Volume 30 Number 1 January 2006 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue:

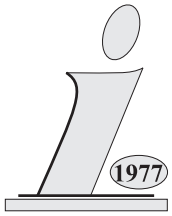
Hot Topics in European Agent Research II

Guest Editors:

Andrea Omicini

Paolo Petta

Matjaž Gams



The Slovene Society Informatika, Ljubljana, Slovenia

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

The Second AgentLink III Technical Forum: Main Issues and Hot Topics in European Agent Research — Part 2

1 Introduction

Together with the previous edition, the present number of *Informatica* presents topics out of the leading edge of European research in agent-oriented systems. The contributions collected in this double special issue originated in the presentations and discussions at and surrounding the Second AgentLink III Technical Forum (AL3-TF2) hosted by the Jozef Stefan Institute in Ljubljana, Slovenia, from February 28 to March 2, 2005. We refer the interested reader to the companion issue (29(4)) for details on AgentLink III, a European Commission (EC)-sponsored Coordination Action aimed at supporting and strengthening European research and development in agent-based technologies.

The double special issue includes both, broader survey papers on different research areas, as well as *hot topic* articles complementing the critical characterisation of the consolidated achievements with deeper analyses and some fresh ideas, thereby conveying an impression of the liveliness of (still much needed!) research in this technological area of ever-growing importance.

The first number of the double special issue includes articles from the areas of Agent-Oriented Software Engineering and the emerging research topic of Environments for Multiagent Systems. The contributions in the present issue cover three additional areas (for which the short names of the AgentLink Technical Forum Groups pursuing the research efforts are again given in parentheses[1], as follows.

Multiagent Resource Allocation (TFG-MARA)

- *Issues in Multiagent Resource Allocation*, by Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaître, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodríguez-Aguilar, and Paulo Sousa, surveys some of the most salient issues in Multiagent Resource Allocation, an emerging area of research at the interface of Computer Science and Economics.

Programming Multi-Agent Systems (TFG-PROMAS)

- *A Survey of Programming Languages and Platforms for Multi-Agent System*, by Rafael Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah-Seghrouchni, Jorge J. Gomez-Sanz, João Leite, Gregory O'Hare, Alexander Pokahr and Alessandro Ricci, surveys the most recent research on programming languages and development tools for MASs.

[1] Further information about AgentLink III Technical Fora is available from the AgentLink website at `http://www.agentlink.org/activities/al3-tf/`.

Self-Organisation in MAS (TFG-SELFORG)

- *Self-Organisation and Emergence in MAS: An Overview*, by Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos, aims at defining the concepts of self-organisation and emergence, and also at providing a state-of-the-art survey about the different classes of self-organisation mechanisms applied in the MAS domain.
- *Bio-inspired Mechanisms for Artificial Self-organised Systems*, by Jean-Pierre Mano, Christine Bourjot, Gabriel Lopardo, Pierre Glize, analyses three forms of biological self-organisation (stigmergy, reinforcement mechanisms and cooperation), and discusses some case studies to show how they could be transposed to artificial systems.
- *On Self-Organising Mechanisms from Social, Business and Economic Domains*, by Salima Hassas, Giovanna Di Marzo Serugendo, Anthony Karageorgos and Cristiano Castelfranchi, discusses examples of socially-inspired self-organisation approaches, as well as their use to build socially-aware, self-organising computational systems.
- *Applications of Self-Organising Multi-Agent Systems: An Initial Framework for Comparison*, by Carole Bernon, Vincent Chevrier, Vincent Hilaire and Paul Marrow, provides MAS examples where self-organisation is used to solve complex problems, along with a number of criteria for comparison of self-organisation between different applications.

Acknowledgements

We would like to repeat our thanks to the AL3-TF2 TFG Chairs for their exemplary efforts that most bespeakingly reflect the enthusiasm and thrust in European research in their areas. The lasting result of AgentLink III would not have been possible without the dedication of its staff, the impeccable work of the Management Committe, and the infinite and patient support provided by Catherine Atherton, Becky Earl, Adele Maggs and Serena Raffin. We also wish to thank once again the local organisers and their staff at the Jozef Stefan Institute in Ljubljana, Slovenia, who allowed us to implement the many lessons learned with the previous technical forum.

We hope you enjoy this second part of the double special issue of *Informatica*: please do consider it an open invitation to join the efforts described!

Andrea Omicini, Paolo Petta and Matjaž Gams

Issues in Multiagent Resource Allocation

Yann Chevaleyre
LAMSADE, Université Paris-Dauphine (France)
E-mail: chevaley@lamsade.dauphine.fr

Paul E. Dunne
Department of Computer Science, University of Liverpool (UK)
E-mail: ped@csc.liv.ac.uk

Ulle Endriss
ILLC, Universiteit van Amsterdam (The Netherlands)
E-mail: ulle@illc.uva.nl

Jérôme Lang
IRIT, Université Paul Sabatier, Toulouse (France)
E-mail: lang@irit.fr

Michel Lemaître
ONERA, Centre de Toulouse (France)
E-mail: michel.lemaitre@cert.fr

Nicolas Maudet
LAMSADE, Université Paris-Dauphine (France)
E-mail: maudet@lamsade.dauphine.fr

Julian Padget
Department of Computer Science, University of Bath (UK)
E-mail: jap@cs.bath.ac.uk

Steve Phelps
Department of Computer Science, University of Liverpool (UK)
E-mail: sphelps@csc.liv.ac.uk

Juan A. Rodríguez-Aguilar
Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona (Spain)
E-mail: jar@iiia.csic.es

Paulo Sousa
DEI, Instituto Superior de Engenharia do Porto (Portugal)
E-mail: psousa@dei.isep.ipp.pt

Keywords: resource allocation, negotiation, preferences, social welfare, complexity, simulation

Received: May 12, 2005

The allocation of resources within a system of autonomous agents, that not only have preferences over alternative allocations of resources but also actively participate in computing an allocation, is an exciting area of research at the interface of Computer Science and Economics. This paper is a survey of some of the most salient issues in Multiagent Resource Allocation. In particular, we review various languages to represent the preferences of agents over alternative allocations of resources as well as different measures of social welfare to assess the overall quality of an allocation. We also discuss pertinent issues regarding allocation procedures and present important complexity results. Our presentation of theoretical issues is complemented by a discussion of software packages for the simulation of agent-based market places. We also introduce four major application areas for Multiagent Resource Allocation, namely industrial procurement, sharing of satellite resources, manufacturing control, and grid computing.

Povzetek: Opisana je alokacija virov v sistemu avtonomnih agentov.

1 Introduction

The allocation of resources is a central matter of concern in both Computer Science and Economics. To emphasise the fact that resources are being distributed amongst several agents and that these agents may influence the choice of allocation, the field is sometimes called *Multiagent Resource Allocation* (MARA). The questions investigated by computer scientists are often of a procedural nature (*how* do we find an allocation?), while economists are more likely to concentrate on qualitative issues (*what* makes a good allocation?). A comprehensive analysis of the problem at hand, however, requires an interdisciplinary approach. Here the *multiagent system* (MAS) paradigm offers an excellent framework in which to study these issues.

MARA is relevant to a wide range of applications. These include, amongst others, industrial procurement [45], manufacturing and scheduling [15, 71, 89], network routing [38], the fair and efficient exploitation of Earth Observation Satellites [59, 60], airport traffic management [52], crisis management [62], logistics [49, 77], public transport [16], and the timely allocation of resources in grid architectures [48].

This paper is a survey of some of the most salient issues in MARA. In the remainder of this introduction, we first give a tentative definition of MARA and introduce its main parameters (Section 1.1). To illustrate the interdisciplinary character of the field, we then list some of the research questions that we consider particularly interesting and challenging (Section 1.2). Finally, we give an overview of the content of the main body of the paper (Section 1.3).

1.1 What is MARA?

A tentative definition would be the following:

> *Multiagent Resource Allocation is the process of distributing a number of items amongst a number of agents.*

However, this definition needs to be further qualified: *What* kind of items (resources) are being distributed? *How* are they being distributed (in other words, what kind of allocation procedure or mechanism do we employ)? And finally, *why* are they being distributed (that is, what are the objectives of searching for an allocation and how are these objectives determined)?

1.1.1 Resources

We refer to the items that are being distributed as *resources*, while *agents* are the entities receiving them. We should stress that this terminology is not universally shared. In the context of applications of MARA in manufacturing, for instance, we usually speak of *tasks* that are being allocated to *resources*. That is, in this context, the term "resource" (*i.e.* the resources available to the manufacturer for production) refers to what we would call an "agent" here.

We can distinguish different types of resources. For instance, resources may or may not be divisible. For divisible resources (such as electricity), different agents may receive different fractions of a resource. In the case of indivisible resources, it may or may not be possible for different agents to *share* (jointly use) the same resource (e.g. access to network connections as opposed to items of clothing). For many purposes, *task allocation* problems can be regarded as instances of MARA (if we think of tasks as resources associated with a *cost* rather than a benefit).

1.1.2 Allocations

A particular distribution of resources amongst agents is called an *allocation*. For instance, in the case of non-sharable indivisible resources, an allocation is a partition of the set of resources amongst the agents. The set of resources assigned to a particular agent is also called the *bundle* allocated to that agent.

1.1.3 Agent Preferences

Agents may or may not have *preferences* over the bundles they receive. In addition, they may also have preferences over the bundles received by *other* agents (in the case of network connections, for example, the value of a resource diminishes if shared by too many users). The latter type of preferences are called *externalities*.

Agents may or may not report their preferences truthfully. To provide incentives for agents to be truthful is one of the main objectives of *mechanism design*.

1.1.4 Allocation Procedures

The *allocation procedure* used to find a suitable allocation of resources may be either *centralised* or *distributed*. In the centralised case, a single entity decides on the final allocation of resources amongst agents, possibly after having elicited the agents' preferences over alternative allocations. Typical examples are combinatorial auctions. Here the central entity is the auctioneer and the reporting of preferences takes the form of bidding. In truly distributed approaches, on the other hand, allocations emerge as the result of a sequence of local negotiation steps.

1.1.5 Objectives

The objective of a resource allocation procedure is either to find an allocation that is *feasible* (e.g. to find any allocation of tasks to production units such that all tasks will get completed in time); or to find an allocation that is *optimal*. In the latter case, the allocation in question could be optimal either for the central entity choosing the allocation (e.g. a solution to a combinatorial auction that maximises the auctioneer's revenue); or with respect to a suitable aggregation of the preferences of the individual agents in the system (e.g. an allocation of resources that maximises the average utility enjoyed by the agents).

Combinations are also possible: The objective may be to find an optimal allocation amongst a small set of feasible allocations; and what is considered optimal could depend both on the preferences of a central entity and on an aggregation of the other agents' individual preferences (e.g. auction mechanisms aiming at balancing revenue maximisation and bidder satisfaction). Of course, where computing an optimal allocation is not possible (due to lack of time, for instance), any progress towards the optimum may be considered a success.

1.1.6 Social Welfare

Multiagent systems are sometimes referred to as "societies of agents" and the aggregation of individual preferences in a MARA system can often be modelled using the notion of *social welfare* as studied in Welfare Economics and Social Choice Theory. Examples include utilitarian social welfare, where the aim is to maximise the sum of individual utilities, and egalitarian social welfare, where the aim is to maximise the individual welfare of the agent that is currently worst off.

1.1.7 The Role of Agents

Our discussion shows that the term "multiagent" in Multiagent Resource Allocation can have different interpretations:

- If a *distributed* resource allocation procedure is used, then the term "multiagent" indicates that the computational burden of finding an allocation is shared amongst several agents.
- If an *aggregation of individual preferences* is used to assess the quality of the final allocation, then the term "multiagent" refers to the fact that the choice of allocation depends on the preferences of several agents (rather than on the preferences of a single entity).

Of course, the term "multiagent" could also be derived merely from the fact that resources are being allocated to several different agents. However, if individual agents have no preferences (or such preferences are not taken into account) and the allocation procedure is centralised, then using the term "multiagent" may be less appropriate.

1.1.8 A Computational Perspective

MARA, as introduced at the beginning of Section 1.1, may not seem to differ significantly from what has traditionally been studied in Microeconomics. However, a distinctive feature of MARA is the focus on *computational* issues. For instance, with respect to the preferences of individual agents, we are interested in representations that can be efficiently managed and communicated. Similarly, in the case of allocation procedures, MARA encompasses both the theoretical analysis of their computational complexity and the design of efficient algorithms for scenarios for which this is possible. As a final example, concerning the strategic aspects of negotiation, we may find that classical results in Game Theory fail to hold due to the computational limitations of the participating agents.

1.2 Research Topics

MARA is a highly interdisciplinary field; relevant disciplines include Computer Science, Artificial Intelligence, Decision Theory, Microeconomics, and Social Choice Theory. Research in MARA can take a variety of forms:

- *Preferences:* What are suitable representation languages for agent preferences? Issues to consider include their expressive power, their succinctness, and their suitability in view of preference elicitation.
- *Social welfare:* What are suitable measures of social welfare to assess the quality of an allocation for a given application? Under what circumstances can we expect an optimal allocation to be found?
- *Complexity:* What is the overall complexity of finding a feasible/optimal allocation? What is the complexity of the decision problems that agents need to solve locally? What is the communication complexity (amount of information to be exchanged) of negotiation?
- *Negotiation:* In particular for the distributed approach, what are suitable negotiation protocols? What are good strategies for agents using such protocols?
- *Algorithm design:* How can we devise efficient algorithms for MARA (e.g. algorithms for combinatorial auction winner determination in the centralised case; algorithms to support complex negotiation strategies in the distributed case)?
- *Mechanism design:* How can we devise negotiation mechanisms that force agents to report their preferences truthfully (both to reduce strategic complexity and to allow for a correct assessment of social welfare)?
- *Implementation:* What are best practices for the development of prototypes for specific MARA applications and general-purpose platforms to support quick prototyping?
- *Simulation and experimentation:* How do different optimisation algorithms or negotiation strategies perform in practice? How serious is the impact of theoretical impossibility results in practice? How prohibitive are theoretical intractability results (computational complexity) in practice?
- *Interplay of theory and applications:* What constraints do real-world applications impose on theoretical models for MARA? How can theoretical results inform the development of new tools?

The aim of this survey is to provide a base line for some of these issues. In particular, we present a range of languages for representing preferences, we give an overview of the social welfare measures most relevant to MARA, and we review known complexity results in the area. As it is often difficult to make precise predictions on the performance of a resource allocation procedure by theoretical means alone, we also discuss the requirements to be met by software packages for MARA simulations. To underline the importance of further research in the area, we introduce several prestigious applications and discuss the challenges imposed on MARA models by these applications.

1.3 Paper Overview

The remainder of this survey paper is organised as follows. In Section 2, we introduce four major *application areas* for MARA technology. These are industrial procurement, the joint exploitation of Earth Observation Satellites, manufacturing control, and grid computing. Throughout Section 2, we highlight the specific challenges raised by these applications.

Next we review three important parameters that are relevant to the *definition* of a MARA problem. Firstly, in Section 3, we discuss generic properties of resources, such as being *indivisible* or *sharable*, and how such properties would affect the design of a concrete MARA system. We then move on, in Section 4, to the issue of *preference representation* for individual agents. Each agent needs to be endowed with a suitable representation of preferences over alternative allocations and it is important to be able to express these preferences in a compact way. We discuss both *quantitative* and *ordinal* preference languages. A third parameter in the definition of a MARA problem is the *social welfare* measure (or a similar tool) we employ to assess the overall quality of a given allocations. A range of different concepts—including *collective utility functions*, *Pareto optimality*, and *envy-freeness*—are reviewed in Section 5.

In Section 6, we attempt to give a short overview of the parameters that are relevant when one chooses (or designs) an *allocation procedure*. We discuss the respective merits and drawbacks of *centralised* and *distributed* approaches to MARA, and we briefly introduce some (centralised) *auction protocols* as well as (distributed) *negotiation protocols*. We also report on results that establish under what circumstances allocations can be expected to *converge* to a socially optimal state in a distributed negotiation setting. Section 7 is a survey of relevant *complexity results*. We mostly concentrate on the *computational complexity* of problems such as finding a socially optimal allocation, but we also briefly discuss issues in *communication complexity* for MARA, which is concerned with the length of negotiation processes.

Our presentation of theoretical issues is complemented by a discussion of software packages for the *simulation* of agent-based market places in Section 8. We start by giving an overview of the typical requirements to be met by such packages and then list the most relevant software products available to MARA researchers interested in simulation. Finally, Section 9 concludes.

2 Application Areas

As mentioned already in the introduction, MARA is relevant to a wide range of application domains. In this section, we introduce four of these problem domains, all of which have recently been addressed by (some of) the authors of this survey.

