# An Abductive Multi-Agent Framework for Distributed Service Coordination and Reasoning in Emergency Scenarios

Sergio Storari
ENDIF - University of Ferrara
Via Saragat, 1 - 44100 - Ferrara - Italy
Email: sergio.storari@unife.it

Anna Ciampolini and Paola Mello
DEIS - University of Bologna
Viale Risorgimento, 2 - 40136 - Bologna - Italy
Email: {aciampolini,pmello}@deis.unibo.it

***Abstract*—In this paper we will show how an abductive multi-agent approach could be successfully exploited in a emergency scenario. To this purpose we will present MeSSyCo, a multi-agent system that integrates and coordinates heterogeneous medical and non medical services. MeSSyCo agents can perform different tasks such as diagnosis, service coordination and intelligent resource allocation. The MeSSyCo coordination infrastructure based on abductive reasoning allows several distinct specialized service providers to be aggregated and properly coordinated in order to perform more complex medical and non medical tasks.**

## I. Introduction

Emergency scenario management is usually a complex and dynamic task since it requires the coordination of several different organizations and resources (e.g., resource depots, hospitals, fire departments, etc.). Such organizations are typically distributed on the territory and provide different sets of services so, in order to perform such complex tasks, each of these entities needs to be properly coordinated with the others. In this way the results obtained by the execution of different services could be collected, integrated and possibly returned as a unique final response. Therefore, in such cases it should be very important to make different independent organizations interact and coordinate.

The literature has proposed multi-agent systems as an appropriate solution to face these issues (e.g., [1]). These systems should guarantee the autonomy of involved entities, facilitate the communication and the coordination between them, and proactively provide information and services to the involved entities.

Following these considerations, this paper presents MeSSyCo, a multi-agent system whose main purpose is the integration of heterogeneous knowledge-based services. This system can be used to represent virtual organizations: each service provider is encapsulated into a MeSSyCo agent; the coordination infrastructure allows the interaction of several (possibly heterogeneous) agents. Services may be implemented either by traditional programming technologies or by using knowledge-based systems with automatic reasoning mechanisms. In this way, the MeSSyCo system is able to perform complex tasks such as intelligent resource allocation, distributed service coordination and distributed diagnosis.

MeSSyCo agents may take advantage from some well known automatic hypothetical reasoning mechanisms such as logic based abduction [2], [3] that allows reasoning in presence of incomplete knowledge. This is the case, for instance, of medical diagnosis: given a set of symptoms, abductive reasoning can produce a set of plausible diagnosis for them. However, the MeSSyCo architecture allows to integrate also non abductive agents.

In this paper we exploit how MeSSyCo can be used to face several issues related to emergency scenario management.

## II. Medical Services Synergy and Coordination

MeSSyCo is a multi-agent system for Medical Services Synergy and Coordination that can accomplish several tasks, such as, intelligent resource allocation, distribute service coordination and distributed diagnosis.

Intelligent resource allocation is performed by choosing, among entities offering the same service in a specific context, the most *suitable* one.

Distributed service coordination is used to answer to complex service requests which require the identification of the agents capable of providing a particular set of services and resources.

In distributed diagnosis, the knowledge bases of different agents (each representing a virtual single expert) are consulted in order to produce a final diagnosis that will take advantage from a more comprehensive, wide and heterogeneous knowledge and can be more precise. This is very useful, for example, in the medical domain when a single agent is maybe unable to explain all the patient symptoms.

MeSSyCo can be considered a JADE [4] implementation of ALIAS [5] with some extensions regarding the distributed probabilistic reasoning [6] and the identification of most appropriate service provider among the available ones.

Its architecture, shown in Figure 1, is characterized by two kind of agents: the application agents and the system agents.

