UPC
Universitat Politècnica de Catalunya

ETSETB
Escola Técnica Superior d'Enginyers de Telecomunicació de Barcelona

Demonstrating communication services based on autonomic self-organization

Author: Albano Bernadas Zurdo

Director: Antonio Manzalini
Co-director: Dr. Josep Solé Pareta, Dr. Salvatore Spadaro

December 2013

Demonstrating communication services based on autonomic self-organization

Acknowledgments Agradecimientos Gratitudine

En primer lugar quiero dar las gracias a las personas más importantes en mi vida, que son mis padres y mi hermana Laura. Ellos me han dado la vida, me han acompañado en todas sus etapas y me han convertido en la persona que soy, con todos sus defectos y virtudes.

Des de mi primer año en la universidad he tenido la suerte y el placer de compartir amistad, vivencias, apoyo y crecer en muchos sentidos con un grupo de personas increíbles. Gracias Duran, Albert, Oscar, Raquel, Javi, Patri, Víctor, Maite, Raúl, Ana, Roser, Chico, Pepo, Sara, Jenny y Jon por todos estos años juntos, y por los que quedan por venir. Con vosotros he descubierto el verdadero sentido de la amistad.

El año vivido en Italia ha sido una las experiencias que más me han aportado de todas las que he vivido. Definirlo podría resumirse con la palabra aventura. Aventura por todo lo que significó comenzar a vivir fuera de tu ciudad, país, cultura e idioma. Aventura por tener la oportunidad de comenzar de cero en un nuevo lugar y con gente desconocida. Por el reto de entrar por primera vez en una gran empresa como Telecom Italia. Desafíos que hacen madurar cuando pensabas que ya lo habías hecho y abrir la mente donde creías tenerla ya abierta. Desde entonces no he podido sino recomendar a todo aquel que estuviera en disposición de vivir una experiencia similar a no dudarlo. En esta aventura he tenido la suerte de compartirla con grandes compañeros de travesía. Gracias Julio, Héctor, Claudia y Santy por ser las personas generosas y auténticas que sois. Aún no deja de sorprenderme cómo un año de nuestra vida nos ha unido con tanta fuerza para el resto de años que llevamos. Estoy seguro que esa fuerza seguirá intacta, de la misma manera que el tiempo compartido en Italia.

Durante el año en Italia también he podido conocer gente muy diversa que también se ha hecho un hueco en mi vida y me han brindado su amistad. No quería olvidarme de Vicky, Luca, Xavi, Alex, Fabio, Miguel, Andrea y Carmelo. Espero tener el placer de seguir compartiendo algún momento con vosotros que, aunque los años los separen, siguen significando un recuerdo de los buenos momentos pasados juntos.

También quería hacer especial mención a Stefano y Mauro por todo su apoyo en el desarrollo de este proyecto y su ayuda para adaptarme al idioma y a la empresa. Los dos fueron sin quererlo unos grandes profesores de profesión.

No podría tampoco dejarme a la persona que más ha marcado mi vida los últimos seis años. Parte de culpa de haber finalizado este proyecto es suya. Gracias Anna por las horas dedicadas ayudándome con la adaptación al inglés. Sin tu ayuda y apoyo no sé cuánto tiempo más hubiera estado pendiente de presentar. Y sobre todo gracias por los años que me has dado. Aunque en todo hay un final, contigo he descubierto el sentido del amor y del cariño, algo que me acompañará y con lo que espero quedarme a lo largo de mi vida.

Finalmente quiero dar las gracias a las personas que me han permitido vivir esta experiencia: Josep Solé, Salvatore Spadaro, Antonio Manzalini y Rosario Alfano. Gracias por vuestro apoyo, consejos y paciencia.

Este proyecto ha sido una etapa importante en el largo camino de la vida, y espero sólo a su fin poderme sentir orgullosos de cada paso dado.

Non possiamo raccontare il momento preciso in cui si forma un'amicizia. Come quando si riempie il vaso goccia a goccia, e alla fine c'è una goccia che lo fà straripare, così in una serie di gentilezze ce n'è un'ultima che fa traboccare il cuore.

J.Boswell

Abstract

The next generation of service framework should be able to address challenges in the delivery of innovative service scenarios, such as high-dynamicity, situation awareness, open “ecosystems”, high-scalability, and reduced infrastructure and operation costs.

This thesis analyses the applicability of autonomic and self-organization capabilities in the execution of ICT services in distributed environments. Autonomicity is one of the key technologies able to address such requirements. In particular, it reports the exploitation of autonomic self-organization solutions to develop solutions for handling wireless communication services even in critical disconnected situations (such as a catastrophic event in a city) and for handling the gathering, correlation and distribution of data. Finally the document describes a prototype of the service scenario, designed by means of autonomic components enriched with an autonomic aggregation protocol.

Keywords

Distributed Service Framework, Autonomic Computing, Agents, Self-organising system

Demonstrating communication services based on autonomic self-organization

Table of Contents

Index of Figures

Demonstrating communication services based on autonomic self-organization

Acronyms

Several acronyms are used throughout this document. The listing below is an attempt to explain some of these.

ACE: Autonomic Communication Element

ANA: Autonomic Network Architecture

AP: Access Point

API: Application Programming Interface

BIONETS: Biologically inspired Network and Services

CASCADAS: Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services

CISIS: Conference on Complex, Intelligent and Software Intensive Systems

CPU: Central Processor Unit

DIET: Decentralised Information Ecosystem Technologies

GA: Goal Achievable

GANA: Generic Autonomic Network Architecture

GN: Goal Needed

GPL: General Public License

GPS: Global Positioning System

GUI: Graphical User Interface

ICT: Information and Communications Technology

IDE: Integrated Development Environment

JDK: Java Development Kit

JDT: Java Development Tools

JRE: Java Runtime Environment

JVM: Java Virtual Machine

RuleML: Rule Markup Language

R&D: Research and Development

SDK: Software Development Kit

SME: Small and Medium-sized Enterprises

SOCRATES: Self-optimization and self-configuration in wireless networks

XML: eXtensible Markup Language

UML: Unified Modelling Language

UTM: Universal Transverse Mercato

CHAPTER I: Thesis environment

1 Thesis introduction

This thesis has been developed in R&D department in Telecom Italia S.p.A (TI) in Torino in the framework of the collaboration between UPC and TI on Autonomic Communication research topic.

TI is a top ICT company in Italy and a Europe-wide leader in offering high quality, value added, and innovative services. The Group is a leading multimedia enterprise, operating in fixed and mobile telecommunications, internet and media, office and system solutions, research and development.

Technological innovation for the Telecom Italia Group constitutes an essential and differentiating element in the development of its competitive advantage and in maintaining its leadership in an increasingly competitive market.

TI technological innovation is the result of strategic partnerships with leading producers of telecommunications equipment and systems, and with top research centres in highly qualified national and international academic institutions (the Polytechnics of Torino and Milan, UC Berkeley, MIT, etc.).

More specifically, the technological innovation activities range from review of base technologies in a logic of increased operational efficiency of networks and systems, to complex and radical reviews of the platforms, services and architectures; intense efforts in the operations and business units is therefore essential in ensuring that new services meet customer services and requirements and for constant improvement in service quality levels.

This thesis, as a part of the work carried out for Telecom Italia, is included inside a European level project called CASCADAS [1], as a part of the survey about Autonomic Communication carried by this project.

Next generation Web 3.0 and Telco 2.0 give a great chance to develop new services, applications and different types of content. The evolution of ICT's needs to overcome current limitations and address emerging trends including: mobility, the mass digitization of media, the emergence of software as services, new models of services and their interaction, etc. These trends offer a motivation and opportunity to develop the thesis.

The purpose of the thesis is to contribute to keep updated the state-of-art of autonomic agents/components technologies at the same time as designing and developing a prototype for demonstrating the applicability of autonomic technologies/solutions for innovative services in a specific use-case of interest.

2 Objectives and results

This chapter will explain the objectives of this thesis and the results carried out.

Technology is currently offering a wide set of portable digital devices (e.g. smartphones, tablets, laptops, digital cameras, music players) at relatively low cost. Penetration of these and other miniaturized digital devices is becoming deeper and deeper in modern cities. This driver is opening new opportunities to produce and access ubiquitously cross-media applications and services.

The next generation of service framework should be able to address challenges in the delivery of innovative service scenarios, such as high-dynamicity, situation awareness, open “ecosystems”, high-scalability, and reduced infrastructure and operation costs. For instance Web 3.0 refers to a next generation of services on the Web allowing people to collaborate and share information and data online, and in the telecommunication service context, Telco 2.0 is emerging to embrace the principles of this Web getting models from the Internet world.

Objectives

In order to meet the challenges and the opportunities offered by this service context, innovative technologies and solutions are required for exploiting highly distributed environments.

- In particular, this document analyses the applicability of autonomic and self-organization capabilities in the execution of ICT services in distributed environments.

The aim of this thesis would like to address one of the autonomic characteristics that should be provided by the innovative distributed service frameworks: namely the autonomic self-organization and aggregation.

- The thesis would like to demonstrate, through the development of a prototype, the applicability of the key principles of autonomic self-organization to the development of solutions for handling wireless communication services even in critical disconnected situations (such as a catastrophic event in a city) and for handling the gathering, correlation and distribution of data.

As autonomicity is one of the key technologies able to address such requirements, the use-case is a city where some catastrophic event has impacted the communication infrastructure causing faults limiting normal wire and wireless connectivity, but with the premise that all agents are endowed with mobile devices with wireless capacity [2].

- The prototype therefore describes a distributed adaptable complex system, realized by means of a population of lightweight autonomic components interacting with each other through self-organizing algorithms.

Results

The demonstration prototype shows the collaborative ambient among elements (survivors and rescue teams) in a critical situation with limited connectivity, such as mobility, data distribution and high probability of disconnection.

- It reports how these factors represent strong challenges for current software architecture and how the distributed lightweight components can self-organize themselves in order to face these challenges.

In addition, the thesis reports the main results achieved in the development of the prototype. The achieved results have demonstrated that the solution, designed by means of autonomic components enriched with self-organising protocols, is meeting some challenging requirements such as adaptability to dynamic situations and robustness, even in environments with high churn rate and/or disconnected situations.

- Mainly, this work has proved the efficiency of to include a self-organising protocol of communication in autonomic agent environments. It makes possible to communicate the autonomic agents in a distributed environment in order to create a totally decentralized organization, making more achievable next digital cities.

This thesis, as a part of the work carried out for Telecom Italia, is included inside a European level project called CASCADAS [1]. The results reported by this document are under further investigations.

The purpose of CASCADAS is to define a new generation of composite, highly-distributed, pervasive services, with underlying technology, that addresses configuration and complexity problems. Its central objective is to identify, develop and evaluate a general-purpose abstraction for autonomic communication services, in which components autonomously achieve self-organization and self-adaptation towards the provision of adaptive and situated communication-intensive services.

- Finally, a paper with the results of the thesis has been written and presented in Second International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2008): “Demonstrating Communication Services Based on Autonomic Self-organization” **¡Error! No se encuentra el origen de la referencia.**. The aim of the conference is to deliver a platform of scientific interaction between the three interwoven challenging areas of research and development of future ICT-enabled applications:
    - Software Intensive Systems

- Complex systems
- Intelligent Systems

CHAPTER II: Autonomic Self-Organization

1 Integrated Project CASCADAS

1.1 Project Summary

CASCADAS is a three-year European integrated project driven by a clear research vision, which is to define a new generation of composite, highly-distributed, pervasive services, with underlying technology, that addresses these configuration and complexity problems. It is proactive in seeking to carry out research encompassing security, resilience, and the interaction of new paradigms on society, which are all key call focuses. Important universities and companies like Telecom Italia, British Telecommunications, Institut Eurecom, Universität Kassel, Fraunhofer Institute, Politecnico di Milano, etc. are working together to develop this project.

The overall goal of CASCADAS is identifying, developing, and evaluating architectures and solutions based on a general-purpose component model for autonomic communication services; specifically in such context autonomic service components autonomously achieve self-organization and self-adaptation towards the provision of adaptive and situated communication-intensive services. In other words, the project is driven by the ambition of identifying a fundamental, uniform abstraction for situated and autonomic communication entities, at all levels of granularity. This abstraction is called an ACE (Autonomic Communication Element), and it represents the cornerstone of the component model, in which the four driving scientific project principles (situation awareness, semantic self-organization, self-similarity, autonomic component-ware) will properly converge.

The study of ACEs is also the basis for achieving a number of other ambitious objectives that will be explicitly tackled by the project. These objectives derives from the need of providing ACEs with the necessary support of algorithms, knowledge, tools and infrastructures (to be realized again as sorts of ACE based middle-services) to make ACEs a practical and trust-worth paradigm. On the other hand, they derives from the willingness to attack and explore some crucial aspects related to the complexity and dynamism challenges that stand in situated and autonomic communication vision. These main research objectives, each conceived in terms of a separated scientific WP and each aimed at delivering specific methodological and software tools, include:

- The development of pervasive supervision functionalities across ensembles of interacting ACEs;
- The development of algorithms and techniques to achieve dynamic adaptation and enforce given service;

- Properties through self-organized component aggregation of ACEs;
- The development of trust, security and self-preservation techniques;
- The identification of models and tools for the organization, correlation and composition of knowledge networks, according to which ACEs can exploit all the available information about their situation, however sparse and diverse.

1.2 Technical Approach

The technical approach is based on four key scientific principles as key enablers for the CASCADAS vision, and around which the future communication services infrastructures should be designed and built:

- Situation awareness: the capability of services to autonomously adapt to the context from which they are requested and in which they execute demands the technologies to capture situational data and at the same time the ability of the system to effectively exploit it. What is still missing is the investigation of the principles and the algorithms with which such growing amount of distributed information can be organized in proper, strongly distributed “networks of knowledge”, and exploited for the purpose of situated and adaptive service provisioning.
- Semantic Self-organization: Self-organization and the algorithms underlying the emergence of global patterns in complex systems have been (and still are) extensively studied in communications, e.g., in P2P computing, ant-based optimization, social networks. There is the need to explore their potential as enablers for service composition and aggregation, drawing inspiration from biological models, and employing proven techniques to abstract from their “organic” implementation and derive design principles adapted to the requirements of artificial systems.
- Self-similarity: to realize the vision and make its embodiment manageable, the communication infrastructure and services must be fully scalable. One promising option is to explore the potential of self-similarity, whereby individual components self-organize and self-aggregate so as to reproduce nearly identical structures over multiple scales. Self-similarity may indeed be a key enabler also for the composition of complex communication-intensive services and for the structuring of the possibly enormous and multi-faceted networks of knowledge items they will have to exploit.
- Autonomic Component-ware: All the above principles are to be federated by a sound “autonomic component” model, to provide both a general model and a robust framework for building autonomic, self-organizing, semantic services. This component model will supply the basic mechanisms and interfaces to support self-similarity, self-organization and situation awareness. Therefore, our autonomic service components will be explicitly conceived as situated in a knowledge network, fitted with mechanisms for self-aggregation and composition, and designed so as

to promote the emergence of high-level ensembles that exhibit self-similarity independently of scale.

The project is structured into 5 work packages (Figure II.1), each dealing with specific research thrusts recognized to be critical elements for the situation-aware and autonomic communication services of the future.

Figure II.1 Vision of the CASCADAS Project

CASCADAS considers a scenario in which dynamic and heterogeneous networks, possibly enriched with sensors and devices connecting with the physical world, have to host the dynamic deployment and execution of applications and services. Such applications and services have to serve users according to both their social situation and the current network and physical situations.

a) Autonomic Communication Elements

The key idea is to identify and rely on a new model of distributed components (ACEs), able to autonomously self-organize with each other towards the provisioning of specific user communication services, and able to self-adapt such provisioning to social and network contexts. These features are likely to dramatically reduce the costs associated to the development and configuration of complex communication services, to leverage the exploitation of distributed computing and communication resources, and to make services more usable and more fitted to user needs. The most important result of the project will be an Open Source toolkit with a set of well-integrated abstractions, algorithms, tools, and application demonstrations.

b) Pervasive supervision

Pervasive supervision addresses the runtime construction of an ad hoc and dynamic runtime structure that encompasses a set of cooperating ACEs, and exerts a fully automated and de-centralized control of the communication-intensive service provisioned collectively by those ACEs.

c) Component aggregation

The development of algorithms and techniques to achieve dynamic adaptation and enforce given service properties through self-organized component aggregation of ACEs. That kind of aggregation will be the basis for identifying and exploring opportunities for co-operation within ensembles of ACEs, which would allow the collective system to exhibit certain desired properties, for example hit situation dependent QoS targets.

d) Trust, security and self-preservation

The development of trust, security and self-preservation techniques, which are of are paramount importance because of the very assumptions upon which the idea of ACEs relies: the heterogeneous nature of the network, the varied capabilities of ACEs, their ability to self-organize and cooperatively supervise each other, which implies the lack of centralized administrative control. Since an ensemble of ACEs possesses those highly dynamic adaptation characteristics, we intend to exploit them to make sure that the resulting system is highly robust and secure, and trust-worth.

e) Knowledge networks

The identification of models and tools for the organization, correlation and composition of knowledge networks, according to which ACEs can exploit all the available information about their situation, however sparse and diverse. Situation is intended here as context, considered in the broadest sense, relating to both (i) the social-organizational context from which services are invoked; (ii) the technological and physical environment in which ACEs live and execute.

1.3 Contribution

The contribution of this thesis within CASCADAS project has been a theoretical follow-up of the implementation of the ACE Toolkit and collaboration in the development of the Self-Organised Component (WP3) with the self-organising algorithm implemented in this thesis.

A set of clearly defined requirements are demanded to be fulfilled by the ACE Toolkit of the CASCADAS project. This includes, as major points, that the Toolkit should enable autonomic, situation-aware, and self-similar development of services based on ACEs as core components.

The ACE Toolkit offers components and mechanisms to develop lightweight Autonomic Communication Elements that support the concepts of self-similarity, semantic self-organization, and situation-aware behaviour. Service developers can use this framework to create distributed, modular, and reusable applications.

The work of this thesis has been developed to analyse algorithms and techniques to achieve dynamic adaptation and enforce given service properties through a self-organized aggregation component include into the ACEs. That kind of component will be the basis for identifying and exploring opportunities for co-operation within ensembles of ACEs, which would allow the collective system to exhibit certain desired properties.

In order to achieve this aim, an aggregation protocol has been designed. The protocol contains the rules and interactions to implement the basic communication behaviour of the ACE. These interactions and rules are included following the “On-demand” clustering basis, in order to preserve homogeneous node degree in the realistic, local rules-based scenario.

The prototype designed in this thesis develops the aggregation protocol and it has been used to include a communication part in the ACEs, making possible introduce real-time interactions to face changes in the environment conditions.

Chapter III explains in great detail the Communication prototype.

2 Autonomic System

The definition of the autonomic system is taking inspiration from the self-governing behaviours of some natural autonomic systems, such as the human autonomic nervous system. Once launching the Autonomic Computing initiative, IBM defined four general properties a system should have to constitute self-management: self-configuring, self-healing, self-optimizing and self-protecting. Since the launch of Autonomic Computing initiative, the self-* list of properties has grown substantially. Now it includes also features such as self-anticipating, self-adapting, self-critical, self-defining, self-destructing, self-diagnosis, self-governing, self-organized, self-recovery, self-reflecting and so on.

The extension of the autonomic technology principles from computing to network and services resources has still the meaning of developing solutions that are capable of hiding operational complexity to both Operators and Users. Autonomic systems are capable of making decisions on their own, by using high-level policies from operators, checking and optimizing their status in order to adapt themselves to change environment conditions at the same time (Figure II.2).

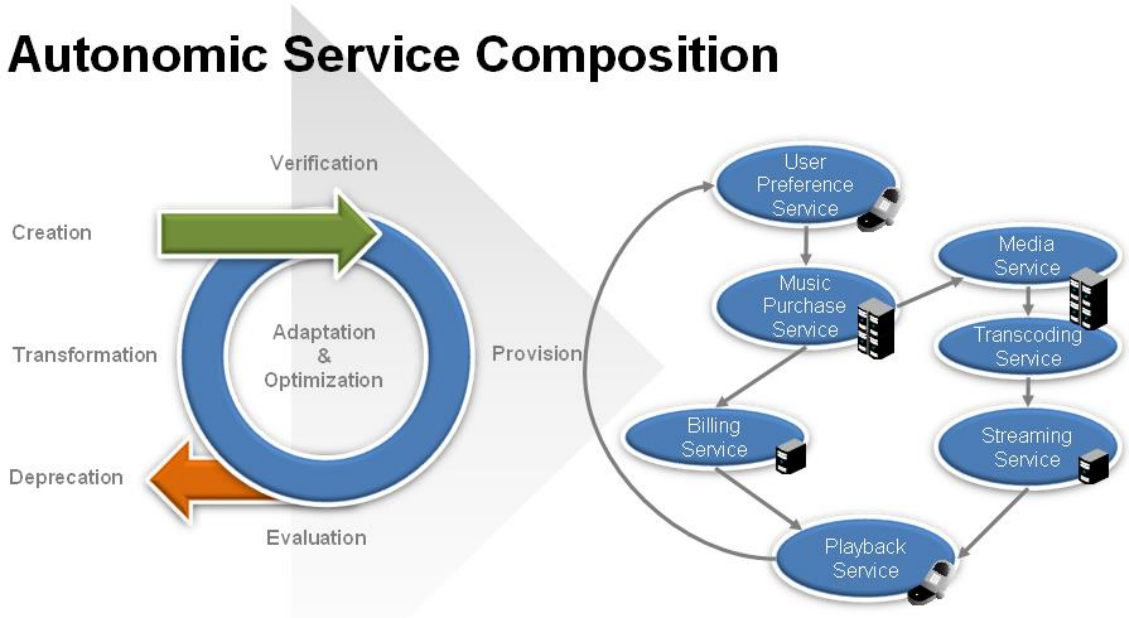


Figure II.2 Continuous monitoring and optimization methods help maintaining composed services across different physical systems, service domains and vendors

Autonomic systems are typically distributed, complex and concurrent, comprised of multiple interacting autonomic elements and all the resultant issues have already been faced in different fields of autonomous agents [5].

This autonomic scenario cannot be developed in a hybrid network, as usually the today's networks are. Then, the infrastructure-based mode that has been a great success of service composition in infrastructure-based environments should turn into an infrastructure-based mobile mode (Figure II.3).

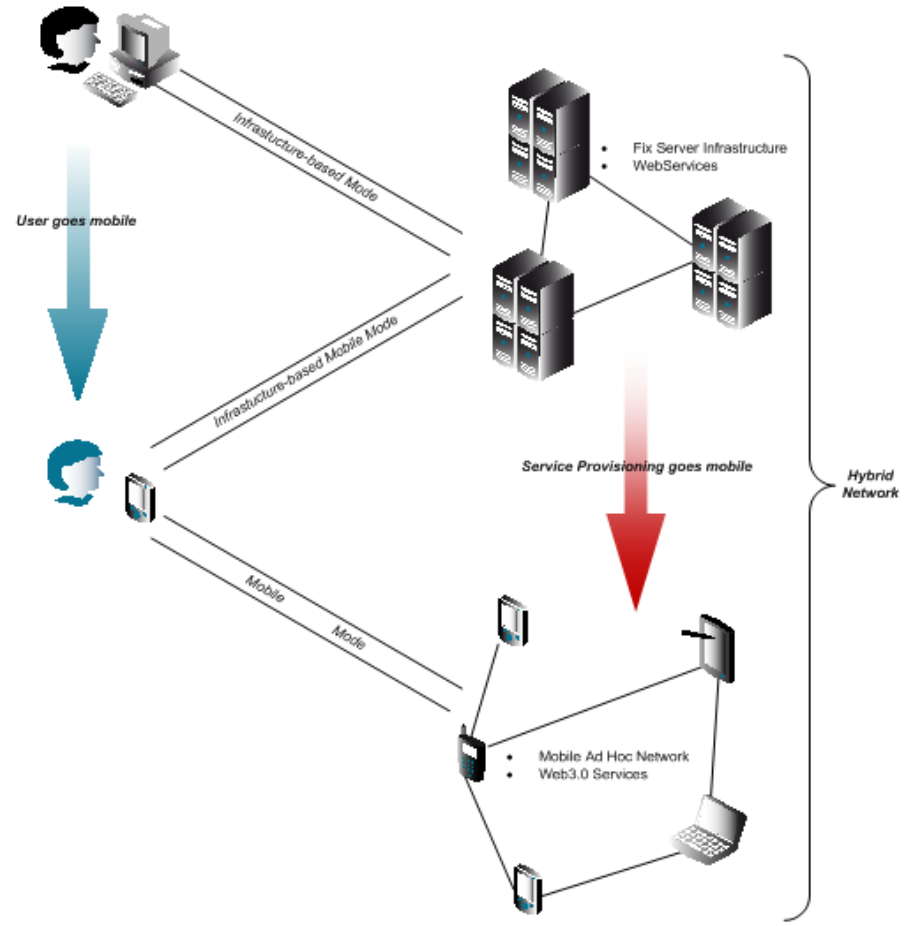


Figure II.3 Infrastructure Requirements for Autonomic Service Composition

As it has already said there is a great need of autonomic behaviour in order to hide complexity and minimize the human intervention. An autonomic system should realize an efficient communication in distributed environments without centralized control, and a reduction of complexity through autonomic and self-managed behaviour.

3 Self-organising System

A key challenge of the autonomic computing initiative has been to draw upon self-* properties in systems other than computational ones in order to develop new computing systems. Biology has been a key source of inspiration. Group-living animals have provided inspiration for the field of collective, or swarm intelligence which models problems through the interactions of a collection of agents cooperating to achieve a common goal. For example, eusocial insects (ants, bees, wasps, termites, etc.) form colonies presenting a high degree of social organization. In these systems, problems are “self-solved” in real time through the emergence of the appropriate collective behaviour, which arises from the sum of all interactions occurring between the agents and with their environment.

The mechanisms that lead to self-organization in biological systems differ from those occurring in physical systems in that they are influenced by biological evolution. Thus there is the possibility of using algorithms inspired by biological evolution. These are part of what has been described as biologically-inspired or “nature-inspired” computing (the latter term chosen because of the breadth of natural complexity that does not usually fall into the area of biology). A diverse range of fields have been analysed, including evolution and genetics, bacterial adaptive mechanisms, morphogenesis and self-organising principles more broadly.

Further interest in developing self-organising systems has considered the role that decentralized control of distributed systems can play in self-organization. Peer-to-peer networks can be used as a basis for self-organization among elements, in order to develop complex adaptive systems or self-organising multi-agent systems. These draw upon the emergence of novel properties at the whole system level when many elements are brought together automatically, without being programmed in by system developers and inspiration is drawn from biological systems.

Many recent developments in distributed computing has been the move from centralised to decentralised systems, with the understanding that decentralisation may not mean complex loss of control. Decentralised approaches have attracted a lot of interest as it has become possible to run highly flexible applications over complex networks in a reliable manner. This is important when demand for services or applications may be unpredictable and network properties may be heterogeneous.

The Agent-based approach has been, and remains, a rich area for the study of the emergence of self-organization. For example, “artificial markets” have been studied for their potential in market-based control. The aspiration is that if the appropriate interaction/trading rules are encoded into a population of agents, then the agents will be able to self-organise into “useful” structures/networks, where “useful” is defined in terms of an application context e.g. Supply chains, or trading markets.

Di Marzo et al.[6] review different aspects of self-organization in Multi-Agent Systems. They show how inspiration derives from natural systems (complex physical systems as well as natural systems). For example, the concept of stigmergy, derived from the behaviour of social insects, has also been important in inspiring the design of Multi-Agent Systems. Bernon et al. [7] review several examples of applications of self-organising multi-agent systems. They show how Multi-Agent Systems can self-organise to carry out tasks, even though individual agents have very simple properties. The emergent properties of the self-organising system support each application.

Another important area of investigation of self-organization has focused upon how individuals can cooperate to produce self-organising behaviour. Like other aspects of the analysis of self-organization, this has been inspired by cooperation in nature, in systems ranging from microorganisms to human beings.

Overall the concept of self-organization has motivated a great deal of research that promises to have considerable impact on emerging autonomic computing and communications technologies.

4 Emergent collective behaviour

Self-organized behaviour represents collective behaviour that emerges at the level of the group from the numerous interactions among individuals and between the individuals and the environment. Moreover, the rules presiding to the interactions among individuals and between the individuals and the environment are executed by using only local and/or incomplete information, without reference to the global pattern (i.e. without relying on global maps or global representations) [8][9].

The mechanisms that lead to self-organization in biological systems differ from those occurring in physical systems in one important respect: the rules that determine the interactions among the agents and between the agents and the environment are influenced by biological evolution, i.e. the properties emerging from these interactions are shaped by natural selection. In effect, evolution simply favours individuals following rules that give rise to adaptive behaviour, thus increasing their fitness.

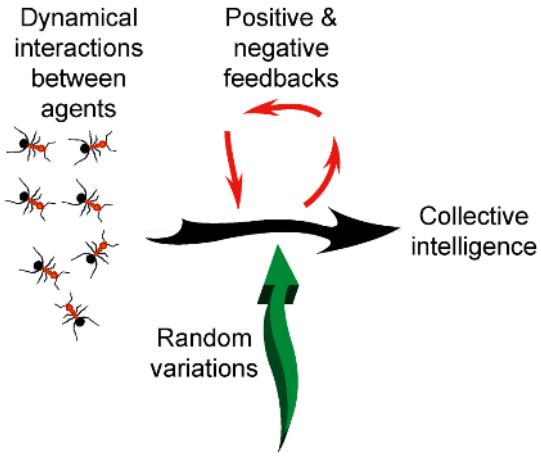


Figure II.4 Core ingredients for collective intelligence.

The essential features in collective intelligent systems are:

- Dynamical non-linear interactions between agents.
- Random variations.
- Positive and negative feedback mechanisms (Figure II.4).