2.1 Industrial Procurement

The sourcing process of multiple goods or services usually involves complex negotiations that include discussion of product features as well as quality, service, and availability issues. Consequently, several commercial systems to support online negotiation (e-sourcing tools) have been developed. In fact, e-sourcing is becoming an established part of the business landscape [90]. However, there are still enormous challenges confronting users who want to get the maximum value out of e-sourcing.

2.1.1 Problem Description

Traditionally, the core of the sourcing process comprises the following tasks:

- request for quotation/proposal (RFQ/RFP);
- provider selection for RFQ/RFP delivery;
- offer generation;
- negotiation through offer/counter-offer interaction or reverse auction; and
- selection of best offers.

Typically a buyer creates an RFQ by sequentially adding items. Each item specifies a product, be it a good or service. A paradigmatic example of multi-item RFQ occurs in industrial settings. The production plan outlined by a company's ERP (Enterprise Resource Planning) or SCM (Supply Chain Management) application comes in the shape of a list of items to be produced along with the parts required for each product, the so-called bill of material. This is the basis for the buyer to initiate multiple sourcing events, each devoted to the procurement of the parts for each of the items to be produced.

Although several commercial systems to support online negotiations have been released, to the best of our knowledge, not a single system can claim to address the full complexity of online negotiation. The first generation of sourcing tools merely incorporate single-item, price-quantity reverse auction mechanisms. Others only offer basic negotiation capabilities that are usually reduced to a demand-offer matching tool. In general terms, there is a lack of

decision support functionalities (decision making in sourcing can involve a few hundred offers, each of which is described by several dozen attributes). Furthermore, there is a lack of technology support for computationally complex negotiation paradigms, which inhibit the application of promising mechanisms such as combinatorial reverse auctions [24, 54].

2.1.2 Challenges

Although the degree of automation, namely of delegation to trading agents, in industrial procurement settings is still low, we do believe that MARA techniques can contribute to improve this situation. In what follows we identify several challenges that any commercial tool aiming at the successful implementation of resource allocation amongst several (human or software) agents in an industrial procurement setting must address.

- *Preferences of buyers and providers.* How do we best capture and represent trading agents' preferences so that they can effectively value their trading partners' offers, counter-offers, and RFQs? While recent advances in preference elicitation are encouraging (see, for instance, the work of Bichler et al. [6]), this still remains as the Achilles' heel of industrial procurement applications.

- *Business rules to constrain admissible allocations.* While in direct auctions, the items to be sold are physically concrete (they do not allow configuration), in a negotiation involving highly customisable goods, buyers need to express relations and constraints between attributes of different items. On the other hand, multiple sourcing is common practice, either for safety reasons or because offer aggregation is needed to cope with high-volume demands. This introduces the need to express constraints on providers and on the contracts they may be awarded. Providers may also impose constraints on their offers. Therefore, highly expressive languages for both buying and providing agents are required.[1] Incorporating business rules into allocation procedures can lead to more *balanced* and *safer* allocations.

- *Automated negotiation strategies.* There are several dimensions to take into account when designing negotiation strategies. Agents may negotiate over multiple attributes of the same item, over a bundle of multiple items, or they may hold separate but interdependent negotiations. Negotiation techniques such as trade-off [37] or partial-order scheduling [102] are candidate techniques put forward from the research arena. The current procurement practices tell us that the possibility of automatic offer submission is seen with interest for repetitive sourcing events in private e-sourcing platforms where providers and business rules are well-known or result from a provider qualification procedure or a frame contract. Nonetheless, the full application of such automated trading still faces barriers, such as providers not wanting to reveal their capabilities/preferences to third parties.

- *Choice of mechanism.* Commercial sourcing tools offer an ever increasing number of customisable negotiation mechanisms. Nonetheless, market design is a highly complex, intricate task. New trends in automated mechanism design [22] as well as evolutionary mechanism design [73] may prove valuable in assisting in the design of market scenarios that ensure certain global properties.

- *Winner determination algorithms.* Further research into algorithms capable of identifying the optimal set of offers in multi-attribute, multi-item negotiation scenarios with side constraints representing business rules is required [45, 83].

- *Bundling.* Should a buyer (seller) conduct a single negotiation or auction for an entire bundle of goods he or she is interested in purchasing (selling) or should they group items into bundles and conduct several negotiations? Unfortunately, for complexity reasons, combinatorial bidding capabilities are rarely found on commercial systems. To overcome this problem, we can think of a third approach: Based on past market real data and knowledge, the whole bundle of items can be divided into separate negotiations for which the appropriate providing agents are invited and for which certain properties are satisfied (e.g. invite providing agents that can offer at least 90% of the items in the bundle). These properties model the expertise of e-sourcing specialists in the form of rules of thumb [76].

Some of these challenges are already being tackled by recently developed negotiation support tools. *iBundler* [44, 44], for instance, is an agent-aware decision support service acting as a combinatorial negotiation solver for both multi-item, multi-unit negotiations and auctions that can integrate business rules to constrain admissible solutions. *iAuctionMaker* [76] is a novel decision support tool for mixed bundling that can help an auctioneer determine how to group items into *promising* bundles that are likely to produce a high revenue. Promising bundles are those that satisfy certain properties believed to be present in competitive sourcing scenarios. These properties are defined by

[1] Consider a buyer who wants to buy 200 chairs (any colour/model is fine) for the opening of a new restaurant and who uses an e-procurement solution that launches a reverse auction. If we employ a state-of-the-art combinatorial auction solver, a possible solution might be to buy 199 chairs from provider A and 1 chair from provider B, simply because this is 0.1% cheaper than the next best allocation and it has not been possible to specify that, in case of buying from more than one provider, a minimum of 20 chairs purchase is required. In a different scenario, the optimal solution might tell us to buy 150 blue chairs from provider A and 50 pink chairs from provider B. Why? Because, although we had no preferences over the regarding colour, we could not specify that all chairs should be of the same colour. Although simple, this example shows that without modelling natural constraints, solutions obtained may be mathematically optimal, but unrealistic.

e-sourcing professionals and capture their experience and knowledge in the domain.

2.2 Earth Observation Satellites

Next we consider another real-word application, namely the exploitation of Earth Observation Satellites (EOSs) [10, 59, 60]. This application pertains to the problem of allocating a set of indivisible goods to some agents with no possible monetary compensation between them. As we will see, this is a typical case of a sharing problem, different from an auction situation, especially because fairness is a key issue.

2.2.1 Problem Description

Due to their high cost, space projects such as EOSs are often co-funded and then exploited by several agents (countries, companies, civil or military agencies, etc.). The mission of an EOS is to acquire images (photos) of specified areas on the earth surface, in response to observation demands from users. Such a satellite is operated by an Image Programming and Processing Center. Each day, the Center collects a set of observation demands from agents. Usually a demand can be covered by a single image, but more complex demands may arise, as we will see below. Each demand is given a *weight* (a positive integer), reflecting the importance the requesting agent assigns to the satisfaction of the demand. The daily task of the Center is, amongst others, to build the imaging workload of the satellite for the next day, by selecting the images to be acquired from the set of agent demands.

Naturally, the exploitation of the satellite must obey a set of *physical constraints*, such as time window visibility constraints, minimum transition times between successive image acquisitions, or memory and energy management. Due to these exploitation constraints, and due to the large number of (possibly conflicting) demands, a set of demands, each of which could be satisfied individually, may not be satisfiable as a whole on a single day. All these physical constraints define the set of *admissible* allocations of images to agents. The exploitation of an EOS must also meet the following requirements:

- *Efficiency:* The satellite should not be underexploited.
- *Equity:* Each agent should get a return on investments that is proportional to its financial contribution.

2.2.2 Modelling

Let us first consider the simple problem where only one agent exploits the resource. In this case, the allocation problem consists of selecting, each day, an admissible sequence of images that will be acquired by the satellite over the next day (and allocated to the agent). This agent measures its satisfaction by a utility function which may be defined as the sum of the weights of the allocated images. The efficiency requirement comes down to a simple optimisation problem: the utility function of the agent must be maximised over the set of admissible allocations (see Lemaître et al. [61] for the description of some algorithms for solving this mono-agent allocation problem).

We now turn to the case where several agents exploit the satellite. For simplicity, we assume in this paper that the agents have equal rights over the resource (we may assume, for example, that they have funded the satellite equally). Of course, each agent wants to maximise its own utility function, but generally they are antagonistic: increasing the utility of one agent can lead to decreasing the utility of others. So a fair compromise must be found, the realisation of which is the role of a suitable preference aggregation mechanism. Such mechanisms will be discussed in detail in Section 5. Here, the *min* function (egalitarian social welfare) fits our requirements, as it naturally conveys the equity requisite: we try to make the agent least happy as happy as possible (a refinement of this approach is given by the so-called *leximin* ordering; see Section 5.4).

As mentioned before, weights of demands are freely fixed by agents. In order to be able to compare individual utilities between agents, a common utility scale must be set and used; that is, the same number should express the same level of satisfaction. To this end, Lemaître et al. [59, 60] have adopted an approach known as the *Kalai-Smorodinsky* solution (see Section 5.6), whereby individual utilities are compared relative to the maximum utility that each agent can receive. It should be noted that, unlike for auction problems, there are no preemptive constraints in this application: the same image could be requested by several agents, and allocated to them all (*i.e.* resources are sharable).

This application is also of interest because it offers real-word examples of dependencies between demands. As a first example, a request may involve a pair of stereoscopic images; receiving only one image would result in a poor satisfaction level for the agent. A second example comes from the fact that, for earth areas situated in high latitudes, several images of the same area can be taken from distinct angles during the same day. Consider a stereoscopic demand concerning such an area, and suppose that it could be photographed from two angles. Let $o11$ and $o12$ be the pair of stereoscopic images from angle 1, $o21$ and $o22$ the images from angle 2. The demand can be quite naturally formulated as $(o11 \wedge o12) \vee (o21 \wedge o22)$.

To sum up, our EOS multiagent fair resource allocation problem can be formally stated in the following way. Agents express their (weighted) demands as simple logical propositions. An agent's individual utility is the sum of the weights of the satisfied demands. The global utility is an aggregation of normalised individual utilities, the aggregation function being the *min* function (or, better, the *leximin* ordering).

2.3 Manufacturing Systems

Since the second half of the 20th century, the organisation of mass production has been shifting towards flexible manufacturing and customised products. From a technological point of view, it has been observed that current manufacturing systems (e.g. computer-integrated manufacturing architectures) have several drawbacks, in particular excessive rigidity and centralisation [50, 71]. Furthermore, future manufacturing systems are expected to be characterised by globally distributed production units, small quantities of a large variety of products, the provision of individual solutions tailored to each customer's specific needs, and concurrent execution of all the activities in the manufacturing process [95].

2.3.1 Problems and Requirements

Future manufacturing systems therefore require coordination amongst production units and it is expected that rigid, static, and hierarchical manufacturing systems will give way to systems that are more adaptable to rapid change [15]. In order to overcome the identified problems with current manufacturing systems and prepare them for the expected future scenarios, the new generation of systems must possess such attributes as decentralisation, distribution, autonomy, adaptability, and incomplete information handling [88].

In manufacturing, the term *resource allocation* is usually synonymous for *task scheduling*. Furthermore, the term *resource* is understood as a physical resource, *i.e.* a machine, of the manufacturing plant. One of the problems in this area is that a task is a step of a production plan for a specific order (e.g. manufacture 100 chairs of type `P12-5`), and there are usually dependencies between tasks that must be obeyed (e.g. operation "drill hole 2" must be done before "cool surface" but after "drill hole 1").

To further complicate things, the tasks involved in a production plan will probably be done on different production resources, thus creating a network of dependencies amongst resources.

One issue in the manufacturing area is that the schedule itself is only valid until the first disturbance (e.g. machine or tool breakdown, rush order, etc.). Since manufacturing control and execution is a real time application, the need to find a *feasible* solution is much greater than to find one that is *optimal*. The system as a whole must reach a stable and feasible schedule without too much interruption of the shop floor.

2.3.2 Manufacturing and Agents

Physically, a manufacturing system involves several resources (numeric control machines, robots, automated guided vehicles, conveyors) and several tasks can be carried out at the same time. The number and configuration of these may change of the lifetime of the system. Since the manufacturing process is dynamic (e.g. suppliers and consumers in a supply chain may change many times) it is impossible to know the exact structure or topology of the system in advance. The number of products and orders, as well as different alternative production routes, account for the highly complex nature of manufacturing systems.

All of the above make the design of manufacturing systems an excellent candidate for the application of agent-based technology. In many implementations of multiagent systems for manufacturing scheduling and control, the agents model the resources of the plant and the scheduling and control of the tasks is done in a distributed way by means of cooperation and coordination of actions amongst agents [15, 53, 72]. As such, manufacturing scheduling and control touches the areas of distributed planning and distributed artificial intelligence. Nonetheless, there are also approaches that use a single agent for scheduling (usually with a well-known centralised scheduling algorithm) that dictates the schedules that the resource agents will execute [92]. The rationale for modelling resources as agents is to better mimic the actual real-world environment and to allow for the modelling of the characteristics of each resource (e.g. available operations, own agenda of tasks to execute, cost of performing each operation, etc.)

When responding to disturbances, the distributed nature of multiagent systems can also be a benefit to the rescheduling algorithm by involving only the agents directly affected, without disturbance to the rest of the community that can continue with their work. Typical approaches to rescheduling include the removal of a late order, reallocation of low priority orders to make room for rush orders, shifting of tasks from one resource to a similar one, etc.

An example for a MARA system for manufacturing control is the *Fabricare* prototype suite [89]. This a multiagent system for dynamic scheduling of manufacturing orders. The agents are modelled as extended logic programs with the ability to handle negative and incomplete knowledge [88]. The system is very dynamic in what concerns its agents, *i.e.* resource agents depend on the system description file; task agents depend on the existing tasks (dynamic events). Each negotiation uses the set of agents that are present and available at that time, thus giving the system a high degree of adaptability to the dynamic nature of the manufacturing arena.

2.4 Grid Computing

Perhaps one of the most pressing applications for MARA techniques is *grid computing* [40]. It is true that there are functioning systems for grid resource allocation, but these largely operate in benevolent, cooperative subnets where participants know and trust one another and there is typically no charge for the utilisation of resources, although perhaps some artificial accounting system is applied. Such frameworks are exactly what is needed in order to test out many grid middleware functions where the objective is to see a job executed across a range of grid resources. In many respects, grid resource allocation—as distinct from

scheduling—and payment for resource usage is an orthogonal problem to the actual processing of a job.

However, at some stage, if the vision of grid computing as a commodity not unlike energy is to become reality, then resource allocation, payment and job processing will have to come together and current research in MARA technology aims to lay the foundations for this union.

2.4.1 Scalability Issues

If the benevolent, cooperative network of mutually trusting participants is discarded, the client is faced with the problem of piecing together a range of disparate resources that are required to complete the processing of a particular job. The parallels with markets, and especially commodity markets, as efficient (by economic measures) resource allocation mechanisms in the presence of large numbers of traders and where possibly complex packages of goods are required, are striking. Grid networks have not yet become so large as to make such approaches essential, but that time is not far off, even in cooperative scientific research networks, if one considers the grid that is foreseen to support the analysis of results emerging from CERN's Large Hadron Collider [17].

The issues could be seen as a function of scale: Existing grids can handle resource allocation through single centralised mechanisms and (economic) efficiency of allocations may not be important. As grids become larger with a wider range of resources, and used for broader classes of tasks, centralised allocation and inefficient allocation of resources are likely to become less tolerable. In response to this, various approaches need to be evaluated and contrasted under carefully controlled conditions, from centralised systems seeking optimal allocation to distributed mechanisms involving bilateral negotiation. Intuition—which should of course be treated with circumspection—suggests that neither of these can be entirely satisfactory, but each may act in different ways as benchmarks against which to measure the rest:

- Centralised systems relying on combinatorial auction clearing algorithms can deliver optimal allocations, but are currently limited by computation costs to hundreds of items and thousands of bids [81].

- Distributed systems relying on bilateral negotiation between consumer and service provider for each component—that is, the consumer constructs their own bundle—will almost certainly scale, but the results are much less likely to be "good". The risks inherent in such an approach are significant: The order in which to undertake negotiation, the possibility that contracting for one resource constrains the choice of subsequent resources, perhaps leading to incomplete bundles, the difficulty in assessing the quality of a bundle or indeed the valuation of a bundle are all surrounded by uncertainty.

Implicit in both scenarios is that a client will need to combine a range of resources from the grid in order to carry out their computation.

2.4.2 Market-based Allocation

In between the two extremes of centralised and distributed lie the many variants of market-based allocation. And given the essentially decentralised nature of (geographically dispersed) grids, potentially with many administrative centres and relatively weak control over individual nodes, the grid seems well suited to market-based schemes, where the twin benefits of reputation and decentralised negotiation can facilitate the trading of computational resources.