Each entity providing services within an organization is modeled by an application agent (shown in Figure 1 as *Ag1*,

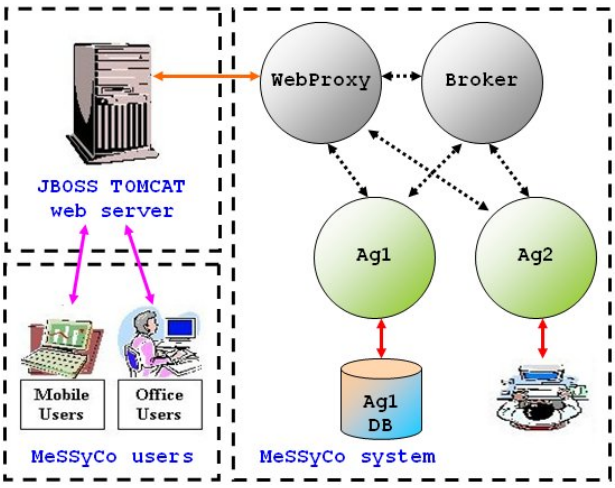


Fig. 1. Schema of the MeSSyCo system architecture.

*Ag2*) which provides several services. Each application agent contains a reasoning module, described in Section III, which stores the knowledge used to provide each agent service. This knowledge may be elicited, for example, from clinician interviews or medical literature. It is also necessary to express how these entities interacts with the others in order to accomplish their objectives.

System agents provide the services necessary to the correct functioning of the whole system. These agents, shown in Figure 1, are the *Broker* agent and the *WebProxy* agent.

The *Broker* agent is an extension of the *Directory Facilitator* agent defined by FIPA [7] that registers all services provided by virtual organization agents and offers a "yellow pages" service. A MeSSyCo agent can also specify a set of service requirements that the Broker uses to identify the most suitable agents among the available ones.

The *WebProxy* agent allows web users to access MeSSyCo services.

## III. Logic-based Abduction for Distributed Service Coordination and Reasoning

In MeSSyCo we use abduction as the main form of agent reasoning. To this purpose, each application agent encapsulates an *abductive logic program* [2], [3]. An abductive logic program is a triple $\langle P, \mathcal{A}, IC \rangle$ where $P$ is a logic program possibly with abducible atoms in clause bodies; $\mathcal{A}$ is a set of *abducible predicates*, i.e., *open* predicates which can be used to form explaining sentences; $IC$ is a set of integrity constraints: each constraint is a denial containing at least one abducible.

Given an abductive program $\langle P, \mathcal{A}, \mathcal{IC} \rangle$ and a formula $G$, the goal of abduction is to find a (possibly minimal) set of atoms $\Delta$ which together with $P$ entails $G$. It is also required that the program $P \cup \Delta$ is consistent with respect to $IC$.

Abduction has been extended toward a multi-agent approach in ALIAS [5]. The MeSSyCo architecture designs and implements a coordination multi-agent based framework which follows the ALIAS approach. To this purpose, we equip each agent with a distinct abductive logic program.

Agents can dynamically join into groups (from now on, we refer to the group of agents with the term *bunch*), with the purpose, for instance, of finding the solution of a given goal in a *collaborative* way. In this perspective, although the set of program clauses and integrity constraints might differ from agent to agent, we assume that the set of abducible predicates (default predicates included) is the same for all the agents in a bunch. This implies that when proving a given goal, if an agent $A$ assumes a new hypothesis $h$, all the agents belonging to the same bunch must check the consistency of $h$ with their own integrity constraints. These checks could possibly raise new hypotheses, whose consistency within the bunch has to be recursively checked. Therefore, in MeSSyCo, the abductive explanation of a goal within a bunch of agents is a set of abduced hypotheses, agreed by all agents in the bunch.

MeSSyCo agent behavior is expressed by means of the *Language for AbductIve Logic Agents* (LAILA, for short). This language, described in details in [8], allows to model agent actions and interactions in a logic programming style. In particular we will focus on agent social behavior, and especially on how each agent can request demonstration of goals to other agents in the system. To this purpose, two composition operators are available: the collaborative AND operator (`&`) and the competitive operator (`;`) that can be used by each agent to express and coordinate abductive queries to other (set of) agents. A competitive query is useful when the same service can be provided by several agents, in order to collect all the answers provided by such agents. A collaborative query is useful when for the the execution of a goal we need a set of services that are not provided by only one agent.