In short, the dynamical interactions within agents and between the agents and the environment constitute the basis for the synthesis of emergent properties. Random fluctuations lead to innovations that might constitute potential effective solutions. Positive feedback mechanisms promote macroscopic changes by reinforcing modifications that “push” the system in the same direction as initial variations. Negative feedback mechanisms keep the amplification process resulting from positive feedback under control. The combination and the interaction between these fundamental mechanisms can lead to ”real-time” selection of self-organized solutions. Communication or interactions are essential to obtain cooperation between individuals. For

example, the combination of these mechanisms can lead to a collective decision process in which an optimal or close to optimal decision is made by the collective, even though no single agent has enough information to make the decision on its own, and despite agents not even being ‘aware’ that a collective choice is being made . In fact, individual behaviour is impossible to understand (and would often be highly damageable to fitness) if considered outside of the social context, which defines its purpose.

The notion of “local” information or rules is also confusing. In real biological (or physical) systems the information used by individuals is often local in the literal, physical sense. But in many cases what the word really means in the context of emergent collective behaviour is that the information used by the agents is incomplete, i.e. that it doesn’t contain accurate details on the global pattern.

Finally, it is also important to understand that collective phenomena don’t necessarily involve a large number of individuals but rather a large number of interaction events involving the agents and their environment. There are experimental biological examples of self-organized collective behaviour emerging in small groups of animals comprising only around ten individuals.

5 Self-organising algorithms

This section describes two examples of self-organizing algorithms [6]: passive clustering and on-demand clustering. In the context of this thesis the term “aggregation” refers to the process by which nodes form associations (“links”) with each other. It is a “clustering” process during which each node, characterised by a certain “type” establish links with nodes of the same type. From this point of view, the efficiency of a self-aggregation algorithm can be read in terms of convergence capabilities of increasing the fraction of links connecting nodes of the same type.

5.1 “Passive” clustering

A first set of basic local rules has been devised requiring only direct interaction between first neighbours yet susceptible to give rise over time to spontaneous system-wide aggregation of elements. The basic idea involves two nodes being notified by a third (the “match-maker”), which are interconnected through an overlay network, even if those two nodes have no direct part in the decision process (“passive” clustering). The rules are as follows:

- match-maker node is randomly selected. This is equivalent to say that every node in the system has a chance of “waking-up” and initiating a rewiring procedure, provided that this procedure is brief enough (and/or infrequent enough) that a situation in which two concurrent rewiring affect the same nodes is extremely unlikely, and so every attempt can be considered as an independent event.

- match-maker randomly selects two of its own neighbours and, if they happen to belong to the same type, instructs them to link together
- if the two chosen nodes were not already directly connected (through the overlay) a new link is established between them (i.e. they become first neighbour of each other).
- if conservation of the total number of links is in force (optional) and a new connection is successfully established, the match-maker terminates one of its own links with one of its two selected neighbours.

5.2 "On-demand" clustering

In passive clustering technique, in order to preserve homogeneous node degree in the realistic, local rules-based scenario, the rewiring procedure has to be modified: there is the need of eliminating the indirect positive feedback leading to the emergence of scale-free topology. It may be objected that the heterogeneous node degree can be highly beneficial to the network operation if the higher connectivity of some vertices can be made to reflect their superior capability. However, in our case, such correlation is effectively absent: the emergence of hubs in the "passive rewiring scenario" results from the amplification of random fluctuations. As it cannot be guaranteed that those nodes ending up with a higher degree effectively have some specific features that designate them as efficient "super-peers", the result could be disastrous and generate critical bottlenecks, which is why we aimed at maintaining node degree as homogeneous as possible throughout the system's history.

This has been achieved by distinguishing between the initiator of a rewiring procedure and the match-maker. Basically, upon "waking-up", the initiator requests a new link from one of its existing neighbours, which will then act as the match-maker. Since with this logic, the probability for a node to be appointed match-maker is obviously a direct function of its own degree (and the match-maker still ends losing one neighbour in the process of a successful rewiring operation), it introduces a negative, "rich becomes poorer" feedback similar to the one observed in the abstract model.

The detailed algorithm governing key node behaviour in the three roles of "initiator", "match-maker" and "candidate" involved in a rewiring operation following the "on-demand" clustering procedure is shown in Figure II.5. It involves exchanging five types of messages (plus the link termination message which isn't discussed here). The "neighbour request" (NRQ) message is sent by the initiator to the chosen match-maker and specifies the type of node desired. The "neighbour reply" (NRP) message is sent by the match-maker to the initiator to inform it to a potential candidate. The "link" (LNK) message is sent by the initiator to the candidate to ask for the establishment of a new link, which will only be effectively created if it is compatible with the goals of the candidate, as evidenced by the receipt of an "acknowledgement" (ACK) message by the initiator. Notice that, for most of the results presented in this section, this will always be the case as all nodes in the system share the same objective, i.e. they are all assumed to be simultaneously in clustering (or reverse-clustering) mode. Finally, after a successful handshake between the initiator and the candidate, the match-maker is informed via the "success" (SCC) message so

that the match-maker can be able to determine whether or not its own connection to the candidate has to be terminated, in order to conserve the total number of links.

This last communication algorithm described is chosen from the whole sort of self-organising algorithms defined in the WP3 [10] of IST CASCADAS by has one of the best statistics behaviours to define our communication protocol. Therefore, implement “On-demand” clustering algorithm into our has been one of the strong reasons.

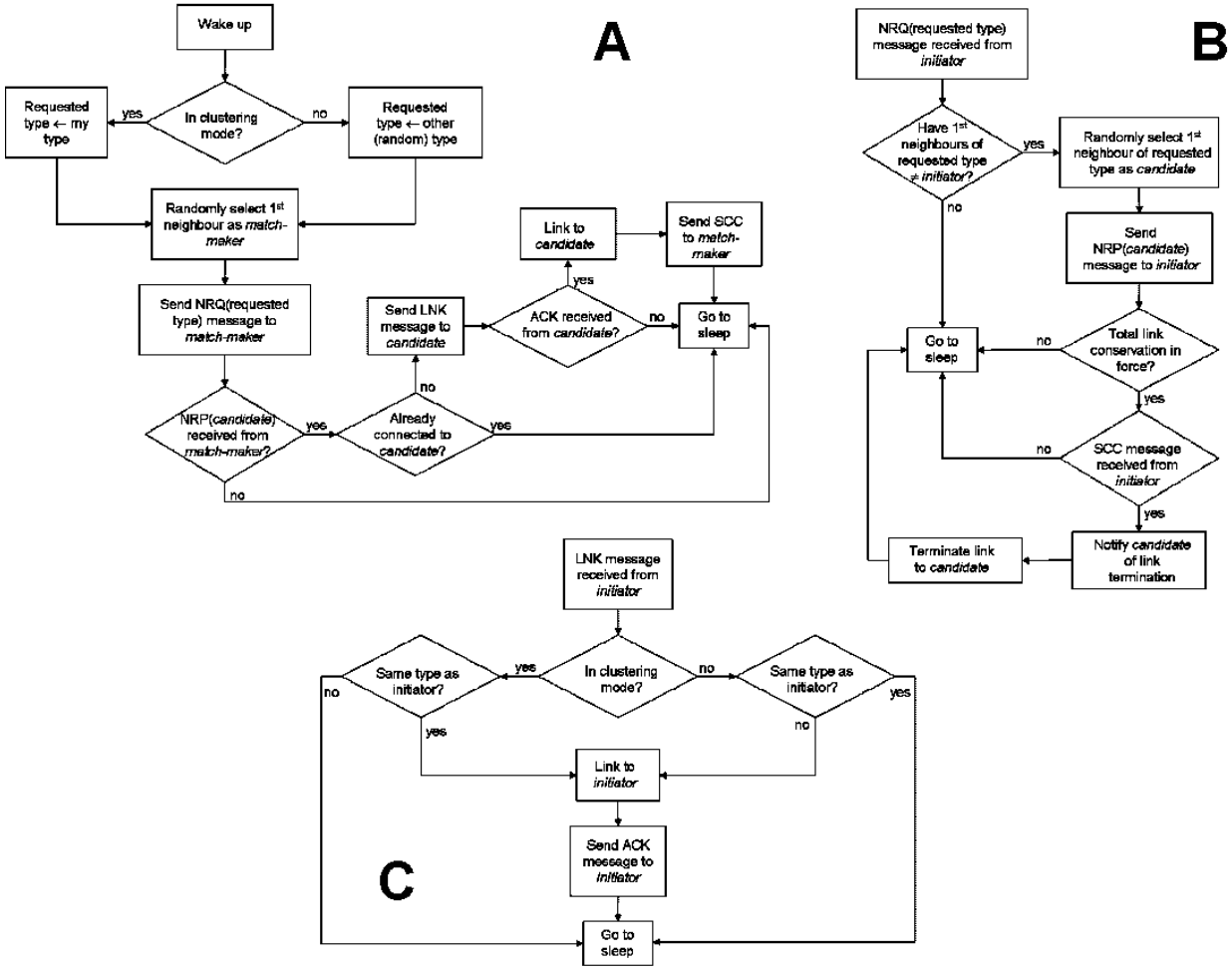


Figure II.5 Algorithms involved in rewiring operation following the “on-demand” clustering procedure with the three roles of “initiator”, “match-maker” and “candidate”.

6 The digital city scenario

A digital city is a possible scenario where autonomic and self-aggregation capabilities can be exploited in order to enable service providers to offer innovative situated services to end-users. The identified scenario is a dynamic

and data intensive digital environment, such as a city with pervasive digital devices. In such a scenario citizens can retrieve and process information and use them to collaborate with each other. This scenario requires technologies and solutions to manage large amounts of highly distributed data items, which need to be transformed into meaningful, reliable and available information for each mobile user. An example is “wikicity”, scenario under study by the Sensible Consortium [11].

Such a scenario is enabled by the technology evolution which is offering today a wide set of portable digital devices (e.g., smartphones, tablets, laptops, digital cameras, music players, etc.) at relatively low cost. This is fostering a wider and wider adoption of such devices by people. Moreover also sensors and other tiny digital devices are being more and more distributed in our cities to monitor the environments and providing several services. This is a trend that will gain even further intensity in the future. The use of all these digital devices during human (and machine) daily activities is generating huge data-clouds describing the activities and the environmental contexts in our cities: dynamics of a city can be captured in real-time by collecting and correlating data and information (anonymous localization, traffic, pollution, cultural sites, events, etc) provided by heterogeneous sources.

This information becomes an instrument for city inhabitants, enabled to base their actions and decisions on such better and more synchronized information. For instance, on a “digital” city map the different information and data layers that coexist can be represented by:

- Events occurring around the city indicated at the corresponding location and time;
- How people is concentrated and moving in the town following the cell phone usage;
- The public transportations’ real time position and time;
- Location tagged news feeds from different places in town.

People moving and acting in a city base their decisions on information that is in most cases not synchronized with the time and place they find themselves in when taking those decisions. Experiencing a shift between decisions and information it is very common in everyone experience: arriving at the airport just to find out that the flight has been delayed, being surprised by a traffic jam. The dynamic city is concerned with the real-time mapping of city dynamics and data: real-time requirements are relevant for controlling entities in an environment characterized by uncertainty and dynamic evolution, and for collecting and elaborating information acquired by sensors on entities and environment states. Moreover, relevance of some information could depend on time, and therefore, data should be associated with the relevant time information (e.g., acquisition time, deadline, etc.).

As example, Figure II.6 shows a Digital City where it is explored different interface modalities that create connections between the virtual data and the actual physical world where users access these data. The system is based on a

common, semantically defined format for interchange of location data and a distributed platform that can collect and manage such data in real time.

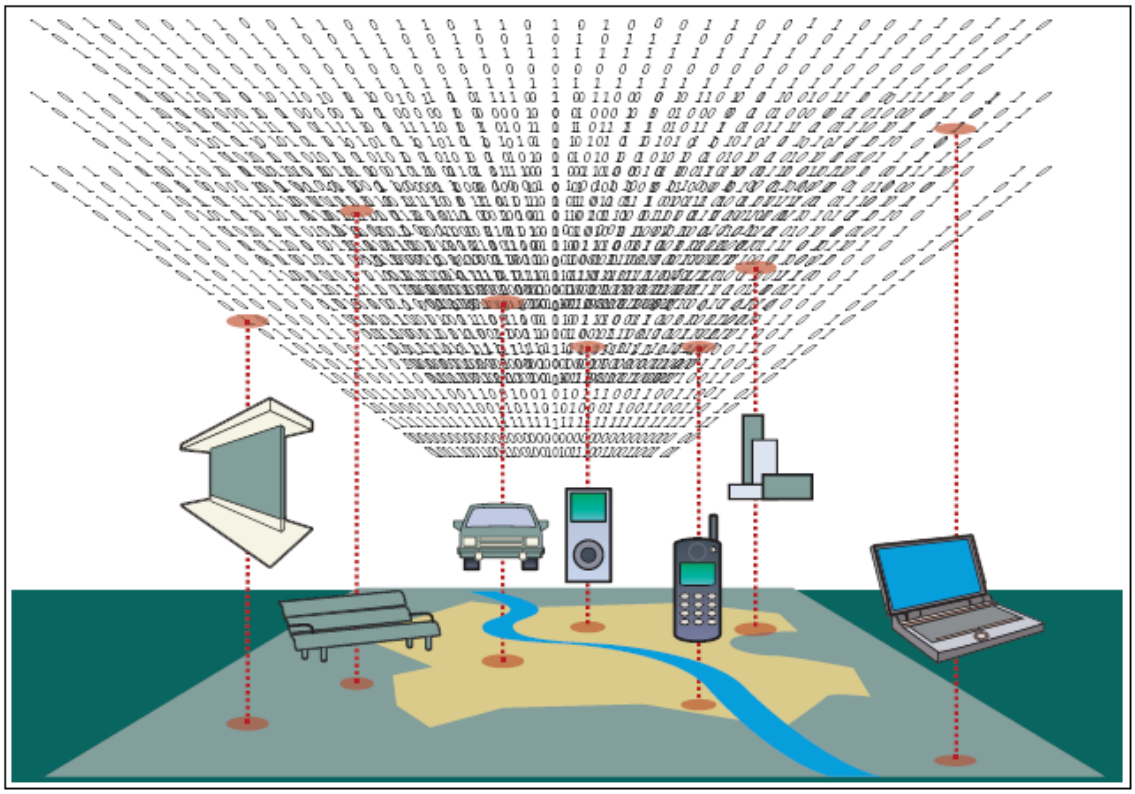


Figure II.6 A Digital City

The potential benefits of such a “digital city” can be observed from different angles:

- Citizens: better knowledge of events, opportunities and environmental conditions concerning their local surroundings. Figure II.7 shows an example;
- Local authorities: better understanding of the urban system and its dynamic evolution, e.g. distribution of pedestrians, vehicles, tourists,… in the city at different times of the day and their correlation with the events happening at specific locations (based on communication network analysis);
- Companies: ability to better promote and distribute their services/products to the local population (location and time based eBusiness);
- Mobility: better use of public and private transports based on real time information about schedules, paths, delays, traffic condition.

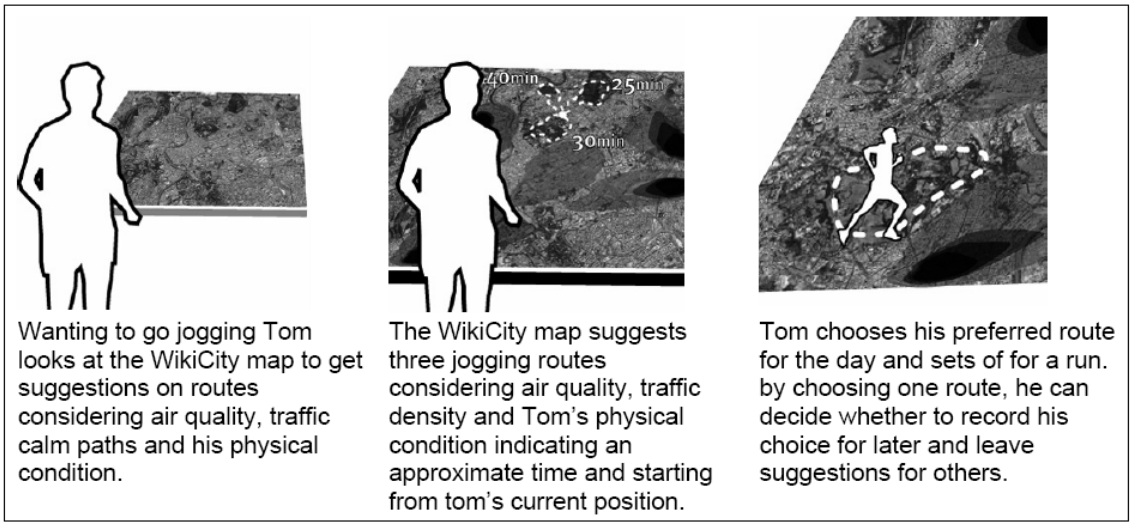


Figure II.7 Jogging path scenario

Deployment of a system implementing a Digital City requires different thread of research (ranging from sensors to dynamics maps, from semantics/folksonomies for tagging information to access modalities to situated services, etc.); autonomic and self-aggregation technologies could contribute in order to handle the dynamicity of data-clouds overlooking a "real" city, and the complexity of their relationship.

In fact, a digital city is the environment of dynamic and evolutionary processes. Its dynamics can be captured and monitored in real-time by collecting and correlating data (anonymous localization, traffic, pollution, cultural events, etc.) provided by heterogeneous sources. By observing these dataclouds overlooking a "real" city, it can be realized how a real city is like an ecosystem with self-adaptive and self-organizing properties. Social patterns, generated by human relationships, can be thus explained in terms of the organization processes of complex self-adaptive systems. Also these social patterns represent valuable information for providing situated services and even anticipating citizen's needs.

In such a context, autonomic and self-aggregation solutions could be used to address real time correlation of huge amount of real time, and dynamically changing data associated to a digital city: self-aggregation and autonomic capabilities can be used in order to provide the right information, at the right level of details and/or aggregation, and at the right time, to services and end-user devices. Pieces of data and information, also associated to location, time, semantics information, can be dynamically retrieved from heterogeneous sources, including end-user devices, correlated, elaborated, and routed to the service components and end-users devices requiring them. Moreover, autonomic capabilities could be used also in order to provide optimized communication services needed for information retrieval, aggregation and distribution among digital devices.

As a simple use-case that involves a distributed collection and elaboration of data and information, Citizens in a city may want to run services on their mobile handsets that require digital maps (e.g. of the area where the Users are

located). In particular maps can be downloaded either from distributed access-points or from neighbour devices (avoiding access to any centralized server). Standards maps are downloadable from access points; richer Maps, containing also other contents and information personally created (audio files, images, commercial text), can be shared by Citizens in peer-to-peer mode in each area. Real time information (e.g. traffic jam) can be further added and correlated to maps. Let's assume a Citizen would like to find the nearest restaurant and the best route to reach it. The map of the area (containing also commercial information of restaurants) is downloaded from the device of a neighbour. Best route, taking also into account traffic information, may be correlated to its location (Figure II.8).

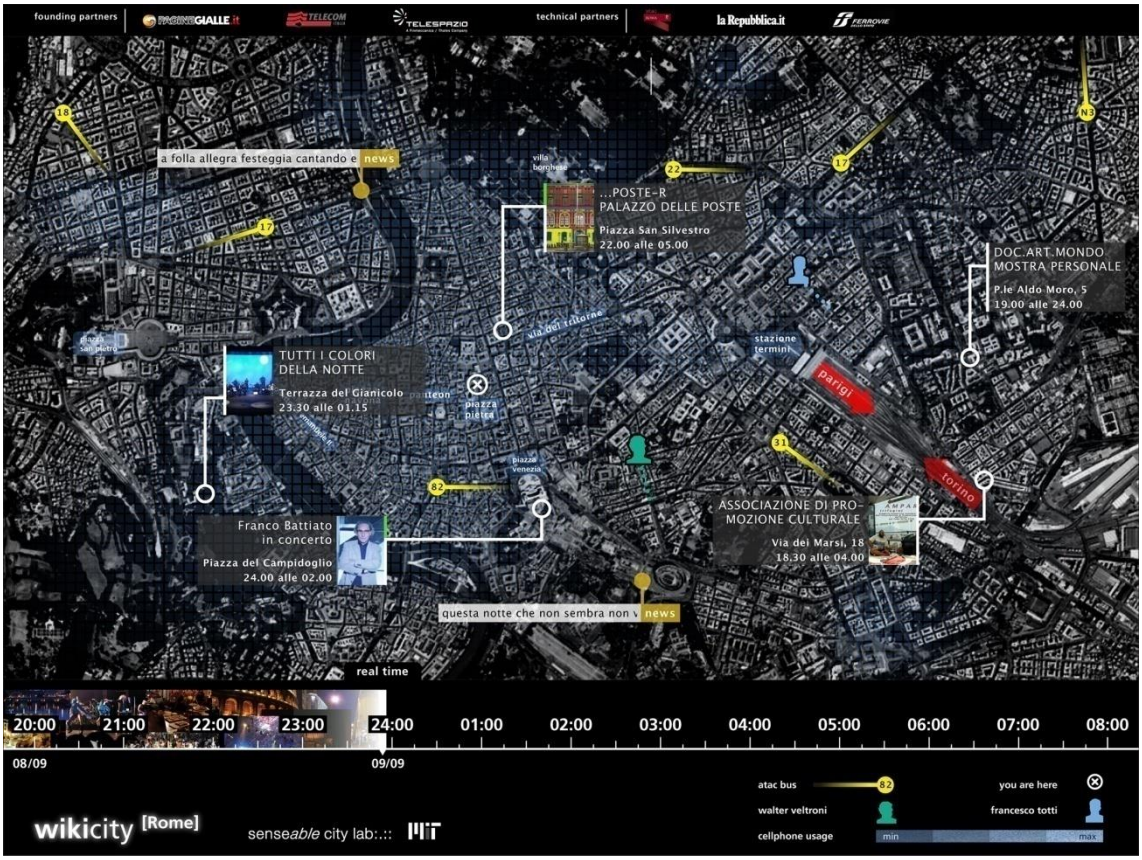


Figure II.8 Real Time Rome *project using cell phones and GPS devices to collect the movement patterns of people and transportation systems*

In this Digital City scenario, the prototype developed in this thesis tries to analyse and demonstrate the applicability and the benefits of autonomic and self-aggregation capabilities in wireless communication services, in critical unpredictable situations (e.g. catastrophic event in a city causing faults that are limiting normal mobile and fixed services). In above conditions, factors such as users' mobility, data disperse distribution and high probabilities of disconnection represent challenges for current software service architecture.

CHAPTER III: Experimental Prototype

1 Introduction

The use-case considers a city where some catastrophic event has impacted the communication infrastructure causing faults limiting normal wire and wireless connectivity, but with the premise that all the involved entities are endowed with mobile devices with wireless capacity [2]. Such a use-case is ideal for demonstrating how autonomic self-organization meets requirements for communications in critical situations between the survivors and rescue teams, with the aim of providing first aid as soon as possible.

Two basic groups are involved in the service: the rescue teams and the survivors, as Figure III.1 shows. The interaction between groups of survivors and between survivors and rescue teams should test the efficiency of the self-management system. Rescue teams and survivors are placed in two differentiated interaction environments.

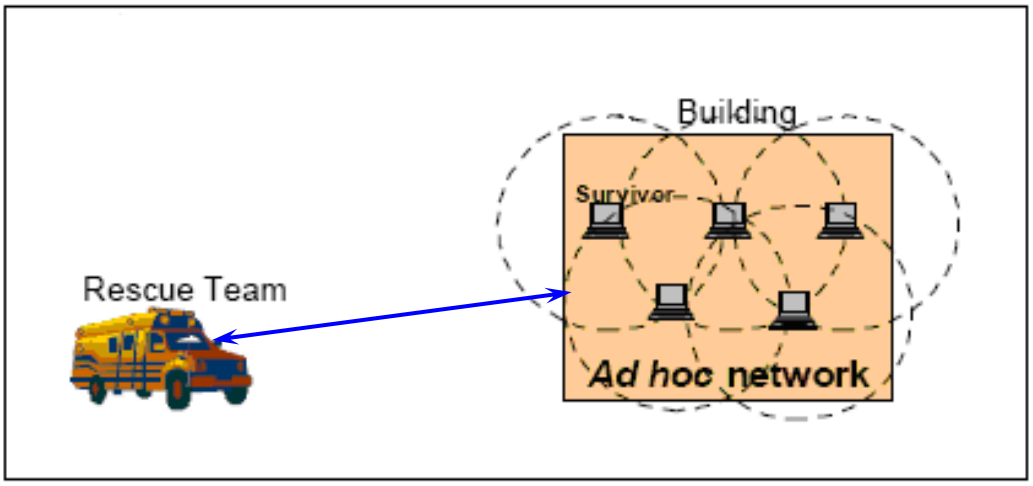


Figure III.1 Representation of the main entities involved in the Use Case

Inside each building the communication is managed in a completely distributed way without any central control. This is likely to be a communication network between peers based on an ad-hoc system. The last objective is to achieve the major spread degree of available information in the environment, in order to make each survivor able to provide and gather the information from its neighbours.

Moreover, the rescue teams will interact with the different environments where the survivors are situated. Therefore, a rescue team will be able to communicate with a group of survivors when it is in the covering area where some of them are situated.

Figure III.2 provides a possible representation of the scenario, the map of a city with groups of survivors inside buildings and rescue teams running around the town.

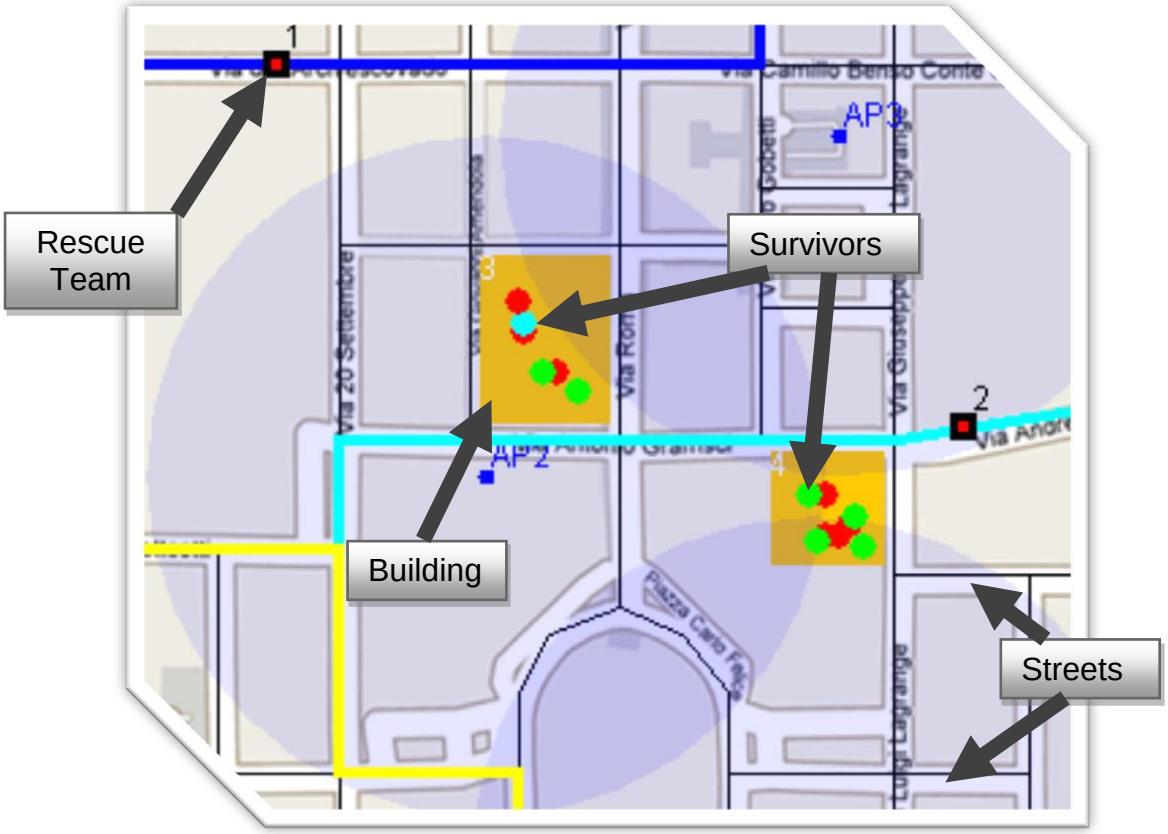


Figure III.2 Possible scenario of the Use Case

The basic rules that the agents must fulfil are:

- **Survivors:** communicate the information to each other in the same environment.
- **Rescue Teams:** decide between a group of environments (buildings) which rescue first according to the rang of priorities (survivors, ages, healthy state, etc.).

2 Agent's Description

Each agent is composed by an autonomic component named ACE (Autonomic Communication Element). Then, the environment will be formed by a population of ACEs capable to offer different kind of services, i.e. ACEs are building as blocks of autonomic self-organising services (Figure III.3).

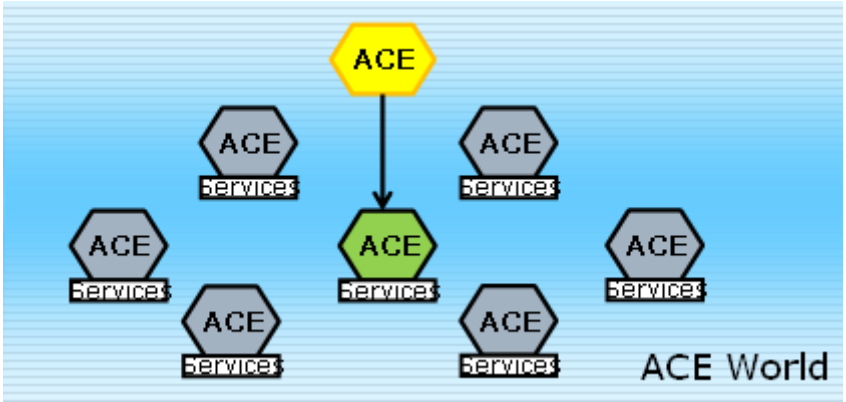


Figure III.3 Environment is formed by a set of ACEs

Each ACE includes the main self-* principles, paying special attention to support the concepts of self-similarity, semantic self-organization, self-description, self-healing and situation-aware behaviour:

- Self-Similarity
  - Lightweight simple components with common interface and functionalities
  - Transparent aggregation into complex services
- Self-Organization
  - Completely distributed way without any central control
- Self-Description
  - Self-Model describes ACEs possible states, transitions and reaction to events
- Self-Healing
  - Pervasive supervision allows to monitor and control ACE behaviour
- Self-Awareness and Situation Awareness
  - Knowledge of current execution state and contextual conditions
  - Knowledge network integration

Two Kinds of ACEs have been developed in the prototype: survivors and rescue teams. Each one shares a Common Part and adds a Specific Part with its own Self-Model (diagram state) and functions. Thus, as many ACEs as survivors and rescue teams are defined.

