which aborted their mission, trying to understand why that occurs.

(+) Mission adaptation: WFDM tool helps to see where the mission is failing. Then, user can do a fast adaptation of the mission, putting or deleting activities or transitions in the participants roles in WFDM. For example, when the domain expert sees that in a determinate *path following* activity always occurs an exception, because the robot leaves the path trajectory. Domain expert can put on *path following* activity an outgoing transition with the exception termination code, connecting that to a *goto XY* activity, passing by parameter the *path following* name, which in *goto XY* robot activity will used to fill its XY coordinates, where the robot will go. At the finish of *goto XY* activity a transition can be linked to *path following* activity, to start that activity again.

However some limitations were found. Operator's personal assistant GUI needs a better design to facilitate operator's work. Another limitation found, it is when the number of tasks and participants increase, the visual information of the workflow is overwhelming. User confusion and messages delay were both induced when in key moments of the mission network drop-outs occurred.

Chapter 5

Conclusions and Future Work

In this chapter of this dissertation are providing a set of conclusions, where it is mentioned what are the contributions of this dissertation and what are the weaknesses. After, in the future work section is pointed some directions for future research.

Before proceeding, it is fundamental to say that the author is fully aware of the limitations of this work, which were deduce from a set of experiments on simulation and in tests field.

5.1 Conclusions

This dissertation contributes with a pioneering step toward exploitation of knowledge based techniques in human-robot teamwork. The goal was to enable cooperative execution of stereotyped tasks, essential in demanding scenarios, where timely decisions are required. The cooperative workflow formalism, usually employed for business oriented human organisations, was selected.

Its cooperative workflow based nature allows a seamless integration between plan specification, execution, adaptation and monitoring. This is of particular importance to enable on-field knowledge update and reuse. Clear distinctions on the way humans and robots interact required the workflow formalism to be adapted. Some adaptations were suggested, with particular focus in data-flow links. Activities in different participants are allowed to exchange messages while

executing, enabling the implementation of tightly coupled coordination. This ability is usually disregarded in theoretical teamwork and cooperative workflow fields works, which typically focus on high-level tasks with sporadic interactions. Although multi-robots literature is more concerned with tightly coupled coordination, it lacks a structural approach to cope with the human factor.

The design of demining and surveillance missions, as complex as presented in the case studies, posed no major challenges to the user. However, for more complex tasks it was clear need for workflow nesting capabilities.

Stereotyped mission awareness information provided to operators is sufficed for a proper decision making, although it was notorious the need for some improvements. In particular, operator's personal assistant GUI needs a better design to facilitate operator's work.

Flexible mission awareness accessed through the WFDM tool allows operators to have a quick grasp of the overall mission status. However, when the number of tasks and participants increases, the visual information of the workflow is overwhelming. It is simply too much information to be managed by the user.

User confusion and messages delay were both induced when in key moments of the mission network drop-outs occurred. Confusion can be mitigated with adjustable autonomy [Goodrich et al., 2001] techniques (e.g. automatically changes from direct motor control to reflexive control).

This multi-agent system presented here explicitly considers the human. First, the workflow formalism is usually employed by humans and consequently natural to them. Second, by considering different message exchanging protocols and system level activity parameters, both human and robot asymmetries are explicitly taken into account. Third, human readable information is formally attached to the ontology concepts used by the human participant. Being supported by JADE, agents run in JAVA virtual machines. Users should access the system in the operating system that they feel more comfortable with (e.g. Windows or Linux). In addition, it facilitates the porting of system's components, in particular those related to human-machine interfaces, to unconventional computational units (e.g. PDAs).

Furthermore, to validate the proposed knowledge-based teamwork model and its architecture implementation, were realized two realistic case studies.

5.2 Future Work

Below, some research opportunities based on this dissertation are enumerated:

1. Nested workflows and automatic hiding of details are required to increase the modularity and abstraction level of the mission design and monitor.

2. Reallocating of team members in different teams. This problem is closely related to previous item.

3. More robustness against communication channels degradation will also be tackled.

4. A systematic analysis of the system's ability to generate awareness is still lacking, too. Awareness analysis techniques (e.g. [Drury et al., 2007]) can be a way to fill this gap.

5. Also in the horizon is the integration of learning and planning mechanisms in this framework. Dynamic invocation of team sub-plans will be pursued as a way of applying well known stereotyped problem solvers (i.e. mission templates) to the situation at hand.

Bibliography

[Arkin, 1998] Arkin, R. C. (1998). *Behavior-Based Robotics*. The MIT Press.