Among the different market schemes that exist, one approach is to mimic ideas seen in commodity trading [48]. While analogies are both risky and seductive, there do seem to be sufficient parallels to make more detailed exploration—and simulation—desirable. Commodity markets are a blend of centralised and distributed in that there are many commodity markets around the world, such that at any one time a significant subset are trading, giving a 24/7 market, but within any given market trades take place through bilateral mechanisms, typically continuous double auctions. However, a trader may participate in more than one market at a time, giving rise to communication between markets as to current valuation trends along with the publication of "closing prices".

But, commodity markets typically trade in lots of a single kind and depending on the market, traders may be direct buyers and sellers with no middle-men or market-makers. Economic analyses and simulations indicate that market-makers increase liquidity and enable the market to remain (economically) efficient at lower levels of participation than in the presence of buyers and sellers alone [7]. Furthermore, in the case of bundles (lots of varying quantities of several kinds of goods), market-makers become repositories of market memory, learning what bundles work (potentially a combination of reputation and fit of resources) and identifying trends as new kinds of bundles emerge. Thus they become more than mediators between buyer and seller, fully justifying the epithet of "market-maker". A trading framework such as this seems highly applicable to grids and resource allocation within grid systems.

3 Types of Resources

A central parameter in any resource allocation problem is the nature of the resources themselves. In this section, we give a brief overview of the (abstract) properties of different types of resources. Some of these properties are characteristics of the resources (such as being perishable rather than static, or continuous rather than discrete), while others are better understood as being characteristics of the chosen allocation system (for instance, whether or not a given item is sharable amongst several agents will typically depend on

the allocation procedure rather than on characteristics of the item itself).

3.1 Continuous vs. Discrete

A resource may be either *continuous* (e.g. energy) or *discrete* (e.g. fruit). This "physical" property will typically influence how the resource is being traded, although this need not be the case. For instance, a continuous resource will typically be regarded as being (infinitely) divisible. Still, in a particular negotiation setting, it may only be possible to buy or sell a certain quantity of such a continuous resource as a whole. Individual units of a discrete resource, however, are always indivisible (an apple that can be sold in small pieces would not count as a discrete resource).

In a setting with several continuous resources, a bundle can be represented as a vector of non-negative reals (or, alternatively, numbers in the interval $[0, 1]$ to denote the proportion of a particular resource owned by the agent receiving the bundle). Bundles of discrete resources can be represented as vectors of non-negative integers. If there is just a single item of each resource in the system, then vectors over the set $\{0, 1\}$ suffice.

A continuous resource may be *discretised* by dividing it into a number of smaller parts to be traded as indivisible units. For instance, rather than treating 10.000 litres of orange juice as a truly continuous resource that could be divided into ever smaller subparts, we may agree to negotiate over 200 units of 50 litres each. This means that methods developed for discrete MARA are often also applicable in the continuous case (although they may not be as efficient as methods specifically tailored to continuous resources).

The allocation of continuous resources (often just a single continuous resource), has been studied in depth in the classical literature in Economics. More recent work in Computer Science and Artificial Intelligence, on the other hand, has often focussed on discrete resources. In this paper, we also concentrate on discrete resources.

3.2 Divisible or not

As discussed above, resources may treated as being either *divisible* or *indivisible*. While being either continuous or discrete is a property of resources themselves, the distinction between divisible and indivisible resources is made at the level of the allocation mechanism. In this survey, we concentrate on indivisible resources.

3.3 Sharable or not

A *sharable* resource can be allocated to a number of different agents at the same time. An example of such sharable resources can be found in the context of the Earth Observation Satellite application discussed earlier (see Section 2.2): A single picture taken by the satellite can be allocated to several different agents (no preemptive constraints). The canonical case, however, considers resources as being *non-sharable* and in the rest of this paper we also make this assumption.

3.4 Static or not

A resource may be *consumable* in the sense that the agent holding the resource may use up the resource when performing a particular action. For instance, fuel is consumable. Also, resources may be *perishable*, in the sense that they may vanish or lose their value when held over an extended period of time. Food is a classical example of a perishable resource.

We call resources that do not change their properties during a negotiation process *static* resources. In general, resources cannot be assumed to be static. In MARA however, it is often assumed that they are (that is, that resources are neither consumable nor perishable). The rationale behind this stance is the fact that the negotiation process is not really concerned with the actions agents may undertake outside the process itself. That is, even if a resource is either consumable or perishable, we can often assume that it remains static throughout a particular negotiation process. In this paper, in particular, we concentrate on static resources.

3.5 Single-unit vs. Multi-unit

In a *multi-unit* setting it is possible to have many resources of the same type and to refer to these resources using the same name. Suppose, for instance, there are a number of bottles of champagne available in the system, but that agents cannot distinguish between these bottles. In a *single-unit* setting, on the other hand, every item to be allocated is distinguishable from the other resources and has a unique name.

The differentiation between single- and multi-unit settings is a matter of representation. Any multi-unit problem can, in principle, be transformed into a single-unit problem by introducing new names for previously indistinguishable items. Vice versa, clearly, any single-unit problem is also a (degenerate) multi-unit unit problem. An important advantage of working within a multi-unit setting is that it may allow for a more compact way of representing both allocations and the preferences of agents over alternative bundles. On the downside, a richer language (variables ranging over non-negative integers, rather than binary values) is required in this case.

3.6 Resources vs. Tasks

At a sufficiently high level of abstraction, a task allocation problem can be reduced to a resource allocation problem. Indeed, *tasks* may be considered *resources* to which agents assign a negative utility. However, an important characteristic of tasks as opposed to resources is the fact that tasks are often coupled with constraints regarding their coherent

combination. For instance, a task may require the achievement of another task as a precondition. In this respect, treating allocations merely as assignments of bundles of items to agents (without associated time constraints, for instance) would be too simple a model.

In this paper, however, we concentrate on general resource allocation problems rather than issues that are specific to task allocation (and exception is our discussion in Section 2.3).

4 Preference Representation

Preferences express the relative or absolute satisfaction of an individual when faced with a choice between different alternatives.[2] In the context of MARA, these alternatives are the different potential allocations of resources, or more concretely, the bundle of resources received by an agent for each of the alternative allocations.

A *preference structure* represents an agent's preferences over a set of alternatives X. There are several choices that can be made regarding the definition of a mathematical model for preference structures (this is an important question that has been discussed by researchers in decision theory for a long time). We can distinguish four families of preference structures:

- A *cardinal* preference structure consists of an evaluation function (generally called *utility*) $u : X \rightarrow Val$, where Val is either a set of numerical values (typically, $\mathbb{N}$, $\mathbb{R}$, $[0, 1]$, $\mathbb{R}^+$, etc.), or a totally ordered scale of qualitative values (e.g. linguistic expressions such as "very good", "good", etc.). In the former case the preference structure is called *quantitative*, in the latter it is called *qualitative*.

- An *ordinal* preference structure consists of a binary relation on alternatives, denoted by $\preceq$, which is reflexive and transitive (and usually, although not necessarily, complete).[3]

  We write $x \prec y$ (strict preference) if and only if $x \preceq y$ but not $y \preceq x$, and $x \sim y$ (indifference) if and only if both $x \preceq y$ and $y \preceq x$.

- A *binary* preference structure is simply a partition of X into a set of *good* and a set of *bad* states. A binary preference structure can be seen as both a (degenerate) ordinal preference structure and a (degenerate) cardinal preference structure.

- A *fuzzy* preference structure is a fuzzy relation over X, *i.e.* a function $\mu : X \times X \rightarrow [0, 1]$. $\mu(x, y)$ is the degree to which x is preferred over y. Fuzzy preferences are more general than both ordinal and cardinal preferences.

Since fuzzy and qualitative preferences have not been used much as far as resource allocation is concerned, we are going to neglect these in this survey, and focus on quantitative and ordinal preferences instead.

Observe that we have three "levels" for preference modelling, according to the possible operations allowed by the preference structure: Ordinal preferences allow only for comparing the satisfaction of a given agent for different alternatives, but cannot express preference intensity and do not allow for interpersonal comparison of preferences (that is, expressing statements such as "agent i is happier with x than agent j with y"). Qualitative preferences do allow for interpersonal comparison of preferences, and can express a weak form of intensity, but they do not allow for any "metric" use of preferences such as computing the difference between two utility degrees so as to allow for a monetary compensation—while quantitative preferences do.

Note that any cardinal preference induces an ordinal preference, namely for a utility function u we can define the complete weak order $\preceq_u$ given by $x \preceq_u y$ if and only if $u(x) \leq u(y)$.

The *explicit* representation of a preference structure consists of the data of all alternatives with their associated utilities (for cardinal preferences) or the whole relation $\preceq$ (for ordinal preferences). These representations have a spatial complexity in $O(|X|)$ for cardinal structures and $O(|X|^2)$ for ordinal structures, respectively.

In many real-world domains, the set of alternatives X is the set of assignments of a value to each of a given set of variables. In such cases, the alternatives are exponentially many. It is not reasonable to ask agents to report their preference in an explicit way when the set of alternatives is exponentially large, as this amounts to listing the exponentially many alternatives together with their utility assessment or their ranking. This is the case, in particular, when alternatives are allocations of resources (assignments of resources to agents). For this reason, the MARA project needs *languages for preference representation* aiming at enabling a *succinct* representation of the description of the problem, without having to enumerate a prohibitively large number of alternatives. Such preference representation languages often allow for a much more concise representation of the preference structure than an explicit enumeration.

In this section, we are going to give a brief survey of languages for preference representation. We begin by discussing several ways of representing compactly quantitative preferences (that is, utility functions), including languages specifically introduced for combinatorial auctions, and then we move on to languages for representing ordinal preferences.

[2] This is the decision-theoretic view of preferences, shared by many communities, from mathematical economics to multi-criteria decision making.

[3] Some work in preference modelling has also addressed non-transitive preference relations, arguing that humans often exhibit non-transitive preferences—for the sake of brevity we will omit this issue here.

4.1 Quantitative Preferences

Let $\mathcal{R} = \{r_1, \ldots, r_m\}$ be a set of indivisible resources. A quantitative preference structure for a resource allocation problem is a utility function $u : 2^{\mathcal{R}} \rightarrow Val$ mapping bundles of resources (subsets of $\mathcal{R}$) to numerical values (such as the reals). By defining utilities over bundles, we assume that the preferences of agents are free of so-called *allocative externalities*. That is, the value that an agent assigns to a bundle R does not depend on the allocation of the remaining resources amongst the other agents.

In the case of *task allocation* (as opposed to resource allocation), we may model the preferences of agents using *cost functions* rather than utility functions. At the level of abstraction being considered in the present survey, there is no effective difference in the representation of utility functions and cost functions. In the former case, agents would usually aim at maximising their utility, while in the latter case they would aim at minimising their costs.

Next we are going to review several languages for representing utility functions.

4.1.1 Bundle Enumeration

The most basic form of representing a utility function is to enumerate the bundles to which it assigns a non-zero value. That is, a utility function u can be presented as the set of pairs $\langle R, u(R)\rangle$ with R those bundles of resources for which $u(R) \neq 0$. We call this the *explicit form*, or the *bundle form*.

The bundle form is obviously *fully expressive* in the sense that any utility function may be so described. A serious drawback, however, is that the length of such descriptions will typically be exponential in the number of resources. This has prompted researchers to develop more succinct languages for utility representation.

4.1.2 The k-additive Form

For some (but not all) utilities it is possible to exploit *regularities* in the function structure in order to build succinct and efficiently computable descriptions. Given $k \in \mathbb{N}$, a utility function u is said to be *k-additive* if and only if there exists a coefficient α_T for each set of resources T of size at most k such that:

$$u(R) \quad = \quad \sum_{T \subseteq R} \alpha_T$$

The coefficient α_T represents the synergetic value of owning all of the items in T together, beyond the utility associated with any of its proper subsets. If a utility function is presented in terms of such coefficients, then we say that it is given in *k-additive form*.

The k-additive form is also fully expressive, but only in the sense that it can describe any utility function provided k is chosen large enough (for any k less than the overall number of resources there are functions that cannot be represented). It is typically considerably more succinct than the simple bundles form (think of a function mapping bundles to the number of items in a bundle), although there are also counterexamples (such as functions mapping only bundles with a single element to a non-zero utility value) [18].

In many application domains, it will be reasonable to assume that utility functions are k-additive with a relatively small value of k (which would allow for a very succinct representation). Indeed, the larger a bundle of resources, the more difficult for an agent to estimate the additional benefit incurred by owning all the resources in that bundle *together* (*i.e.* beyond the benefit incurred by the relevant subsets).

The k-additive form of representing utility functions is inspired by work in fuzzy measure theory [47]. It has been introduced into the MARA domain by Chevaleyre et al. [18] and, independently and in a combinatorial auction setting, by Conitzer et al. [23].

4.1.3 Weighted Propositional Formulas

Many languages for compact preference representation make an explicit use of *logic* (for a survey of such languages we refer to the work of Lang [57]). The basic idea of logic-based preference representation for MARA is that each resource r can be identified with a propositional variable p_r, which is *true* if the agent whose preferences we are modelling owns the corresponding resource, and *false* otherwise.[4] That is, every bundle R corresponds to a model. Agents can then express their preferences in terms of propositional formulas (or *goals*) that they want to be satisfied. We write $R \models G$ to express that the goal G is satisfied in the model corresponding to the bundle R.

The simplest (and prototypical) logical representation of preferences simply consists of giving a single propositional formula G (representing the agent's goal). The utility function u_G generated by G is extremely basic: $u_G(R) = 1$ if $R \models G$, $u_G(R) = 0$ if $R \models \neg G$. One possible refinement of this consists of considering a *goal base* $GB = \{G_1, \ldots, G_n\}$ and counting the number of goals satisfied by R.

In a further refinement, goals are associated with numerical weights, which tell how important the satisfaction of the goal is considered to be. Formally, the preferences of an agent are expressed by means of a finite set of such weighed goals: $GB = \{\langle G_1, \alpha_1\rangle, \ldots, \langle G_n, \alpha_n\rangle\}$, where each α_i is an integer and each G_i is a propositional formula. For every bundle R, we define the *penalty* of R as follows:

$$p_{GB}(R) \quad = \quad \sum\{\alpha_i \mid R \not\models G_i\} \qquad (1)$$

The penalty of R can be viewed as its disutility, that is, $u_{GB}(R) = -p_{GB}(R)$. Many other operators can be used, in place of the sum, for aggregating weights of violated (or symmetrically, satisfied) formulas [56].

[4] In a multi-unit setting (see Section 3.5), we would have to consider atomic sentences such as $x \geq 50$, signifying a bundle with at least 50 units of type x.

4.1.4 Straight-line Programs

A further representation form for utility functions is based on *straight-line programs* (SLPs). SLPs may be viewed as directed acyclic graphs consisting of two distinguished types of vertex: *inputs* which are sources (have in-degree 0) and *gates*, each of which has in-degree exactly 2. A subset of the gates (with out-degree 0) are distinguished as the program *outputs*. In addition to the graph structure an SLP is fully defined by associating a binary Boolean operation with each gate vertex. For an SLP, C, with m inputs, ordered as $\langle x_1, x_2, \ldots, x_m\rangle$ and p gates, a *topological labelling* of the vertices assigns a unique integer in the range $[1, m+p]$, $\lambda(v)$, to each vertex of C in such a way that: $\lambda(x_i) = i$; if v is a gate with $\langle w_1, v\rangle$ and $\langle w_2, v\rangle$ edges in C then $\lambda(v) > \max\{\lambda(w_1), \lambda(w_2)\}$. A topological labelling may be efficiently computed for C using depth-first search.

An SLP, C with inputs $\langle x_1, x_2, \ldots, x_m\rangle$ and p gates, t of which are outputs labelled $\langle s_0, s_1, \ldots, s_{t-1}\rangle$ computes its result by executing the program consisting of exactly $m+p$ lines, at each of which a single bit value ($res(i)$) is computed. Given an instantiation of the inputs $\langle \alpha_1, \ldots \alpha_m\rangle$, the ith line, l_i computes: $res(i) := \alpha_i$ if $1 \leq i \leq m$; and $res(j_1)\theta_i res(j_2)$ if $m+1 \leq i \leq m+p$, where θ_i is the binary operation associated with the ith gate and whose inputs are the vertices labelled j_1 and j_2. The *numerical* value computed by C as a consequence of a particular instantiation $\underline{\alpha}$ of its inputs is $val(C, \underline{\alpha}) = \sum_{i=0}^{t-1} res(s_i)\cdot 2^i$.