Every survivor tries to find others to share information and collaborate with each other, searching for the GPS position, identifying a doctor and making a list with the main characteristics like blood type, age, health state, etc. Once the (ACE associated to a) rescue team starts, it searches for buildings in its signal coverage and decides which building rescue. This decision is taken from the lists received, making a scale of importance according the survivors' age, health state, etc.

An agent, such as a survivor or rescue team, has to be uniquely specified by its address, and depending on the kind of the agent, it will have more attributes. A rescue team only is specified by its address, while a survivor has the following attributes (Figure III.4):

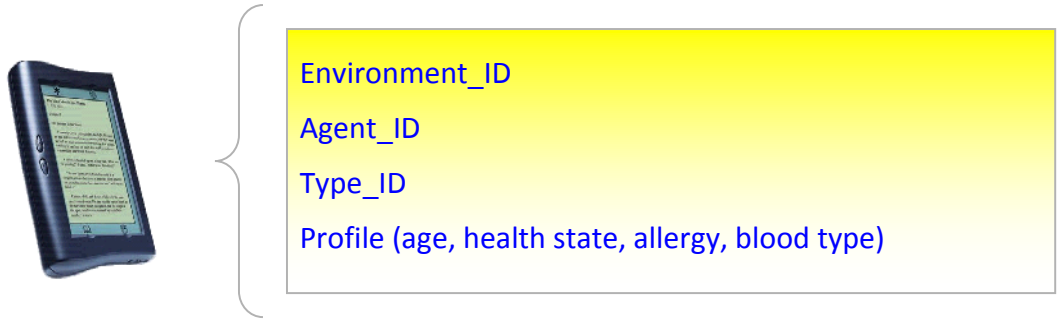


Figure III.4 Survivor's attributes

- Environment identity. The identity of the environment (building) where the agent resides in. It would change if the agent migrates to a different environment.
- Agent identity. The identity is established when an agent is created, and it remains fixed throughout its lifetime.
- Type identity. Element to describe the agent's type according to its capabilities. Next Figure III.5 describes with a numerical classification the utilities and services for each type.
- Profile. In order to prioritize the rescue, each survivor has a profile with its age, health state, allergies and blood type.

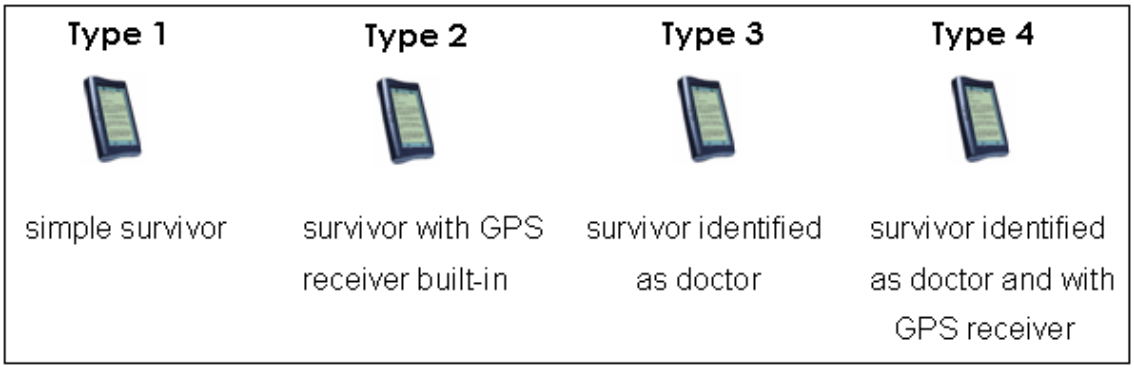


Figure III.5 Survivor's types

Setup files

XML is the format of the file's structure that has been chosen to setup all the elements of the prototype. In it there are described the number of rescue teams, survivors and their proprieties. The two following files have been devised for that scope:

1. **MapGps.xml:** It is a definitions file with an xml structure where all the elements that appear in the map are defined but the rescue teams. Within this file the vertex and the lines that make them join are defined as well, in order to give a logical description of the different streets and avenues that the chosen map's section has. The definition is very simple in order to be possible to apply an enrouting GPS algorithm [12]. We also have the buildings' locations, as well as the agents located there. In the area of each agent their profile features (age, heath, medical data and so on) are defined as attributes. Finally, the number or positions of the different Access Points, which are distributed all around the city, are defined. A sample of the used typology can be seen in the picture below (Figure III.6).

2. **Elements.xml:** In this file we can find a definition of all the different Rescue Teams that will be involved in the simulation as well as its starting position point. As its starting position, we must refer to one of the different kinds of vertex, where two ways meet and that is defined within the MapGps.xml file.

```
<map>

    <buildings>

        <rectangle>
            <survivor age="15" type="2" state="healthy" allergic="Dermatitis" blood_type="O+"/>
            <survivor age="40" type="1" state="healthy" allergic="No" blood_type="AB-"/>
            <survivor age="57" type="1" state="healthy" allergic="Asthma" blood_type="A+"/>
        </rectangle>

        <rectangle>
            <survivor age="15" type="1" state="healthy" allergic="Latex" blood_type="O+"/>
            <survivor age="40" type="1" state="healthy" allergic="No" blood_type="AB-"/>
            <survivor age="80" type="2" state="healthy" allergic="Asthma" blood_type="B+"/>
            <survivor age="43" type="1" state="injured" allergic="No" blood_type="A+"/>
            <survivor age="12" type="2" state="healthy" allergic="Shellfish" blood_type="O-"/>
            <survivor age="63" type="1" state="healthy" allergic="No" blood_type="O+"/>
            <survivor age="10" type="1" state="healthy" allergic="Penicillin" blood_type="AB+"/>
            <survivor age="51" type="1" state="healthy" allergic="No" blood_type="B-"/>
        </rectangle>

        <rectangle>
            <survivor age="6" type="1" state="healthy" allergic="Dermatitis" blood_type="O+"/>
            <survivor age="40" type="4" state="healthy" allergic="No" blood_type="AB-"/>
            <survivor age="80" type="2" state="healthy" allergic="Salicylates" blood_type="B+"/>
            <survivor age="10" type="1" state="injured" allergic="No" blood_type="A+"/>
            <survivor age="12" type="3" state="healthy" allergic="Pollen" blood_type="O-"/>
        </rectangle>

                    ..................................................

    </buildings>

    <rescueTeams num="2">
    </rescueTeams>

</map>
```

Figure III.6 Setup file for survivors and rescue teams

The files can be found in the path Reasoning/conf/.

3 Prototype Architecture

A prototype has been designed and developed in order to demonstrate the applicability of the main principles of autonomic self-organization in terms of interactions of a population of autonomic components through self-organizing algorithms.

The scenario used for demonstrating the prototype is a dynamic and data intensive digital environment, such as the city described above with pervasive digital devices, where everyone can exchange information and collaborate with each other. A scenario like this requires technologies and solutions to manage large amounts of highly distributed data items, which need to be transformed into meaningful, reliable and available information for each mobile user.

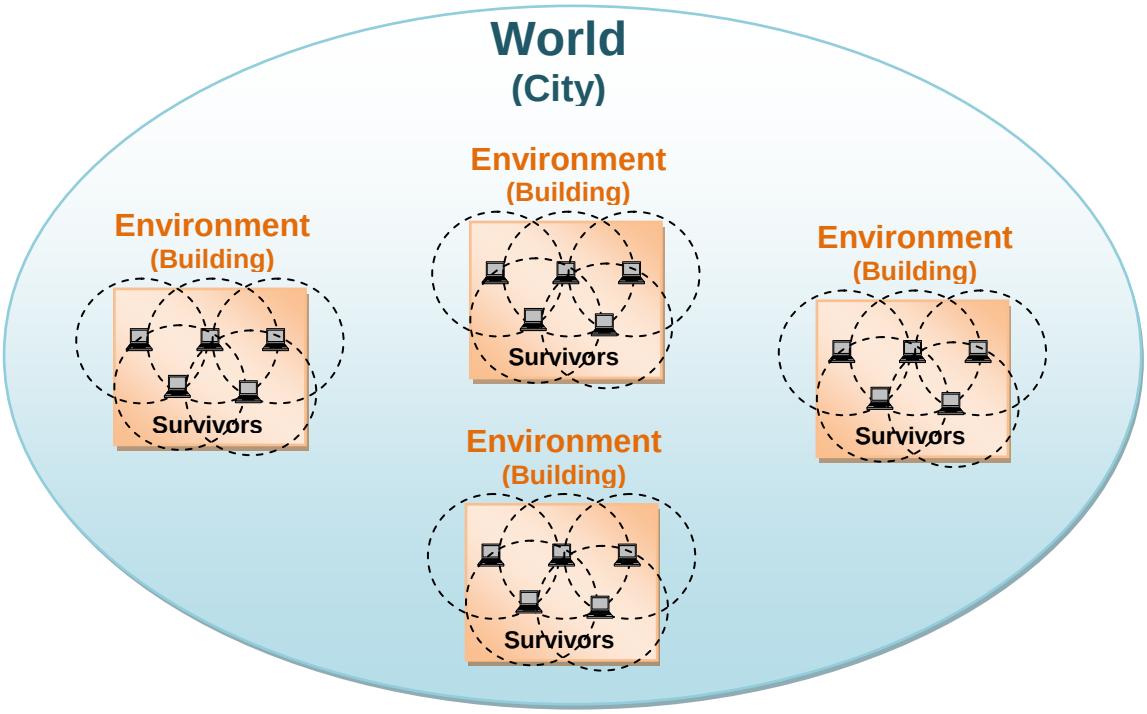


Figure III.7 City devised in terms of the DIET platform parameters

In order to be platform independent to a certain extend, the Java programming language is used to develop the prototype. The prototype provides all the components necessary to build ACEs which are the core elements used to develop services.

As Java has been chosen as the language to develop the prototype, Eclipse [13] is the Integrated Development Environment (IDE) selected for building, deploying and managing the software. It is an open source software framework

for Java developers, although it is able to extend its capabilities by installing plug-ins, such as development toolkits for other programming languages.

4 Start-up Protocol

This protocol is defined as messages' transaction produced when agent is created (Figure III.8). Using this protocol, the agent gives to know its existence and capability, as well as it scans devices that has within its reach. This notification only takes place with agents who are within the action area, therefore, its neighbours. The connections created will be useful to make work the Aggregation Protocol, as the Aggregation Protocol needs some pre-existent connections to be able to destroy old connections and create new ones.

It is possible to make a parallelism by activating the WIFI device that every single simulated agent has in the practise case, in which, they perform a broadcasting of their presence in order to localize the closest agents. The peer to peer connections created towards these agents allow us to reach our objectives.

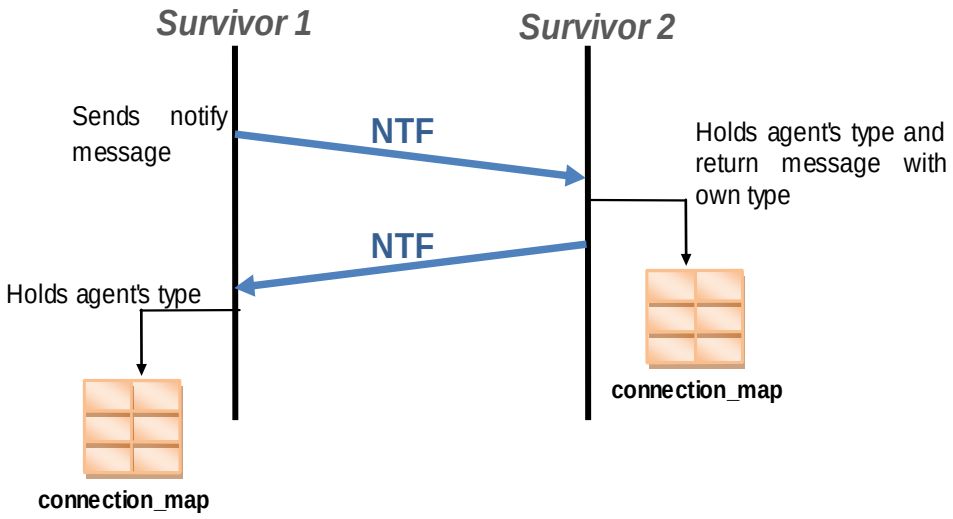


Figure III.8 Diagram of the messages transaction involved into the Start-up Protocol

Agent uses a Hashtable mapping connections to type the new connections added (Figure III.9). So, in the moment when the agent creates a new connection, it checks whether the connection is included within its Hashtable, checking through its functionality, which is indicated by the type_id value, if it is included inside the proper Connectionlist. If it is not included, the new connection will be included.

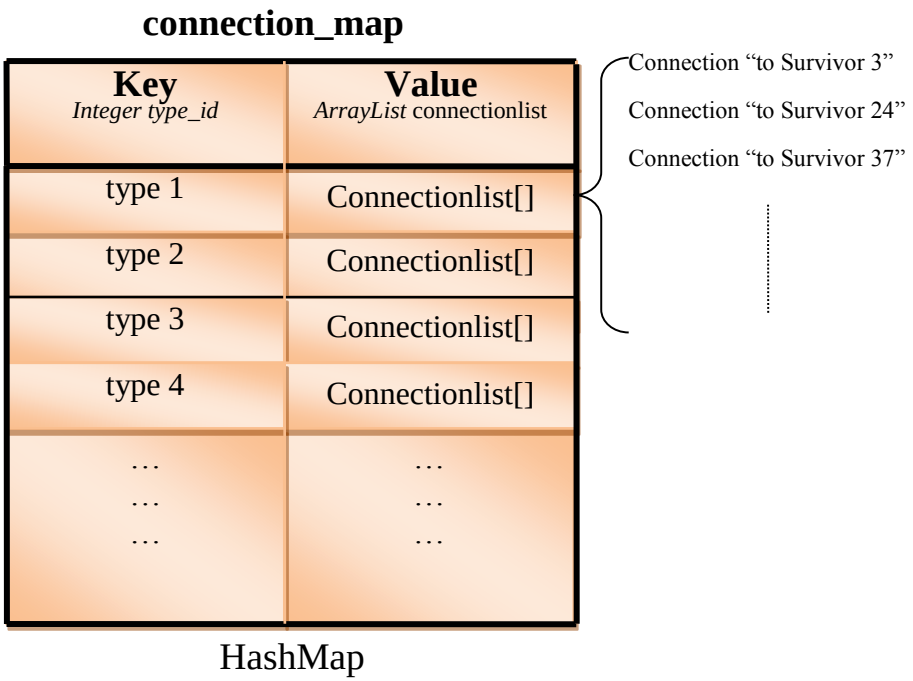


Figure III.9 Structure of the HashMap in charge of storing all connections discovered mapping them according to their type

The description and structure of the message involved into is described below:

5.3 NTF (Notify Message)

Message that is in charge of notify the existence of the agent to other attainable agents, likewise ordering the identification of the attainable agents. There are two types of notify messages, the go ones and the return ones. The go messages are the ones who start the discovery process and are identified by a "return_needed" value, which is equal to one, that indicates the necessity of an answer provided by the receptor agent. The messages with a "return_needed" value equal to zero indicate that they are the answer to a Notify message so they do not need any answer. Apart from the direction already included in the issuer agent by the DIET platform, the other information included in that message is the integer identifier, which indicates the type of funtionality of the agent (Figure III.10).

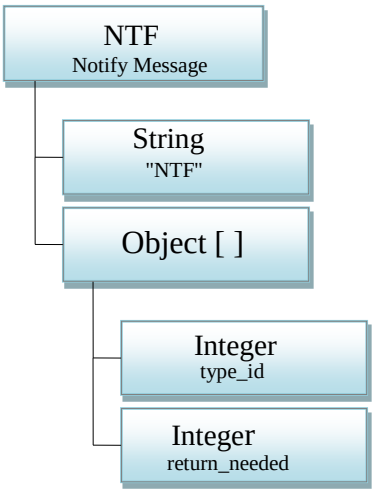


Figure III.10 Structure of the Notify Message

5 Aggregation Protocol

This protocol is defined as messages' transaction produced when agent is requesting a certain type of functionality. It is based in "On-demand" clustering algorithm from WP3 of CASCADAS' Project [10].

The Figure III.11 shows the interaction among the three behaviours included in the "On-demand" clustering protocol and implemented in each agent.

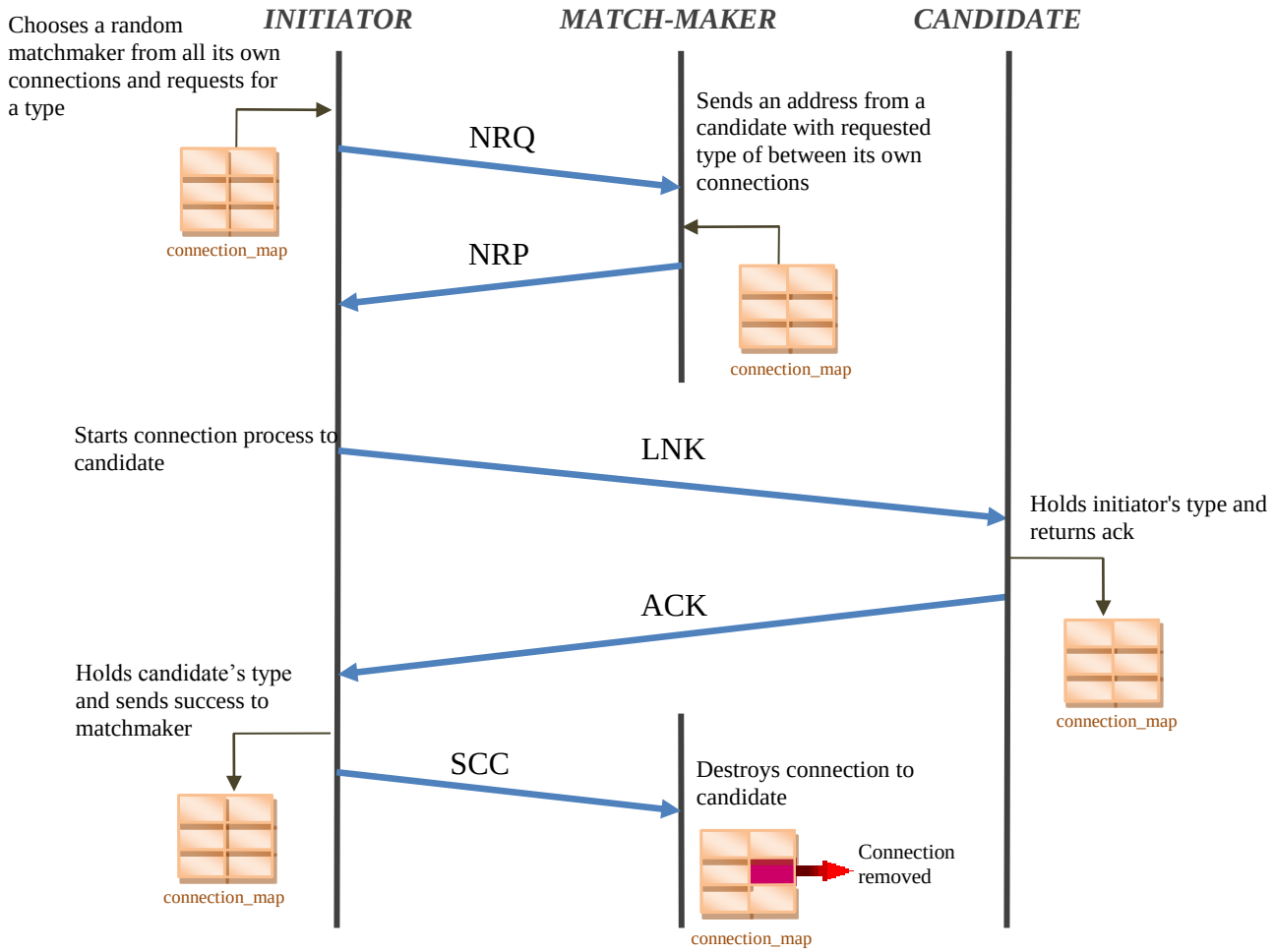


Figure III.11 Diagram of the messages transaction involved into the Aggregation Protocol

5.1 Protocol development stages

The protocol starts to work once an agent has been initialized by the Start-up Protocol and has a certain number of initial connections, so that it can start working. These connections will be modified, created and destroyed, according to the agent needs. The aggregation protocol is developed throughout the six following stages:

1. An agent wakes up and, according to the orders of its state machine, it may need to find another agent able to give an information or service. This is the moment when the protocol starts. In order to develop the requests, the protocol uses the NRQ message (Neighbour Request), in which, the agent required is described. That message is dedicated to a match-maker agent randomly chosen between all the available connections. If, after a while, the match-maker agent does not give any response with a candidate, the request is re-sent by the initiator agent to another match-maker randomly chosen between the rests of available

connections. This process will be repeated successively, as many times as needed, until an answer is obtained (Figure III.12).

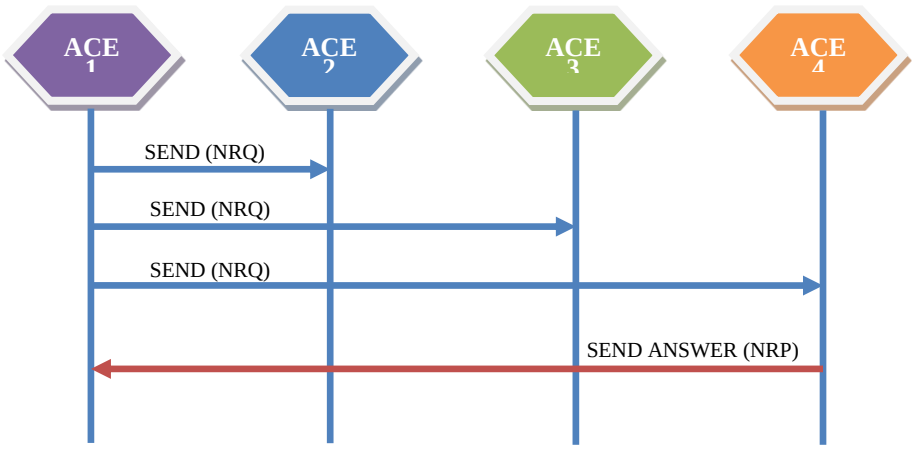


Figure III.12 Requests Messages are re-send by the initiator agent to the different match-makers chosen till receive an answer

2. When the NRQ message reaches the agent selected as match-maker by the initiator, the match-maker checks through its connection map if the agent contains any connection referring to an agent with the required typology. If that occurs, the match-maker will return a NRP message (Neighbour Reply) including the address of that agent.
3. Once the initiator agent gets the NRP message containing the address of the candidate proposed by the match-maker, the initiator sends to that candidate a LNK (link) message for the connection creation and keeps waiting an affirmative answer ACK. In the case of no response, and after waiting for a while, the aggregation algorithm is re-started.
4. When the candidate receives the LNK message it consults its internal reasoning machine in order to decide if it is able to create the connection with the candidate y, consequently, allow its creation by sending the ACK message. In the simulation, the no sending of the message only happens if those moments coincide with the creation of other connection with another agent, which depends on the power of the machine in which the process is executed and the internal limitations of the DIET platform [X].
5. Once the initiator receives the ACK message, allowed by the candidate agent, the connection is finally established and then it will be kept in the connection map of the two implicated agents. After that, the initiator will send a SCC (success) message to the match-maker. That is the moment when the match-maker role ends within the process.
6. When the SCC message is received by the match-maker, it will destroy the connection previously established with the candidate agent, the one who had proposed before to the initiator. This action is also framed inside "On-demand" clustering algorithm to conserve the homogeneity through the maintenance of number of connections in the system.

5.2 Messages involved

This section includes an explanation of the main messages that are involved in the protocol previously described.

All the messages are formed by two main parts:

- **A String:** defines and classifies the messages by its name.
- **An objects array**: includes the pointers to the information that takes part in the communication between two agents.

By the way the DIET platform works, all the messages are accompanied by the address of the agent who sends the message. Then, all the agents are able to know who sends every single message and, consequently, able to give an answer it that was necessary.

NRQ (Neighbour Request Message)

This message is in charge of ordering agents of a certain typology. It is sent by the agent whose role is initiator within the Aggregation Protocol.

As the information, it sends an integer message including a number that identifies the kind of agent required. That number indicates the functionality of the agent.

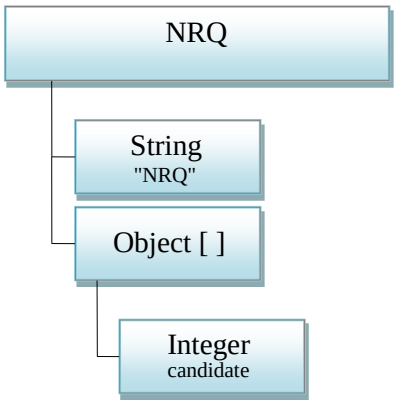


Figure III.13 Structure of the Neighbour Request Message

NRP (Neighbour Reply Message)

This message is in charge of answering an order about the type of agent. This is the answer to the NRQ message sent by the initiator. The NRP message is sent by the agent whose role is match-maker.

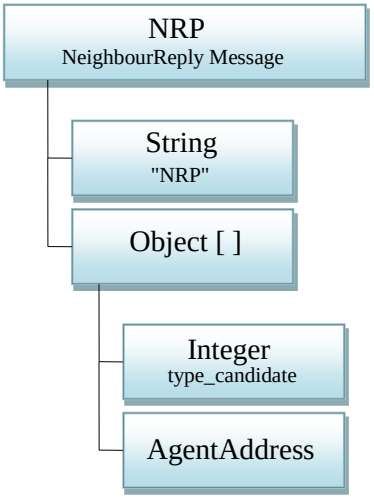


Figure III.14 Structure of the Neighbour Reply Message

LNK (Link Message)

This message is sent by the initiator agent and is in charge of the connection establishment with the agent selected by the match-maker (candidate). As identification, information about the initiator agent's functionality is included so that it can be taken and memorized by the candidate by the moment in which the connection is created.

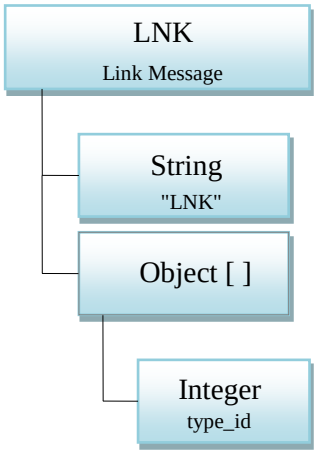


Figure III.15 Structure of the Link Message

ACK (Ack Message)

This message is in charge of transmitting the acceptation and availability of the Candidate message to the Initiator in order to establish the connection. The Candidate agent, the one who send the message, will take the last decision to finish with the establishment of the connection. Therefore, if the Initiator agent does not receive that message it should restart the search of other agent with the same functionality.

As information, that message includes the number that identifies the functionality of the Candidate agent, which is the one that sends the message

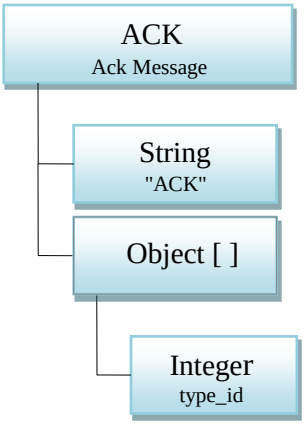


Figure III.16 Structure of the Ack Message

SCC (Success Message)

This message is in charge of introducing to the Match-Maker the connection's establishment on behalf of the Initiator and Candidate agents. So, following the steps for the connection's global maintenance, established by the Aggregation Protocol, the Match-Maker can finish the connection with the Candidate. The information included in that message is just formed by the candidate's address.

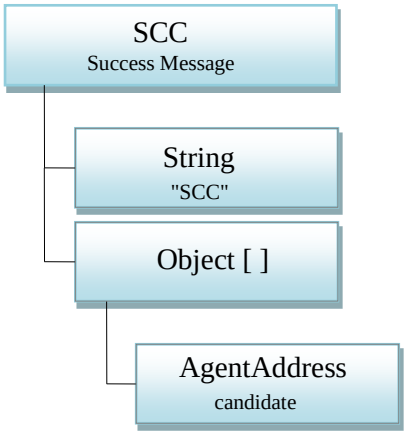


Figure III.17 Structure of the Success Message

6 Main architecture

The architecture of each agent is structured into the following functional blocks.

6.1 Reasoning Part

It has the task of managing the lifecycle of an agent. It describes the possible state and invokes the proper specific features, if specified, running all of them as a state machine. It works within the system as a DIET's job, in a parallel execution way.

6.2 Communication part

It is in charge of implementing communication among the agents. It includes the three behaviours described before in the "On-demand" clustering algorithm ("initiator", "match-maker" and "candidate"). It also includes blocks that are in charge of the information exchange contemplated in Reasoner's logic (messages including survivor's list, GPS position and so on). Jobs are the way of distributing the functionalities of an agent. Therefore, agents can compose their behaviour by combining multiple jobs. To implement the behaviour that responds to the communication protocol mentioned a structure of different jobs has been created. The main jobs' that intervenes in management of the communication protocol are described below:

- **NotifyNeighboursJob**: Job that on the start-up notifies in broadcast of its type of ID to the neighbours by creating connections. It also handles notifications of the neighbours.
- **RandomNeighbourRequestJob**: Job that initiates the request process for type items (Initiator behaviour).
- **HandleNeighbourRequestJob**: It handles requests, returning a random address of a candidate with the type requested (Match-Maker behaviour).
- **HandleNeighbourReplyJob**: It handles notifications of the type requested, starting the link process with the chosen candidate (Initiator behaviour).
- **HandleLinkJob**: It handles "link" requests from the initiator in the link process (Candidate behaviour).
- **HandleAckJob**: It handles "ack" responses from the candidate finishing the link process (Initiator behaviour).
- **HandleSuccessJob**: It handles "successful links" notifications to the candidate from the initiator agent destroying the connection with the candidate (Match-Maker behaviour).