[Barata, 2005] Barata, J. (2005). *Coalition Based Approach For Shop Floor Agility*. ORION, Lisboa.

[Barata and Camarinha-Matos, 2004] Barata, J. and Camarinha-Matos, L. M. (2004). A methodology for shop floor reengineering based on multiagents. In *BASYS*, pages 117–128.

[Bass, 1980] Bass, B. M. (1980). Individual capability, team performance, and team productivity.

[Bellifemine et al., 2003] Bellifemine, F., Caire, G., Poggi, A., and Rimassa, G. (2003). Jade: A white paper. *TILAB*, 3(3).

[Bellifemine et al., 1999] Bellifemine, F., Poggi, A., and Rimassa, G. (1999). Jade - a fipa-compliant agent framework. In *In Proceedings of PAAM-1999*, pages 97–108.

[Bonabeau et al., 1999] Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA.

[Camarinha-Matos, 2003] Camarinha-Matos, L. M. (2003). Infrastructures for virtual organizations - where we are. In *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, volume 2, pages 405 – 414.

[Cannon-Bowers et al., 1995] Cannon-Bowers, J. A., Tannenbaum, S. I., Salas, E., and Volpe, C. E. (1995). Defining team competencies: Implications for training requirements and strategies.

[Carlson and Murphy, 2005] Carlson, J. and Murphy, R. (2005). How ugvs physically fail in the field. *IEEE Transactions on Robotics*, 21(3):423–437.

[Chalupsky et al., 2001] Chalupsky, H., Gil, Y., Knoblock, C., Lerman, K., Oh, J., Pynadath, D., Russ, T., and Tambe, M. (2001). Electric elves: Applying agent technology to support human organizations.

[Chen and Sycara, 1998] Chen, L. and Sycara, K. (1998). WebMate: A personal agent for browsing and searching. In Sycara, K. P. and Wooldridge, M., editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 132–139, New York. ACM Press.

[Cohen and Levesque, 1991] Cohen, P. R. and Levesque, H. J. (1991). Teamwork. 25:487–512.

[Correia and Garção, 1995] Correia, L. and Garção, A. S. (1995). Behavior based architecture with distributed selection. *NATO Advanced Study Institute – The Biology and Technology of Intelligent Autonomous Systems*, 144:377–389.

[Drury et al., 2003] Drury, J., Scholtz, J., and Yanco, H. (2003). Awareness in human-robot interactions.

[Drury et al., 2007] Drury, J. L., Keyes, B., and Yanco, H. A. (2007). Lassoing hri: analyzing situation awareness in map-centric and video-centric interfaces. In *HRI*, pages 279–286.

[Farquhar et al., 1997] Farquhar, A., Fikes, R., and Rice, J. (1997). The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6).

[Gennari et al., 2003] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., and Tu, S. W. (2003). The evolution of protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123.

[Gerkey et al., 2003] Gerkey, B., Vaughan, R., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems.

[Gerkey et al., 2001] Gerkey, B., Vaughan, R., Sty, K., Howard, A., Sukhatme, G., and Mataric, M. (2001). Most valuable player: A robot device server for distributed control.

[Godart et al., 2000] Godart, C., Charoy, F., Perrin, O., and Skaf-Molli, H. (2000). Cooperative workflows to coordinate asynchronous cooperative applications in a simple way. In *ICPADS '00: Proceedings of the Seventh International Conference on Parallel and Distributed Systems (ICPADS'00)*, page 409, Washington, DC, USA. IEEE Computer Society.

[Goodrich et al., 2001] Goodrich, M., Olsen, D., Crandall, J., and Palmer, T. (2001). Experiments in adjustable autonomy.

[Grosz and Kraus, 1996] Grosz, B. and Kraus, S. (1996). Collaborative plans for complex group action.

[Grosz and Kraus, 1998] Grosz, B. and Kraus, S. (1998). The evolution of sharedplans.

[Hollingsworth, 1995] Hollingsworth, D. (1995). The workflow reference model. Technical report, Workflow Management Coalition.

[Jennings, 1995] Jennings, N. R. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240.

[Lenox et al., 2000] Lenox, T., Hahn, S., Lewis, M., Payne, T., and Sycara, K. (2000). Agent-based aiding for individual and team planning tasks.

[Long et al., 2005] Long, M. T., Gage, A., Murphy, R. R., and Valavanis, K. P. (2005). Application of the distributed field robot architecture to a simulated demining task. In *ICRA*, pages 3193–3200. IEEE.

[Murphy et al., 2001] Murphy, R., Lisetti, C., Irish, L., Tardif, R., and Gage, A. (2001). Emotion-based control of cooperating heterogeneous mobile robots.