This model provides an alternative representation for utility functions, $u : 2^{\mathcal{R}} \rightarrow \mathbb{N}$ by a suitable SLP, C: a subset S defines an instantiation of the inputs via its m-bit characteristic vector $\alpha(S)$; the value $u(S)$ is then simply $val(C, \alpha(S))$. It is noted that, although this definition uses $\mathbb{N}$ as the range, it is a trivial matter to extend to $\mathbb{Z}$ (allow an additional output to act as a *sign* bit) and to $\mathbb{Q}$ (interpret the output bits as two groups, one defining the numerator, the other the denominator). As with the bundle form, the SLP form has the property of being fully expressive. In addition, however, there are the following advantages:

- The number of bits needed to encode utility functions can be exponentially smaller than that required in the bundle form.
- If the function $u : 2^{\mathcal{R}_m} \rightarrow \mathbb{Q}$ is computable by a deterministic Turing Machine in time T, then u may be represented by an SLP, C containing $O(T \log T)$ lines.

The first of these is easily seen by considering the function with value 1 if $|S|$ is odd, and value 0 if $|S|$ is even: the number of bundles to be listed is exactly 2^{m-1}. The same function, however, is described by the program with $2m-1$ lines corresponding to the computation $\oplus_{i=1}^{m} x_i$. The second property is a consequence of the constructions presented by Schnorr [85] and Fischer and Pippenger [39]. These simulations are effective (*i.e.* not simply existence arguments) and can be efficiently implemented.

In principle, other "program-based" formalisms can be defined, however, in order to be effective it must be possible efficiently to validate that a given bit-string does describe a syntactically correct program *and* to have an effective method of determining the program output. For the SLP approach above, both of these are satisfied, the latter since the runtime of a given SLP is exactly the number of program lines contained within it.

Extensive complexity-theoretic treatments of the SLP model (described in its usual terminology of *combinational logic networks*) may be found in the monographs of Savage [84], Wegener [96] and Dunne [28]. In the context of MARA, the SLP form has been considered by Dunne et al. [32].

4.1.5 Bidding Languages

Bidding languages are used in combinatorial auctions to allow agents to communicate their preferences to the auctioneer.[5] Preferences structures here are *valuation functions* or, equivalently, positive and monotonic utility functions on $2^{\mathcal{R}}$.

Bids are expressed as combinations of atomic bids of the form $\langle R, p\rangle$, where p is the amount the bidder is prepared to pay for the bundle R. Two prominent bidding languages are the OR and the XOR languages:

- The OR language is probably the most widely used bidding language. Here the valuation of a bundle is taken to be the maximal value that can be obtained when computing the sum over *disjoint* bids for subsets of the bundle. For instance, a bid of the form

  $$\langle\{a\}, 2\rangle \text{ OR } \langle\{b\}, 2\rangle \text{ OR } \langle\{c\}, 1\rangle \text{ OR } \langle\{a, b\}, 5\rangle$$

  expresses that the bidder is willing to pay 2 for a alone, 2 for b alone, 5 for both a and b, and 6 for the full set. Clearly, this language is not fully expressive since it cannot represent subadditive utility functions (for example, there is no way to specify that you would only be prepared to pay 4 for the full set).
- In the XOR language [80], atomic bids are assumed to be mutually exclusive. In this case, the valuation of a bundle is simply the highest value offered for any of its subsets. The XOR language can express any (normalised) monotonic utility function.

While the XOR language is more expressive than the OR language, it can also prove to be far less compact for certain types of preferences. For instance, the utility function $u(R) = |R|$ requires an exponential number of atomic bids in the XOR language, but only a linear number of OR bids.

Because the OR language is widely considered a simple and natural bidding language, there have been several attempts to extend its expressiveness without requiring an

[5] Of course, strategic considerations may cause agents not to report their *true* preferences, but this issue is not relevant from the viewpoint of preference *representation*.

exhaustive listing of XOR bids. It is, for instance, possible to combine the types of bids, and to thus to obtain OR-of-XOR and XOR-of-OR bidding languages. For an extensive discussion of such languages we refer to the review article by Nisan [68].

An interesting alternative is to simulate XOR bids by means of OR bids. The idea is simply to introduce "fake" resources (or phantom goods, or dummy items), that have no function other than making bundles mutually exclusive, because the resource appears in both bundles [41]. For instance, if one wanted to express that the set $\{a, b, c\}$ should be valued at 4, it would be possible to add the fake resource f to obtain both $\langle\{c, f\}, 1\rangle$ and $\langle\{a, b, f\}, 5\rangle$, and to bid in addition on $\langle\{a, b, c\}, 4\rangle$. This language, known as the OR$*$ bidding language (or OR with dummy items), is as expressive as the XOR language.

4.2 Ordinal Preferences

Next we are going to discuss the representation of ordinal preferences. Again, let $\mathcal{R} = \{r_1, \ldots, r_m\}$ be a set of indivisible resources. An ordinal preference structure $\preceq$ is a binary relation over $2^{\mathcal{R}}$. Here, logic-based languages play a central role (see also our discussion of weighted propositional formulas in Section 4.1.3).

4.2.1 Prioritised Goals

Prioritised goals are the ordinal counterpart to weighted goals: instead of numerical weights attached to goals (expressed as propositional formulas), we have a priority relation on goals, from which a preference relation on the set of bundles can be drawn.

While some approaches make use of partial priority preorders, most of them make the assumption that the priority relation is complete. When this is the case, then priorities on formulas can be expressed by a function r from integers to integers. A goal base is then a finite set of formulas with an associated function: $GB = \langle\{G_1, \ldots, G_n\}, r\rangle$. If $r(i) = j$, then j is called the rank of the formula G_i. By convention, a lower rank means a higher priority. The question is now how to extend the priority on goals to a preference relation over alternatives. The following three choices are the most frequent ones:[6]

- *best-out ordering* [5]: $R \preceq^{bo}_{GB} R'$ iff $\min\{r(i) \mid R \not\models G_i\} \leq \min\{r(i) \mid R' \not\models G_i\}$
- *discrimin ordering* [5, 14, 43]: Let $d(R, R') = \min\{r(i) \mid R \not\models G_i \ \&\ R' \models G_i\}$. $R \preceq^{dis}_{GB} R'$ iff $d(R, R') < d(R', R)$ or $\{G_i \mid R \models G_i\} = \{G_i \mid R' \models G_i\}$
- *leximin ordering* [5, 58]:[7] Let $d_k(R) = |\{G_i \mid R \models G_i \ \&\ r(i) = k\}|$.
  $R \prec^{lex}_{GB} R'$ iff there exists a k such that $d_k(R) < d_k(R')$ and $\forall j < k$, $d_j(R) = d_j(R')$
  $R \preceq^{lex}_{GB} R'$ iff $R \prec^{lex}_{GB} R'$ or $\forall j$, $d_j(R) = d_j(R')$

Note that $\preceq^{lex}_{GB}$ and $\preceq^{bo}_{GB}$ are complete preference relations while $\preceq^{dis}_{GB}$ is generally not. We moreover have the following chain of implications: $R \prec^{bo}_{GB} R'$ entails $R \prec^{dis}_{GB} R'$ entails $R \prec^{lex}_{GB} R'$.

4.2.2 Ceteris Paribus Preferences

In this language, preferences are expressed in terms of statements like: "all other things being equal, I prefer these alternatives over those other ones." Formally, let C, G and G' be three propositional formulas and V a set of propositional variables including those occurring in G and G'. The *ceteris paribus* desire $C : G > G'[V]$ means: "when C is true, all irrelevant things being equal, I prefer $G \wedge \neg G'$ to $\neg G \wedge G'$", where the "irrelevant things" are the variables that are not in V. The preference relation induced by a set of such preference statements is then the transitive closure of the union of preference relations induced by individual preference statements. This language can also be extended so as to allow for indifference statements.

An important sublanguage of *ceteris paribus* preferences is the language of (binary) *CP-nets* [9], which is obtained by imposing the following syntactical restrictions:

- Goals G and G' are literals speaking about the same propositional variable.
- The variables mentioned in the context C of a preference statement about variable p must belong to a fixed set, called the *parents* of p.
- For each variable p and each possible assignment π of the parents of p, there is one and only one preference statement $C : p > \neg p$ or $C : \neg p > p$ such that $\pi \models C$.

Various extensions of CP-nets have been proposed so as to be more expressive. For instance, TCP-nets [12] are CP-nets with a dominance relation between variables. Languages for *cardinal* preference representation in the style of CP-nets have been defined as well, for instance UCP-nets [8], which are based on generalised additive independence.

4.3 Discussion

At least five very important issues should be addressed when investigating preference representation languages:

- *Elicitation:* How hard is is to elicit preference from an agent so as to obtain a statement expressed in a given preference language L?
- *Cognitive relevance:* How close is a given language L to the way in which humans would express their preferences?

[6] We are using the convention $\min(\emptyset) = +\infty$.

[7] Not to be confused with (although related to) the *leximin* ordering for the aggregation of individual preferences in a society of agents presented in Section 5.4.

- *Expressive power:* Given a representation language L, a relevant question is whether L can express all preorders and/or all utility functions, or only complete preorders, or only a strict subclass of them, etc.
- *Computational complexity:* For a given language L, what is the computational complexity of comparing two alternatives, of deciding whether a given alternative is optimal, or of finding an optimal alternative?
- *Comparative succinctness:* Given two languages L and L', determine whether every preference structure that can be expressed in L can also be expressed in L' without a significant (that is, supra-polynomial) increase in size (in which case L' is said to be at least as succinct as L).

A detailed discussion of these issues in view of all the different representation languages we have covered would be beyond the scope of this survey. We limit ourselves to a few indicative remarks.

With the exception of bidding languages, all the languages for quantitative preferences presented above are fully expressive and we have already discussed several examples of comparative succinctness results for such languages. A problem with quantitative preferences in general is the well-known difficulty of eliciting numerical preferences from agents. *Ceteris paribus* preferences, being rather close to human intuition and comparatively easy to elicit, are interesting from a cognitive point of view. However, they have a high computational complexity in the general case, and furthermore, they generally leave many pairs of alternatives incomparable. As for prioritised goals, their lack of expressive power (no compensation allowed between goals) somewhat limits their range of use.

5 Social Welfare

A typical objective in MARA is to find an allocation that is optimal with respect to a metric that depends, in one way or another, on the preferences of the individual agents in the system. The aggregation of individual preferences can often be modelled using the notion of *social welfare* as studied in Welfare Economics and Social Choice Theory. This view is in line with the widely used metaphor of multiagent systems as "societies of agents". For instance, assuming that individual agents model their preferences using utility functions mapping bundles of resources to numerical values, the concept of *utilitarian social welfare*, defined as the sum of individual utilities, can be used to measure the quality of an allocation from the viewpoint of the system as a whole. This is probably the most widely used interpretation of the term "social welfare" in the multiagent systems literature [79, 97].

In Welfare Economics and Social Choice Theory, on the other hand, many different *notions of social welfare* and related concepts have been studied [2, 65, 86] and many of these are also applicable to MARA systems [33]. In the context of an e-commerce application, our aim may be to maximise the average profit generated by the negotiating agents. In this case, utilitarian social welfare provides a suitable metric for assessing system performance. In an application such as that introduced in Section 2.2, where agents need to agree on the access to an Earth Observation Satellite that has been jointly funded by the owners of these agents, on the other hand, it is important that each agent receives a *fair* share of the common resource (possibly reflecting the size of the financial contribution made by its owner). In this case, average utility is clearly not a good indicator of performance.

Generally speaking, before sending a software agent into a system to negotiate on our behalf, we would like to know under what (social) rules that system operates. If these rules are not satisfactory, we may not be prepared to agree to be bound by the outcome of a negotiation.

In this section, we are going to review some of the notions of social welfare proposed in the literature on Welfare Economics and Social Choice Theory that are relevant to MARA. More specifically, we are going to present and discuss different approaches to defining a *social welfare ordering*, *i.e.* a mapping from the preferences of the agents in a society to the "preferences" of society as a whole. Good references in this area are the *Handbook of Social Choice and Welfare*, edited by Arrow, Sen and Suzumura [2], and the textbook by Moulin [65]. We are going to cover preference aggregation mechanisms for both ordinal and cardinal agent preferences (utility functions). Given that every utility function also induces an ordinal preference relation, any concept defined for ordinal preferences also extends to the cardinal case.

5.1 Notation

Let $\mathcal{A} = \{1, \ldots, n\}$ be a set of agents. Depending on whether we assume cardinal or ordinal preference structures, each of these agents i is equipped with either a utility function u_i or a preference relation $\preceq_i$. An allocation P is a mapping from agents to bundles of resources; that is, $P(i)$ is the bundle held by agent i in allocation P.

Our presentation is independent from the exact nature of the resources used (divisible or not, sharable or not, etc.). In most cases, we only assume that agents have preferences over alternative *allocations* (only in the case of envy-freeness, discussed in Section 5.7, we need to assume that agents have preferences over alternative *bundles*). For instance, $P \preceq_i Q$ states that agent i likes allocation P no more than allocation Q. Despite such generality, it makes sense to think of preferences as being defined over bundles of resources (as discussed in Section 4), *i.e.* to assume that there are no allocative externalities. That is, $P \preceq_i Q$ may be considered an abbreviation for $P(i) \preceq_i Q(i)$ and $u_i(P)$ is short for $u_i(P(i))$.

5.2 Pareto Optimality

An allocation P is *Pareto-dominated* by another allocation Q if and only if the following hold:

- $P \preceq_i Q$ for all agents $i \in \mathcal{A}$; and
- $P \prec_i Q$ for at least one agent $i \in \mathcal{A}$.

An allocation is *Pareto optimal* (or Pareto efficient) if and only if it is not Pareto-dominated by any other allocation. That is, an allocation is Pareto optimal if and only if it is not possible to (strictly) improve the individual welfare of an agent without making any of the others worse off.

Pareto optimality is generally regarded as the most fundamental criterion for efficiency. Note that the concept of Pareto optimality is purely ordinal: It does not require preferences to be numerical, not even interpersonally comparable. Also observe that the notion of Pareto dominance only gives rise to a partial (rather than a complete) ordering over alternative allocations.

5.3 Collective Utility Functions

If individual agents use utility functions to represent their preferences, then every allocation P gives rise to a utility vector $\langle u_1(P), \ldots, u_n(P)\rangle$. A *collective utility function* (CUF) is a mapping from such vectors to numerical values (e.g. the reals). Given that every allocation P determines a utility vector, a CUF may also be regarded as a function from allocations P to numerical values. Every CUF sw induces a *social welfare ordering:* The allocation Q is socially preferred over allocation P if and only if $sw(P) \leq sw(Q)$.

In the sequel, we list several examples for such CUFs and indicate the kind of MARA applications where they may be useful.

5.3.1 Utilitarian Social Welfare

The *utilitarian social welfare* is defined as the sum of individual utilities:

$$sw_u(P) \quad = \quad \sum_{i\in\mathcal{A}} u_i(P) \tag{2}$$

The utilitarian CUF is independent of the zeros of individual utilities. It can provide a suitable metric for overall (as well as average) profit in a range of e-commerce applications.

5.3.2 Egalitarian Social Welfare

The *egalitarian social welfare* is given by the utility of the agent that is currently worst off:

$$sw_e(P) \quad = \quad \min\{u_i(P) \mid i \in \mathcal{A}\} \tag{3}$$

This CUF offers a level of *fairness* and may be a suitable performance indicator when we have to satisfy the minimum needs of a large number of customers. Fair division [13, 66, 101] is an important area with many potential applications in the field of MARA.

5.3.3 Nash Product

The *Nash product* is defined as the product of individual utilities:

$$sw_N(P) \quad = \quad \prod_{i\in\mathcal{A}} u_i(P) \tag{4}$$

This notion of social welfare favours both increases in overall utility and inequality-reducing redistributions. In this sense, it may be regarded as a good compromise between the utilitarian and the egalitarian agendas. Another interesting aspect of this CUF is that it is independent of the individual scales of agent utility functions.

Observe that the Nash product can only provide a meaningful metric of social welfare if all individual utilities are non-negative (or better even, if they are all positive).

5.3.4 Elitist Social Welfare

The *elitist social welfare* is given by the utility of the agent that is currently best off:

$$sw_{el}(P) \quad = \quad \max\{u_i(P) \mid i \in \mathcal{A}\} \tag{5}$$

The elitist CUF is clearly *not* a fair measure for social welfare, but it can be useful in cooperation-based applications where we require only one agent to achieve its goals.