The language provides also a communication operator (`>`) that is used to submit service request (queries) to other agents.

## IV. Distributed Coordination in an Emergency Scenario

Emergency scenarios are usually very difficult to manage since they are characterized by an extreme complexity and dynamism. MeSSyCo can be used to support the emergency manager in the identification and coordination, among the available resources (e.g. resource depots, hospitals, fire departments, etc.), of the most appropriate and consistent ones. This is made by using the distributed abductive reasoning approach described in Section III.

This section describes, by mean of a simple but meaningful example, the implications introduced by MeSSyCo in emergency scenario management.

First of all, we introduce the example, then we present how MeSSyCo manages the single service reasoning and several issues related to the coordination of different services.

*Example Scenario.* In example scenario, we suppose to be in the Emilia Romagna region of Italy, an emergency manager is in charge of reaching an emergency site, evaluating the problems related to the emergency and requesting the

resources (e.g., vehicles, work teams) needed to solve the emergency. Resources are located in several Civil Protection Center/Depots within the Emilia Romagna region, available for the emergency Control Center if needed.

We have modeled this example scenario with three MeSSyCo agents, the Control Center, the Bologna depot and the Civil Protection of Ferrara. The Control Center Agent (CC) models the Emergency Control Center. It offers the `emCond(COND)` service that is used to adapt the resources used to particular emergency condition and/or requirements (e.g., environment conditions, special instruments). The Civil Protection Ferrara Agent (CP-FE) models the resources offered by the Civil Protection of Ferrara. In particular, it offers the `cat(DAY)` service, that is used to request if there is a caterpillar free on DAY, and the `team(DAY,MINWORK)` service, that is used to request if there is a work team of at least MINWORK workers free on DAY. The Depot Bologna Agent (DP-BO) models the resources offered by the vehicle Depot of Bologna. In particular, it offers the `cat(DAY)` service.

*Single Service Reasoning.* MeSSyCo agents use abductive reasoning in order to hypothesize solutions for their services. Suppose for example that the emergency manager asks the `cat('11:09:07')` service to CP-FE through the query:

```
PC-FE > cat('11:09:07')
```

PC-FE receives the request and starts and abductive derivation process on its knowledge base in order to hypothesize solutions for this service.

The part of its knowledge base related to the `cat(DAY)` service is the following:

```
cat(DAY) ← catN('cat1'), free('cat1',DAY).
cat(DAY) ← catN('cat2'), hvyCat, free('cat2',DAY).
free(X,DAY) ← not work(X,DAY).
work('cat1','11:09:07').
```

The abductive derivation process works as follows. Initially the set $\Delta_0$ of abduced assumptions is empty and `catN`, `hvyCat` and `noGPSCat` are abducible predicates. Given the `cat('11:09:07')` goal, named $G_0$, the abductive resolution process unifies it with the head of the first rule and collects the body literals of this rule as new query to prove. The new goal is then:

$G_1$= ← `catN('cat1'), free('cat1','11:09:07')`.

and $\Delta_1 = \Delta_0$.

The first literal `catN('cat1')` is then removed from $G_1$ and the resolution process start again on the new single literal goal $G_2$= ← `catN('cat1')`. The knowledge base for this literal is incomplete (does not have any information about this predicate). But in the given example this is an abducible predicate, so the abductive derivation instead of failing to prove this goal, adds the literal `catN('cat1')` temporarily to $\Delta_1$ which becomes $\Delta_2 = \{$`catN('cat1')`$\}$. A consistency derivation is activated on $\Delta_2$. Since PC-FE contains no integrity constraints, the consistency check is successful and the abductive derivation process continues its proof on the remaining goal $G_3$= ← `free('cat1','11:09:07')` and the set of abduced predicates $\Delta_3 = \Delta_2 = \{$`catN('cat1')`$\}$.

Given the `free('cat1','11:09:07')` goal, the abductive resolution process unifies it with the head of the fourth rule and the new query to prove becomes $G_4$= ← `not work('cat1','11:09:07')` and $\Delta_4 = \Delta_3 = \{$`catN('cat1')`$\}$.