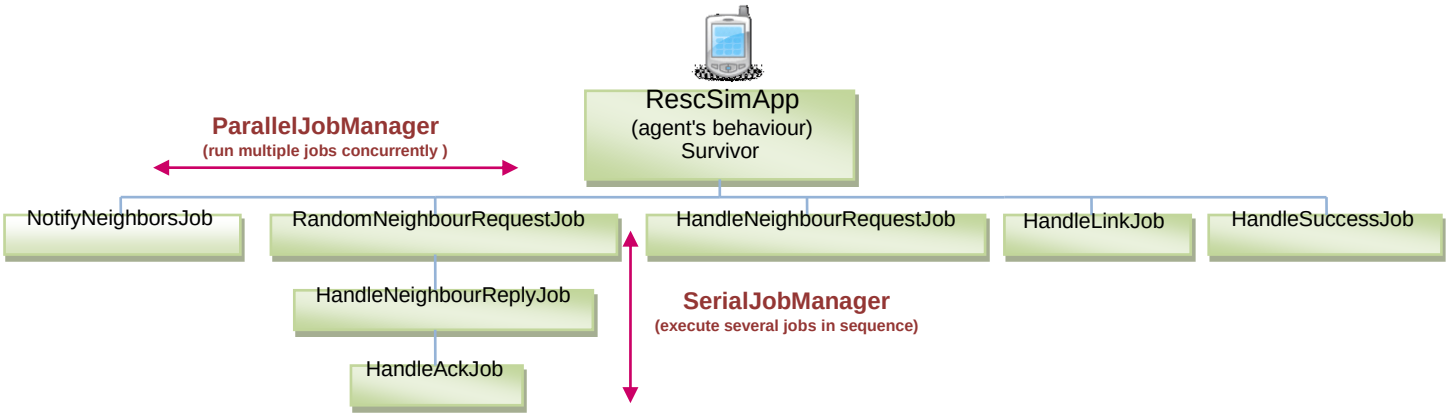


Figure III.18 Jobs' Structure working managed by the ParallelJobManager and the SerialJobManager which compose the Agent's behaviour

.

The created jobs works to the unison in two different forms: in serial or in parallel, controlled through DIET by their corresponding job manager:

- The **SerialJobManager** is used to execute several jobs in sequence. Once the first job has finished, it will start the second job, the third, the fourth and so on. This is useful when an agent's behaviour can be split into various stages. This sequence system is used to implement all the steps of the initiator agent behaviour ("wake-up", link creation, send success and so on).
- The **ParallelJobManager** is used to run multiple jobs concurrently. For instance, an agent may use a scheduler job to manage its schedule events, and another job that implements the specific behaviour to the agent which requires scheduling functionality. Therefore, the system designed in the simulation is served by this scheduler job to manage the tree jobs structure because it requires combinations of sequence and parallel execution.

A correct implementation of these jobs provides us an important feedback about the applied behaviour of the algorithm. Figure III.18 shows the tree structure distribution of the different jobs for the proper work of the communication algorithm.

6.3 Specific Part

It executes a normal code, depending on Reasoning Part's decisions, and returns the results so that they can be checked and sent back. For instance, there are specific functions for the survivors' list managing that confronts the information contained in them as well as the management of information that refers to the GPS positioning.

7 Application’s internal structure

To look for a more efficient mode of operation we have divided the development of application in different functional parts (packages). Each one takes charge of vital parts in the simulation like for instance: communication, positioning, graphic representation... Later we have united all these parts managed by a main class, to work in a cooperative way.

In the future the part relative to the reasoning will be added as another extension of the program.

Within the following points an explanation of the different packages is included, as well as a description the different kinds and main functions of them. Therefore, we could see an overview about how the internal application works.

The Figure III.19 represents the distribution of different parts including all of their classes.

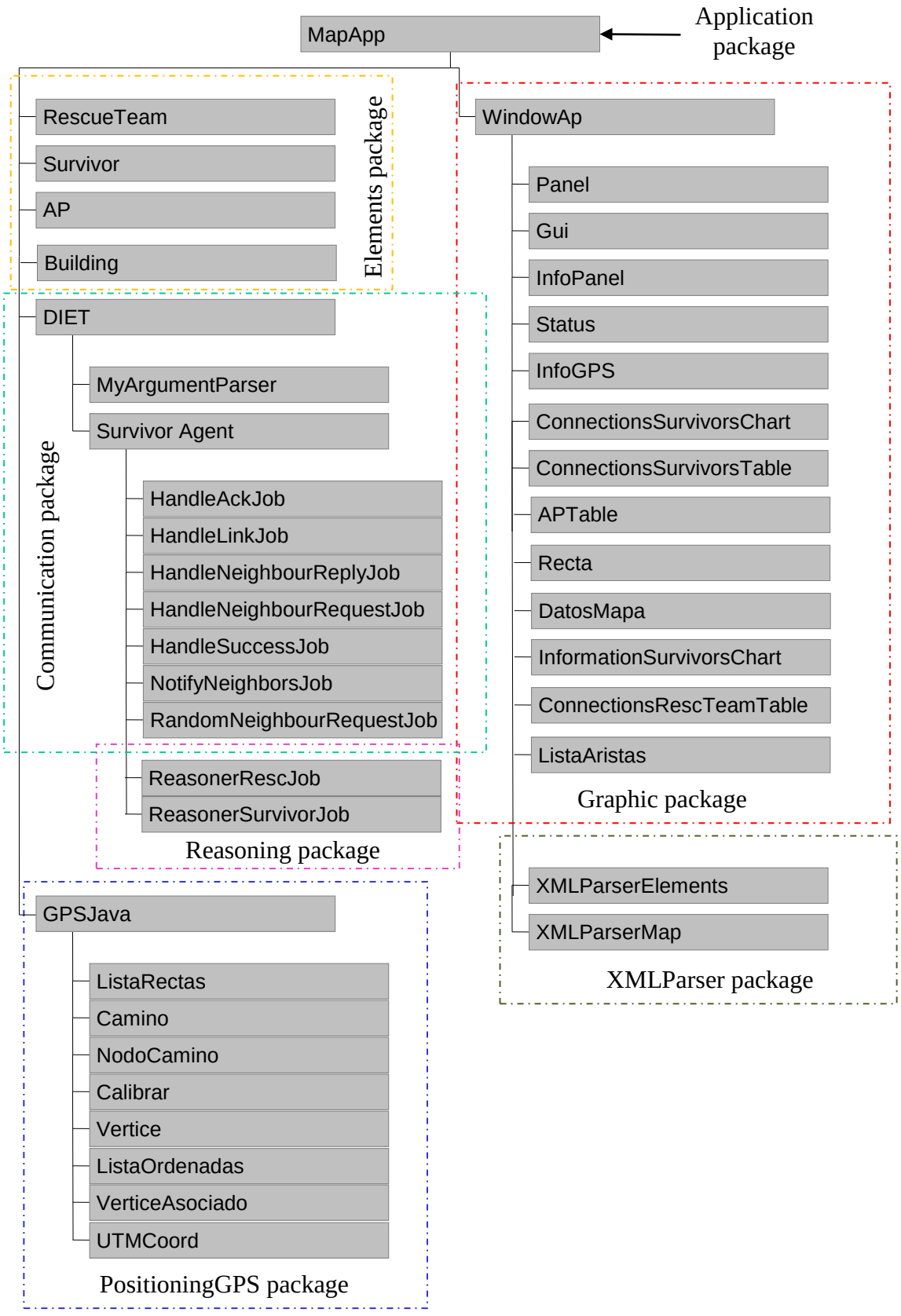
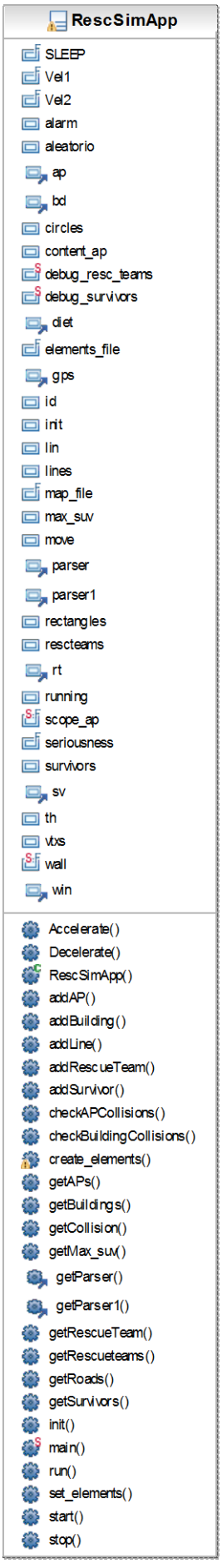


Figure III.19 Class Map of all classes composing the different packages

7.1 Application Package (simulResc.app)



In this package lies the main class, “RescSimApp.java”, from where the thread that controls the rest of the classes born and dies. A deep description of the parts of this main class will be included, as it is the class of which they spread the others. In this class there are located the global variables that form the following principal parameters:

- **String map_file/elements_file**: variable where the XML configuration file’s path can be found. It is read by the “XMLParser” package.
- **double Vel1/Vel2**: variable where it is located the numerical value for the displacement speed of the rescue teams’ agents (value expressed in pixels per time refresh). The first variable indicates the speed with the refresh deactivated acceleration and in the second activated variable.
- **int SLEEP:** variable that configures the repaint speed in milliseconds.
- **int scope_ap:** variable that indicate scope of the Access points distributed on the simulation map. It indicates the radius of a circular coverage, expressed in pixels. Actually, it is an approximation of a WiMax coverage carried out without bearing in mind the extenuation provoked by the buildings and the other elements, as well as obviating the losses of spread.
- **int wall:** it indicates the numerical value for the of the margin of thickness of the external walls of the buildings in order to get a major random distribution of the survivor agents in it. This value is proportional to the thickness of the graphical representation of the survivors.
- **boolean debug_survivors/debug_resc_teams:** boolean variable that control the visualization of the process information for its monitoring.

This class is also responsible for the creation of all the elements that will take part in the simulation (e.g. survivors, rescue teams, buildings, access points…) throughout the reading of the XML configuration files, where their number and attributes are described. It also includes some interesting functions in order to obtain information about the elements as well as the interaction between all of them. Within the following lines we can find a description of the most important functions:

- **RescSimApp**(): It is the builder class. It is in charge of reading the XML configuration files, starting with the DIET type that implement all the communication part and the GPSJava class that is in charge of the information's process and positioning routes of the different elements.
- **create_elements():** This function is in charge of creating all the elements that take part in the simulation. The XML configuration files contains all the information of the structure of the simulation map, which serve to construct the vertexes that represent the crossings of streets, the lines that represents the streets, the buildings spread around the city's map, the survivors who are located in the map as well as the access points and rescue teams in the indicated position.
- **checkAPCollisions():** This function controls the rescue team's access to any access point network area. That information its being updated inside the content_ap[j][i] counterfoil, where the j elements are the access points and the i elements the rescue teams.

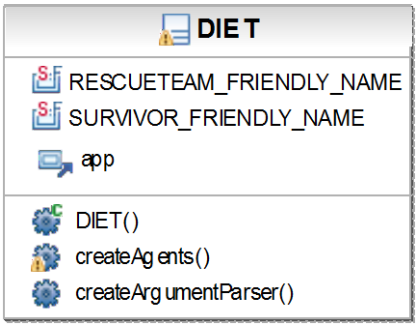
7.2 Communication Package (simulResc.communication)

This is the package that is in charge of establishing all the part that refers to the agent's communication that takes part in the simulation, the survivors and the rescue teams. Within the package, all the classes in charge of the reception and emission of messages are included, once per type. The whole functioning of all these classes gives the shape to the Self-organising communication protocol used by the agents.

The most important types of this package are described as follows:

DIET.java

This is the main class included in the package. It comes from the BasicApp class included in the DIET package that, as commented before, it is the open source application chosen that gives the support to establish the Self-organising protocol. Its function consists of creating the communicative elements associated to every single agent. In other words, it creates the communication part of each agent. This part is established following the functioning DIET directives. Therefore, it is focused by following the rules established on the tutorials and existing applications. A brief explanation of the main general variables would be the following:



- **SURVIVOR_FRIENDLY_NAME**: String variable that defines the visible name for the agent's survivor user.
- **String RESCUETEAM_FRIENDLY_NAME:** String variable that defines the visible name for the recue team's user.

The main functions of the type are:

- **createAgents():** This function is in charge of creating the DIET part of all agents. It creates as many agents as it is indicated in the XML configuration files and, within each one, every Job class in charge of the reception and sending of a specific type of message. As commented before, the jobs are arranged inside a functioning hierarchy in which they have to work in parallel and series according to their participation within the general protocol of communication. Consequently, and before the Jobs' class creation, the different JobManagers are created (parallel and serial) in order to be able later to arrange all the different types in a specific order. Once all the Jobs and its respective variables of access and information mapping are created, the following step is to create the correspondent DIET agent throughout the DIET function called createAgent.

  This process will be repeated for the rescue team agents as well as for the survivors, not forgetting the fact that each of them has a 128 bits Tag that tells the difference between them.

- **createArgumentParser():** It creates a new class called MyArgumentParser, which is in charge of reading all the configuration data throughout the XMLParser to be only used from the communications package. The class configure many variables which spread from the DIET library, like number of environments (as buildings in our prototype), as well as other variables regarding to the agents' type of communication, the agent's friendly name, its message buffer's size, etc…

Handle + type of message +Job.java

These classes are in charge of managing the kind of messages that its own name indicates, as well as replying with the specific answer in case of being required. The Figure III.20 shows a list including the different handle types and all the messages it is in charge of, both reception and sending.

All of them have a similar structure in which they do the superior function handleMessage from the DIET package. They detect the arrival of every new message throughout events, but they only deal with the message they are ready to deal with. By reading the internal string of the message and checking the structure of its contents, they are only able to serve the message they are programmed to attend. From that moment, they extract the information of the message, process the information of that message and reply with an answer if necessary, a message.

Java class	Message received	Message replied
RandomNeighbourRequestJob		NRQ NeighbourRequest Message
HandleNeighbourRequestJob	NRQ NeighbourRequest Message	NRP NeighbourReply Message
HandleNeighbourReplyJob	NRP NeighbourReply Message	LNK Link Message
HandleLinkJob	LNK Link Message	ACK Ack Message
HandleAckJob	ACK Ack Message	SCC Success Message
HandleSuccessJob	SCC Success Message	

Figure III.20 Table with different handle types and all the messages in charge of reception and sending

7.3 Elements Package (simulResc.elements)

This is the package where we can find the different classes related to all the simulation elements. All of them are created at the beginning of the application from the RescSimApp.java, once the configuration XML files have been read, and there are as much classes as elements are in the map. All of them are equipped with getter and setter methods in order to manipulate the data, besides other certain functions. The classes we can find are the following:

- **AP.java:** It refers to the Access Points allocated all around the city. They include information of their location, scope and, also, an arraylist containing all the environment addresses corresponding to every building they serve according to the DIET description.
- **Building.java:** it refers to the buildings that are allocated in the map. Every single object that it creates by the main class refers to every single building of the simulation. This class includes information referring to the location, size and number of survivors as well as the environment addresses that define the building.

- **RescueTeam.java:** This class refers to the existing rescue teams. It includes information about its GPS position, environment address as well as the objective and it state. That state may vary whether it goes to an objective or comes back to the hospital. This class implements the functions of its movement by using the positioningGPS package.
- **Survivor.java:** This class refers to the survivor agents. The class contains information about their position in pixels inside the map, supplied randomly by the main class within the frame that define the area where the building is located. It also includes information about each survivor profile, their type defined by the service it is able to provide, their age, health condition, blood type…

  Through the GPS package functions, the agent will have its positioning from the moment of its creation, just if the agent has the GPS capability built-in. Also, a survivors list is included, which refers to the knowledge of the survivors' agents about the rest of agents that are located inside the same building where the agent is located. In order to manage, it disposes the corresponding functions to confront the survivors' list that have been acquired by its neighbours, throughout the Aggregation Protocol, and thus be able to include the new found agents in the list. Finally, we cannot fail to mention that, we can also find the function to find out the percentage of knowledge of each survivor, being the following its calculation:

  - GPS positioning: 40% knowledge
  - Complete survivors list: 60% knowledge, being calculated through as an average between the number of the new found

agents and the number of the total of agents that we can find inside the building:

$$0.6 \times \left\lceil \frac{\text{number of holded\_list\_survivors}}{\text{number of building\_survivors}} \right\rceil$$

7.4 Graphics Package (simulResc.graphics)

This is the package where the different classes corresponding to the graphical data output elements are, as well as the interface application. The data collection represented in the screen includes different data like data to control the number of connections, the status list of the survivor agents, status of the rescue teams' agents, survivors' environment knowledge chart, graph including the section of the city we are studying, agents movement routes, GPS positioning data… up to the access points coverage data.

Within the interface section a description of its functioning will be included, as well as the data output interpretation. The main class of this package is the WindowAp.java, where the rest of classes are created. The java classes included are the following:

- APTable.java
- ConnectionsRescTeamTable.java
- ConnectionsSurvivorChart.java
- ConnectionsSurvivorTable.java
- DatosMapa.java
- Gui.java
- InfoGPS.java
- InfoPanel.java
- InformationSurvivorsChart.java
- ListaAristas.java
- Panel.java
- Recta.java
- Status.java
- WindowAP.java

7.5 GPS Positioning Package (simulResc.positioningGPS)

The GPS positioning Package includes classes that help the rest of the application classes. The agents use it in order to obtain a GPS positioning in

UTM (Universal Transverse Mercator) coordinates as well as a route calculation to get to a certain position through an algorithm.

In order to obtain a more or less realistic positioning of the agents, an initial calibration is needed, using the section edges data from the map used within the Calibrar.java class. Thus, by squaring, a transformation of the position into pixels can be done (the number of horizontal and vertical pixels from the top-left corner) within the graphical representation of the agents to another represented by UTM coordinates (longitude, latitude), regardless the altitudes in this case.

The calculation for the quickest routes to get to a position in the map is the assistance to the rescue teams' agents in order to reach their aims in the most efficient way. In order to do that, an algorithm inspired by the Dijkstra [12] algorithm has been used, as weights for every section of the street between junctions, depending on their distance, have been assigned. In this case, the distance has been the only variable taken into account, regardless realistic algorithms where the kind of lane (quick and slow), information about the traffic, road works and so on are taken into account. Our algorithm has been called "a-star algorithm" and it is located within the GPSJava.java class.

The classes that are included in this package are the following:

- Calibrar.java
- Camino.java
- GPSJava.java
- ListaOrdenada.java
- ListaRectas.java
- NodoCamino.java
- UTMCoord.java
- Vertice.java
- VerticeAsociado.java

7.6 Reasoning Package (simulResc.reasoning)

This is the package where the reasoning of the agents that take part in the application is. Depending on the existing types of agents, there are two behavioural models, which are defined by two classes of java:

- **ReasonerRescTeam.java:** Refers to the rescue teams' behaviour design. We can see the status diagram about the logic that has been used in figure III.21:

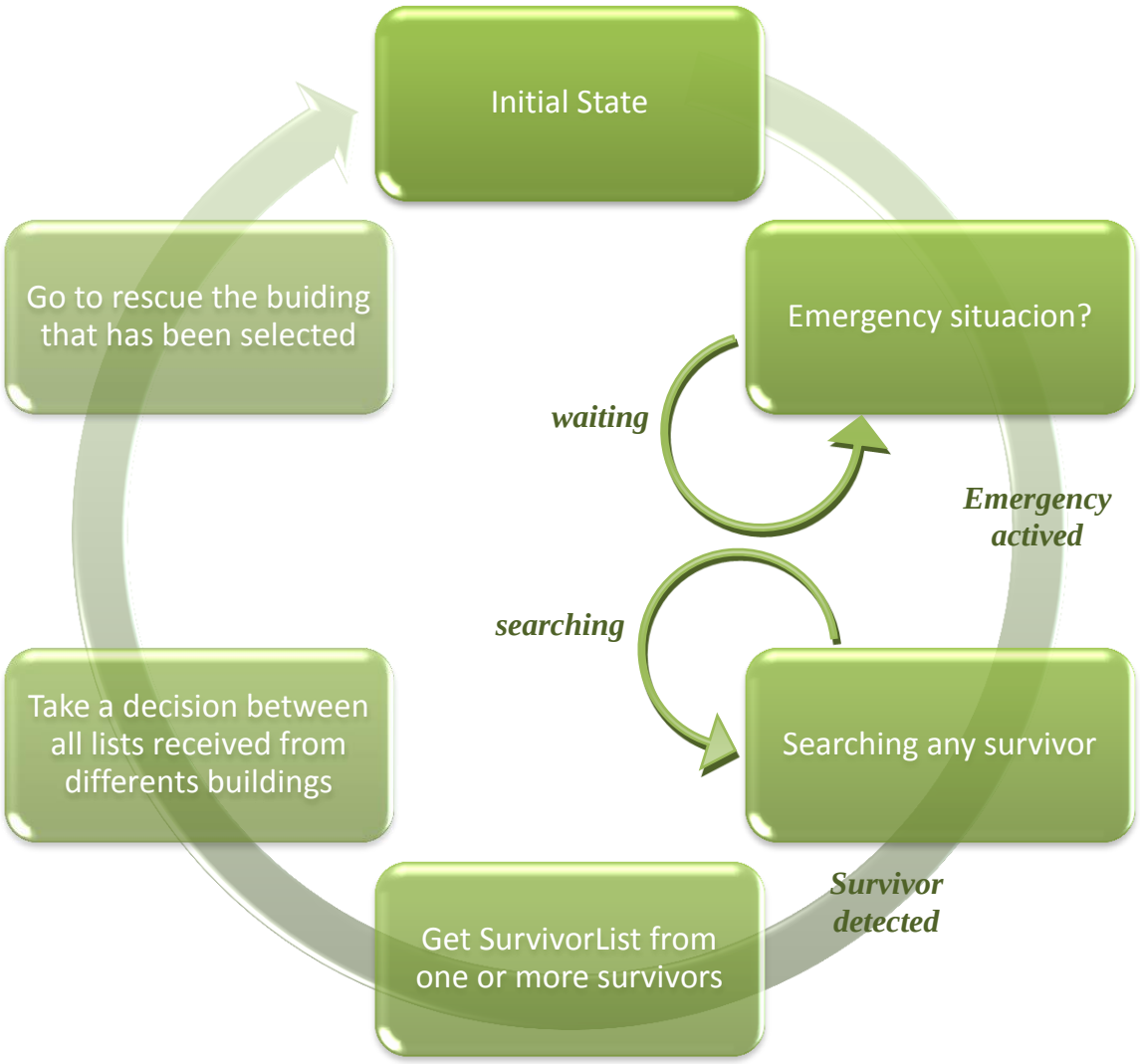


Figure III.21Self-model designed to configure the rules of the Rescue Team's reasoning

- **ReasonerSurvivor.java:** refers to the survivors' behaviour design. We can see the status diagram about the logic that has been used in figure III.22:

Figure III.22 Self-model designed to configure the rules of the Survivor's reasoning

These classes work as jobs within the DIET architecture. Therefore, they work in parallel and independently from the rest of the internal modules of the agent. They are in charge of evaluating the environment and the agent's situation in order to take the proper decision to attain their objectives.

This first development has been done in a relatively simple way in which a status pattern has been defined, where the transitions occur once the needed features have been reached. In future implementations, as it will be defined with all details within the future work section, the basis to give more dynamism and flexibility to that logic have been established, introducing the so-called an environment's adaptation in which the system will be able to vary its status machine before any unexpected situation. This will be done throughout the introduction of a logical programming language (Rule Markup Language [4]).

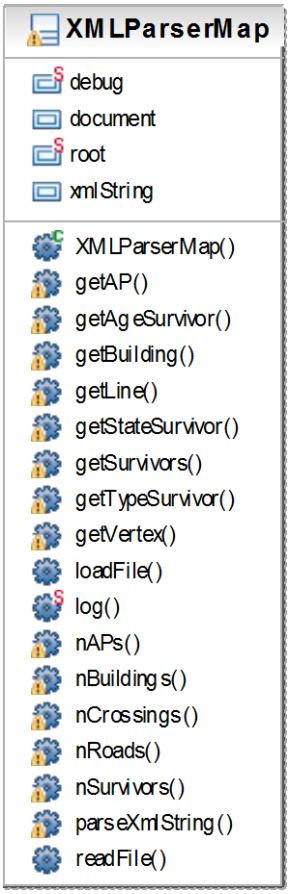
7.7 XMLParser Package (simulResc.XMLParser)

In this package we can find the two different classes that are in charge of get back the XML configuration files' data. Its methods are in charge of, through an intelligent way, recover, process and provide to the application all the data contained in the configuration files. Most of the methods are called from the main class RescSimApp.java at the beginning of the application, given that it is in that concrete moment when the simulation map including all the elements (streets, buildings, Access Points, Rescue Teams, survivors and so on) is set up.

The two classes that form this package are the following:

XMLParserMap.java

This class is in charge of both get back and process all the data that is included in the configuration file MapGps.xml. Within its methods, the most important are the following:

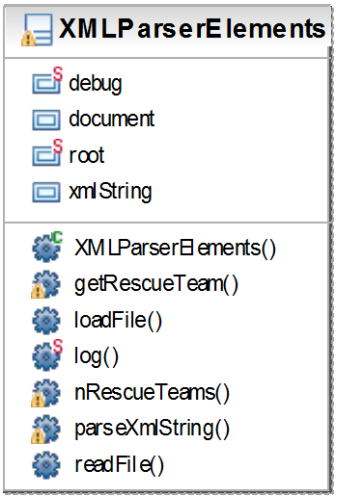


- **getAP():** brings back the Access Points list taken from the configuration file with all its position details.
- **getAgeSurvivor():** brings back the survivor age value, from the indicated survivor.
- **getBuilding():** brings back the position and shape of the indicated building.
- **getLine():** brings back the start and end vertex of the indicated line. That line corresponds to a certain street in the map.
- **getStateSurvivor():** brings back the health status of the indicated survivor.
- **getSurvivors():** brings back a vector including all survivors that are included in the configuration file, as well as its location building.
- **getTypeSurvivor():** brings back the type of functionality that the indicated survivor is able to develop.
- **getVertex():** brings back a vector including all vertex that form the map, as well as the position of each of them with high and wide pixels value. The vertexes correspond to the street's junctions in the map, which allow establishing the start and end of all the streets.
- **nAPs():** brings back the number of Access Points.
- **nBuildings():** brings back the number of buildings.

- **nCrossings():** brings back the number of junctions in the map.
- **nRoads():** brings back the number of streets.
- **nSurvivors():** brings back the number of survivors

XMLParserElements.java

This class is in charge of both get back and process all the data included in the configuration file Elements.xml. Within its methods, the most important are the following:



- **getRescueTeams():** brings back a vector including all the rescue teams included within the configuration file. Within the data that brings back, we can find the initial position where they will appear in the simulation. That information is included in a concrete vector that forms the map.
- **nRescueTeams():** brings back the number of rescue teams that are detailed in the configuration file.

8 Technological Approach

This section provides a description of the technology and algorithms used for the prototype development, as well as the necessary steps for the configuration of the prototype.

8.1 Prerequisites to set-up the ACE environment

The prototype requires an installation of the Java programming language runtime environment [14] in the version 6.0 SE; it is recommended to install the most recent JDK, as well as the source files and the API documentation for debugging purposes. This software can be found at http://java.sun.com/javase/downloads/index.jsp.

The prototype has been developed with the integrated development environment Eclipse [13], but other tools should also be fine.

The code source of the prototype and all the libraries required can be found attached to this document.

8.2 DIET Agents Platform

DIET (Decentralised Information Ecosystem Technologies) is a platform for developing agent-based applications created in Java. DIET platform [15], developed in the EU-founded DIET project, is an Open Source framework released under GPL license and downloadable from source forge web site.

DIET goal is providing an ecosystem-inspired approach to the design of agent applications. A bottom-up design was used to ensure that the platform is lightweight, scalable, robust, adaptive and extensible. It is especially suitable for rapidly developing peer-to-peer prototype applications and/or adaptive, distributed applications that use bottom-up, nature-inspired techniques. It is scalable at a local and at a global level. Local scalability is achieved because DIET agents can be very lightweight. This makes it possible to run large numbers of agents, up to several hundred thousands, in a single machine. DIET is also globally scalable, because the architecture is such that it does not impose any constraints on the size of distributed DIET applications. This is mainly achieved because the architecture is fully decentralised, thus not imposing any centralised bottlenecks.

The architecture of DIET software is layered, incorporating modularity that allows for the flexible extension of the framework (Figure III.23). The kernel of the DIET software resides in the bottom layer, the Core Layer. It provides the fundamental functionality available to all implementations in the DIET architecture, but also embodies the constraints under which all DIET agents must operate. The application reusable component layer (ARC Layer) includes optional components that are useful to various applications. It also contains general components that allow for the validation and testing of DIET applications. The Application Layer is the third layer and contains application - specific code. Associated with this layer may be validation components, to enable validation of applications developed using the DIET platform.

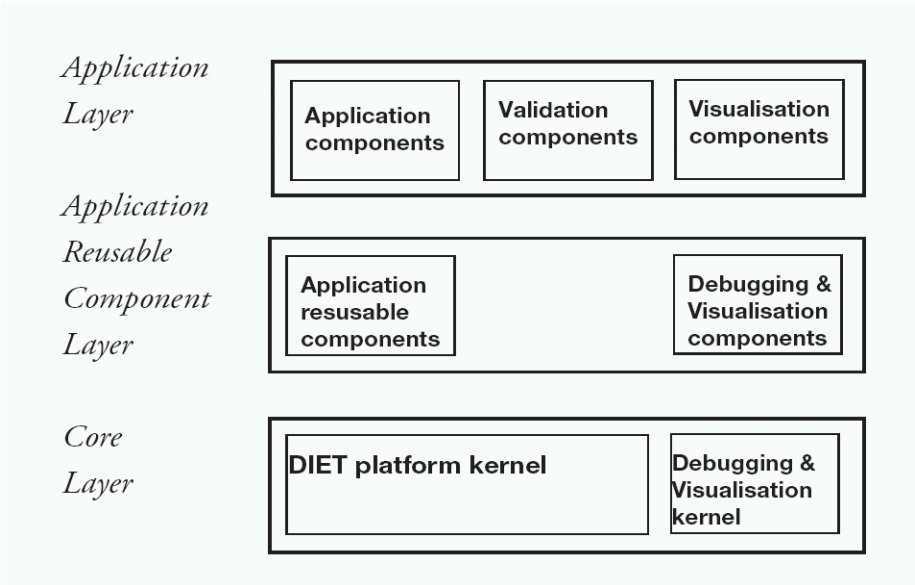


Figure III.23 DIET Architecture

Moreover it is robust and supports adaptive applications. The DIET kernel itself is robust to hardware failure and/or system overload. The effects of these failures are localised, and the kernel provides feedback when failure occurs allowing applications to adapt accordingly. The decentralised nature of DIET also makes the platform less susceptible to failure.