5.3.5 Rank Dictators

The egalitarian and the elitist CUFs are both representatives of the family of k*-rank dictator* CUFs, which we are going to define next. Let $(v_P^{\uparrow})_k$ denote the kth smallest utility assigned to allocation P by any of the agents in $\mathcal{A}$ (this is the kth coordinate in the *ordered utility vector* for allocation P; see also Section 5.4). Then the k-rank dictator CUF sw_k is defined as follows:

$$sw_k(P) \quad = \quad (v_P^{\uparrow})_k \tag{6}$$

A special case of particular interest is the *median rank dictator* CUF which is defined as sw_k with $k = \frac{n}{2}$ in case n is even and $k = \frac{n+1}{2}$ in case n is odd. Indeed, for certain applications the individual level of welfare on an agent that does at least as well as half of the agents in the system but not better than the other half may be considered as suitable indicator for overall system performance.

5.4 The *leximin* Ordering

The *leximin* ordering is a social welfare ordering that refines egalitarian social welfare. It works by comparing first the utilities of the least satisfied agents, and in case these utilities coincide, compares the utilities of the next least satisfied agents, and so on. This idea is formalised as follows.

Suppose agents use utility functions to express their preferences. Then every allocation P gives rise to an *ordered utility vector* $v_P^{\uparrow}$, which is the result of first computing

$u_i(P)$ for every agent $i \in \mathcal{A}$ and then arranging these values in ascending order. For example, $v_P^{\uparrow} = \langle 3, 5, 20 \rangle$ means that the agent worst off enjoys utility 3, the one best off utility 20, and the third one utility 5.

Then Q is *leximin*-preferred to P if and only if there exists an integer $k \in \{1, \ldots, n\}$ such that:

- $(v_P^{\uparrow})_i = (v_Q^{\uparrow})_i$ for all $i < k$; and
- $(v_P^{\uparrow})_k < (v_Q^{\uparrow})_k$.

In other words, the *leximin* ordering is the lexicographic ordering over ordered utility vectors. It favours the reduction of inequalities between agents. An allocation is *leximin-optimal* if and only if it is not *leximin*-preferred by any other allocation.

5.5 Generalisations

It is possible to build families of parametrised CUFs able to induce a continuous collection of social welfare orderings, including most of those defined above. Let us describe briefly two such families. The first one is defined by the following additive CUF [65]:

$$sw_{(p)}(P) \quad = \quad \sum_{i \in \mathcal{A}} g_{(p)}(u_i(P)) \tag{7}$$

The parameter p is a real number, $p \neq 0$, and $g_{(p)}(x) = \mathrm{sgn}(p) \cdot x^p$ (where $\mathrm{sgn}(p) = 1$ if $p > 0$ and $\mathrm{sgn}(p) = -1$ if $p < 0$), with the convention $g_{(0)}(u) = \log u$. Obviously, $sw_{(1)}$ measures utilitarian social welfare, and $sw_{(0)}$ induces the same social welfare ordering as the Nash product. The *leximin* ordering is the limit of the social welfare ordering induced by $sw_{(p)}$ as p goes to $-\infty$.

The other family of CUFs is a particular case of what is known as *ordered weighted averaging* (OWA) operators [99]. With the notation introduced above, let us define:

$$sw_w(P) \quad = \quad \sum_{i \in \mathcal{A}} w_i \cdot (v_P^{\uparrow})_i \tag{8}$$

Here, $w = (w_1, w_2, \ldots, w_n)$ is a vector of real numbers. Let us consider the vector w such that $w_i = 0$ for all $i \neq k$ and $w_k = 1$, then we have exactly the k-rank dictator CUF (including the egalitarian and the elitist CUFs, which are special cases of rank dictators). Consider now the vector w such that $w_i = \alpha^{i-1}$, with $\alpha > 0$, then the case $\alpha = 1$ corresponds to the utilitarian CUF, and the *leximin* ordering is the limit of the social welfare ordering induced by sw_w as α goes to 0.

5.6 Normalised Utility

It can often be necessary to *normalise* utility functions before aggregating individual preferences using any of the methods presented here, because many of them require individual utilities to be intercomparable. For instance, if P_0 is the initial allocation of resources, then we may restrict our attention to allocations P that Pareto-dominate P_0 and use the utility gains $u_i(P) - u_i(P_0)$ rather than the utilities $u_i(P)$ themselves as input to either a collective utility function or the *leximin* ordering.

A further normalisation step would be to evaluate an agent's utility gains *relative* to the gains it could expect in the best possible case. More precisely, let us define the maximum individual utility for each agent as:

$$\widehat{u}_i \quad = \quad \max\{u_i(P) \,|\, P \in Adm\} \tag{9}$$

Here, Adm is the set of admissible allocations. That is, $\widehat{u}_i$ is the utility that agent i could enjoy if it were the sole agent exploiting the available resources. Then we define the normalised individual utility of an agent i as follows:

$$u'_i(P) \quad = \quad \frac{u_i(P)}{\widehat{u}_i} \tag{10}$$

Observe that $\max\{u'_i(P) \,|\, P \in Adm\} = 1$, for all agents i. In other words, the maximum normalised utility is the same for all agents.

The optimum of the *leximin* ordering with respect to normalised utilities is known as the *Kalai-Smorodinsky* solution [66].

5.7 Envy-freeness

An allocation is *envy-free* if and only if each agent is at least as happy with its share than it would be with any of the bundles allocated to one of the other agents [13]. That is, an allocation P is envy-free if and only if $P(j) \preceq_i P(i)$ holds for all agents i and j. Envy-freeness is a property that does not require the intercomparability of the utilities of different agents.

If we require all items to be allocated, then an envy-free allocation does not always exist (consider, say, a an allocation problem with a single resource that is desired by all agents in the system). But even when not all items need to be allocated, it is well-known that there are allocation problems for which there exists no allocation that is both Pareto optimal and envy-free. One could therefore aim at finding (Pareto optimal) allocations that would, at least, minimise the overall "degree of envy" as much as possible. There are several candidate definitions for *minimal envy*. Two possible approaches would be the following:

- Minimise the number of envious agents.
- Minimise the average degree of envy (the distance to the most envied competitor) of all envious agents.

5.8 Example

To exemplify some of the concepts introduced in this section, consider a scenario with two agents, 1 and 2, and a set of three resources $\{a, b, c\}$ that are indivisible and cannot

be shared. Suppose the preferences of the two agents are represented by the utility functions u_1 and u_2:

$$\begin{array}{lll} u_1(\{a\}) = 18 & u_1(\{b\}) = 12 & u_1(\{c\}) = 8 \\ u_2(\{a\}) = 15 & u_2(\{b\}) = 8 & u_2(\{c\}) = 12 \end{array}$$

Furthermore, suppose u_1 and u_2 are *additive*, *i.e.* $u_i(R) = \sum_{r \in R} u_i(\{r\})$, and thereby fully specified by the above values. Let P be the allocation giving a to 1 and b and c to 2.

Allocation P has maximal egalitarian social welfare (18). Utilitarian social welfare, on the other hand, is not maximal for this allocation (38 rather than 42) , and neither is elitist social welfare (20 rather than 38).

P is Pareto optimal as well as *leximin*-optimal, but not envy-free, since agent 1 would be happier with the share of 2 than with its own. In fact, there is no allocation that would be both Pareto and and envy-free for this problem. On the other hand, for the slightly different problem where $u_1(\{a\}) = 20$ instead of 18 (leaving the rest unchanged), allocation P would be both Pareto optimal and envy-free.

5.9 Welfare Engineering

The insight that very different notions of social welfare may be appropriate for different applications of MARA has provided the impetus for the development of the *Welfare Engineering* framework [19, 33], which addresses two issues:

- the systematic choice of suitable social welfare orderings for a given application of MARA (and possibly the application-driven design of new orderings); and
- the design of appropriate rationality criteria and social interaction mechanisms for negotiating agents in view of different notions of social welfare.

By "appropriate" we mean criteria and mechanisms that ensure the convergence of the negotiation process to an allocation that is optimal with respect to the chosen social criterion (see also Section 6.4). Of course, depending on the application in question, such criteria need to be balanced with the autonomy requirements of individual agents.

An example for the first aspect of Welfare Engineering would be the *elitist* collective utility function discussed earlier, which seems unethical for human society, but it may be just the right performance indicator for a distributed computing application where several agents are working towards their own goals, but the system designer is only interested in (at least) one of them achieving their objective as quickly as possible. This aspect of Welfare Engineering may be characterised as "welfare economics for *artificial* agent societies".

An example for the second aspect would be the following convergence result: To achieve Pareto optimal outcomes in negotiation without monetary side payments, ask agents to negotiate mutually beneficial deals involving any number of agents or resources, but also to participate in deals that do at least not lower their own level of utility [35]. This aspect of Welfare Engineering can be summarised as "*inverse* welfare economics", alluding to the characterisation of mechanism design as "inverse game theory" [70].

6 Allocation Procedures

Generally speaking, the allocation procedure used to find a suitable allocation of resources could be either *centralised* or *distributed*. In the centralised case, a single entity decides on the final allocation of resources amongst agent, possibly after having elicited the preferences of the other agents in the system. Typical examples for the centralised approach are combinatorial auctions [24]. Here the central entity is the auctioneer and the reporting of preferences takes the form of bidding. In truly distributed approaches, on the other hand, allocations emerge as the result of a sequence of local negotiation steps. Such local negotiation is often restricted to bilateral trading as in the classical *Contract-Net* approach [87], but systems allowing for multilateral exchanges of resources between more than two agents are also possible.

A comprehensive survey on allocation procedures for MARA would be beyond the scope of this paper. Any such survey would have to address at least the following three issues:

- *Protocols:* At this level, we need to address ontological issues (what types of deals are possible?) and devise communication protocols accordingly (what messages do agents have to exchange to agree on one such deal?).
- *Strategies:* When designing individual agents, we need to devise strategies for agents that allow them to best exploit a given negotiation protocol. This can also provide feedback to the first level: Where possible, protocols should be designed in such a way that they provide incentives to the negotiating agents to adopt a particular desirable profile of behaviour (mechanism design).
- *Algorithms:* At this level, we need to provide algorithms to solve the computational problems faced by agents when engaged in negotiation. This includes both algorithms to decide how to respond to a proposal in a distributed negotiation scenario and winner determination algorithms for combinatorial auctions. Again, this level may provide feedback to the other two levels: If a particular computational problem proves too hard to be solved in a reasonable amount of time then this may call for a simplification of the negotiation protocol (or strategy).

In this paper, we concentrate on the first of these issues. The most fundamental question to consider before devising a protocol for a MARA system is whether to adopt a

centralised or a distributed design. We therefore start with a short discussion of the respective merits and drawbacks of centralised and distributed approaches to MARA. This is followed by an introduction to protocols for combinatorial auctions and an overview of the Contract-Net and related protocols for distributed resource allocation. Finally, we make a connection to our discussion of social welfare measures in Section 5 and review a number of results concerning the convergence to a socially optimal allocation for different protocols in the distributed setting.

6.1 Centralised vs. Distributed

Both the centralised and the distributed approach to MARA have their advantages and disadvantages. Possibly the most important argument in favour of auction-based mechanisms concerns the simplicity of the communication protocols required to implement such mechanisms. Another reason for the popularity of centralised mechanisms is the recent push in the design of powerful algorithms for combinatorial auctions that, for the first time, perform reasonably well in practice [41, 80]. Of course, such techniques are, in principle, also applicable in the distributed case, but research in this area has not yet reached the same level of maturity as for combinatorial auctions. An important argument *against* centralised approaches is that it may be difficult to find an agent that could assume the role of an "auctioneer" (for instance, in view of its computational capabilities or in view of its trustworthiness).

The distributed model seems also more natural in cases where finding optimal allocations may be (computationally) infeasible, but even small improvements over the initial allocation of resources would be considered a success. Step-wise improvements over the *status quo* are naturally modelled in a distributed negotiation framework.

6.2 Auction Protocols

Auctions [24, 54, 55, 94, 98] are centralised mechanisms for the allocation of goods amongst several agents. Agents report their preferences and wait for the final allocation to be made by the auctioneer (whether there is an initial allocation of goods, as in combinatorial exchanges, or not, as in regular combinatorial auctions). The act of reporting preferences is called *bidding* and, naturally, agents are not required to reveal their true preferences during bidding, but they may submit whatever bid(s) they believe to best serve their own interests.

Bidding may be public (*open-cry*) as in the well-known English auction model or private (*sealed bids*). In the case of open-cry bidding, we can further distinguish between *ascending* bids (English auction) and *descending* bids (Dutch auction) [94]. In combinatorial domains, which is what we are interested in here (*i.e.* there are many goods and agents can submit bids for different combinations of goods), typically, most auction protocols foresee only a single round of bidding using sealed bids. The *bidding language* (see Section 4.1.5) determines what types of bids are admissible (and how to interpret them).

The auction protocol also specifies which agent would be awarded which goods, based on the bids received in time, and what price they should pay for the bundles allocated to them. In some cases, this decision can be left entirely to the auctioneer (who will seek to maximise her revenue). In other cases, it is important that the auctioneer follows the rules specified by the protocol, as these rules have been designed in such a way as to provide incentives to the bidders to bid truthfully. This is the case for the Vickrey auction model [94], and its extensions to combinatorial scenarios, where the winning agents pay less then the prices they specified in their bids.

For an extensive review of different auction models for resource allocation in combinatorial domains we refer to the forthcoming book on *Combinatorial Auctions*, edited by Cramton, Shoham and Steinberg [24], and the review article on the same topic by Kalagnanam and Parkes [54].

6.3 Negotiation Protocols

We now give a brief overview of some of the protocols developed for negotiation over resources in a distributed setting.

6.3.1 Contract-Net

Perhaps the most popular negotiation protocol is the *Contract-Net* protocol [87]. Although the protocol was primarily designed for *task* allocation, it is also perfectly suited to MARA. The protocol consists in four interaction phases, involving two roles (manager and bidder):

- *Announcement* phase: The manager advertises the resource to a number of partner agents (the bidders).
- *Bidding* phase: The bidders send their proposals to the manager.
- *Assignment* phase: The manager elects the best bid and assign the resource accordingly.
- *Confirmation* phase: The elected bidder confirms its intention to obtain the resource.

Any agent can initiate an interaction following the protocol by assuming the adequate role. The protocol is really a one-to-many protocol, leading to the assignment of a single task (or resource) to a single contractor (that is, the resulting deal is a one-to-one agreement regarding a single item).

6.3.2 Extensions

Many different extensions to this protocol have been proposed and we briefly review some of these here. The TRACONET system developed by Sandholm [77], for instance, uses a variant of the classical Contract-Net protocol to allow negotiation over the exchange of *bundles* of resources.

Golfarelli et al. [46] have proposed an extension where the bidders have no explicit mechanism for utility transfer (in other words, they cannot use money). The first phase remains the same as in the original Contract-Net: the manager announces a (bundle of) resource(s). But the protocol is based on exchanges: instead of bidding money, the agents will bid for one or more resources they are interested in exchanging. This extension allows agents to agree on *swapping* resources (rather than buying them from each other).

Sousa et al. [89] have designed a version of the Contract-Net protocol where bidders first propagate *constraints* between them in order to guarantee the coherence of different operations related to the same task.

6.3.3 Concurrent Contract-Net

As pointed out by Aknine et al. [1], when many managers negotiate simultaneously with many contractors, using the Contract Net protocol can lead to unsatisfactory results. In particular, because contractors are required to answer a single bid at a time, they may miss some contracts. To overcome this, they have proposed an extension to in which a pre-bidding and a pre-assignment phase are added before the final bidding and assignment phase of classical Contract-Net protocols. During the pre-bidding and pre-assignment phases, which can last a long time, agents propose temporary bids and managers temporarily accept (or reject) these bids. These new phases have several positive effects:

- After a deal has been temporarily accepted, if the manager receives a better offer, this deal can be turned into temporarily rejected offer. It turns out that when many negotiations are conducted simultaneously, by delaying the final acceptance, better deals (from the manager's point of view) may be negotiated.
- Contractors can modify their offers many times by making temporary offers. If the contractor receives a better new offer from another manager, it can modify its temporary bids before sending a definitive bid.
- The pre-bidding phase may be quite long. This has the positive effect of reducing the risk of decommitment.

An alternative way to tackle this latter problem is to allow agents to decommit, but to apply penalties when they do so. This route has been followed in the *levelled commitment* approach proposed by Sandholm and Lesser [82].

6.4 Convergence Properties

As discussed earlier, once a particular negotiation protocol has been fixed, we need to devise strategies for the agents using that protocol. Work in this area is often of a game-theoretical nature. A different line of research has analysed how the negotiation behaviour of individual agents affects the quality of the overall distribution of resources (with respect to some of the social welfare measures introduced in Section 5) by abstracting away from the details of individual negotiation strategies [35, 78].