The prove of $G_4$ fails as `work('cat1','11:09:07')` is a fact in the knowledge base so the current abductive derivation fails (the caterpillar named 'cat1' cannot be hypothesized as a solution of the `cat('11:09:07')` service), and the abductive derivation process backtracks to $G_0$ and starts a new derivation considering the second rule.

The new goal to prove becomes

$G_5$=← `catN('cat2'), hvyCat, free('cat2','11:09:07')`.

with $\Delta_3 = \Delta_2 = \{\}$.

The abdcutive derivation process for $G_5$ succeeds as `catN('cat2')` and `hvyCat` (hvyCat stands for heavy caterpillar) are abducibles predicates in our example, and `'cat2'` is not working on `'11:09:07'`.

Therefore, PC-FE responds to the `cat('11:09:07')` request by notifying to the emergency manager one (minimal) set of hypotheses:

$\Delta_6$ = {`catN('cat2'), hvyCat`}

This solution expresses that PC-FE hypothesizes the provision of the `cat('11:09:07')` service by mean of the heavy caterpillar named 'cat2'.

*Distributed Service Coordination.* As described in Section III, local abductive reasoning has been extended to face distributed service coordination. This is useful in the context of emergency management when more than one service is required. Usually, resource coordination can be manually performed by emergency managers, constructing step by step resource combinations capable of covering all the necessary scenario requirements. During this construction step, partial resource combinations may become inconsistent with the ones necessary to face all the requirements, therefore backtracking should be performed in order to identify the final solution. This time consuming step increases its importance for scenarios in which the reaction time is a critical point.

Given scenario requirements, MeSSyCo interacts with all its agents in order to automatically find consistent solutions: it checks which agents can provide specific services, combines the resources they offers to provide that services and verifies the consistency of such combinations.

Consider, for example, that a landslide is happened. An emergency coordinator reaches the landslide location, observes the emergency context and issues a service request to block and repair the landslide. An example of service request can be the following:

```
{cat('11:09:07'), team('11:09:07',3), emCond('wet')}
```

MeSSyCo, interacting with the Directory Facilitator, identifies the agents which provide the requested services and try to

accomplish the whole service request by creating the following LAILA query Q:

```
(PC-FE > cat('11:09:07') ; DP-BO > cat('11:09:07') )
& PC-FE > team('11:09:07',4) & CC > emCond('wet')
```

The `cat` service is asked in competition between PC-FE and DP-BO because they can provide this service.

MeSSyCo, using the distributed abductive reasoning, answers to this complex service request hypothesizing two possible solutions, both consistent with the knowledge bases of the participating agents CC, PC-FE, and DP-BO:

```
SOL1: {catN('cat3'), teamN('FE1',5), not hvyCat}
SOL2: {catN('cat4'), noGPSCat, teamN('FE1',5), not
hvyCat}
```

where `catN` is used to identify a specific caterpillar, `teamN` a specific team, `noGPSCat` is used to indicate that the caterpillar proposed in the solution is not equipped with a GPS and `not hvyCat` is used to indicate that since the landslide location is wet, an heavy caterpillar is unusable.

*Intelligent Resource Allocation.* The emergency manager can choose among the previous solutions its preferred one. MeSSyCo also support such choice by mean of a suitability value. In particular, a knowledge base can be associated to the Broker in order to allow the computation of the suitability of an agent in providing a specific service in a specific context. For example, let us suppose that the emergency manager should ward an injuried man with heavy burns in a hospital and that two hospitals, one specialized in the management of burnt patients and one without specific skills, provide a warding service. When the emergency manager agent requires the warding service to the MeSSyCo Broker, specifying in the service request information that the patient is burnt, the Broker uses, if provided, its knowledge base to associate an higher suitability level to the agent which represents the hospital specialized in burnt management.