DIET Agents are not assumed to be highly intelligent and/or to use complex communication protocols. Instead, agents can be very small and simple, allowing intelligent behaviour to emerge from the interactions between large numbers of agents. Although the capability of each lightweight component itself may be very simple, the collective behaviours and the overall functionality arising from their interactions exceed the capacities of any individual organism.

Key features of the platform are:

- A clean layered architecture, with a kernel that is lightweight, simple and general.
- The fail-fast kernel constrains and minimises the use of threads, sockets and memory.
- Agents are autonomous yet lightweight, making it possible to run 100,000s of agents in a single VM.
- A model-event infrastructure provides sophisticated visualisation support.
- A thread-safe agent execution model makes programming new agent behaviours straightforward.
- Provision of extensible and modular agent behaviours, using jobs and event managers.
- Various implementations of remote communication are provided, built on top of the kernel.
- Because of all these features, DIET is an appropriate platform to develop the test and implementation of the communication protocol required in the prototype.

8.3 Java

The whole application has been developed taking Java [14] as programming language in its version 6.0 SE and the Java Development Kit (JDK) J2SE 6.0.

Java is a programming language originally developed by Sun Microsystems and released in 1995 as a core component of Sun's Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode which can run on any Java virtual machine (JVM) regardless of computer architecture.

Sun distinguishes between its Software Development Kit (SDK) and Runtime Environment (JRE) which is a subset of the SDK, the primary distinction being that in the JRE the compiler is not present.

The Java Runtime Environment or JRE is the software required to run any application deployed on the Java Platform. End-users commonly use a JRE in software packages and Web browser plugins. The SDK, more commonly known as the JDK, includes development tools such as the Java compiler, Javadoc, and debugger.

Java Platform, Standard Edition or Java SE (formerly known up to version 6.0 as Java 2 Platform, Standard Edition or J2SE), is a collection of Java programming language APIs useful to many Java platform programs. The prototype is implemented using J2SE JDK1.6.0_01, which represents J2SE 6.0.

The prototype also uses different tools to implement and improve its characteristics:

Java Reflection

Reflection is commonly used by programs which require the ability to examine or modify the runtime behaviour of applications running in the Java virtual machine. This is a relatively advanced feature and should be used only by developers who have a strong grasp of the fundamentals of the language. With that caveat it mind, reflection is a powerful technique and can enable applications to perform operations which would otherwise be impossible.

The prototype uses Java Reflection to call an action when a transition is running.

Java Swing

Swing is a GUI toolkit for Java and it is one part of the Java Foundation Classes (JFC). It provides facilities to help to the developer to construct Graphical User Interfaces (GUIs). For that, Swing includes GUI widgets such as text boxes, buttons, split-panes, and tables.

Refactoring

Refactoring a source code module often means modifying without changing its external behaviour, and is sometimes informally referred to as "cleaning it up". In extreme programming and other agile methodologies, refactoring is an integral part of the software development cycle: developers alternate between adding new tests and functionality and refactoring the code to improve its internal consistency and clarity. Automatic unit testing ensures that refactoring does not make the code stop working.

Refactoring neither fixes bugs nor adds new functionality. Rather it improves the intelligibility of the code or changes its internal structure and design, and removes dead code, to make it easier for human maintenance in the future. In particular, adding new behaviour to a program might be difficult with the

program's given structure, so a developer might refactor it first to make it easier, and then add the new behaviour. Refactoring is also a tool for removing bad code smells that exist in code.

Code refactoring alone does not change or add functionality to a system, but it can simplify future additions, application of design patterns, and reuse through polymorphism, which otherwise could be impractical, expensive, or unmaintainable.

Thinking about the future developers and to facilitate the intelligibility of the code, refactoring has been used in the prototype development.

8.4 Required libraries

After initial check-out the project needs to be configured: make sure to add the existing JUnit4 library to the build path plus all libraries from the lib folder of the distribution and to set the project's java version to 6.

Following libraries are used by the prototype and provided attached to this document:

- diet-agents-0_97.jar – DIET Agents platform library;
- elvis.jar – Needed by DIET Agents platform to run the agent's connection representation;
- jdom-B9.jar – An API to provide a complete, Java-based solution for accessing, manipulating, and outputting XML data from Java code;
- xercesImpl.jar – Needed by Jdom;
- jfreechart-1.0.5.jar – Free Java chart library that makes easy for developers to display professional quality charts in their applications;
- jcommon-1.0.9.jar – Needed by JFreeChart.

9 Interface

For development of the application we have chosen Java as programming language, so we take the advantages that its flexibility and its easy development capacity offers. Taking profit of these advantages we have designed a user-friendly interface to monitor all the variables that intervene in the simulation.

The interface will be described by means the real progress of the simulation so we will see the interaction between the simulation and the interface, and to realize the provided information by the different tables and graphs.

Start–up state

This is the initial state of the simulation. All elements are situated in its initial location as are determined in the configuration XML file. The Init button makes start the Start-up communication protocol.

The figure III.24 shows the graphical representation of real time simulation's main characters and belongs to the Map's lash. Moreover positioning UTM coordinates of each rescue teams are shown, as well as information about buildings' number, survivor's number and so on.

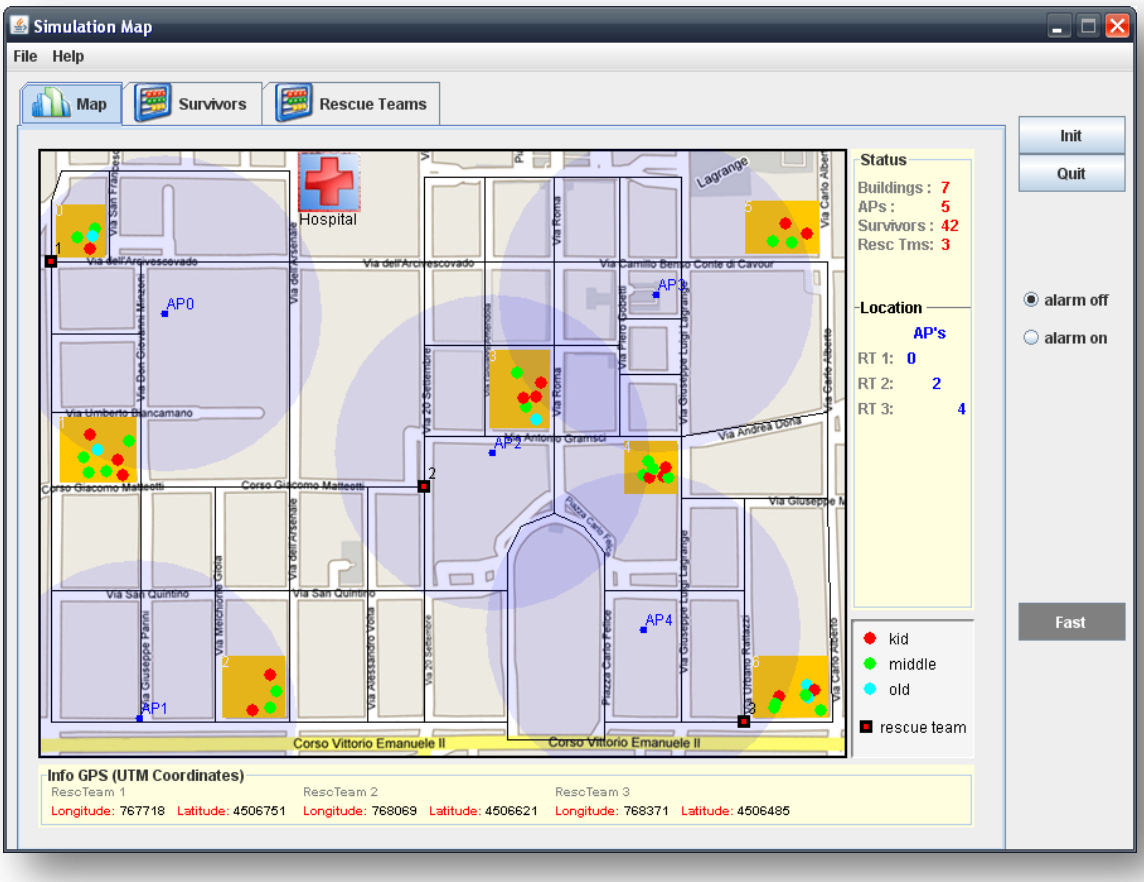


Figure III.24 Interface

Into the figure III.25 is possible to observe the initial knowledge of survivors with location capability. Therefore the GPS positioning is granted by a forty percent of knowledge. The rest of knowledge belongs to the number of survivors discovered from total which are placed in the same building as it's described before.

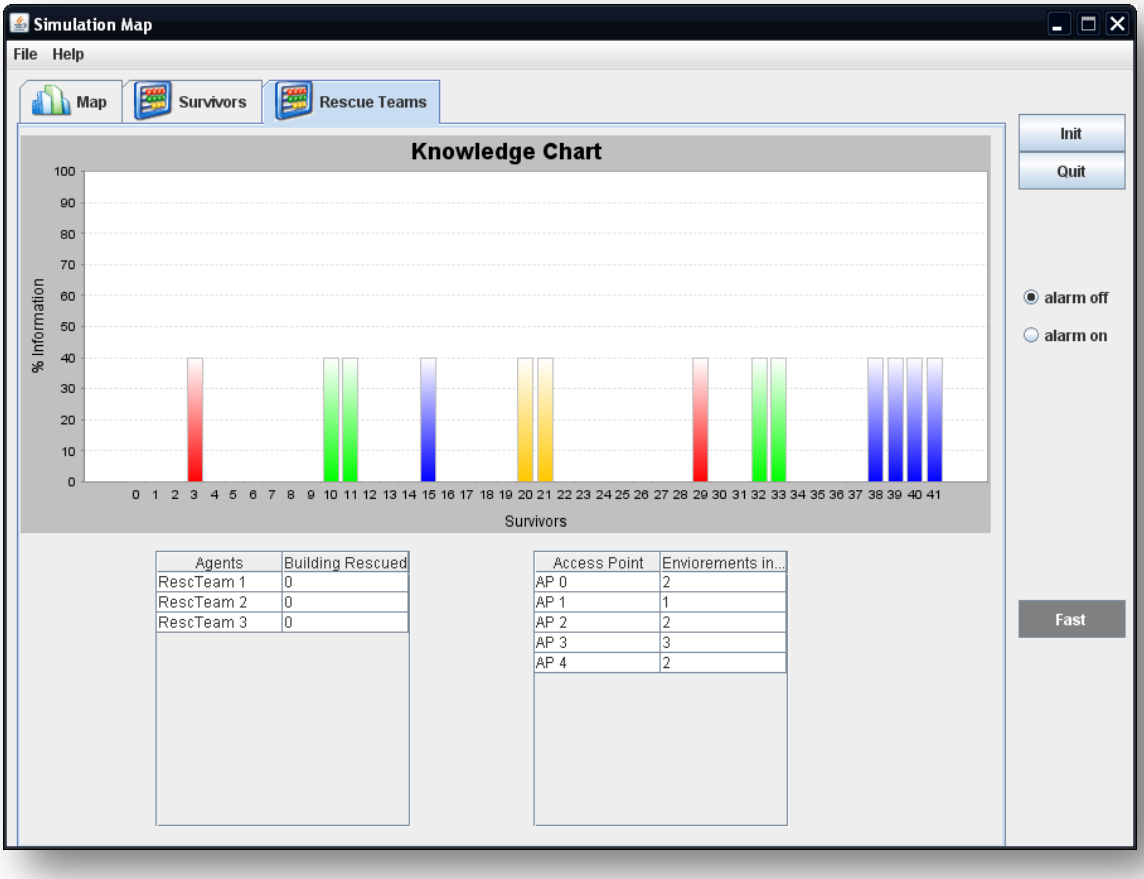


Figure III.25 Interface

First connections state

The image in figure III.26 corresponds to the lash in charge to show variations of connections' number that each agent has. That information come represented both a graph and a table. Their temporary variation can give us information about correct functioning of the communication protocol as well as the evolution of every agent. They also indicate the number of initial connections created by the Start-up protocol which are necessary to be ready to start the aggregation protocol into the survivor's devices.

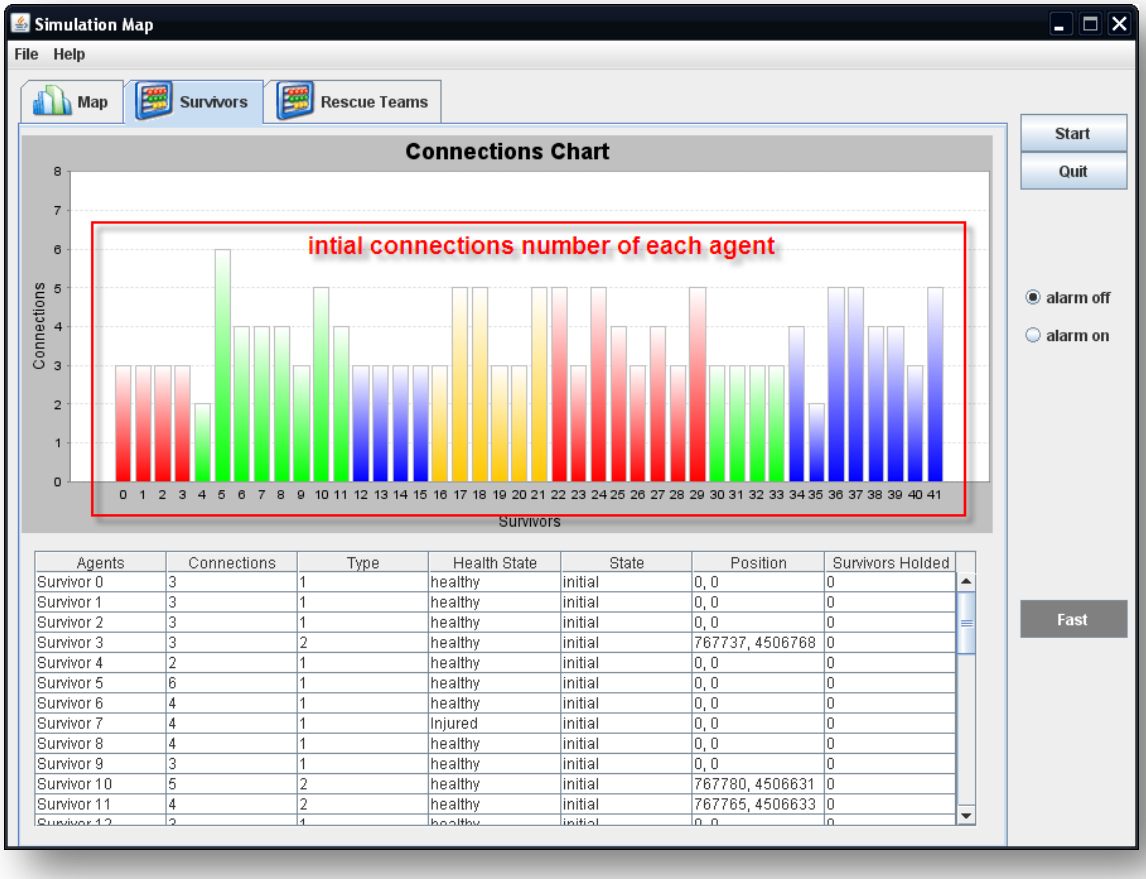


Figure III.26 Interface

Searching state

In figure III.27 we can observe the lash corresponding to the information table of the percentage of survivor‘s knowledge.

The calculation of this percentage has already been explained previously. In outline, the growth of every agent’s bar represents an increase in the environment’s knowledge which, in the current simulation devised, is the discovery of a new building’s agent.

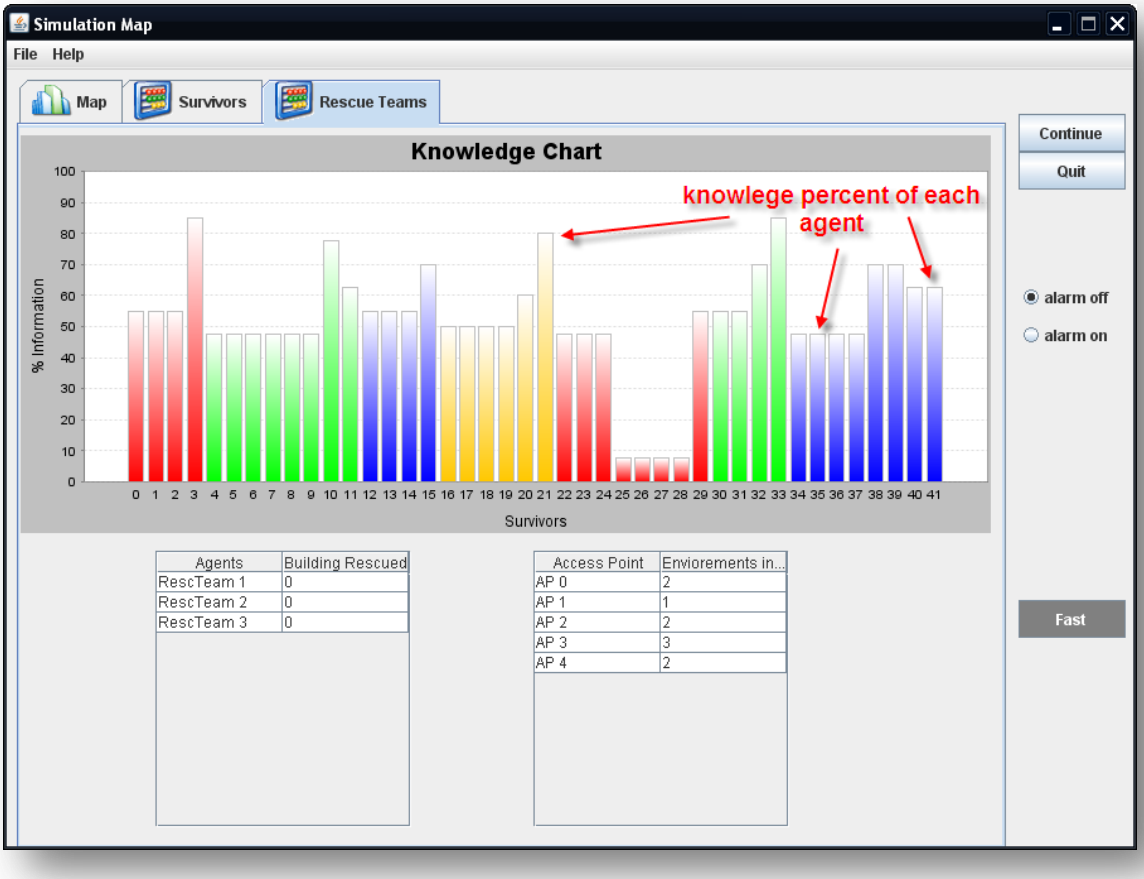


Figure III.27 Interface

Emergency situation

It is possible to activate the emergency status at any moment, being advisable to do it once the first interactions between the survivors have been produced. That occurs because the process times have been designed for its study, not being a real representation of how the real devices will be used.

In order to activate that status, the interface has a radio-button with the indications “alarm on” and “alarm off”. As a concept, they make a simulation of the emergency system’s activation, so all the rescue teams start to work. From the Map flap we can state that it has started working. Following the logic we have designed in the Reasoning package, they start to circulate through the streets in the map, which we are going to study, in order to identify possible survivors.

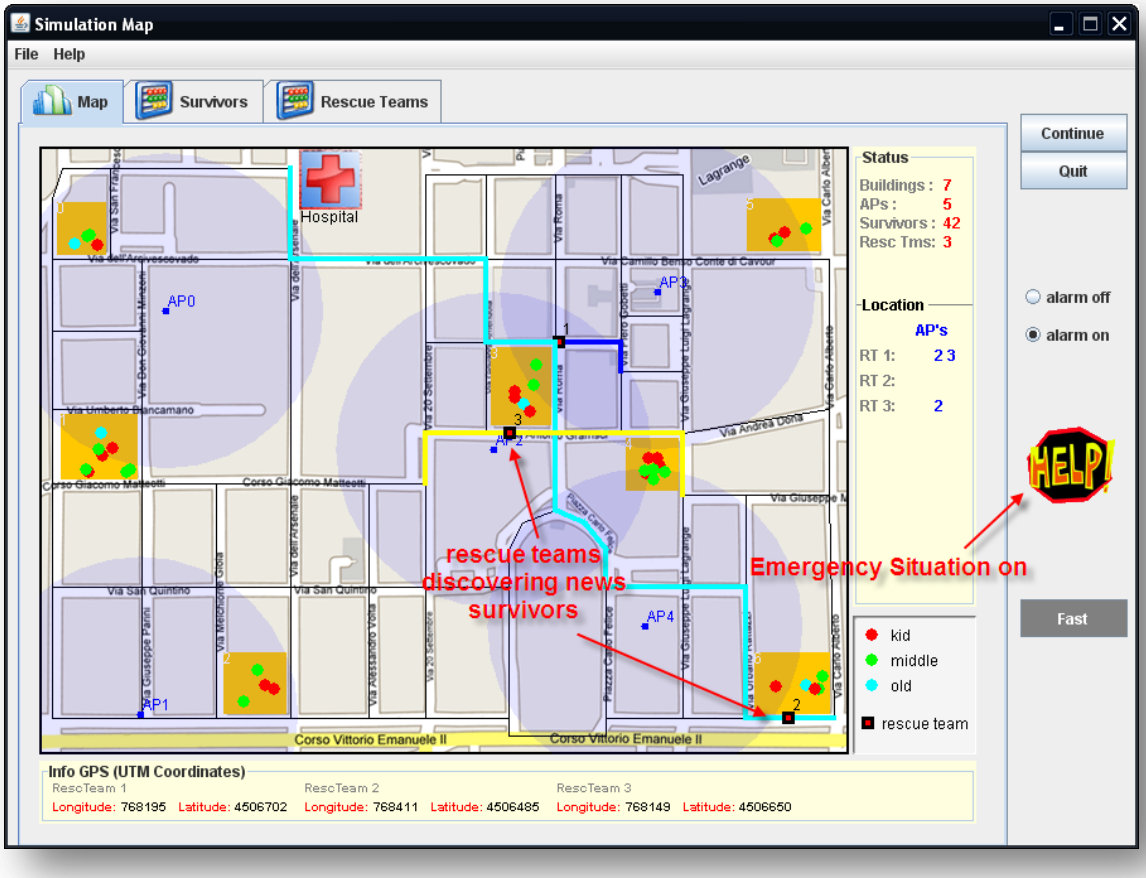


Figure III.28 Interface

Once the buildings that contain the survivors have been identified, they contact with any of them in order to get the information about the number and status of those survivors. Once the have got that information, they evaluate and decide which building from all the building's information gathered will be the objective to rescue. This decision can be seen in the table, which contains the flap, in picture III.28. As well as we can affirm that, in this simulation, the rescue team is able to gather all information from the buildings that are included within the Access Point's action scope.

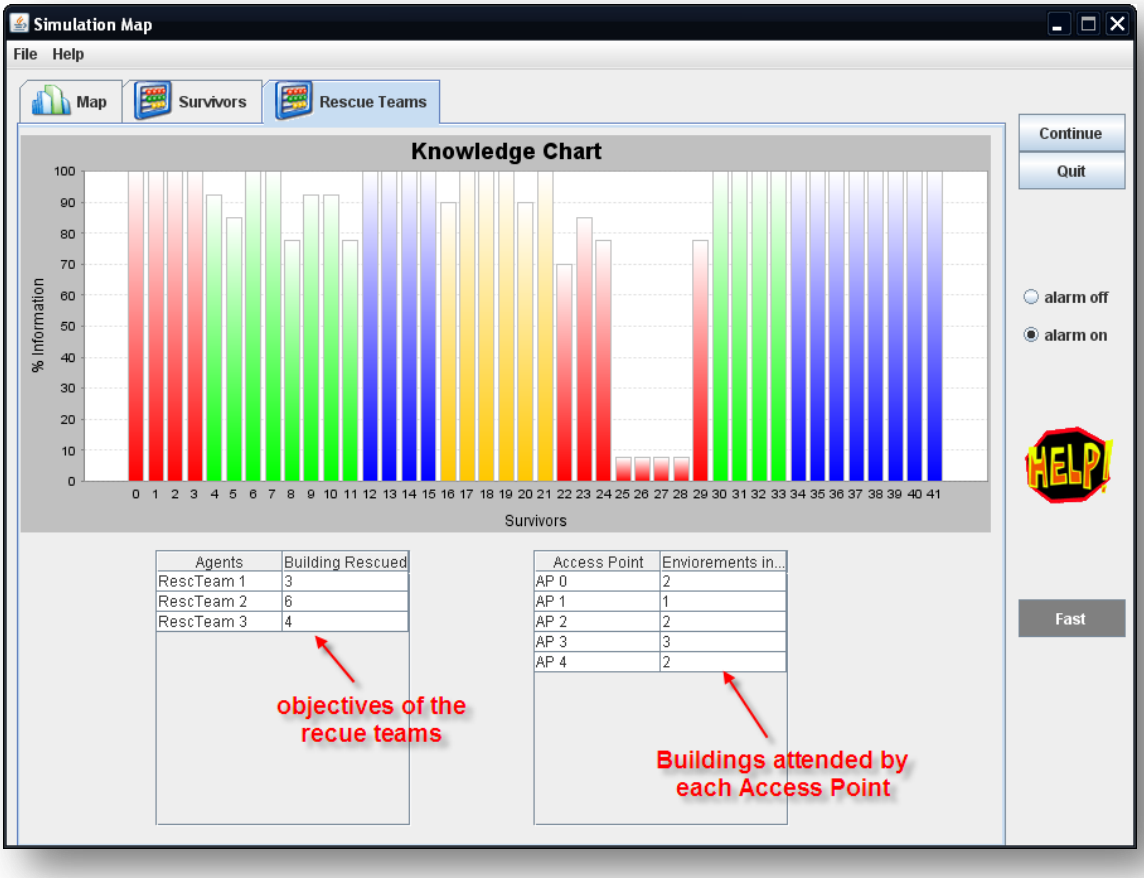


Figure III.29 Interface

Elvis

It is also possible to control network topology formed in each instant thanks to Elvis application included in DIET. In the figure III.30 is possible to observe different survivors distributed in groups (environments) just as it has been described previously.

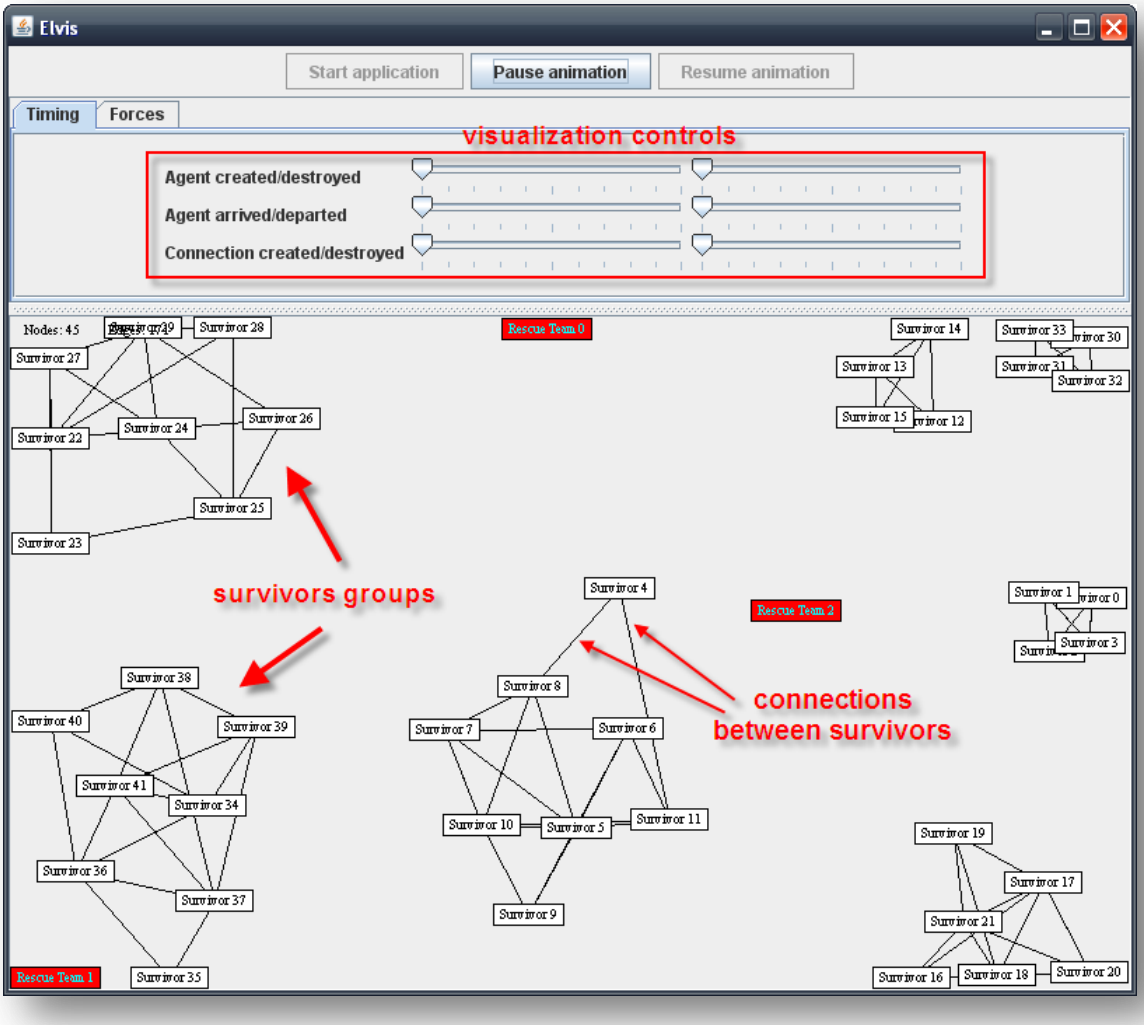


Figure III.30 Interface

10 Test results

This section provides some results carry out through different tests of the prototype. The first chapter presents the results of a validation test and the second analyses some features of the prototype.