For instance, a *rational* agent may be defined as an agent that will only agree to deals that result in a positive payoff for itself. That is, a set of rational agents will only agree on mutually beneficial deals. Which of the possibly many mutually beneficial deals agents will actually agree on depends on the concrete strategies they use, and overly aggressive negotiation strategies may even prevent agents from identifying any mutually beneficial deal at all [67]. However, in cases where it is admissible to assume that agents will agree on *some* deal meeting certain rationality criteria (such as resulting in a strictly positive payoff for everyone involved) whenever such a deal exists, it is sometimes possible to prove so-called *convergence properties* of a negotiation framework.

For instance, in the context of negotiation over finitely many indivisible resources, an important result, due to Sandholm [78], states that *any* sequence of deals that are mutually beneficial will eventually result in an allocation with maximal utilitarian social welfare, provided that agents can use monetary side payments to compensate their trading partners for otherwise disadvantageous deals (and each agent's payoff is linear in the amount of money received). That is, there can be no infinite sequence of mutually beneficial deals, and if agents keep on making such deals the system will converge to an allocation that maximises the sum of individual utilities. A similar result states that any sequence of mutually beneficial deals without side payments will converge to a Pareto optimal allocation [35].

An important caveat is that these results apply to negotiation settings where agents can agree on truly *multilateral* deals: A single deal may involve any number of agents (as well as any number of resources). Decomposing such a multilateral deal into a sequence of bilateral deals is not always possible, because some of the bilateral deals making up the overall deal may not be mutually beneficial to both agents. Hence, myopic agents that require a positive payoff for every single deal they take part in will not accept such a deal.

Given the difficulty of implementing such general deals, it is important to understand under what circumstances sequences of structurally simple deals suffice to guarantee convergence to a socially optimal allocation of resources. Recent results in this area show that mutually beneficial deals with side payments that involve only a single resource each (and thereby only two agents at a time) suffice to reach allocations with maximal utilitarian social welfare in case all agents use *modular* utility functions [35].[8] In fact, the class of modular utility functions is also *maximal* in the

[8] A utility function u is said to be *modular* if and only if we have $u(R_1 \cup R_2) = u(R_1) + u(R_2) - u(R_1 \cap R_2)$ for all bundles R_1 and R_2. This means that the utility assigned to a bundle of resource can be computed as the sum of the utilities of the individual resources in that bundle, *i.e.* the classes of modular and 1-additive functions coincide.

sense that for no class of functions strictly including that class it would still be possible to guarantee that agents using utility functions from this larger class and negotiating only mutually beneficial deals over single resources will eventually reach an allocation with maximal utilitarian social welfare in all cases [21]. Related work has also identified classes of utility functions (and ordinal preference relations) that guarantee the convergence to optimal allocations for sequences of deals involving at most k resources each [20].

7 Complexity Results

A growing body of work within the study of MARA considers various concepts of complexity, not only in the standard sense of *computational complexity theory* but also in terms of concepts such as *communication complexity*. Such work comprises both *positive* results—e.g. algorithms with provably efficient performance characteristics, properties of restricted classes of allocation settings, etc.—and a large collection of *negative* results that suggest many naturally arising decision and optimisation problems are unlikely to admit generally applicable algorithmic solutions. Within this section our aim is to review extant work that has addressed such questions and to catalogue related open problems.

7.1 Models and Assumptions

The structure we consider in the subsequent text will be referred to as a *resource allocation setting*, by which we mean a triple $\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle$ where:

- $\mathcal{A} = \{1, 2 \ldots, n\}$ is a set of n agents;
- $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$ is a collection of m resources; and
- $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ describes the utility function $u_i : 2^{\mathcal{R}} \rightarrow \mathbb{Q}$ for the agent $i \in \mathcal{A}$.

We assume that each $r \in \mathcal{R}$ is *indivisible* and *non-shareable*, *i.e.* at most one agent at a time will "own" r (see also Section 3). An allocation of the resources in $\mathcal{R}$ among the agents in $\mathcal{A}$ is a mapping $P : \mathcal{A} \rightarrow 2^{\mathcal{R}}$ with $P(i) \cap P(j) = \emptyset$ for any $i \neq j$. The set of all allocations of $\mathcal{R}$ among $\mathcal{A}$ will be denoted by $\Pi_{n,m}$. From the fact that there are n choices of agent for each of the m resources, it is easily seen that $|\Pi_{n,m}| = n^m$.

7.2 Computational vs. Communication Complexity

In very informal terms, traditional computational complexity theory is concerned with the issue of classifying computational problems with respect to how much of a particular computational resource is required for their solution. Typically, *computational problems* are phrased as *decision questions*, *i.e.* given an input instance I, is it the case that some property ϕ holds true of I? For example, given a directed graph $H(V, E)$ and a vertex s in V, is it the case that every vertex in V can be reached by some path that starts in s? The concept of *computational resource* is modelled via some formal *model of computation*. Thus, *time* (*space*) as the (worst-case) number of *moves* (*tape cells*) made by a (deterministic) 2-tape Turing machine (DTM) that correctly classifies input instances, *i.e.* accepts if $\phi(I) = \top$, rejects if $\phi(I) = \bot$. For further introductions to computational complexity theory we refer the reader to the textbook by Papadimitriou [69].

In the context of MARA problems, computational complexity results have tended to address what might be termed *"global"* properties of given resource allocation settings, e.g. whether allocations satisfying particular criteria exist. Recent work, however, has begun to address computational properties of abstract high-level *negotiation protocols* as reviewed in Section 6.4 above, e.g. given some constraint, χ, that allowed deals must satisfy, a number of decision problems may be formulated regarding allocations that are reachable from a starting allocation via sequences of χ-*deals*.

This view of complexity has not, in general, needed to be concerned with *"localised"* questions, e.g. the overheads involved in describing and implementing proposed deals; how many deals may be needed in order to reach an allocation with desirable properties, etc. In the work of Endriss and Maudet [34] the term *communication complexity*, deriving from the model put forward by Yao [100], is introduced to capture the combination of *number of deals* and *communication to agree a deal* that could be needed in order for an allocation to be finalised. While the bulk of the survey below is concerned with complexity issues from the perspective of computational complexity, we also discuss some results related to communication from the works of Endriss and Maudet [34] and Dunne [29], that consider upper and lower bounds on the *number of deals* needed in various contexts.

7.3 Allocations with Given Properties

Given a resource allocation setting, $\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle$, the agents concerned seek to bring about an allocation that will satisfy certain criteria. As discussed in Section 5, such criteria may be purely quantitative (e.g. the sum of the individual utility valuations (utilitarian social welfare) is maximal (or is above a given amount), but so-called qualitative properties (Pareto-optimal or envy-free outcomes, for instance) are also of interest.

7.3.1 Representation Issues

Standard computational complexity theory considers properties of algorithms implemented within some "well-defined" model of computation, e.g. Turing machines. In order sensibly to consider the performance of a specific al-

gorithm, this is reported as a function of the algorithm's *input length*. This convention presumes that, in comparing different algorithmic approaches to a particular problem, such comparisons are only "reasonable" if the representation of input instances is similar, or that (at worst) different formats can be translated between efficiently.

In considering how instances are to be represented in the case of decision problems concerning resource allocation settings, a significant issue that arises is the encoding of the collection of utility functions $\mathcal{U}$. The domain of a utility function is $2^{\mathcal{R}}$: thus (from the viewpoint of upper bounds on complexity) the characteristics of algorithms employing an enumerative form (listing all subset/value pairs) may not be comparable with algorithms employing some *compact* representation. We therefore give complexity results for the three different forms of representing utility functions discussed in Section 4.1: the bundle form, the SLP form, and the k-additive form (here the 2-additive form is of particular interest).

7.3.2 Quantitative Criteria

Two natural decision questions regarding the measure sw_u of utilitarian social welfare, have been considered with regard to each of the three formalisms for representing utility functions:

Welfare Optimisation (WO)	
Instance:	$\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle;\ K \in \mathbb{Q}$
Question:	$\exists\, P \in \Pi_{n,m} : sw_u(P) \geq K$?

Welfare Improvement (WI)	
Instance:	$\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle;\ P \in \Pi_{n,m}$
Question:	$\exists\, Q \in \Pi_{n,m} : sw_u(Q) > sw_u(P)$?

WI and WO are both NP-complete for the representation of utility functions in bundle form (reduction from SET PACKING [18]); for the SLP form (reduction from 3-SAT [32]); and for 2-additive functions (the simplest proof is via a reduction from MAX-2-SAT [18]). Both the 2-additive and SLP results apply even in systems containing only 2 agents; the SLP reduction shows that the problems remain NP-complete when (both) utility functions are monotonic.

7.3.3 Qualitative Criteria

The qualitative measures of Pareto optimality and envy-freeness give rise to the following decision problems:

Pareto Optimality (PO)	
Instance:	$\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle;\ P \in \Pi_{n,m}$
Question:	Is P Pareto optimal?

Envy-Freeness (EF)	
Instance:	$\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle$
Question:	$\exists\, P \in \Pi_{n,m} : P$ is envy-free?

Deciding PO is coNP-complete for both SLP and 2-additive utility functions. The former was shown by Dunne et al. [32] (reduction from 3-UNSAT restricted to instances with clause and variable numbers equal); the latter, although not explicitly stated by Chevaleyre et al. [18], is an immediate consequence of their proof that WI is NP-complete. Again both continue to hold in 2-agent contexts, with the SLP reduction also applying to monotonic utility functions.

EF is examined in a variety of cases in the work of Bouveret and Lang [11]. They consider a representation based on concise logic-based descriptions of agent preferences (as discussed in Section 4.1.3 above). In addition to the basic question of whether envy-free allocations are possible—shown to be NP-complete even within 2-agent settings—the question of allocations that combine envy-freeness with Pareto optimality is examined (termed *efficient envy-free*, or EEF, allocations). For such decision problems they demonstrate completeness results ranging from NP-complete up to Σ_2^p-complete, depending on the restrictions placed on the preference relations. That NP-completeness also holds for the question EF within the SLP model in 2-agent settings has been shown by Dunne [30] (reduction from 3-SAT).

7.4 Path and Convergence Properties

The collection of results referred to above, hold independently of the regime used to negotiate allocations. There are, however, a number of questions that arise specifically in the context of distributed negotiation when the structure of admissible deals is constrained. Thus suppose that only *individually rational* deals may be used, *i.e.* deals that are beneficial to all the agents involved. If monetary side payments are allowed, then individually rational deals are deals $\langle P, Q \rangle$ under which $sw_u(Q) > sw_u(P)$ [35]. As has been shown by Sandholm [78], if additional constraints, such as "all deals are bilateral and involve exactly one resource changing" (sometimes called the class of O-contracts), then there are cases where some rational deals cannot be implemented. A further problem that arises is that, even when there is a *rational O-contract path* to go from P to Q this may involve an agent repeatedly making deals involving the same resource, *i.e.* such paths may contain more than m distinct deals.

In general, given some predicate Φ on deals, the following decision problem arises:

Φ-Path	
Instance:	$\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle;\ P^{(s)}, P^{(t)} \in \Pi_{n,m}$ with $sw_u(P^{(t)}) > sw_u(P^{(s)})$
Question:	$\exists \Delta = \langle P^{(0)}, P^{(1)}, \ldots, P^{(r)} \rangle$ s.t. $P^{(0)} = P^{(s)}$ and $P^{(r)} = P^{(t)}$ and $\forall 1 \leq i \leq r, \Phi(P^{(i-1)}, P^{(i)})$?

Dunne et al. [32] consider the complexity of Φ-Path where $\Phi(P, Q)$ holds only if the deal is individually rational and

involves *at most* some given number k of the resources being passed from one agent to another. Within 2-agent settings using SLP representation it is shown that Φ-Path is NP-hard for all $k \leq \frac{m}{3}$ (and for the case $k = \frac{m}{2}$). In the special case of O-contracts (*i.e.* $k = 1$) NP-hardness holds with both utility functions being monotonic. Recent work, presented in Dunne and Chevaleyre [31], improves this NP-hardness lower bound and obtains an exact complexity classification: Φ-Path is PSPACE-complete for $\Phi(P, Q)$ holding if the deal is an individually rational O-contract.

Introducing the idea of a *maximal* Φ-path (from an initial allocation P) as a sequence of deals every one of which satisfies Φ and with the final allocation, $P^{(t)}$ being such that for every Q it is the case that $\neg\Phi(P^{(t)}, Q)$, leads to the following related problem:

Φ-Convergence	
Instance:	$\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle$
Question:	Is it the case that $\forall P \in \Pi_{n,m}$, for all maximal Φ-paths Δ starting from P, the allocation $last(\Delta)$ these terminate in, is one which maximises sw_u?

For instance, the basic convergence result first proved by Sandholm [78] (discussed in Section 6.4) shows that the answer to the above question is *always* "yes" when Φ does not restrict the range of admissible deals in any way. Φ-Convergence is the subject of ongoing work which has already established the following: For Φ corresponding to individually rational O-contracts, Φ-Convergence is coNP-complete for the SLP model and for 4-additive utility functions [31] Both results hold in 2-agent settings. If all utility functions are modular (*i.e.* 1-additive), then the answer to Φ-Convergence is always "yes" [21].

We now return to an issue relating to the ideas of *communication complexity* discussed above. The question of interest also has a bearing on establishing *upper bounds* on the complexity of Φ-Path. Given a resource allocation setting and Φ, consider the (rational) deals that *can* be implemented by Φ-paths. Dunne [29] has introduced the following measures:

- $L^{opt}(P, Q)$: the length of the *shortest* Φ-path realising $\langle P, Q \rangle$.
- $L^{\max}(\mathcal{A}, \mathcal{R}, \mathcal{U})$: the maximum value of $L^{opt}(P, Q)$ over those deals for which a Φ-path exists.
- $\rho^{\max}(n, m)$: The maximum value (taken over all choices of utility function) of $L^{\max}(\mathcal{A}, \mathcal{R}, \mathcal{U})$.
- $\rho^{\max}_C(n, m)$: As $\rho^{\max}$, but with the maximisation taken over utility functions belonging to some class C.

A related study (employing different terminology) is given by Endriss and Maudet [34], where attention is focussed on utility functions which allow any rational deal to be implemented via some sequence of rational O-contracts; the main case being considered in this respect is that of 1-additive functions.

Let $\Phi(P, Q)$ hold if and and only if the deal $\langle P, Q \rangle$ is an individually rational O-contract:

- $\rho^{\max}(n, m) \leq n^m - m(n-1)$ [78]
- $\rho^{\max}(n, m) \geq \frac{77}{256} 2^m - 1$ [29]
- $\rho^{\max}_{1-add}(n, m) = m$ [34]
- $\rho^{\max}_{mono}(n, m) \geq \frac{77}{128} 2^{\frac{m}{2}} - 3$ [29]

The latter two results pertain to the classes of 1-additive and monotonic utility functions, respectively. The constructed rational paths in the general and monotonic lower bound cases are unique.

7.5 Open Problems and Conjectures

Given the existing results concerning the measure sw_u wherein exact complexity classifications have been derived for each of the three representation styles for utility functions, the following conjectures seem plausible and ought to be straightforward to verify.

Conjecture 1 *Deciding if there is an allocation P with $sw_e(P) \geq K$ is NP-complete (whether $\mathcal{U}$ is given in bundle form, SLP form, or is 2-additive).*

Conjecture 2 *Given $K \in \mathbb{Q}$, deciding if $\max\{sw_u(P) \mid P \in \Pi_{n,m}\} = K$ is D^P-complete (again in all three representation formalisms).*

Conjecture 3 *EF is NP-complete for 2-additive utility functions.*

8 Simulation Platforms

Theoretical work in Microeconomics and Auction Theory provides a very strong foundation for analysing many resource allocation problems. However, on occasion we may be faced with a problem in which some of the assumptions underlying the theory are violated. This is especially the case in MARA scenarios where computational concerns are prominent [25]. For example, mechanism design as originally developed in Economics is not concerned with computational issues such as algorithmic or communication complexity. In a conventional auction design scenario issues such as the speed of winner determination and the communication costs of submitting bids are often not of significant concern since they are not typically a bottleneck with respect to the entire auction process which can involve protracted and lengthy decision making by human traders. However, in a market place run entirely by automated trading agents, such issues are likely to be of more concern since their performance costs can sometimes be of similar order of magnitude as the overall computational costs of running the auction. Once these costs are taken

into account many of the results in auction theory become somewhat brittle. For example, the revelation principle no longer applies when transaction-throughput and reduction in communication complexity are adopted as design goals [74].

In such cases experimental work using simulations of agent-based market places—*Agent-based Computational Economics* (ACE) [93]—can shed light on some of the grey areas that are difficult to analyse using existing theoretical tools.