[Nair et al., 2003] Nair, R., Tambe, M., and Marsella, S. (2003). Role allocation and reallocation in multiagent teams: Towards a practical analysis. In *Proceedings of Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS-03)*.

[Nourbakhsh et al., 2005] Nourbakhsh, I., Sycara, K., Koes, M., Young, M., Lewis, M., and Burion, S. (2005). Human-robot teaming for search and rescue.

[Okamoto et al., 2006] Okamoto, S., Scerri, P., and Sycara, K. (2006). Toward an understanding of the impact of software personal assistants on human organizations. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 630–637, New York, NY, USA. ACM.

[Paris et al., 2000] Paris, C. R., Salas, E., and Cannon-Bowers, J. A. (2000). Teamwork in multi-person systems: a review and analysis. Technical Report 8.

[Parker, 1998] Parker, L. (1998). Alliance: An architecture for fault-tolerant multi-robot cooperation.

[Pechoucek et al., 2000] Pechoucek, M., Marík, V., and Stepankova, O. (2000). Coalition formation in manufacturing multi-agent systems. In *DEXA '00: Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, page 241, Washington, DC, USA. IEEE Computer Society.

[Rich and Sidner, 1997] Rich, C. and Sidner, C. L. (1997). COLLAGEN: When agents collaborate with people. In Johnson, W. L. and Hayes-Roth, B., editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 284–291, New York. ACM Press.

[Salas et al., 1992] Salas, E., Dickinson, T. L., Converse, S., and Tannenbaum, S. I. (1992). Toward an understanding of team performance and training.

[Santana et al., 2007] Santana, P., Barata, J., and Correia, L. (2007). Sustainable robots for humanitarian demining. *Int. Journal of Advanced Robotics Systems (special issue on Robotics and Sensors for Humanitarian Demining)*, 4(2):207–218.

[Santana et al., 2008a] Santana, P., Cândido, C., Santos, P., Almeida, L., Correia, L., and Barata, J. (2008a). The ares robot: case study of an affordable service robot. In *Proc. of the European Robotics Symposium 2008 (EUROS'08)*, Prague.

[Santana et al., 2008b] Santana, P., Salgueiro, M., Santos, V., Correia, L., and Barata, J. (2008b). Awareness in human-robot teams: A multi-agent approach. Technical report, University of Lisbon and IntRoSys S.A., Lisboa, Portugal.

[Santana et al., 2008c] Santana, P., Salgueiro, M., Santos, V., Correia, L., and Barata, J. (2008c). A knowledge-based component for human-robot teamwork. In *Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics*, Madeira, Portugal. Springer.

[Santos et al., 2007] Santos, V., Cândido, C., Santana, P., Correia, L., and Barata, J. (2007). Developments on a system for human-robot teams. In *Proc. of the 7th Conf. on Mobile Robots and Competitions*, Paderne, Portugal.

[Scerri et al., 2003] Scerri, P., Pynadath, D., and Tambe, M. (2003). Towards adjustable autonomy for the real world.

[Sierhuis et al., 2005] Sierhuis, M., Clancey, W. J., Alena, R. L., Berrios, D., Buckingham Shum, S., Dowding, J., Graham, J., van Hoof, R., Kaskiris, C. A., Rupert, S., and Tyree, K. S. (2005). NASA's Mobile Agents Architecture: A Multi-Agent Workflow and Communication System for Planetary Exploration. In *'i-SAIRAS 2005' - The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, volume 603 of *ESA Special Publication*.

[Sukhatme et al., 2001] Sukhatme, G., Montgomery, J., and Vaughan, R. (2001). Experiments with cooperative aerial-ground robots.

[Sycara and Lewis, 2002] Sycara, K. and Lewis, M. (2002). Integrating agents into human teams. In *Proc. of the Human Factors and Ergonomics Society's 46th Annual Meeting*.

[Sycara and Sukthankar, 2006] Sycara, K. and Sukthankar, G. (2006). Literature review of teamwork models. Technical Report CMU-RI-TR-06-50, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[Tambe, 1997] Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124.

[Traum et al., 2003] Traum, D., Rickel, J., Gratch, J., and Marsella, S. (2003). Negotiation over tasks in hybrid human-agent teams for simulation-based training.

[WfMC, 2001] WfMC (2001). *Workflow Process Definition Interface - XML Process Definition Language*. Number WFMC-TC-1025.

[Yanco and Drury, 2004] Yanco, H. and Drury, J. (10-13 Oct. 2004). "where am i?" acquiring situation awareness using a remote robot platform. *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 3:2835–2840 vol.3.