*Dynamic Solution Adaptation.* Adaptation to changing conditions is another feature allowed by the distributed abductive reasoning infrastructure. Suppose, for example, that, due to the worsening of the weather conditions on the landslide site, a GPS becomes a necessary equipment for all the caterpillars. Adding `emCond('GPSCat')` to the previous service coordination request, expressed by mean of the LAILA query Q, and supposing the constraint ← `emCond('GPScat'), noGPSCat.` to be part of the CC knowledge base, `SOL2` becomes inconsistent and the only applicable service request solution becomes `SOL1`.

MeSSyCo can also handle distributed diagnosis, but due to lack of space we cannot show an example of such functionality. Interested readers can find more detail in [6].

## V. Conclusion and Future Works

In this paper we focused on the definition and development of a multi-agent architecture for the management of heterogeneous medical knowledge-based services.

Abduction is an important technique in the MeSSyCo implementation as it is used both to express agent reasoning and to manage the coordination between different agents.

The literature proposes other multi-agent systems for health care service management (e.g., [9]) and a lot of systems for service coordination [10], [11]. The distinguishing feature of MeSSyCo is the use of distributed abductive reasoning. Thanks to this approach MeSSyCo can manage complex service request identifying, among different agents, the ones that combined can provide consistent solutions. It can also identify, among these solutions, the most suitable one w.r.t. the emergency scenario characteristics. Moreover, it can manage the dynamic aspects of an emergency scenario changing appropriately the proposed solutions.

We experimented MeSSyCo in the development of a simple emergency scenario in which distributed services are coordinated to support the emergency management.

Performances have been evaluated for distributed diagnosis and intelligent resource allocation scenarios in [12]. In the future we will complete the implementation of the system and the evaluation of its performance by using real emergency scenarios.

## Acknowledgment

This work has been partially supported by the PRIN 2005 project "Specification and verification of agent interaction protocols".

## References

[1] G. Cabri, F. De Mola, and R. Quitadamo, "Supporting a territorial emergency scenario with services and agents: A case study comparison," in *Proceedings of WETICE '06*. IEEE Computer Society, 2006, pp. 35–40.

[2] K. Eshghi and R. A. Kowalski, "Abduction compared with negation by failure," in *Proceedings of ICLP-89*, G. Levi and M. Martelli, Eds. MIT Press, 1989, pp. 234–255.

[3] A. C. Kakas and P. Mancarella, "Generalized stable models: a semantics for abduction," in *Proceedings of ECAI-90*. Pitman, 1990.

[4] "Jade framework," available at: http://sharon.cselt.it/projects/jade/.

[5] A. Ciampolini, E. Lamma, P. Mello, C. Stefanelli, and P.Torroni, "An implementation for abductive logic agents," in *Proceedings of AI*IA-99*, ser. LNAI, vol. 1792. Springer, 2000, pp. 61–71.

[6] A. Ciampolini, P. Mello, and S. Storari, "Distributed medical diagnosis with abductive logic agents," *AI*IA Notizie*, vol. XV, no. 3, pp. 12–21, Sep. 2002.

[7] "FIPA," available at: http://www.fipa.org/.

[8] A. Ciampolini, E. Lamma, P. Mello, and P. Torroni, "LAILA: A language for coordinating abductive reasoning among logic agents," *Computer Languages*, vol. 27, no. 4, pp. 137–161, February 2002.

[9] B. Lopez, S. Aciar, B. Innocenti, and I. Cuevas, "How multi-agent systems suport acute stroke emergency treatment," in *IJCAI Workshop on Agents Applied in Health Care*, 2005, pp. 51–59.

[10] E. Simperl, R. Krummenacher, and L. Nixon, "A coordination model for triplespace computing." in *COORDINATION*, ser. LNCS, vol. 4467. Springer, 2007, pp. 1–18.

[11] H. Helin, M. Klusch, A. Lopes, A. Fernandez, M. Schumacher, H. Schuldt, F. Bergenti, and A. Kinnunen, "Context-aware business application service co-ordination in mobile computing environments." in *AAMAS05 workshop on Ambient Intelligence - Agents for Ubiquitous Computing*, 2005.

[12] A. Ciampolini, P. Mello, and S. Storari, "Integration of medical services in the messyco agent system," Demo session at AAMAS 2004, 2004.