As with any software engineering problem, in choosing an appropriate software framework in which to implement an ACE simulation it is important to consider the requirements that the software needs to meet. In this section, we give an overview of the typical requirements addressed by ACE software, and we then proceed to give an overview of some commonly-used simulation frameworks.

8.1 Simulation vs. Implementation

Software for *simulating* multiagent systems typically addresses different requirements from that designed to *implement* multiagent systems. Although it is natural to view a MAS implementation as its own simulation, there are a number of problems with such an approach, which we shall address in turn.

Firstly, ideally we would like the outcome of a simulation experiment to be exactly reproducible given the initial conditions of the experiment. This is not always possible in a MAS implementation since many environmental factors will be beyond the experimenter's control. For example, the precise outcome of an experiment may depend on the exact timing with which an agent responds to a particular message, and this time interval will depend on factors beyond the experimenter's control, such as the memory and CPU currently available to the agent.

Secondly, when we come to analyse the results of a simulation, we often need to generalise beyond a single run of an experiment with a single set of initial conditions. Typically, we generalise by taking many samples of free initial variables and running the experiment many times for each sample. Simulation frameworks are equipped to log data from the outcome of each experiment to a format suitable for analysis using statistical analysis software, such as MATLAB.

Thirdly, the performance considerations of a simulation are qualitatively different to that of an implementation. The software architecture of a MAS implementation is driven by real-world requirements that do not always hold in a simulation context. For example, trading agents need to be able to run on different machines due to commercial and practical considerations. This distributed parallelism is detrimental to raw system-level performance however, since inter-host network communication overheads dominate other performance considerations. By running all agents on the same host we can achieve several orders of magnitude performance increase. This would be an impractical solution for a real MAS trading implementation. However, such considerations do not apply in a simulation context, and by relaxing these constraints we can achieve a significant gain in performance.

Similarly, much of the technical complexity of a real MAS implementation addresses requirements that are not present in a simulation context. For instance, MAS implementations need to be robust against system failures, and they need to respond quickly to real-time asynchronous events. This necessitates a highly parallel software architecture, involving, for example, many threads of execution running simultaneously. Such considerations do not apply in agent-based simulation, since real-time parallelism can be simulated using a sequential program, and this greatly reduces the complexity of the software (and hence the potential for bugs).

Finally, any MAS interacts at some point with the environment. In a MARA scenario, for example, the environment might constitute economically relevant characteristics of the human owners of agents, such as their utility functions. Unlike the agents in a MAS implementation, the environment is not a software entity in a MAS implementation, and cannot be directly ported to an agent-based simulation. Rather, the environment itself must be simulated. Agent-based simulation toolkits allow for the abstract statistical simulation of environmental factors. Hence a key feature of any simulation toolkit is a library of pseudo-random number generators (PRNGs). A good simulation toolkit will provide high quality PRNGs, such as the Mersenne Twister PRNG [64], with extremely large periods, low statistical correlation, and the ability to produce random numbers according to arbitrary (non-uniform) distributions.

In summary, when developing a system to simulate a MARA scenario, it is important to choose a framework or toolkit that is specifically designed for agent-based *simulation*, as opposed to toolkits such as JADE [51] that are designed for *implementing* multiagent systems.

8.2 Simulating Time

For practical purposes we often prefer to simulate the parallelism of events using sequential computation, rather than execute the simulation of multiple simultaneous events in parallel in real-time. This necessitates a framework for computing the outcome of events that occur simultaneously. There are several approaches to simulating time in a model.

8.2.1 Continuous Time Models

Many physical processes are characterised by smooth and continuous changes in time-dependent variables. Differential equation models are common in analytical microeconomics. Such models are applicable approximations of real market places when there are very large numbers of participants in the market since individual characteristics

of the participants play a less significant role and the entities in the system can be treated as simple and homogeneous particles. However, these models break down when the number of participants becomes very small and the individual and strategic characteristics of the participants become more prominent.

Agent-based models address this issue by providing a richer structure with which to model market participants. In such models, macro-level variables describing the ensemble of agents no longer vary smoothly with time. This necessitates alternative approaches to temporal modelling.

8.2.2 Discrete-event Simulation

Discrete-event simulation frameworks [4, 42] model time in discrete quanta called *ticks*. Intuitively, a tick can be thought of as an "instant" of time. During the simulation of a tick—the *tick cycle*—entities (agents) in the simulation signal which agents they interact with during that instant of time by sending *events* to each other. Individual events specify the exact nature of the interaction between agents. In an auction simulation, for example, an auctioneer agent may send an *end-of-auction* event to all trading agents in the auction when it has closed. At the end of a tick cycle, once events have been exchanged, each entity updates its internal state in response to any events it has received.

8.3 Agent Modelling

In a MARA simulation, agents often need to make intelligent decisions in their resource utilisation and acquisition behaviour. The intelligent agents community has traditionally favoured symbolic approaches, such as the class of BDI (Belief-Desire-Intention) models. In a MARA scenario, however, an agent's goals are often quantitative in nature; for example, agents act to maximise their expected utility. In the field of agent-based electronic commerce, this has led to the adoption of Bayesian approaches to agent's decision problems such as (multiagent) reinforcement learning.

Many agent-based simulation frameworks have been developed by the Artificial Life (ALife) community. Agents in ALife models are imbued with very little intelligent behaviour at the outset; rather, intelligent behaviour emerges collectively from the complex interactions between agents equipped with relatively crude decision making machinery. Connectionist approaches such as neural networks and evolutionary approaches such as genetic algorithms, are popular in such models.

Since simulation is the main methodology used in ALife research, ALife software toolkits tend to be the most mature in terms of simulation functionality. Correspondingly, since empirical methods are relatively rare in MAS research, there are few frameworks for *simulating* BDI agents, as opposed to implementing BDI agents.

8.4 Extensibility and Integration

When conducting research via simulation it is often necessary to *extend* the existing functionality of the system. Although all frameworks provide the ability to configure simulations, the desired behaviour cannot always be implemented by configuring the existing components provided by the framework. In this case it is necessary for the researcher to implement the desired behaviour by writing their own code. Toolkits take two main approaches to allowing extensibility: They allow either for scripting in a custom language or for the introduction of new classes and methods via inheritance.

8.5 Software Listing

We are now going to give a brief overview of some commonly used general-purpose simulation frameworks that might be suitable for analysing MARA problems.

8.5.1 Swarm

Swarm [91] is one of the most famous ALife software toolkits and has been continually improved by an active community of users and developers since the early 1990s. It provides an API for discrete-event simulation, uses high-quality PRNGs, allows for spatial modelling, and includes real-time visualisation tools. Swarm is an open-source project written in the Objective-C programming language.

8.5.2 Extensions to Swarm

The Evo toolkit [36] is an extension to Swarm that provides agents with the ability to mate and evolve new behaviour over time using a system similar to genetic programming.

MAML [63] is an extension to Swarm that provides a higher-level scripting language that is simpler to use than Objective-C. The goal is to allow researchers from the social sciences, who are not necessarily skilled programmers, to quickly develop simulations.

8.5.3 RePast

RePast [75] is another toolkit inspired by Swarm, but is written entirely in Java, and the ultimate design goals of this system are more MAS- rather than ALife-oriented. It offers similar features as Swarm (discrete-event simulation, high-quality PRNGs, spatial modelling, visualisation tools) and it is also open-source and extensible.

The core simulation functionality of RePast is particularly mature and robust (it use the COLT library for high-performance scientific computing). The MAS-oriented features, on the other hand, are still relatively immature (no explicit reinforcement learning, no BDI support).

8.5.4 Desmo-J

Desmo-J [26] is implemented in Java and provides raw discrete-event simulation functionality. While only pro-

viding minimal functionality, the system comes with a highly flexible, well-designed API. It uses the standard Java PRNG, but the API should allow other (more advanced) PRNGs to be plugged in as well.

8.5.5 AScape

AScape [3] is a Java-based discrete-event simulation framework with an emphasis on spatial modelling of agents.

8.5.6 DEx

DEx [27] is a high-performance toolkit providing high-quality PRNGs, discrete-event simulation, spatial modelling, and real-time visualisation tools (including 3D representation).

9 Conclusion

We have presented a survey of salient issues in Multiagent Resource Allocation (MARA), a timely and fast-developing area of research at the interface of Computer Science and Economics. Naturally, the choice of topics selected for detailed presentation has been driven, at least in part, by personal interests and preferences. Nevertheless, we are confident that this material will prove useful to many researchers working on different aspects of MARA and related disciplines.

In the first part of the paper, after a short introduction to the field, we have highlighted four major application domains, which together both demonstrate the wide scope of MARA and underline the urgent need to further advance the field to meet the enormous challenges still posed by these applications. The second part of the paper serves as a catalogue of fundamental concepts in MARA: generic properties of resources characterising a MARA problem at hand; languages for preference representation to model the interests of individual agents; and social welfare measures and related tools to assess the overall quality of an allocation of resources. The third part of the paper then addresses actual MARA techniques. This includes, in particular, an introduction to allocation procedures and a selection of relevant complexity results. Where theoretical results alone are not sufficient, our survey of simulation platforms can serve as a starting point for experimental work.

Two important issues that we have *not* covered are the *algorithmics* of MARA and the *game-theoretical analysis* of negotiation (and bidding) strategies. The former includes the design of algorithms for the winner determination problem in combinatorial auctions, and a survey of recent work in this area is available elsewhere [81]. The literature on game-theoretical issues in negotiation, multiagent systems, and Computer Science in general is vast and fast-developing. A good starting point for readers interested in the computational approach to Game Theory (and the game-theoretic approach to Computer Science) is the short paper by Papadimitriou [70].

Acknowledgements. This survey has been written in the context of the activities of the AgentLink Technical Forum Group on Multiagent Resource Allocation (TFG-MARA).

References

[1] S. Aknine, S. Pinson, and M. F. Shakun. An extended multi-agent negotiation protocol. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(1):5–45, 2004.

[2] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*. North-Holland, 2002.

[3] AScape System. `http://www.brook.edu/es/dynamics/models/ascape/main.htm`.

[4] J. Banks, J. Carson, B. Nelson, and D. Nicol. *Discrete-event System Simulation*. Prentice Hall, 4th edition, 2005.

[5] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI-1993)*, pages 640–647. Morgan Kaufmann, 1993.

[6] M. Bichler, J. Kalagnanam, and H. S. Lee. RECO: Representation and evaluation of configurable offers. In H. K. Bhargava and N. Ye, editors, *Computational Modeling and Problem Solving in the Networked World: Interfaces in Computing and Optimization*. Kluwer, 2003.

[7] R. A. Bourne and R. Zaidi. A quote-driven automated market. In *Proc. AISB Symposium on Information Agents for E-Commerce*. AISB, 2001.

[8] C. Boutilier, F. Bacchus, and R. I. Brafman. UCP-networks: A directed graphical representation of conditional utilities. In *Proc. 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2001)*, pages 56–64. Morgan Kaufmann, 2001.

[9] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Pool. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[10] S. Bouveret, H. Fargier, J. Lang, and M. Lemaître. Allocation of indivisible goods: A general model and some complexity results. In *Proc. 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, pages 1309–1310. ACM Press, 2005.

[11] S. Bouveret and J. Lang. Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. In *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, pages 935–940. Morgan Kaufmann, 2005.

[12] R. I. Brafman and C. Domshlak. Introducing variable importance tradeoffs into CP-nets. In *Proc. 18th Conference in Uncertainty in Artificial Intelligence (UAI-2002)*, pages 69–76. Morgan Kaufmann, 2002.

[13] S. J. Brams and A. D. Taylor. *Fair Division: From Cake-cutting to Dispute Resolution*. Cambridge University Press, 1996.

[14] G. Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proc. 11th International Joint Conference on Artificial Intelligence (IJCAI-1989)*, pages 1043–1048. Morgan Kaufmann, 1989.

[15] H. van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry*, 37(3):255–274, 1998.

[16] E. Cantillon and M. Pesendorfer. Auctioning bus routes: The London experience. In P. Cramton et al., editors, *Combinatorial Auctions*. MIT Press, 2006. To appear.

[17] CERN. LHC: The Large Hadron Collider. `http://lhc.web.cern.ch`.

[18] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Multiagent resource allocation with k-additive utility functions. In *Proc. DIMACS-LAMSADE Workshop on Computer Science and Decision Theory*, Annales du LAMSADE 3, pages 83–100, 2004.

[19] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Welfare engineering in practice: On the variety of multiagent resource allocation problems. In *Engineering Societies in the Agents World V*, pages 335–347. Springer-Verlag, 2005.

[20] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Negotiating over small bundles of resources. In *Proc. 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, pages 296–302. ACM Press, 2005.

[21] Y. Chevaleyre, U. Endriss, and N. Maudet. On maximal classes of utility functions for efficient one-to-one negotiation. In *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, pages 941–946. Morgan Kaufmann, 2005.

[22] V. Conitzer and T. W. Sandholm. Complexity of mechanism design. In *Proc. 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pages 103–110. Morgan Kaufmann, 2002.

[23] V. Conitzer, T. W. Sandholm, and P. Santi. Combinatorial auctions with k-wise dependent valuations. In *Proc. 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 248–254. AAAI Press, 2005.

[24] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006. To appear.

[25] R. K. Dash, D. Parkes, and N. R. Jennings. Computational mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.

[26] Desmo-J: Discrete-Event Simulation and Modelling in Java. `http://www.desmoj.de`.

[27] DEx: Dynamic Experimentation Toolkit. `http://dextk.org`.

[28] P. E. Dunne. *The Complexity of Boolean Networks*. Academic Press, 1988.

[29] P. E. Dunne. Extremal behaviour in multiagent contract negotiation. *Journal of Artificial Intelligence Research*, 23:41–78, 2005.

[30] P. E. Dunne. Multiagent resource allocation in the presence of externalities. In *Proc. 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS-2005)*, pages 408–417. Springer-Verlag, 2005.

[31] P. E. Dunne and Y. Chevaleyre. Negotiation can be as hard as planning: Deciding reachability properties of distributed negotiation schemes. Technical Report ULCS-05-009, Dept. of Computer Science, University of Liverpool, 2005.

[32] P. E. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation. *Artificial Intelligence*, 164(1–2):23–46, 2005.

[33] U. Endriss and N. Maudet. Welfare engineering in multiagent systems. In *Engineering Societies in the Agents World IV*, pages 93–106. Springer-Verlag, 2004.

[34] U. Endriss and N. Maudet. On the communication complexity of multilateral trading: Extended report. *Journal of Autonomous Agents and Multi-Agent Systems*, 11(1):91–107, 2005.

[35] U. Endriss, N. Maudet, F. Sadri, and F. Toni. On optimal outcomes of negotiations over resources. In *Proc. 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, pages 177–184. ACM Press, 2003.

[36] Evo Artificial Life Framework. `http://sourceforge.net/projects/evo/`.

[37] P. Faratin. *Automated Service Negotiation Between Autonomous Computational Agents*. PhD thesis, Dept. of Electronic Engineering, Queen Mary College, University of London, 2000.

[38] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Selfish routing in non-cooperative networks: A survey. In *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS-2003)*, pages 21–45. Springer-Verlag, 2003.

[39] M. Fischer and N. J. Pippenger. Relations among complexity measures. *Journal of the ACM*, 26:361–381, 1979.

[40] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd edition, 2004.

[41] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-1999)*, pages 548–553. Morgan Kaufmann, 1999.

[42] J. M. Garrido. *Object-oriented Discrete-event Simulation with Java: A Practical Introduction*. Kluwer, 2001.

[43] H. Geffner. *Default Reasoning: Causal and Conditional Theories*. MIT Press, 1992.

[44] A. Giovannucci, J. A. Rodríguez-Aguilar, A. Reyes, F. X. Noria, and J. Cerquides. iBundler: An agent-based decision support service for combinatorial negotiations. In *Proc. 19th National Conference on Artificial Intelligence (AAAI-2004)*, pages 1012–1013. AAAI Press, 2004.

[45] A. Giovannucci, J. A. Rodríguez-Aguilar, A. Reyes, F. X. Noria, and J. Cerquides. Towards automated procurement via agent-aware negotiation support. In *Proc. 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)*, pages 244–253. ACM Press, 2004.

[46] M. Golfarelli, D. Maio, and S. Rizzi. A task-swap negotiation protocol based on the contract net paradigm. Technical Report 005-97, CSITE, University of Bologna, 1997.

[47] M. Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92:167–189, 1997.

[48] P. Gradwell and J. Padget. Distributed combinatorial resource scheduling. In *Proc. AAMAS Workshop on Smart Grid Technologies (SGT-2005)*, 2005.

[49] P. J. 't Hoen and J. A. La Poutré. A decommitment strategy in a competitive multi-agent transportation setting. In *Agent-Mediated Electronic Commerce V*, pages 56–72. Springer-Verlag, 2004.