10.1 Experimental results

This case-study has been chosen due to the fact digital city is one of the main scenarios to develop this technology.

In the digital city scenario, we find a mass of heterogenic and highly distributed information. The first objective set within each surviving building is to centralize information from all the media in each of the individuals.

Therefore, one of the purposes is measure the information propagation to get the full environment knowledge for each survivor. Finally, we calculate the efficiency of the communication protocol used (on-demand clustering) in environments with decentralized connections and self-organizing algorithms.

10.2 Prototype analysis

In an autonomic computing system, the prototype created expects to be a scale and functional model of the self-aggregation protocol.

The tests that have been realized show the information average distribution according to the execution cycles. As explained above, the function to find out the percentage of knowledge of each survivor, being the following its calculation:

- **GPS positioning**: 40% knowledge
- **Complete survivors list**: 60% knowledge, being calculated through as an average between the number of the new found agents and the number of the total of agents that we can find inside the building:

$$0.6 \times \left[\frac{\text{number of holded\_list\_survivors}}{\text{number of building\_survivors}}\right]$$

The number of survivors who have available GPS positioning has remained stable within the 25% in all tests.

Each execution cycle corresponds to a send or response action within the exchange described in the communication protocol, and the actions involved in the message.

To see how information can flow in environments with varying population, results are drawn below with a different number of survivors:

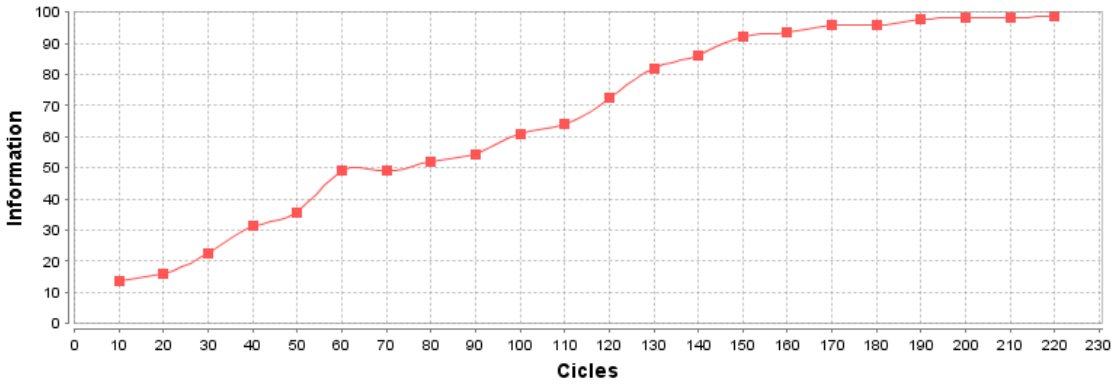


Figure III.31 10 survivors into the building

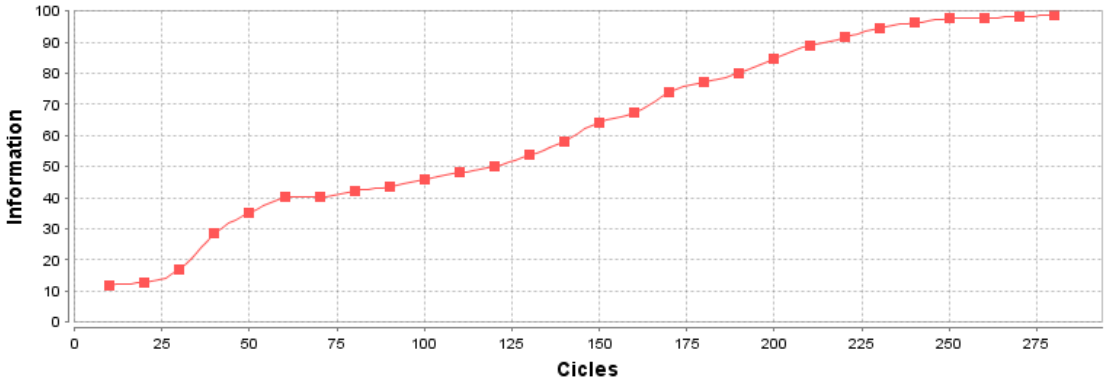


Figure III.32 25 survivors into the building

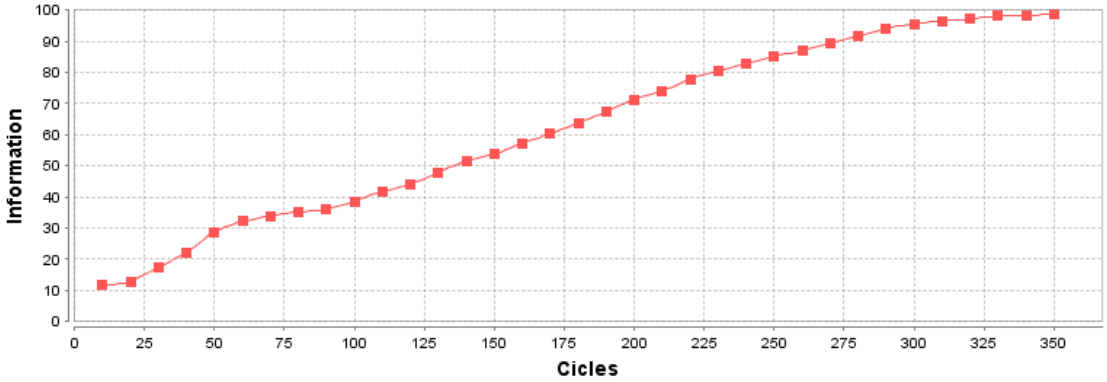


Figure III.33 50 survivors into the building

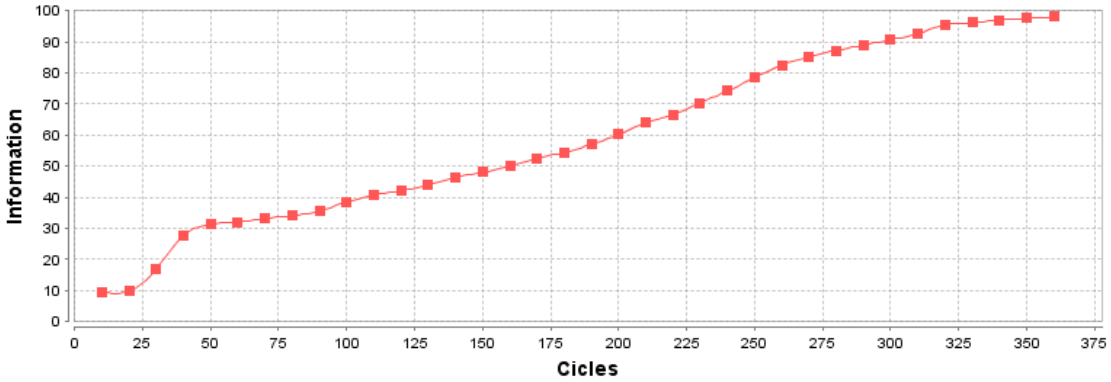


Figure III.34 75 survivors into the building

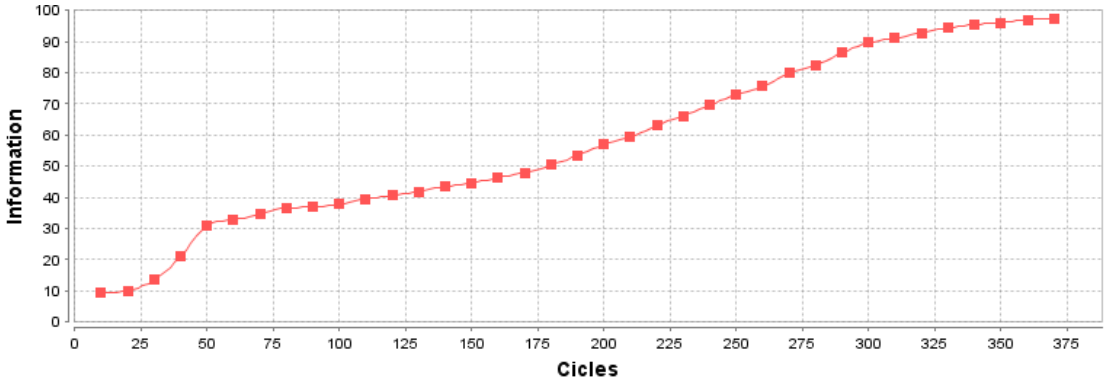


Figure III.35 100 survivors into the building

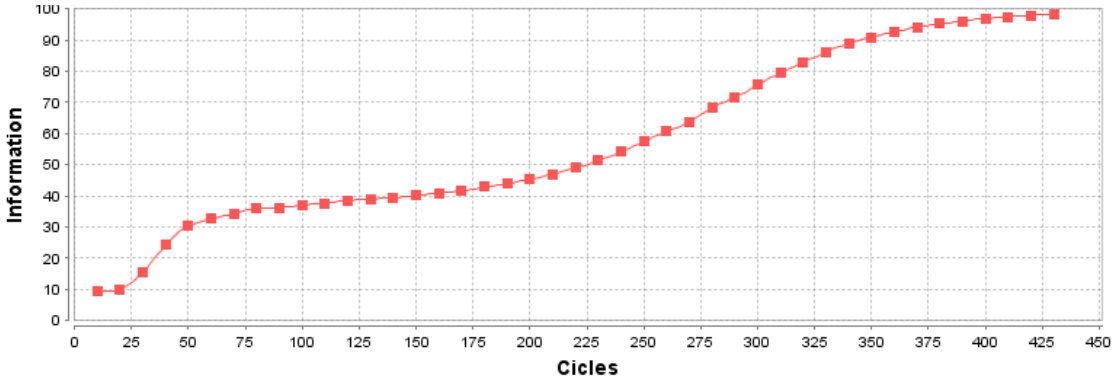


Figure III.36 500 survivors into the building

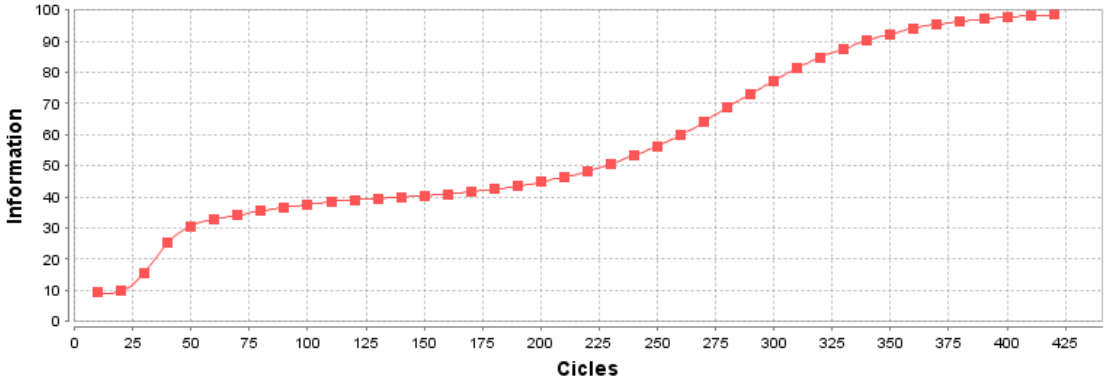


Figure III.37 1000 survivors into the building

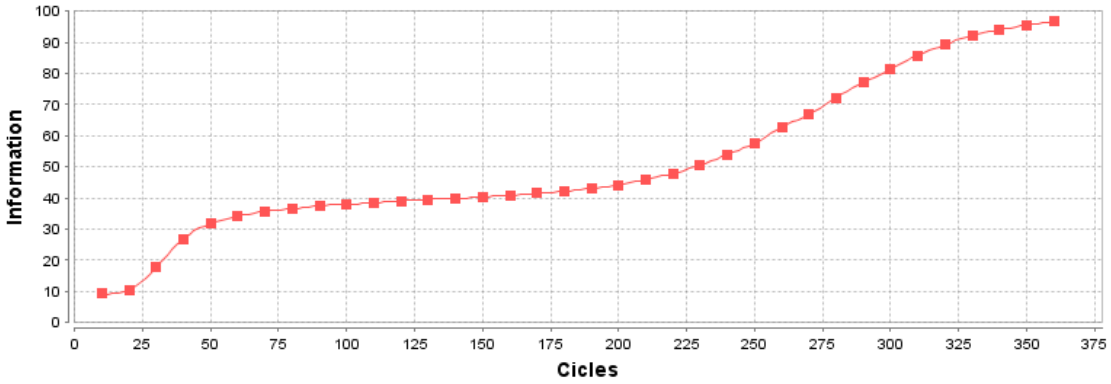


Figure III.38 1500 survivors into the building

Visualizamos ahora una gráfica con los diferentes datos recogidos de manera superpuesta, focalizado en mostrar entornos con un número de supervivientes entre 10 i 1000:

Next graph has been collected in a different data manner. Graph shows superimposed display environments focused on a number of survivors among 10 and 1000:

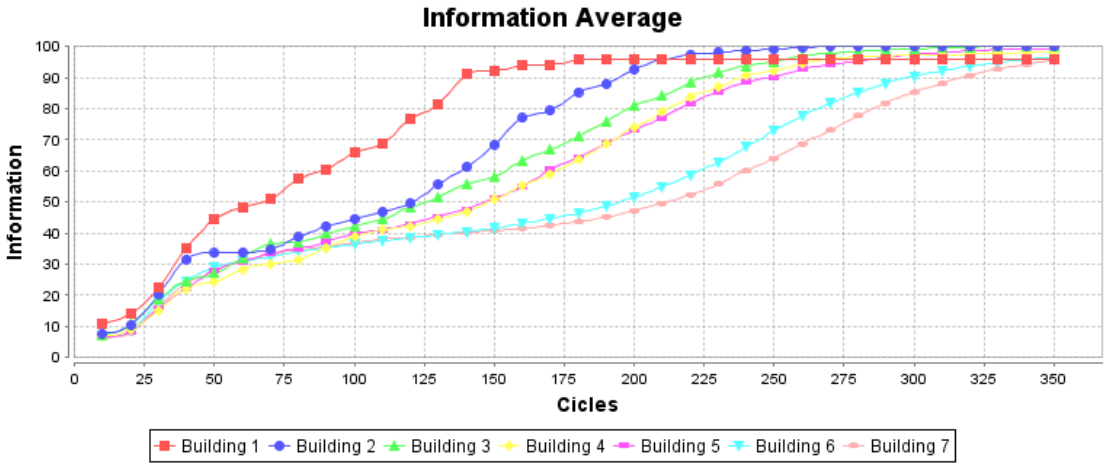


Figure III.39 Buildings among 10 and 1000 survivors

Where we can quantify the number of survivors to the building which interact them (environment) as follows:

- Building 1: 10 survivors.
- Building 2: 25 survivors.
- Building 3: 50 survivors.
- Building 4: 75 survivors.
- Building 5: 100 survivors.
- Building 6: 500 survivors.
- Building 7: 1000 survivors.

Regarding the data shown we can conclude that the growth curve of the average distributed information slows down when the number of agents climbed to 500 elements involved. Once passed the 500 survivors in the same environment, the curve moves into delimited margins. Accordingly, the degradation rate has an asymptotic distribution and not exceeds a number greater than 500 cycles of execution to move between 90% 100% of the information distribution target.

Finally, for the objective in simulation and for a large number of elements in the same environment, the communication protocol used to implement the self-organising algorithms presents a good and strong performance.

CHAPTER IV: Conclusion and Outlook

1 Conclusion

This document described the applicability of autonomic and self-organization capabilities in the execution of ICT services in distributed environments. It exemplified the exploitation key principles of autonomic self-organization to the development of solutions for handling wireless communication services even in critical disconnected situations (such as a catastrophic event in a city) and for handling the gathering, correlation and distribution of data.

In addition, the preceding chapters reported the main results achieved in the development of a prototype. The achieved results have demonstrated that the solution, designed by means of autonomic components enriched with reasoning capabilities, is meeting some challenging requirements such as adaptability to dynamic situations and robustness, even in environments with high churn rate and/or disconnected situations. Mainly, the prototype has proved the efficiency to include a reasoning module in autonomic agent environments. It makes possible introduce real-time reactions to changes in the environment conditions, making more achievable next digital cities.

Beside the description of the Autonomic Communications Element (ACE) design principles, its internal structure and applied communication paradigms which are given in chapter III, the document provides comprehensive overview of how to develop distributed applications using the reasoning module. As example, a rescue teams and survivors' environment has been presented.

This is a first version of a Reasoner. In its current implementation it provides all basic ACE functionalities which allow creating ACEs providing basic services. From the lifecycle point of view, the basic lifecycle functionalities starting, stopping and interaction of ACEs are supported, whereas migration will be implemented in the future. Therefore, in the current implementation ACEs are stationary components. They will remain in the environment of the initial creation.

The ACE autonomic behaviour is specified by the ACE developer within the Self-Model and carried out by the Facilitator. RuleML is used for specifying Plan creation and modification rules. The Facilitator reads the Self Model and creates the Plan accordingly.

ACE Plans are executed by the Reasoner which translates them into RuleML and performs the specified actions. Actions can either invoke a service from a remote ACE with the GN/GA protocol, or local functionalities from the Functionality Repository.

ACE specific functionalities are loaded and maintained by the Functionality Repository. The Functionality Repository invokes specific functionalities when requested by the Reasoner and delivers the results to the requesting party.

Communication among ACEs is event based. It is provided by the ACE organ called Bus and allows implementing the GN/GA protocol.

All of the ACE organs listed above have been implemented. Some of them are in an advanced stage like for example Reasoner, Facilitator and Functionality Repository, whereas the Bus requires additional development and improvement. Nevertheless, with the current prototype it is able to implement ACEs which provide services of a limited complexity.

The current implementation of the prototype has been tested and evaluated with an example scenario following the Digital City idea. It presents a city with two kinds of ACEs: survivors and rescue teams where all interact with each other after a catastrophic situation. Survivors inside buildings try to share all the information between them and rescue teams to take a priority decision about which building to rescue. A brief video clip (Simulation.swf) of the demonstrator and audio guide (Audio_Simulation.wma) are available attached to this document.

2 Outlook

Further investigation and development will be based on the enhancing of the Reasoner. In particular, the enhancements (by adding a rule engine with a set of rules) will improve the behaviour of the survivors and rescue teams in order to obtain the highest efficiency. Added rules can be created, deleted and modified by the previous rules if necessary so the rule engine is able to adapt itself to the changes in the environment. It can also be seen as a rule engine that has a set of states that an agent must achieve by fulfilling some goals.

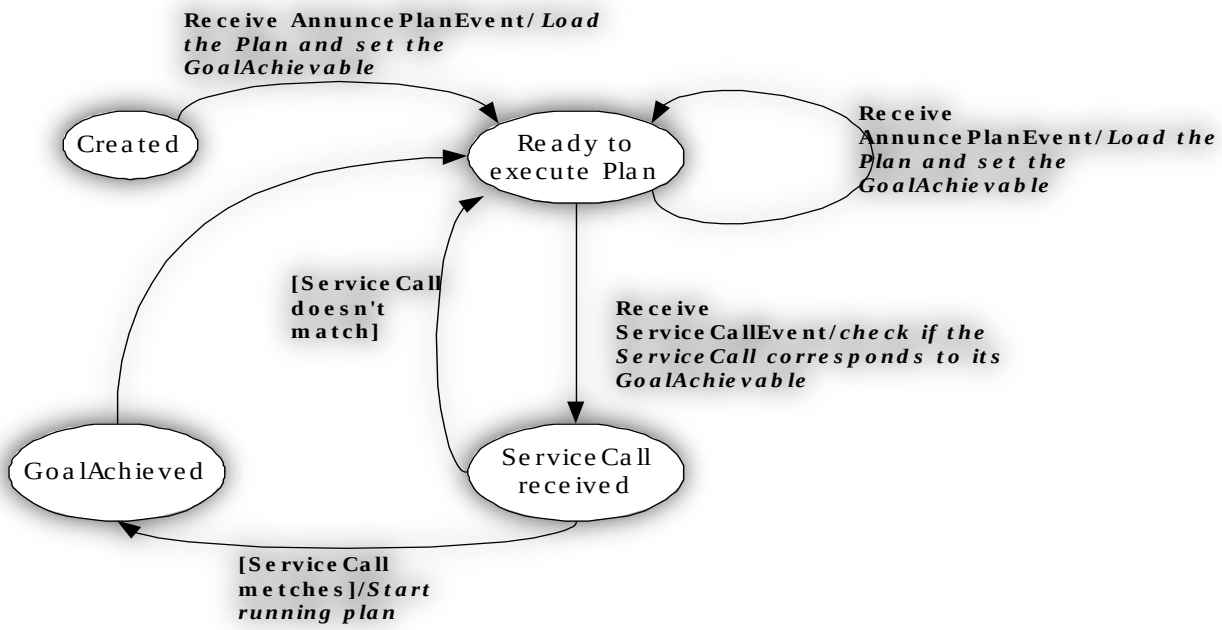


Figure IV.1 Basic process of Reasoner

The state machine tries to understand the different changes in the environment and it gives orders to change some of the internal behaviours. Every survivor and rescue team will have his own state machine implemented in RuleML and attended by the reasoning. RuleML is a shared Rule Markup Language, which allows both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks.

The Figure IV.1 shows the basic state-machine process of the Reasoner.

Above, the simplified state-machine shows the main mechanism used by the agent to fulfil a ServiceCallEvent. The Reasoner for each plan, provided by the Facilitator, extracts a GA, representing the goal the agent achieve after the execution of the plan.

When a Reasoner with an active plan receives a ServiceCallEvent transforms the ServiceCallEvent in a GA and tries to match it with the GA of its active Plan. If the two GAs match, the reasoning on the active Plan starts. In

this way, the ServiceCallEvent will be completely fulfilled when the Reasoner reach the GA.

Within the Reasoner, development has been oriented to modify the plan using a RuleML version and to introduce refactoring in the reasoning. So, Reasoning capabilities allow the agent to take the proper actions in terms of sending GAs (Goal Achievable) and GNs (Goal Needed), invoking specific functions and checking internal conditions, in order to achieve a given Goal.

The Figures IV.2 and IV.3 corresponds to the state machines devised for both survivor and rescue team agents respectively.

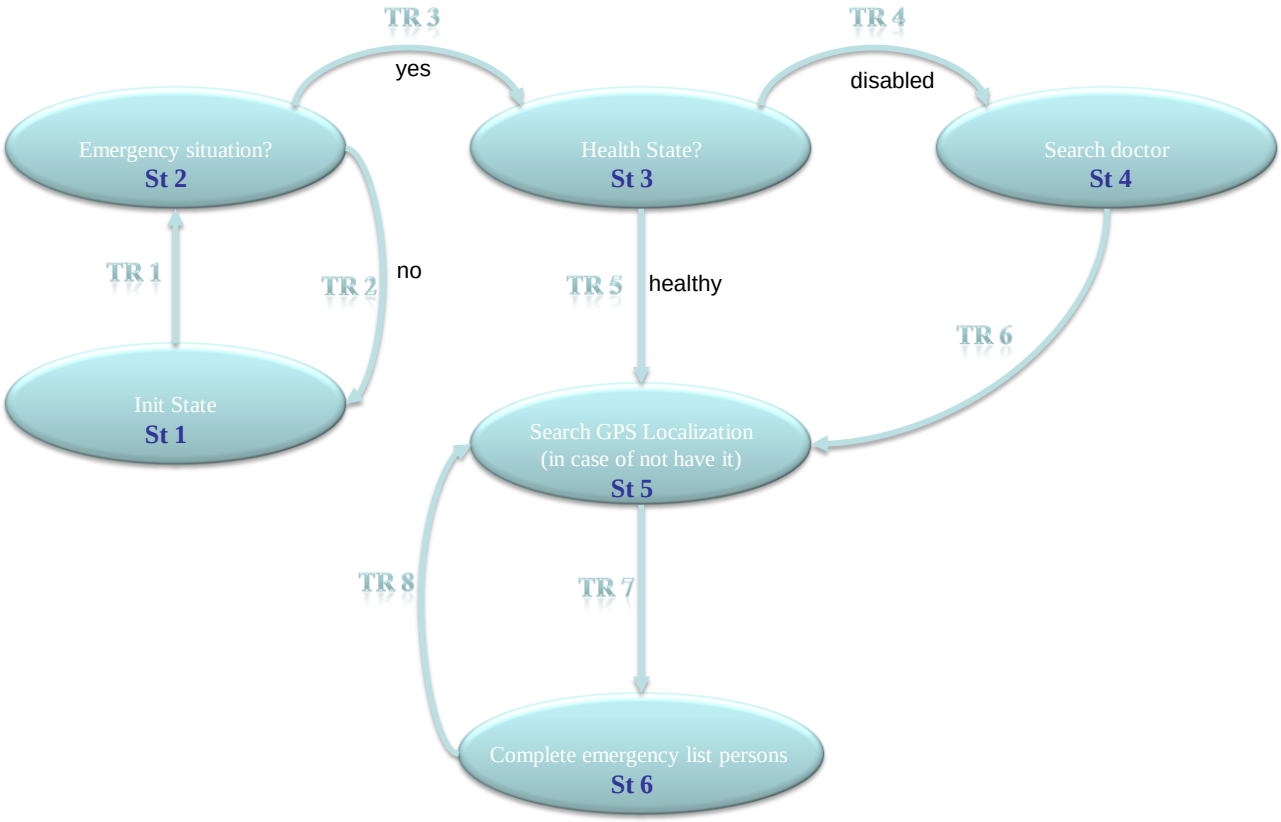


Figure IV.2 Future Survivor's Self-Model

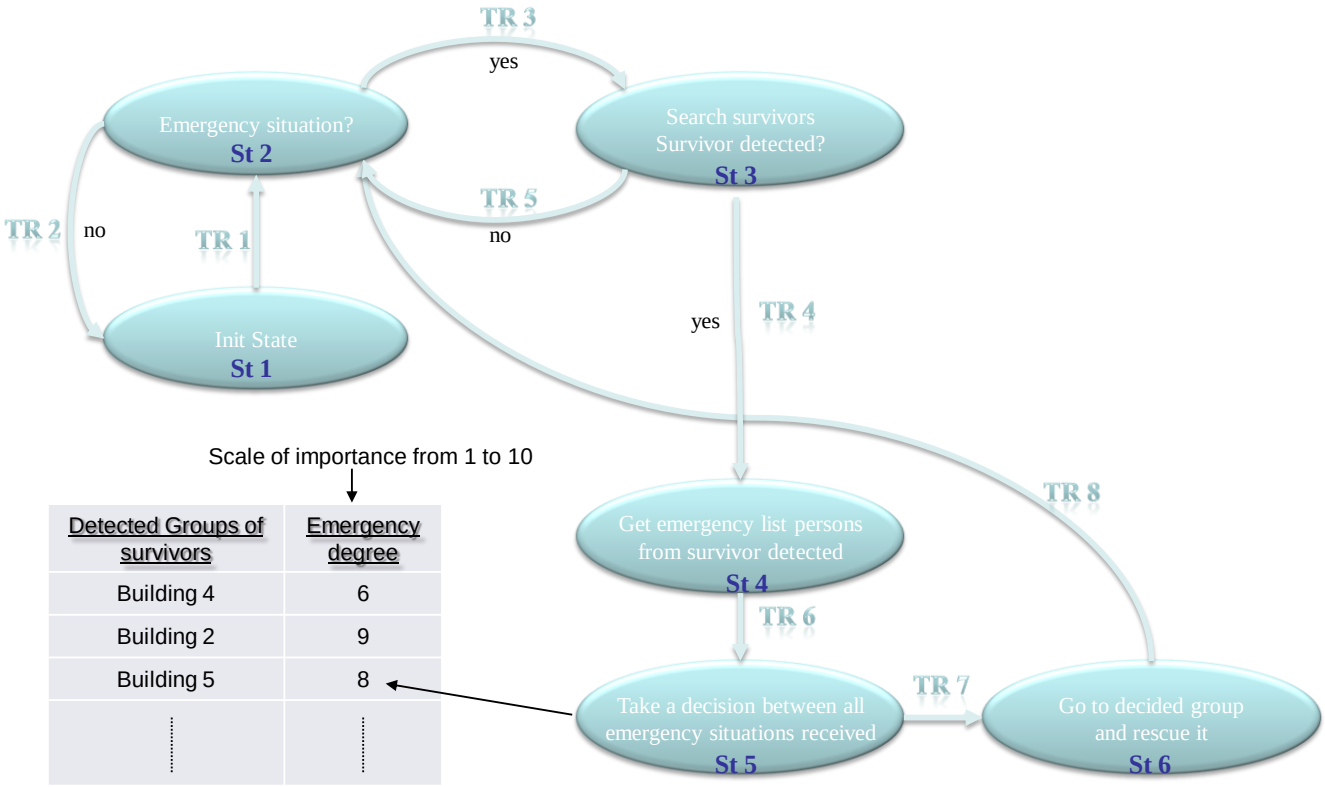


Figure IV.3 Future Rescue Team's Self-Model

CHAPTER V: Annex

1 Paper presented at CISIS 2008

A paper with the results of the thesis has been written and presented in Second International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2008): "Demonstrating Communication Services Based on Autonomic Self-organization" **¡Error! No se encuentra el origen de la referencia.**. The aim of the conference is to deliver a platform of scientific interaction between the three interwoven challenging areas of research and development of future ICT-enabled applications:

a. Software Intensive Systems
b. Complex systems
c. Intelligent Systems

Demonstrating communication services based on autonomic self-organization

A. Bernadas[2], R. Alfano[1], A. Manzalini[1], J. Sole Pareta[2], S. Spadaro[2]

[1]Telecom Italia, e-mail: antonio.manzalini@telecomitalia.it

[2]Universitat Politècnica de Catalunya, e-mail: pareta@ac.upc.edu

***Abstract* —Paper reports main results achieved in the development of a prototype for demonstrating communication services based on the principles autonomic self-organisation. In particular, the prototype has been designed and developed as a distributed complex adaptive systems realized by means of a population of lightweight autonomic components interacting each other through self-organizing algorithms. Prototype demonstration simulates the collaborative behaviour of a rescue team in a critical situation with limited connectivity. In this condition factors such as mobility, data distribution and high probability of disconnection represent strong challenges for current software architecture. Prototype demonstrations show how distributed lightweight components can self-organize in order to face above challenges.**

***Index Terms* — Situated Autonomic Communications, Self-organization algorithm, Multi-Agent, Self-* capabilities,**

I. INTRODUCTION

Definition of autonomic system is taking inspiration from the self-governing behaviors of some natural autonomic systems, e.g. the human autonomic nervous system. Upon launching the Autonomic Computing initiative, IBM defined four general properties a system should have to constitute self-management: self-configuring, self-healing, self-optimizing and self-protecting. Since the launch of Autonomic Computing initiative, the self-* list of properties has grown substantially. Now it includes also features such as self-anticipating, self-adapting, self-critical, self-defining, self-destructing, self-diagnosis, self-governing, self-organized, self-recovery, self-reflecting, etc. Extension of autonomic technology principles from computing to network and services resources still has the meaning of developing solutions capable of hiding operational complexity to both Operators and Users. Autonomic systems are per se capable of making decisions, by using high-level policies from operators, whilst constantly checking and optimizing their status in order to adapt to changing environment conditions.

In Framework Program VI, European Commission has launched the Situated Autonomic Communication initiative (Future Emerging Technologies): vision refers to the self-* characteristics of distributed network and service resources, systems and infrastructures fostering the development of the Information Communication Society.

In this context, the IST Integrated Project Cascadas [http://www.cascadas-project.org/] has the main goal identifying and developing distributed component-ware architecture for development of innovative situation aware and autonomic services. Basic concepts and algorithms adopted for the development of the prototype have been inspired by the activities carried out in the project.

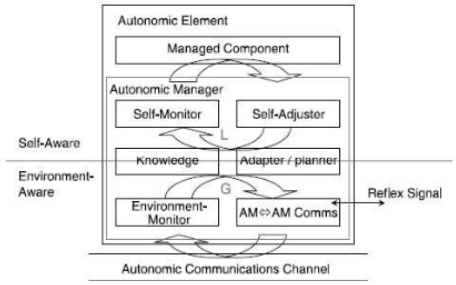


Figure 1 – Example of an Autonomic Element

In IP Cascadas, a distributed autonomic system can be seen as a framework of autonomic components (figure 1) [1], dynamically interacting with each other and self-organizing their activities to serve certain overall goals. Specifically autonomic features of components can be exploited through the introduction of goal-oriented knowledge reasoning capabilities. Concerning the self-organization capabilities, the second key aspect of the prototype, biological algorithms have always been a key source of inspiration [2]. As an example, swarm intelligence principles have been widely used for modeling problems through some simple interactions of a collection of agents cooperating to achieve a common goal. In these systems, problems are "self-solved" in real time through the emergence of the appropriate collective behavior, consequence of interactions occurring between agents and with environment.

Above principles can be ideally applied to model the self-* collective behavior that can be observed also in human social relationships. Cities, for example, can be easily recognized as self-* ecosystems. Even more today, with the wide adoption

of digital devices, communications are generating data clouds overlooking modern cities whose patterns exhibit self-adaptive and self-organizing properties.

Thesis of this paper is demonstrating, through the development of a prototype, the applicability of the key principles of autonomic self-organization for the development of solutions enabling communication services even in critical disconnected situations (e.g. catastrophic event in a city).

A brief video clip (.ogg) of the demonstrator is available at the Cascadas web-site [http://www.cascadas-project.org/].

II. AUTONOMIC SELF- AGGREGATION

This section will provide an introduction on the two basic foundations of autonomic self-organization, as developed into the prototype: i) autonomic agent environments; ii) self-organising algorithms. Attention has been mainly focussed where innovation is needed, i.e. on the applicability of self-organising algorithms in autonomic agent environment. Given the richness of available results, references to literature are provided for further details.

A. Autonomic Agents

Autonomic systems are typically distributed, complex and concurrent systems, comprised of multiple interacting autonomic elements and all the resultant issues have already been faced in different fields of autonomous agents.

Agent-based approaches have been, and remains, a rich area for the study of the emergence of self-organisation. For example, "artificial markets" have been studied for their potential in market-based control. The aspiration is that if the appropriate interaction/trading rules are encoded into a population of agents, then the agents will be able to self-organise into "useful" structures/networks, where "useful" is defined in terms of an application context e.g. Supply chains, or trading markets. Di Marzo et al. [17] review different aspects of self-organisation in Multi-Agent Systems. They show how inspiration derives from natural systems (complex physical systems as well as natural systems). For example, the concept of stigmergy, derived from the behaviour of social insects, has also been important in inspiring the design of Multi-Agent Systems. Bernon et al. [18] review several examples of applications of self-organising multi-agent systems. They show how Multi-Agent Systems can self-organise to carry out tasks, even though individual agents have very simple properties. The emergent properties of the self-organising system support each application.

B. Self-organising algorithms

This section describes two examples of self-organizing algorithms [19]: passive clustering [19] and on-demand clustering [19]. In the context of this paper the term "aggregation" refers to the process by which nodes form associations ("links") with each other. It is a "clustering" process during which each node, characterised by a certain "type" establish links with nodes of the same type. From this point of view the efficiency of a self-aggregation algorithm can be read in terms of convergence capabilities of increasing the fraction of links connecting nodes of the same type.

1) "Passive" clustering

A first set of basic local rules has been devised requiring only direct interaction between first neighbours yet susceptible to give rise over time to spontaneous system-wide aggregation of elements. Basic idea involves two nodes being notified by a third (the "match-maker") that they are interconnected through an overlay network, even if those two nodes have no direct part in the decision process, ("passive" clustering). Rules are as follows:

- match-maker node is randomly selected. This is equivalent to say that every node in the system has a chance of "waking-up" and initiating a rewiring procedure, provided that this procedure is brief enough (and/or infrequent enough) that a situation in which two concurrent rewiring affect the same nodes is extremely unlikely, and so every attempt can be considered an independent event.
- match-maker randomly selects two of its own neighbours and, if they happen to belong to the same type, instructs them to link together
- if the two chosen nodes were not already directly connected (through the overlay) a new link is established between them (i.e. they become first neighbour of each other).
- if conservation of the total number of links is in force (optional) and a new connection is successfully established, the match-maker terminates one of its own links with one of its two selected neighbours.

2) "On-demand" clustering

In passive clustering technique in order to preserve homogeneous node degree in the realistic, local rules-based scenario, the rewiring procedure has to be modified: there is the need of eliminating the indirect positive feedback leading to the emergence of scale-free topology. It may be objected that heterogeneous node degree can be highly beneficial to network operation if the higher connectivity of some vertices can be made to reflect their superior capability. However, in our case, such correlation is effectively absent, the emergence of hubs in the "passive rewiring scenario" resulting from the amplification of random fluctuations. As it cannot be guaranteed that those nodes ending up with a higher degree effectively have some specific features that designate them as efficient "super-peers", the result could be disastrous and generate critical bottlenecks, which is why we aimed at maintaining node degree as homogeneous as possible throughout the system's history.

This has been achieved by distinguishing between the initiator of a rewiring procedure and the match-maker. Basically, upon "waking-up", the initiator requests a new link from one of its existing neighbours, which will then act as the match-maker. Since with this logic, the probability for a node to be appointed match-maker is obviously a direct function of

its own degree (and the match-maker still ends losing one neighbour in the process of a successful rewiring operation), it introduces a negative, "rich becomes poorer" feedback similar to the one observed in the abstract model.

The detailed algorithm governing key node behaviour in the three roles of "initiator", "match-maker" and "candidate" involved in a rewiring operation following the "on-demand" clustering procedure is shown in figure 3. It involves exchanging five types of messages (plus the link termination message which isn't discussed here). The "neighbour request" (NRQ) message is sent by the initiator to the chosen match-maker and specifies the type of node desired. The "neighbour reply" (NRP) message is sent by the match-maker to the initiator to inform it of a potential candidate. The "link" (LNK) message is sent by the initiator to the candidate to ask for the establishment of a new link, which will only be effectively created if it is compatible with the goals of the candidate, as evidenced by the receipt of an "acknowledgement" (ACK) message by the initiator. Note that, for most of the results presented in this section, this will always be the case, as all nodes in the system share the same objective, i.e. they are all assumed to be simultaneously in clustering (or reverse-clustering) mode. Finally, after a successful handshake between the initiator and the candidate, the match-maker is informed via the "success" (SCC) message, so as to be able to determine whether its own connection to the candidate has to be terminated in order to conserve the total number of links.

III. Architecture of the Prototype

A prototype has been designed and developed in order to demonstrate the applicability of the main principles of autonomic self-organization in terms of interactions of a population of autonomic components through self-organizing algorithms.

The scenario adopted for demonstrating the prototype is a dynamic and data intensive digital environment (e.g. a city with pervasive digital devices), where everyone can exchange information and collaborate with each others. A scenario like this requires technologies and solutions to manage large amounts of highly distributed data items that need to be transformed into meaningful, reliable and available information for each mobile user.

A. The Use-case

The selected use-case is a city where some catastrophic event impacted the communication infrastructure causing faults limiting normal wire and wireless connectivity. Use case is ideal for demonstrating how autonomic self-organisation meets requirements for emergency communications between the survivors and rescue teams, having the goals of providing first aid as soon as possible.

Two basic groups are involved in the simulations: the rescue teams and the survivors. The interaction between groups of survivors and between survivors and rescue teams should test the efficiency of system self-management. Rescue teams and survivors are placed in two differentiated interaction environments.

Inside each building the communication is managed in a completely distributed way without any central control. This is likely a communication network between peers based on an ad-hoc. Final goal is to achive the major spread degree of available information in the environment, in order to make each survivor to provide and to gather information from its neighbours.

Moreover, the rescue teams will interact with the different environments where the survivors are situated. Therefore a rescue team will be able to communicate with a group of survivors when it is in the covering area of someone of them.

The basic rules that the agents must fulfil are:

- communicate each other in the same environment.
- migrate to other environment (building).

The picture below provides a representation of the main entities involved:

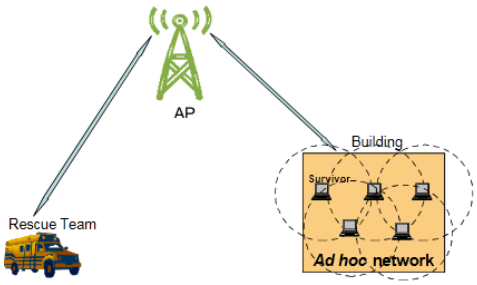


Fig. 1. Main entities involved.

The picture below also shows a snapshot of the demo application developed to test the proposed technique:

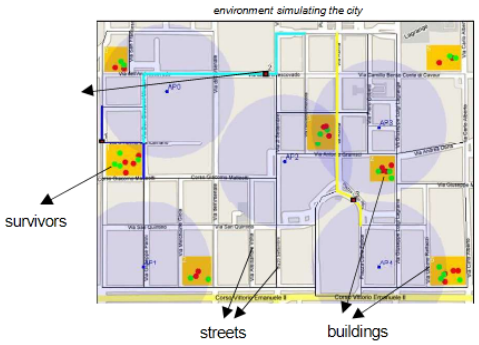


Fig. 2. Demo snapshot

B. Main architecture

The architecture of each agent is structured in following functional blocks:

- *Reasoning Part:* has the task of managing the lifecycle of an agent. It describes the possible states and invokes the proper specific features if specified,

running all as a state machine. It works within the system as a DIET's job in a parallel execution way.

- *Communication part:* in charge of implement communication among agents. It includes the three behaviours described before in the "On-demand" clustering algorithm ("initiator", "match-maker" and "candidate"). It includes blocks in charge of the information exchange contemplated in reasoner's logic (messages including survivor's list, GPS position…). Jobs are the way of distributing the functionalities of an agent. Therefore agents can compose their behaviour by combining multiple jobs. To implement the behaviour that responds to the communication protocol activate-clustering a structure of different jobs has been created. The structure is the following one:

1. *NotifyNeighborsJob:* Job, which on start-up notifies in broadcast of its type ID to neighbors creating connections. Handle also notifications of neighbors.
2. *RandomNeighbourRequestJob:* Job that initiates the request process for type items. (Initiator behaviour)
3. *HandleNeighbourRequestJob:* Handles requests returning a random address of a candidate with the type requested. (Match-Maker behaviour)
4. *HandleNeighbourReplyJob:* Handles notifications of type requested starting the link process to candidate. (Initiator behaviour)
5. *HandleLinkJob:* Handles requests for link from the initiator of link process. (Candidate behaviour)
6. *HandleAckJob:* Handles "ack" responses from candidate finishing the link process. (Initiator behaviour)
7. *HandleSuccessJob:* Handles notifications of successful links to candidate from a initiator agent destroying the connection to candidate. (Match-Maker behaviour)

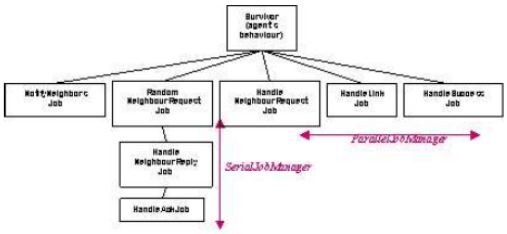


Fig. 3. Jobs' Structure.

The *SerialJobManager* is used to execute several jobs in sequence. After the first job has finished, it will start the second job, and so on. This is useful when an agent's behaviour can be split into various stages. This sequence system is used to implement all the steps of the initiator agent behaviour ("wake-up", link creation, send success…).

The *ParallelJobManager* is used to run multiple jobs concurrently. For instance, an agent may use a scheduler job to manage its schedule events, and another job that implements the behaviour specific to the agent which requires scheduling functionality. Therefore, system designed in the simulation is served by this scheduler job for manage the tree jobs because require combinations of sequence and parallel execution.

A correct implementation of these jobs provides an important feedback to us on the applied behaviour of the algorithm.

- *Specific Part:* it executes normal code depending on Reasoning Part's decisions and returns the results so they can be checked and sent back. For instance, there are specific functions for the survivors' list managing that confronts the information contained in them as well as the managing of information concerning to the GPS positioning.

C. Technological approach

This section provides a description of technology and algorithms used for the prototype development. In particular the prototype has been developed using the DIET (Decentralised Information Ecosystem Technologies) multi-agents framework and implementing the active-clustering self-organization protocol.

1) DIET

DIET

DIET Agents (Decentralised Information Ecosystem Technologies) [http://diet-agents.sourceforge.net] is a platform for developing agent-based applications. DIET platform, developed in the EU-funded project DIET project, is an Open-source framework released under GPL license and downloadable from sourceforge web site 7.

DIET goal is providing an ecosystem-inspired approach to the design of agent applications 7. In this context an ecosystem can be viewed as an entity composed of one or more communities of lightweight components conducting frequent, flexible local interactions with each other and with the environment that they inhabit.

Although the capability of each lightweight component itself may be very simple, the collective behaviours and the overall functionality arising from their interactions exceed the capacities of any individual organism. These higher-level processes can be adaptive, scalable and robust to changes in their environment.

2) Self-organization protocol: active-clustering

In order to achieve a good level of self-organization a bio-inspired self-organizing algorithm has been applied.

A distinction has been made among the initiator of a

rewiring procedure and the match-maker. Basically, upon "waking-up", the initiator requests a new link from one of its existing neighbours, which will then act as the match-maker. Since with this logic, the probability for a node to be appointed match-maker is obviously a direct function of its own degree (and the match-maker still ends losing one neighbour in the process of a successful rewiring operation), it introduces a negative, "rich becomes poorer" feedback similar to the one observed in the abstract model.

The detailed algorithm governing key node behaviour in the three roles of "initiator", "match-maker" and "candidate" involved in a rewiring operation following the "on-demand" clustering procedure is shown in figure below. It involves exchanging five types of messages (plus the link termination message which isn't discussed here). The "neighbour request" (NRQ) message is sent by the initiator to the chosen match-maker and specifies the type of node desired. The "neighbour reply" (NRP) message is sent by the match-maker to the initiator to inform it of a potential candidate. The "link" (LNK) message is sent by the initiator to the candidate to ask for the establishment of a new link, which will only be effectively created if it is compatible with the goals of the candidate, as evidenced by the receipt of an "acknowledgement" (ACK) message by the initiator. Finally, after a successful handshake between the initiator and the candidate, the match-maker is informed via the "success" (SCC) message, so as to be able to determine whether its own connection to the candidate has to be terminated in order to conserve the total number of links.

In order to implement the algorithm above three behaviours have been defined: Initiator, match-maker, candidate. The following picture shows the interaction among the three behaviours:

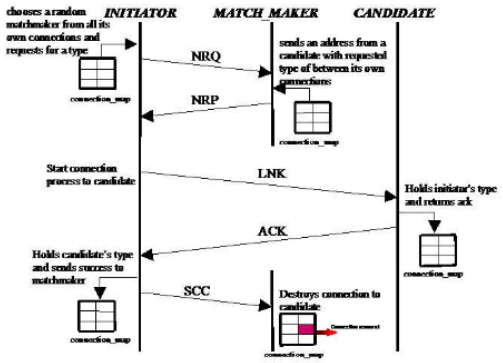


Fig. 4. DIET behaviours interactions

IV. Future Work

Further investigation and development will be based on the enhancing the Reasoner (with RuleML). In particular, the enhancements (by adding a rule engine with a set of rules) will improve the behaviour of the survivors and rescue teams in order to obtain the highest efficiency. Added rules can create, delete and modify previous rules if necessary so the rule engine is able to adapt to the changes in the environment. It can also be seen as that the rule engine to be a set of states that an agent must achieve by fulfilling some goals.

The state machine tries to understand the different changes in the environment and it gives orders to change some of the internal behaviours. Every survivor and rescue team will have his own state machine implemented in RuleML and attended by the reasoning. RuleML is a shared Rule Markup Language, permitting both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks.

Following state-machine shows the basic process of the reasoner:

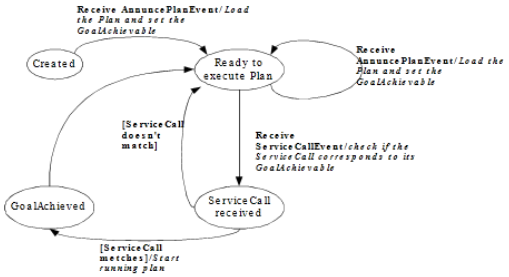


Fig. 5. Basic process of Reasoner.

Above simplified state-machine shows the main mechanism used by the agent to fulfil a ServiceCallEvent. The Reasoner for each plan provided by the Facilitator extracts a GA, representing the goal the agent achieve after the execution of the plan.

When a Reasoner with an active plan receives a ServiceCallEvent, it transforms the ServiceCallEvent in a GA and tries to match it with the GA of its active Plan. If the two GAs match, the reasoning on the active Plan starts. In this way, the ServiceCallEvent will be completely fulfilled when the Reasoner reach the GA.

Within of the Reasoner, development has been oriented to modificate the plan using a RuleML version and introduces refactoring in the reasoning. So, Reasoning capabilities allows the agent to take the proper actions in terms of sending GAs, GNs, invoking specific functions and checking internal conditions in order to achieve a given Goal.

The following figures correspond to the state machines devised for the survivor agent and the rescue team.

Fig. 6. Survivor's Self-Model.

Rescue Team's Self-Model

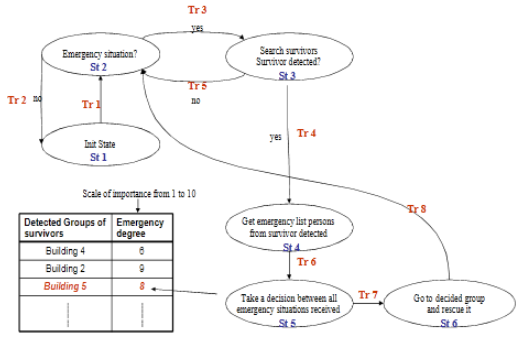


Fig. 7. Rescue Team's Self-Model.

V. Conclusions

Paper reports main results achieved in the development of a prototype for demonstrating communication services based on the principles autonomic self-organisation. In particular, the selected use-case focus on the applicability of above principles for the development communication services even in critical disconnected situations (e.g. catastrophic event in a city).

Achieved results have demonstrated that the solution, designed by means of self-organising algorithms deployed in frameworks of distributed autonomic agent, is meeting some challenging requirements such as: adaptability to dynamic situations, robustness even in environments with high churn rate and/or disconnected situations.

A brief video clip (.ogg) of the demonstrator is available at the Cascadas web-site [http://www.cascadas-project.org/].

Further investigation and development will go in the main direction of enhancing reasoning capabilities of the agents, e.g. by adopting a RuleML approach.

Acronyms

ACE	Autonomic Communication Element
ACF	Autonomic Communication Forum
GA	Goal Achievable
GN	Goal Needed
AutoComm	Autonomic Communication
HSDPA	High-Speed Downlink Packet Access
GPS	Global Positioning System
GSM	Global System for Mobile Communication
HSDPA	High-Speed Downlink Packet Access
HSUPA	High-Speed Uplink Packet Access
PDA	Personal Digital Assistant
P2P	Peer to Peer
QoS	Quality of Service
RFID	Radio Frequency Identification
TMF	Telemanagement Forum
UMTS	Universal Mobile Telecommunications Services
UWB	Ultra-Wideband
Wi-Max	Worldwide Interoperability for Microwave Access

Acknowledgements

Authors would like to acknowledge the European Commission for funding the Integrated Project CASCADAS (Note 1) "Component-ware for Autonomic, Situation-aware Communications, And Dynamically Adaptable Services" (FET Proactive Initiative, IST-2004-2.3.4 Situated and Autonomic Communications).

References

[1] R. Sterritta, M. Parasharb, H. Tianfieldc, R. Unland, A concise introduction to autonomic computing, Advanced Engineering Informatics 19 (Elsevier, 2005), 181–187.
[2] A. Manzalini, P. Marrow, "The CASCADAS Project: A Vision of Autonomic Self-organizing Component-ware for ICT Services" SOAS 2006 (Erfurt, September 2006)
[3] A Manzalini, F Zambonelli, Towards Autonomic and Situation-Aware Communication Services: the CASCADAS Vision, IEEE Workshop on Distributed Intelligent Systems (Prague, June 2006)
[4] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence, Oxford University Press, 1999.
[5] P. Bouquet, L. Serafini, S. Zanobini, "Peer-to-Peer Semantic Coordination", Journal of Web Semantics, 2(1), 2005.
[6] L. Capra, W. Emmerich, C. Mascolo, "CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications", IEEE Trans. on Software Engineering Journal, 29(10), 2003.
[7] T. Choudhury, A. Pentland, "Modeling Face-to-Face Communication Using the Sociometer", Conference on Ubiquitous Computing, Seattle (WA), 2003.
[8] S. Dill, et al., "Self-Similarity in the web", ACM Trans. on Internet Technology, 2(3), 2003.
[9] D. Estrin, D. Culler, K. Pister, G. Sukjatme, "Connecting the Physical World with Pervasive Networks", IEEE Pervasive Computing, 1(1), Jan. 2002.
[10] J. Kephart, D. Chess, "The Vision of Autonomic Computing", IEEE Computer, 36(1), 2003.
[11] M. Mikic-Rakic, N. Medvidovic, "Support for Disconnected Operation via Architectural Self-Reconfiguration", 1st International Conference on Autonomic Computing, 2004.
[12] M. Philipose, et al., "Inferring Activities from Interactions with Objects", IEEE Pervasive Computing, 3(4), 2004.
[13] S. Ratsanamy, et al., "A Scalable Content-Addressable Network", SIGCOMM Conference 2001, Aug. 2001.
[14] N. Ravi, et al., "Accessing Ubiquitous Services using Smart Phones", 3rd International Conference on Pervasive Computing and Communications, Kauai Island (NW), March 2005.
[15] L. Tummolini, et al., "A Shared Environment for Flexible Coordination with Tacit Messages", Environments for Multiagent Systems, Springer, 2005.
[16] F. Zambonelli, N. Jennings, M. Wooldridge, "Developing Multiagent Systems: the Gaia Methodology", ACM Trans. on Software Engineering and Methodology, 12(3), 2003.
[17] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. LectureNotesinComputerScience, 1563:404–413, 1999.
[18] B-G. Chun, R. Fonseca, I. Stoica, and J. Kubiatowicz. Characterizing selfishly constructed overlay routing networks. In Proc.IEEEINFOCOM, 2004
[19] Aggregation Algorithms, Overlay Dynamics and Implications for Self-Organised Distributed Systems. IST CASCADAS Work Package 3 Deliverable Month 12
[20] Filip Perich, Cougaar Software, Inc., Anupam Joshi, University of Maryland, Baltimore County, Rada Chirkova, North Carolina State University Data Management for Mobile Ad-Hoc Networks
[21] DIET - http://diet-agents.sourceforge.net/ProjectBackground.html

Albano Bernadas is a student of the Electrical Engineering School of the UPC (Universitat Politècnica de Catalunya). He was awarded with a grant for

doing a stay at Telecom Italia (TI) within the frame of a Collaboration Agreement between UPC and TI for training UPC students.
Taking advantage of this grant, he is currently doing his Master Thesis there working on the "design and demonstration of autonomic systems based on self-aggregation components". His supervisor at TI is Dr. Antonio Manzalini.

Antonio Manzalini received a Dr. Ing. degree in electronic engineering from the Politecnico of Turin (Italy). In 1990 he joined Telecom Italia Lab (formerly CSELT). He started his activities in the area of transport networks moving to address architectural and functional issues for advanced networking such as IP/GMPLS. He has been awarded 5 patents on networking and services systems and methods. He is author of a book on transport network synchronization and his work has been published in several technical papers and publications in the field of advanced networking. He was active in the ITU standardization for transport networks: from 1997 to 2000, he was appointed Chairman of ITU SG13 Question 19 "Transport network architecture and interworking principles"; he was also Chairman of ITU SG15 Question 12 "Technology Specific Transport Network Architectures". He was also involved in several EURESCOM and European Project (ACTS and IST). From 2000 to 2002 he run as Project Manager the Vth FP IST Project LION "Layers Interworking in Optical Networks". He was also Project Manager of the 6th FP IST Integrated Project NOBEL. In 2003 he was appointed as member of the Scientific Committee of CTTC (Centre Tecnològic de Telecomunicacions de Catalunya). On January 2006 he has been appointed as Project Leader of the FET IST IP CASCADAS "Component-ware for Autonomic, Situation-aware Communications And Dynamically Adaptable Services" whose main goal is identifying, developing and demonstrating an architectural vision based on self-organised distributed components for the provisioning of autonomic and situation-aware communication services.

Dr. Rosario Alfano received a magna cum laude M.Sc degree in Computer Science from Università di Torino. In 2002 he joined Telecom Italia Lab (formerly CSELT). Since the beginning he worked in different projects aiming at designing and developing highly distributed architectures. He also took part in the development of Telecom Italia Voip Platform. On January 2006 he has joined the FET IST IP CASCADAS, whose main goal is identifying, developing and demonstrating an architectural vision based on self-organized distributed components for the provisioning of autonomic and situation-aware communication services.

Prof. Josep Solé-Pareta obtained his M.Sc. degree in Telecom Engineering in 1984, and his Ph.D. in Computer Science in 1991, both from the UPC. In 1984 he joined the Computer Architecture Department of UPC. Currently he is Full Professor with this department. He did a Postdoc stage (1993 and 1994) at the Georgia Institute of Technology. He is co-founder of the UPC-CCABA. His publications include several book chapters and more than 100 papers in relevant research journals (> 20), and refereed international conferences. His current research interests are in Autonomic Communications, Traffic Monitoring and Analysis and High Speed and Optical Networking, with emphasis on traffic engineering, traffic characterisation, MAC protocols and QoS provisioning. He has participated in many European projects devoted to Networking topics.

Dr. Salvatore Spadaro received the M.Sc. (2000) and the Ph.D. (2005) degrees in Telecommunications Engineering from UPC. He also received the Dr. Ing. degree in Electrical Engineering from Politecnico di Torino, (2000). He is currently Assistant Professor in the Optical Communications group of the Signal Theory and Communications Dept. of UPC. Since 2000 he is a staff member of the UPC-CCABA and he is currently participating in NOBEL-2 and e-Photon/One phase 2 projects. He has co-authored about 50 papers in international journals and conferences. His research interests are in the fields of autonomic communications and all-optical networks with emphasis on traffic engineering and resilience in GMPLS networks.

2 New research

Since 2008, the ending year of this thesis, when results and conclusions were presented in CISIS conference, new researches and workshops has been realized in order to continue the progress in the applicability of autonomic and self-organization capabilities.

The main challenge addressed is the growing of complex and dynamic scenarios of network evolution, while the same time, reduce costs and increase revenue. In order to face this challenge, autonomic network architecture allows dynamic adaptation and re-organization of the network according to the working, economic and social needs of the users. This is expected to be especially challenging in a mobile context where new resources become available dynamically, administrative domains change frequently, and the economic models may vary.

As explained above, the self-organization autonomic systems have some common features:

- High-dynamicity
- Situation awareness
- Open “ecosystems”
- High-scalability
- Reduced infrastructure and operation costs

Some of the projects and events involved into autonomic self-organization research are the following:

a. Autonomic Network Architecture (ANA) - February 2009 [16]

i. About the project

The ANA is a project funded by the European Union Information Society Technologies Framework Programme 6.

The project aims at exploring novel ways of organizing and using networks beyond legacy Internet technology. The ultimate goal is to design and develop a novel autonomic network architecture that enables flexible, dynamic, and fully autonomous formation of network nodes as well as whole networks. Universities and research institutes from Europe and Northern America participated in this project.

The scientific objective of this proposal is to identify fundamental autonomic network principles. Moreover, the ANA project build, demonstrate, and test such

autonomic network architecture. The key attribute is that such a network scales in a functional way that is, the network can extend both horizontally (more functionality) as well as vertically (different ways of integrating abundant functionality).

The challenge addressed in this project is to come up with network architecture and to fill it with the functionality needed to demonstrate the feasibility of autonomic.

This integrated project aims at exploring novel ways of organizing and using networks beyond legacy Internet technology. The ultimate goal is to design and develop a novel network architecture that enables flexible, dynamic, and fully autonomic formation of network nodes as well as whole networks. This is expected to be especially challenging in a mobile context where new resources become available dynamically, administrative domains change frequently, and the economic models may vary.

ii. Motivation

The success of the existing Internet architecture is a testimony of the wise design decisions of the early days of the Internet. Indeed, the closely specified protocol suite and the simple basic mechanisms paved the way of this success. However, for today's and future challenges, this architecture may not suffice with the continuous growth of the number of networking devices and their increased diversity in functionality, the role and the properties of the networking elements become challenged.

iii. Objectives

The project has two complementary objectives that iteratively provide feedback to each other: a scientific objective and a technological one.

To identify fundamental autonomic networking principles that enable networks to scale not only in size but also in functionality. The main premise of the project work is that a functionally scaling network is a synonym for an evolving network which includes the various self-x attributes essential to autonomic communication such as self-management, self-optimization, self-monitoring, self-repair, and self-protection. The hypothesis is that, due to these self-x attributes, such functional scaling naturally lead to networks that are not only richer in functionality but which also scale in size. Scientific research in ANA explores the "Internet de-construction" trends of functional atomization, diffusion and sedimentation that replace the current static layering approach.

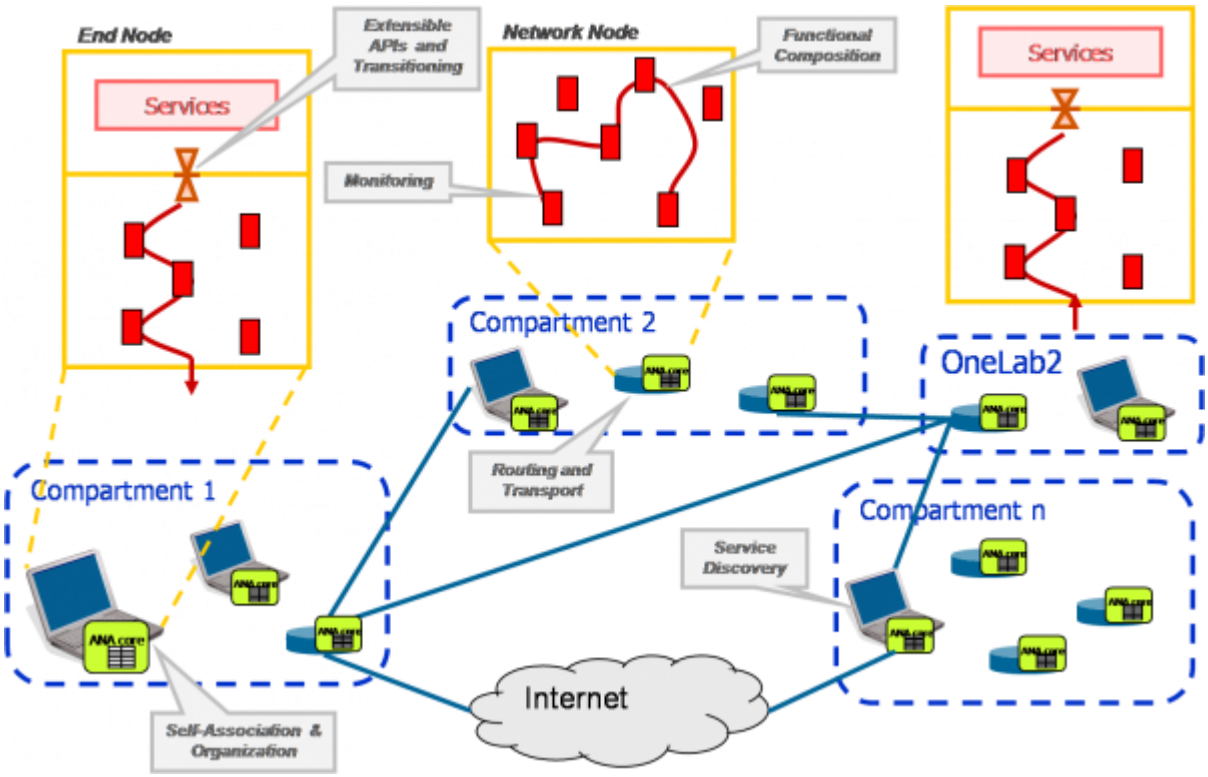


Figure 2.a.1 Autonomic network scenario

New autonomic network architecture emerges as a result of this research. This architecture provides the framework for network function re-composition. The goal is to produce an architectural design that enables flexible, dynamic and fully autonomic formation of large-scale networks in which the functionalities of each constituent network node are also composed in an autonomic fashion. Moreover, it must support mobile nodes and multiple administrative domains.

The second premise is that the only way to make new ideas and concepts succeed is to put them into practice. Therefore, ANA takes on the challenge of not only producing original scientific research results and a novel architectural design, but also showing that they work in real situations, and using the experience gained experimentally as feedback to refine the architectural models and other research results.

The technological objective is therefore to build experimental autonomic network architecture, and to demonstrate the feasibility of autonomic networking.

As a first step, a network based on the predominant infrastructure of Ethernet switches and wireless access points are built. The goal is to demonstrate self-organization of individual nodes into a network. The design of such network should potentially scale to large network meshes in the range of 105 active (routing) elements. Obviously, the consortium alone has not resources to literally build a network of 105 nodes. In order to show scalability, three approaches are envisaged:

a) Overlay for interconnecting the participating sites.
b) Simulations.
c) A distributed open collaborative approach similar to successful initiatives such as “SETI@Home”, “Folding@Home”, to include external experimentations and to disseminate ANA results.

The second step, using insights from the first effort, has lost the constraints and permit wired and multi-hop wireless heterogeneous devices to be integrated in an autonomic way. Here the focus is on the self-organization of networks into a global network. The rationale for a two phase approach is that an architecture can only be developed and its quality be validated if more than one case is explored.

These two (scientific and technical) objectives complement and reinforce each other in a tight feedback loop: Prototypes of research results are implemented in the test bed at an early stage, such that preliminary experimental results can be used as a feedback to steer and refine the architectural design and to obtain more accurate and realistic research results. The research part shapes the test bed in order to maintain it at the fore-front of technology. To help the long term visions to materialize, ANA uses the test bed as an investigative research vehicle while remaining committed to the far looking character of the situated and autonomic networking initiative.

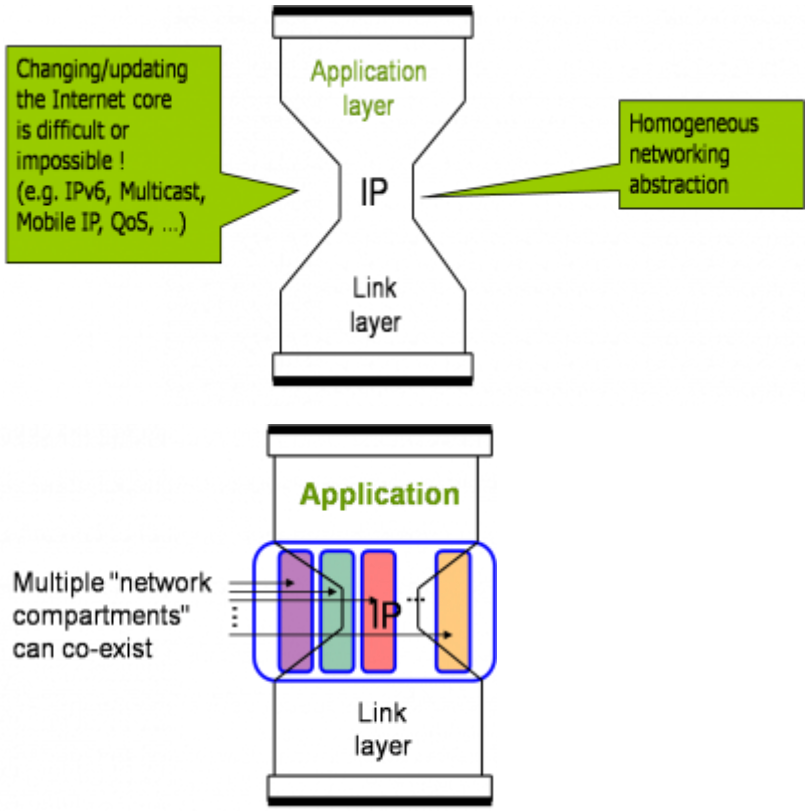


Figure 2.a.2 Architectural design

iv. Institutions involved into

ETH Zurich Communication Systems Group Autonomic Networking, University of Basel, NEC Europe Ltd. Network Laboratories (NEC), University of Lancaster (ULanc), Fraunhofer Gesellschaft, zur Förderung der angewandten Forschung (FOKUS), Université de Liége (ULg), Université Paris VI, Pierre et Marie Curie (UPMC), National and Kapodistrian University of Athens (NKUA), Universitetet I Oslo (UiO) and Telekom Austria and University of Waterloo.

b. Biologically inspired network and services (BIONETS) – March 2009 [17]

i. About the project

BIONETS project is a novel bio-inspired approach to the design of localized services in pervasive communication/computing environments. Conventional networking approaches are not suitable for such scenarios, where they face three main issues, namely:

1. Heterogeneity
2. Scalability
3. Complexity

The proposed solution draws inspiration from the living world in terms of evolutionary paradigms able to drive the adaptation process of autonomic services and social paradigms for the provisioning of the necessary cooperation mechanisms. The net result is the introduction of autonomic self-evolving services that are able to adapt to localized needs and conditions while ensuring the maintenance of a purposeful system. Such a system requires scalable support from the communication standpoint. In networking terms, this results in the introduction of a two-tier architecture based on localized opportunistic exchanges of information. The presented approach is able to achieve better scalability properties when compared to more conventional communication approaches.

ii. Motivation

The motivation for BIONETS comes from emerging trends towards pervasive computing and communication environments, where myriads of networked devices with very different features enhance our five senses, our communication and tool manipulation capabilities. The complexity of such environments does not be far from that of biological organisms, ecosystems, and socio-economic communities. Traditional communication approaches are ineffective in this context, since they fail to address several new features: a huge number of nodes including low-cost sensing/identifying devices, a wide

heterogeneity in node capabilities, high node mobility, the management complexity, and the possibility of exploiting spare node resources. BIONETS aims at a novel approach able to address these challenges. Nature and society exhibit many instances of systems in which large populations are able to reach efficient equilibrium states and to develop effective collaboration and survival strategies, able to work in the absence of central control and to exploit local interactions. The project seek inspiration from these systems to provide a fully integrated network and service environment that scales to large amounts of heterogeneous devices, and that is able to adapt and evolve in an autonomic way.

BIONETS overcomes device heterogeneity and achieves scalability via an autonomic and localized peer-to-peer communication paradigm. Services in BIONETS are also autonomic, and evolve to adapt to the surrounding environment, like living organisms evolve by natural selection. Biologically-inspired concepts permeate the network and its services, blending them together, so that the network moulds itself to the services it runs, and services, in turn, become a mirror image of the social networks of users they serve. This new paradigm breaks the barrier between service providers and users, and sets up the opportunity for "mushrooming" of spontaneous services, therefore paving the way to a service-centric ICT revolution.

iii. Objectives

A new concept of information exchange has been presented in pervasive networks. The major conceptual shift is the use of networks of occasional information exchanges between mobile users, helping to spread information rather than forwarding data packets. The use of genetic models leads to a population of service instances on a set of user nodes. This population can grow if the service is successful or decline if it is not. In addition to the growth of populations BIONETS allow for the evolution of the service itself through mutations and selection of the fittest. The specific fitness criteria are still to be elaborated.

In order to examine the behaviour of the proposed model in a day-to-day scenario the project considers the case of a parking lot application. In such scenario the city is split into blocks and each gene contains the information on the status of a parking spot (free or occupied). The service guides the users towards the nearest free parking place and evolves according to the environment where the users are moving. Since first simulations indicated the great potential of this idea, BIONETS expands the model towards more realistic user behaviour.

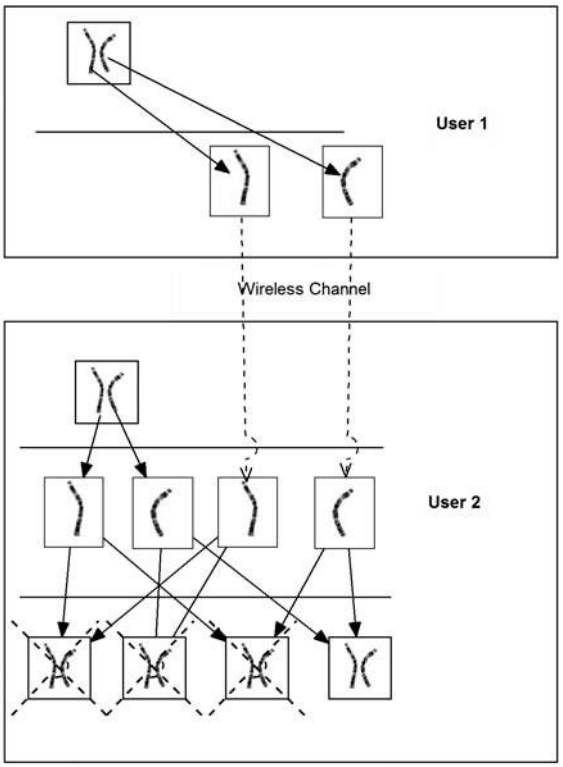


Figure 2.b.1 Gene network behaviour

iv. Institutions involved into

Consiglio Nazionale delle Ricerche - Pisa, University of Trento, Technion, University of Basel, Technische Universitaet Berlin, Institute of IT-Security and Security Law at the University of Passau, Budapest University of Thechnologie and Economics, Nokia Corporation, Valtion Teknillinen Tutkimuskeskus, Institut National de Recherche en Informatique et Automatique, National and Kapodistrian University of Athens, Telecom Italia, London School of Economics and Political Science, and Sun Microsystems Iberica SA.

c. EFIPSANS workshop – November 2010 [18]

i. About the workshop

On November 15th 2010, EFIPSANS has been organized a workshop on the Benefits and Deployment of Autonomics & Self-Management Technologies based on IPv6, and Methodologies Network Operators and Service Providers.

The event aimed to bring together network technology seniors, developers and experts, coming from vendors, operators and ISPs for exchanging knowledge on the possible ways and compatible means to deploy and utilize advanced IPv6-based autonomic mechanisms and Self-Management

methodologies coming from EFIPSANS project in existing networking and service delivery systems. More information can be found on the workshop website.

The EFIPSANS project exposes the features in IPv6 protocols that can be exploited or extended for the purposes of designing or building autonomic networks and services. What this means is, a study of the emerging research areas that target desirable user behaviours, terminal behaviours, service mobility, e-mobility, context-aware communications, self-ware, autonomic communication /computing/ networking will be carried out, and out of these areas desirable autonomic(self-*) behaviours in diverse environments (e.g. end systems, access networks, wireless versus fixed network environments will be captured and specified).

ii. Objectives

The workshop objectives were:

- To communicate vital knowledge and insights on the benefits and usability of the emerging area of Autonomic systems and mechanisms in Network services and Applications.
- To present to the present to the audience the EFIPSANS project context, outcomes, achievements and results.
- To familiarize the Network technology experts and Engineers with Autonomic Networking and Self-Management mechanisms, validation and deployment methodologies in IPv6-based autonomic/self-managing networks.
- To share with the audience the ideas behind the EFIPSANS proposed Extensions to IPv6 (IPv6++) and illustrate a viable Evolution Path for the Internet towards the Self-Managing Future Internet powered by IPv6 and envisaged protocol & network evolution.
- To discuss ways for actual and direct deployment in existing systems and infrastructures.
- To present the new Interface (Network Governance Interface) required by Network Operators to interact with an Autonomic/Self-Managing Network.
- To engage engineering community in discussing a number of crucial issues like compatibility, expandability, security, etc., around actual deployment.
- To get feedback on the practical applicability and usability of the autonomic/self-management mechanisms in all different levels and layers of Telecom industry.
- To estimate scientific trends towards autonomic systems after evaluating the received feedback.

iii. Institutions involved into

Ericsson, Telefónica, Telcordia, Fujitsu, Alcatel-Lucent, Fraunhofer-Fokus, IPv6.

d. Self-optimization and self-configuration in wireless networks (SOCRATES) – February 2011 [19]

i. About the project

The SOCRATES (self-optimization and self-configuration in wireless networks) project aims at the development of self-organization methods to enhance the operations of wireless access networks, by integrating network planning, configuration and optimization into a single, mostly automated process requiring minimal manual intervention.

Regarding the technological scope, SOCRATES primarily concentrates on wireless access networks, as the wireless segment generally forms the bottleneck in end-to-end communications, both in terms of operational complexity and network costs. As a consequence, the largest gains from self-organization can be anticipated here. 3GPP LTE (3rd Generation Partnership Project, Long Term Evolution) is selected as the radio interface of central radio technology in this study. The reason for this choice is that 3GPP LTE is the natural, highly promising and widely supported evolution of the world's most popular cellular networking technologies (GSM/EDGE, UMTS/HSPA).

ii. Motivation

Bringing together a well suited, strong consortium of two of the world's largest equipment vendors (Ericsson, Nokia Siemens Networks), a leading mobile operator (Vodafone), an SME developing support tools for network planning and operations (Atesio) and three renowned research organizations (IBBT, TNO ICT, TU Braunschweig) with a proven record in successful cooperation with the mobile industry, the SOCRATES project has a great opportunity to achieve considerable impact.

The SOCRATES project influences global standardization, by developing solutions for standardised measurements, new or adapted interfaces and new or modified protocols sup- porting self-organization functionalities.

SOCRATES reinforces European industrial leadership by contributing to European dominance in the development of world-wide standards, by creating a 'head start' in the development of self-organizing features for radio networks and support tools, and in providing high-level consultancy. In addition, the strong partnership creates stronger synergies between various sector actors and contributes to new business models.

The findings from the consortium create new industrial and business opportunities within the management and control area of existing and future networks, and have several important spin-offs, e.g. towards the development of new services with a reduced time-to-market.

iii. Objectives

The general objective of SOCRATES is to develop self-organization methods in order to optimize network capacity, coverage and service quality while achieving significant OPEX (and possibly CAPEX) reductions. Although the developed solutions are likely to be more broadly applicable (e.g. to WiMax networks), the project primarily concentrates on 3GPP's LTE radio interface (E-UTRAN). In more detail the objectives are as follows:

- The development of novel concepts, methods and algorithms for the efficient and effective self-optimization, self-configuration and self-healing of wireless access networks, adapting the diverse radio (resource management) parameters to smooth or abrupt variations in e.g. system, traffic, mobility and propagation conditions. Concrete examples of the radio parameters that are addressed include: power settings, antenna parameters, neighbour cell lists, handover parameters, scheduling parameters and admission control parameters.
- The specification of the required measurement information, its statistical accuracy and the methods of information retrieval including the needed protocol interfaces, in support of the newly developed self-organization methods.

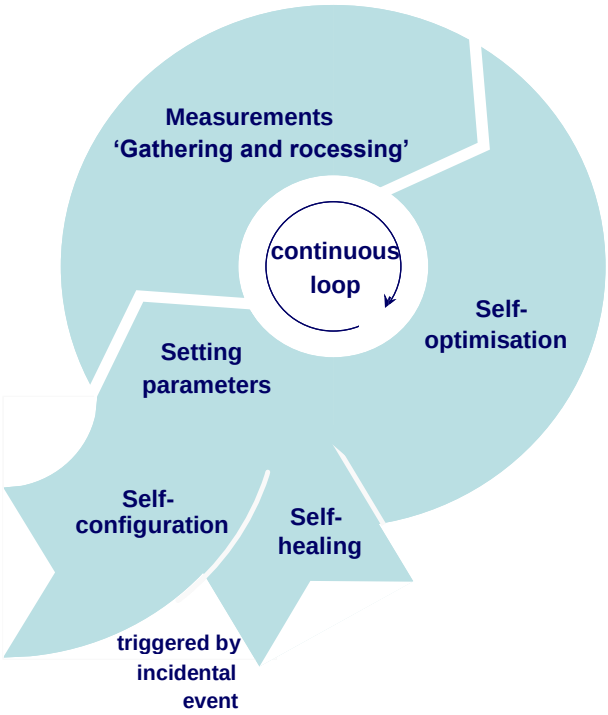


Figure 2.d.1 Self-organization loop

- The validation and demonstration of the developed concepts and methods for self-organization through extensive simulation experiments. In particular, simulations are performed in order to illustrate and assess the established capacity, coverage and quality enhancements, and estimating the attainable OPEX (/CAPEX) reductions.
- An evaluation of the implementation and operational impact of the developed concepts and methods for self-organization, with respect to the operations, administration and maintenance architecture, terminals, scalability and the radio network planning and capacity management processes.
- Influence on 3GPP standardization and NGMN activities.

iv. Institutions involved into

Ericsson AB, Nokia Siemens Networks-Poland/Germany, Atesio, IBBT, TNO Information and Communication Technology, and TU Braunschweig.

e. Generic Autonomic Network Architecture (GANA) – April 2013 [20]

i. About the project

The GANA (Generic Autonomic Network Architecture) reference model is a unified model for autonomic networking, cognition, and self-management. It defines generic functional blocks and associated reference points and characteristic information that are specific to enabling autonomics, cognition, and self-management in target architecture.

Therefore, it can be "instantiated" onto implementation-oriented reference architecture such as the 3GPP architecture, BBF architecture or ITU-T (NGN) architecture. The generic functional blocks and reference Points can also be applied in designing future network architectures that exhibit self-management capabilities from the dawn (outset) of their design.

The reference model is addressed to network architects, researchers, and developers/implementers "refer" to the reference model when reasoning about or applying the concepts and principles defining the domain of autonomic communication, autonomic networking, autonomic and cognitive management and control-all as part of the "big-picture" of self-management.

ii. Motivation

Through analysis of the state-of-the art, the existing standards and current practices, the following have been identified as essential properties needed, foreseen or desirable for the systems and networks intended to apply the Generic Autonomic Network Architecture (GANA) reference model.

The way they shall be realized in future networks and systems is left for further study and will be documented on in the work related to the Instantiation of the Model onto concrete implementation-oriented reference architectures:

- Automation.
- Awareness.
- Adaptiveness.
- Stability.
- Scalability.
- Robustness.
- Security.
- Switchable.
- Federation.

i. Objectives

The general objectives of GANA in more detail are as follows:

- An autonomic system should be able to detect, reconfigure and reregister its managed resources or managed devices such as router or user equipment even if it is mobile and allow session continuity with no disruption.
- An autonomic system should manage and control the mobility of an ambient system, in order to provide session continuity, local mobility decision should take into account the preferences, the capabilities, the objectives of the different players involved in session in order to identify a common decision able to provide session continuity.
- The mobility enabler will be used to retrieve the different monitoring process, security process, configuration process in order to decide and disseminate the mobility decision.
- A mobility enabler should be managed by different type of players. It will avoid today incoherence decision e.g. User Equipment (UE), Wireless Local Access Network (WLAN), 3GPP Mobile access network (e.g. UTRAN), core network(e.g. Home Agent) or application provider (e.g. Service Centralisation and Continuity Server AS-SCC). Each of them takes local decision with no common knowledge.

i. Institutions involved into

European Telecommunications Standards Institute.

3 Topics update

A key challenge of the autonomic computing initiative has been to draw upon self-* properties in systems other than computational ones in order to develop new computing systems. The main objective of this thesis has been to demonstrate how an autonomic system achieves an efficient communication in distributed environments without centralized control. This efficiency leads to a reduction of complexity, and therefore costs, through autonomic and self-managed behaviour.

The self-organized behaviour represents collective behaviour that emerges at the level of the group from the numerous interactions among individuals and between the individuals and the environment. This biological inspiration is applied in a digital city as a scenario where autonomic and self-aggregation capabilities can be exploited, like the emergency use-case used in this thesis. In order to achieve this aim, an aggregation protocol has been analysed, the “On-demand” clustering protocol.

The “On-demand” protocol contains the rules and interactions to implement the basic communication behaviour of the ACE. The prototype designed in this thesis develops the aggregation protocol and it has been used to include a communication part in the ACEs, making possible introduce real-time interactions to face changes in the environment conditions.

One of the main changes in the investigation of self-organized networks is an applied network orientation given the increasing number of devices that are connecting through real Internet daily. Therefore, the research has focused on how to transform the connection layer of Internet in a network capable of applying the self-* properties, like as self-organization. Some of the most important telecom operators have been involved in these new projects

For instance, the ANA’s Project goal has been to design and develop a novel autonomic network architecture that enables flexible, dynamic, and fully autonomous formation of network nodes as well as whole networks. The main premise of the project work is that a functionally scaling network is a synonym for an evolving network which includes the various self-x attributes essential to autonomic communication such as self-management, self-optimization, self-monitoring, self-repair, and self-protection. The ANA’s Project approach has allowed study both scientific and technical, looking recompose existing layer in the current internet connection to make it properly autonomous self-organization. The plan consists in losing the constraints of Internet layer and permit wired wireless heterogeneous devices to be integrated in an autonomic way.

Moreover in a technical research line, the EFIPSANS’ workshop has exposed the features in IPv6 protocols that can be exploited or extended for the purposes of designing or building autonomic networks and services. The EFIPSANS has proposed extensions to IPv6 (IPv6++) and illustrated a viable evolution path for the Internet towards the self-managing future Internet. It will be powered by IPv6 and envisaged protocol & network evolution.

Finally, the GANA's Project has been searching to create a unified reference model for autonomic networking, cognition, and self-management. This model defines generic functional blocks and associated reference points and characteristic information that are specific to enabling autonomics, cognition, and self-management in target architecture. Therefore, it can be instantiated onto implementation-oriented reference architecture such as the 3GPP architecture.

Furthermore, some research projects have done a more extensive investigation of the theoretical basis of the autonomic communication systems. BIONETS is a clear example of this development.

BIONETS' project has drawn inspiration from the living world in terms of evolutionary paradigms able to drive the adaptation process of autonomic services and social paradigms for the provisioning of the necessary cooperation mechanisms. The net result is the introduction of autonomic self-evolving services that are able to adapt to localized needs and conditions while ensuring the maintenance of a purposeful system. Services in BIONETS are also autonomic, and evolve to adapt to the surrounding environment, like living organisms evolve by natural selection. Biologically-inspired concepts permeate the network and its services, blending them together, so that the network moulds itself to the services it runs, and services, in turn, become a mirror image of the social networks of users they serve.

Finally, other theoretical approach is the SOCRATES' project. This project focalize on develop self-organization methods in order to optimize network capacity, coverage and service quality while achieving significant operational complexity and network costs reductions.

The evolution of the research has mainly change from a theoretically study to a more practical one. This has been noticeable thought the creation of different frameworks required to implement the new architecture. In conclusion we could say that the foundations keys of CASCADAS' project have remained intact but have been adapted to be applied in real networks.

A more practical approach to autonomic principles and device scalability has been performed in real networks research. Therefore, it can be considered that the different projects that have emerged are an evolution of theoretical communication model proposed in the test case of this thesis.

The study of the current status of development on self-organized networks has made me realize I have been part of first steps in autonomic networks research. I'm confidence that the use case study of this thesis may contribute in analysis of high-level communication protocols of autonomic systems. However, the "On-demand" communication protocol must have been notorious transformations for being able to be adapted at the devices interactions in a real data network.

The emerging technology based on the IBM's principles of autonomic system has a promising future due the efforts of main telecom operators. The key to their success may come from reduced operation costs in expanding data networks as well as an improvement in the efficiency of these networks with a large number of interconnected devices, having also a service orientation.

REFERENCES

[1] IP CASCADAS “Integrated Project Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services” http://www.cascadas-project.org

[2] N. Ravi, et al., “Accessing Ubiquitous Services using Smart Phones”, 3rd International Conference on Pervasive Computing and Communications, Kauai Island (NW), March 2005

[3] A. Manzalini, R. Alfano, A. Bernadas, J. Sole Pareta, S. Spadaro, Demonstrating communication services based on autonomic self-organization, in Proc. Int. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS 2008), 4-7 March, 2008, Barcelona (Spain)

[4] RuleML (Rule Markup Language), http://www.ruleml.org

[5] R. Sterritta, M. Parasharb, H. Tianfieldc, R. Unland, A concise introduction to autonomic computing, Advanced Engineering Informatics 19 (Elsevier, 2005), 181–187

[6] G. Di Marzo Serugendo, M.-P. Gleizes, A. Karageorgos, “Self-Organization in MAS”, Knowledge Engineering Review 20(2):165-189, Cambridge University Press, 2005

[7] Carole Bernon, Valérie Camps, Marie Pierre Gleizes, Gauthier Picard, “Tools for Self-Organizing Applications Engineering”. Engineering Self-Organising Systems, 2003, pp:283-298 AAMAS 2003

[8] M. Dorigo, T. Stüzle (2004). Ant Colony Optimization. MIT Press, Cambrige (Mass).

[9] Cl. Detrain, J.L. Deneubourg & J.M. Pasteels (1999). Information Processing in social Insects. Birkhauser, Basel.

[10] Aggregation Algorithms, Overlay Dynamics and Implications for Self-Organised Distributed Systems. IST CASCADAS Work Package 3 Deliverable Month.

[11] WikiCity, Sensible Consortium, http://senseable.mit.edu

[12] E. W. Dijkstra, Go To Statement Considered Harmful, Communications of the ACM, Vol. 11 (1968)

[13] Eclipse, Open Source Integrated Development Environment, http://www.eclipse.org

[14] Java, http://java.sun.com

[15] DIET, Decentralised Information Ecosystem Technologies, http://diet-agents.sourceforge.net

[16] ANA, Autonomic Network Architecture, http://www.ana-project.org/

[17] BIONETS, Biologically inspired Network and Services, http://www.bionets.eu/

[18] EFIPSANS workshop, http://www.efipsans.org/

[19] SOCRATES, Self-optimization and self-configuration in wireless networks, http://www.fp7-socrates.eu/

[20] GANA, Generic Autonomic Network Architecture, http://www.etsi.org