[50] M. Höpf. Holonic manufacturing systems: The basic concept and a report of IMS test case 5. In J. K. H. Knudsen et al., editors, *Sharing CIM Solutions*. IOS Press, 1994.

[51] JADE: Java Agent Development Framework. `http://jade.tilab.com`.

[52] G. M. Jonker, J.-J. Meyer, and F. Dignum. Market mechanisms in airport traffic control. In *Proc. 2nd European Workshop on Multiagent Systems (EUMAS-2004)*, 2004.

[53] B. Kádár, L. Monostori, and S. Szelke. An object-oriented framework for developing distributed manufacturing architectures. *Journal of Intelligent Manufacturing*, 9(2):173–179, 1998.

[54] J. Kalagnanam and D. C. Parkes. Auctions, bidding and exchange design. In D. Simchi-Levi et al., editors, *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*. Kluwer, 2004.

[55] V. Krishna. *Auction Theory*. Academic Press, 2002.

[56] C. Lafage and J. Lang. Logical representation of preferences for group decision making. In *Proc. 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000)*, pages 457–468. Morgan Kaufmann, 2000.

[57] J. Lang. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

[58] D. J. Lehmann. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 15(1):61–82, 1995.

[59] M. Lemaître, G. Verfaillie, and N. Bataille. Exploiting a common property resource under a fairness constraint: A case study. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-1999)*, pages 206–211. Morgan Kaufmann, 1999.

[60] M. Lemaître, G. Verfaillie, H. Fargier, J. Lang, N. Bataille, and J.-M. Lachiver. Equitable allocation of earth observing satellites resources. In *Proc. 5th ONERA-DLR Aerospace Symposium (ODAS-2003)*, 2003.

[61] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Sciences and Technology*, 6:367–381, 2002.

[62] B. López, S. Suárez, and J. L. de la Rosa. Task allocation in rescue operations using combinatorial auctions. In *Proc. 6th Catalan Congress on Artificial Intelligence (CCIA-2003)*. IOS Press, 2003.

[63] MAML: Multi-Agent Modeling Language. `http://www.maml.hu`.

[64] M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[65] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[66] H. Moulin. *Fair Division and Collective Welfare*. MIT Press, 2003.

[67] R. B. Myerson and M. A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983.

[68] N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton et al., editors, *Combinatorial Auctions*. MIT Press, 2006. To appear.

[69] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[70] C. H. Papadimitriou. Algorithms, games, and the Internet. In *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC-2001)*, pages 749–753. ACM Press, 2001.

[71] H. Van Dyke Parunak. Applications of distributed artificial intelligence in industry. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*. John Wiley and Sons, 1996.

[72] H. Van Dyke Parunak, A. D. Baker, and S. J. Clark. The AARIA agent architecture: An example of requirements-driven agent-based system design. In *Proc. 1st International Conference on Autonomous Agents (Agents-1997)*, pages 482–483. ACM Press, 1997.

[73] S. Phelps, P. McBurney, S. Parsons, and E. Sklar. Co-evolutionary auction mechanism design: A preliminary report. In *Agent-Mediated Electronic Commerce IV*, pages 123–142. Springer-Verlag, 2002.

[74] S. Phelps, S. Parsons, and P. McBurney. An evolutionary game-theoretic comparision of two double auction market designs. In *Agent-Mediated Electronic Commerce VI*. Springer-Verlag, 2005. To appear.

[75] RePast: Recursive Porus Agent Simulation Toolkit. `http://repast.sourceforge.net`.

[76] A. Reyes-Moro and J. A. Rodríguez-Aguilar. *i*AuctionMaker: A decision support tool for mixed bundling. In *Agent-Mediated Electronic Commerce VI*. Springer-Verlag, 2005. To appear.

[77] T. W. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. 11th National Conference on Artificial Intelligence (AAAI-1993)*, pages 256–262. AAAI Press, 1993.

[78] T. W. Sandholm. Contract types for satisficing task allocation: I Theoretical results. In *Proc. AAAI Spring Symposium: Satisficing Models*, 1998.

[79] T. W. Sandholm. Distributed rational decision making. In G. Weiß, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[80] T. W. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1–2):1–54, 2002.

[81] T. W. Sandholm. Optimal winner determination algorithms. In P. Cramton et al., editors, *Combinatorial Auctions*. MIT Press, 2006. To appear.

[82] T. W. Sandholm and V. R. Lesser. Leveled commitment contracts and strategic breach. *Games and Economic Behavior*, 35(1–2):212–270, 2001.

[83] T. W. Sandholm and S. Suri. Side constraints and non-price attributes in markets. In *Proc. IJCAI Workshop on Distributed Constraint Reasoning*, 2001.

[84] J. E. Savage. *The Complexity of Computing*. John Wiley and Sons, 1976.

[85] C. P. Schnorr. The network complexity and Turing machine complexity of finite functions. *Acta Informatica*, pages 95–107, 1976.

[86] A. K. Sen. *Collective Choice and Social Welfare*. Holden Day, 1970.

[87] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

[88] P. Sousa, C. Ramos, and J. Neves. Manufacturing entities with incomplete information. *Studies in Informatics and Control*, 9(2):79–88, 2000.

[89] P. Sousa, C. Ramos, and J. Neves. The Fabricare scheduling prototype suite: Agent interaction and knowledge base. *Journal of Intelligent Manufacturing*, 14(5):441–455, 2003.

[90] Stephens Inc. Internet Supply Chain Team. *Strategic Sourcing: Applications to Turn Direct Materials Procurement into Competitive Advantage*. Industry Report, 2001.

[91] Swarm. `http://www.swarm.org`.

[92] K. Sycara, S. F. Roth, N. Sadeh, and M. S. Fox. Resource allocation in distributed factory scheduling. *IEEE Expert*, 6(1):29–40, 1991.

[93] L. Tesfatsion. Agent-based computational economics: Growing economies from the bottom up. *Artificial Life*, 8(1):55–82, 2002.

[94] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

[95] Visionary manufacturing challenges for 2020. Committee on Visionary Manufacturing Challenges, National Research Council. National Academic Press, 1999.

[96] I. Wegener. *The Complexity of Boolean Functions*. John Wiley and Sons, 1987.

[97] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, 2002.

[98] P. R. Wurman, M. P. Wellman, and W. E. Walsh. A parametrization of the auction design space. *Games and Economic Behaviour*, 35(1–2):304–338, 2001.

[99] R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.

[100] A. C.-C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proc. 11th Annual ACM Symposium on Theory of Computing (STOC-1979)*, pages 209–213. ACM Press, 1979.

[101] H. P. Young. *Equity in Theory and Practice*. Princeton University Press, 1994.

[102] X. Zhang and V. Lesser. Multi-linked negotiation in multi-agent systems. In *Proc. 1st International Joint Conference on Autonomous Agents And Multiagent Systems (AAMAS-2002)*, pages 1207–1214. ACM Press, 2002.

The validity of all cited URLs has been verified at the time of writing (August 2005).

A Survey of Programming Languages and Platforms for Multi-Agent Systems

Rafael H. Bordini
University of Durham, UK
E-mail: R.Bordini@durham.ac.uk, http://www.dur.ac.uk/r.bordini

Lars Braubach
Universität Hamburg, Germany
E-mail: braubach@informatik.uni-hamburg.de, http://vsis-www.informatik.uni-hamburg.de

Mehdi Dastani
Utrecht University, The Netherlands
E-mail: mehdi@cs.uu.nl, http://www.cs.uu.nl/~mehdi

Amal El Fallah Seghrouchni
University of Paris 6, France
E-mail: Amal.Elfallah@lip6.fr, http://www-poleia.lip6.fr/~elfallah

Jorge J. Gomez-Sanz
Universidad Complutense de Madrid, Spain
E-mail: jjgomez@sip.ucm.es, http://grasia.fdi.ucm.es/jorge

João Leite
Universidade Nova de Lisboa, Portugal
E-mail: jleite@di.fct.unl.pt, http://centria.di.fct.unl.pt/~jleite

Gregory O'Hare
University College Dublin, Ireland
E-mail: Gregory.OHare@ucd.ie, http://www.cs.ucd.ie/staff/gohare

Alexander Pokahr
Universität Hamburg, Germany
E-mail: pokahr@informatik.uni-hamburg.de, http://vsis-www.informatik.uni-hamburg.de

Alessandro Ricci
Università di Bologna, Italy
E-mail: aricci@deis.unibo.it, http://lia.deis.unibo.it/~ari

Keywords: Multi-Agent Systems, Programming Languages, Platforms

Received: April 1, 2005

This paper surveys recent research on programming languages and development tools for Multi-Agent Systems. It starts by addressing programming languages (declarative, imperative, and hybrid), followed by integrated development environments, and finally platforms and frameworks. To illustrate each of these categories, some systems were chosen based on the extent to which European researchers have contributed to their development. The current state of these systems is described and, in some cases, indications of future directions of research are given.

Povzetek: Podan je pregled jezikov in orodij za MAS.

1 Introduction

Research in Multi-Agent Systems (MAS) has recently led to the development of practical *programming languages and tools* that are appropriate for the implementation of such systems. Putting together this new programming paradigm is fast becoming one of the most important topics of research in multi-agent systems, in particular because this is an essential requirement for an eventual technology transfer.

Surveying the MAS literature will reveal a large number of different proposals for agent-oriented languages, ranging from purely declarative, to purely imperative, and various hybrid approaches. Some are designed from scratch,

directly encoding some theory of agency, while others extend existing languages to suit the peculiarities of this new paradigm. Using these languages, instead of more conventional ones, proves useful when the problem is modelled as a *multi-agent* system, and understood in terms of cognitive and social concepts such as beliefs, goals, plans, roles, and norms.

Most agent programming languages have some underlying platform which implements its semantics. However, agent frameworks exist that are not tightly coupled with one specific programming language. Instead, they are concerned with providing general techniques for relevant aspects such as agent communication and coordination. The most mature languages will be accompanied by some Integrated Development Environment (IDE), intended to enhance the productivity of programmers by automating tedious coding tasks. Typically these will provide functionalities such as project management, creating and editing source files, refactoring, build and run process, and testing.

Despite the large number of languages, frameworks, development environments, and platforms recently proposed, implementing MAS is still an often daunting task. To address the problem of managing the inherent complexity of MAS and helping the structuring of their development, the research community has produced a number of methodologies [4]. Nevertheless, even if MAS practitioners follow such methodologies during the design phase, they still find great difficulties in the implementation phase, partly due to the lack of maturity of both methodologies and programming tools. Among others, such difficulties can be traced to the lack of specialised debugging tools; to the lack skills that are necessary in mapping analysis/design concepts to programming languages constructs; to the lack of proficiency in dealing with the specific characteristics of different agent platforms; and also to the lack of understanding of the very foundations as well as practical characteristics of the agent-oriented approach to programming.

Even though most of the languages and tools developed so far have not been tried yet in large-scale, industrial-strength applications, much progress has been achieved in the last few years. This is, therefore, an appropriate time for a *reality check* and a bird's eye view of the field, helping to consolidate existing achievements and guide future developments. To this end, this paper surveys some of the existing approaches situated in the MAS Programming area of research, from programming languages to development infrastructures, chosen in part according to the extent to which European researchers have contributed to their development.

The first part of the paper is devoted to the presentation of agent-oriented programming languages, structured according to the existing paradigm on which they build. In Section 2, we present declarative agent-oriented languages, while Section 3 covers the imperative languages and Section 4 some hybrid languages. The second part will cover various implementations of software infrastructure for agents. These will be structured according to whether they are development environments for MAS, in Section 5, or MAS platforms and frameworks, in Section 6. The paper ends with some reference to further readings on this subject in Section 7, and some final remarks in Section 8.

2 Declarative Languages

Declarative languages are partially characterised by their strong formal nature, normally grounded on logic. This is the case with most of the declarative languages described here: FLUX, Minerva, Dali, and ResPect. Other declarative languages are also grounded on other formalisms, such as CLAIM which finds parts of its roots in the ambient calculus. Declarative languages that allow for easy integration with imperative code will be reviewed in Section 4 below.

CLAIM (**C**omputational **L**anguage for **A**utonomous, **I**ntelligent and **M**obile Agents [23]) is a high-level declarative agent-oriented programming language. It is part of an unified framework called Himalaya [25] (**H**ierarchical **I**ntelligent **M**obile **A**gents for building **L**arge-scale and **A**daptive s**Y**stems based on **A**mbients). It combines the main advantages of agent-oriented programming languages, for representing cognitive aspects and reasoning, with those of concurrent languages based on process algebra, for representing concurrency and agent mobility.

The CLAIM language is inspired by *ambient calculus* [11] and agents are hierarchically organised, thus supporting the design of Mobile Multi-Agent Systems (MMAS) – a set of connected hierarchies of agents – to be deployed on a network of computers. Every agent (i.e., a node of a hierarchy) contains cognitive elements (e.g., knowledge, goals, capabilities), processes, and sub-agents and is also mobile as it can move within its hierarchy or to a remote one. In addition, an agent can dynamically acquire intelligent and computational components from its sub-agents, which can be seen as some sort of inheritance. The mobility and the inheritance as defined in Himalaya framework favour a dynamic adaptability and reconfiguration of systems [50] for coping with the increasing complexity of distributed and cooperative applications. The main elements of CLAIM agents are cognitive, interaction, mobility, and reconfiguration primitives.

The formal semantics of CLAIM is based on Plotkin's [41] structural operational approach consisting of a transition relation, from an initial state of a program to another state resulting from the execution of an atomic operation. At each step of an agent execution, either a message is dealt with, a running process executed, or a goal processed. For a detailed presentation of the semantics, we refer the reader to [24].

As an MMAS within Himalaya is deployed on a set of connected computers, the language CLAIM is supported by a distributed platform called SyMPA [51], which offers all the necessary mechanisms for management of agents, communication, mobility, security, fault-tolerance, and load balancing [30]. SyMPA is implemented in Java and

compliant with the specifications of the MASIF [37] standard from the OMG (Object Management Group). There is a central system providing management functions. An agent system is deployed on each computer connected to the platform.

The Himalaya environment has been used for developing several complex applications that showed the expressiveness of the language and the robustness and strength of the platform, such as: an application for information search on the Web [22], several electronic commerce applications [23, 52], a load balancing and resource sharing application using mobile agents [30], and an application for a network of digital libraries.

FLUX [53] is a high-level programming system for cognitive agents, which can be downloaded from `http://www.fluxagent.org`. It consists of an implementation of the Fluent Calculus, an action representation formalism that provides a basic solution to the classical frame problem using the concept of state update axioms, while addressing a variety of aspects in reasoning about actions (hence the relevance for agents), such as ramifications (i.e., indirect effects of actions), qualifications (i.e., unexpected action failure), nondeterministic actions, concurrent actions, continuous change, and noisy sensors and effectors.

An agent program in FLUX is a logic program consisting of three parts: the kernel providing the agent with the cognitive ability to reason about its actions and acquired sensor data, a background theory providing an internal model of its environment, and a strategy which specifies the task-oriented behaviour in accordance with which the agent reasons, plans, and acts. The full expressive power of logic programming can be used to design strategies while facilitating formal proofs of the correctness of strategies with respect to a problem-dependent specification.

The use of progression, where a (possibly incomplete) initial world model is updated upon the performance of an action, is one of the main characteristics of FLUX. This allows for a computationally efficient solution to the frame problem and, consequently, an efficient agent implementation based on the Fluent Calculus. Further information regarding FLUX can be obtained in [54].

MINERVA [32, 33] is an agent system designed to provide a common agent framework based on the strengths of Logic Programming, to allow for the combination of several existing non-monotonic knowledge representation and reasoning mechanisms. It uses MDLP and KABUL to specify agents and their behaviour. A MINERVA agent consists of several specialised, possibly concurrent, subagents performing various tasks, whose behaviour is specified in KABUL, while reading and manipulating a common knowledge base specified in MDLP.

MDLP (**M**ulti-Dimensional **D**ynamic **L**ogic **P**rogramming) is the basic knowledge representation mechanism of an agent in MINERVA. MDLP is an extension of Answer Set Programming (ASP) where knowledge is represented by logic programs arranged in an acyclic digraph. In this digraph, vertices are sets of logic programs, and edges represent the relations between program. MDLP enjoys the merits of ASP such as default negation. Default negation allows the definition of non-monotonic behaviour thus facilitating the representation of, and reasoning about, incomplete knowledge. MDLP also allows for the simultaneous representation of several aspects such as hierarchies and preferences, as well as the evolution of the represented knowledge.

KABUL (**K**nowledge **A**nd **B**ehavior **U**pdate **L**anguage), as its recent evolution EVOLP [1], is a logic-programming style language that allows the specification of updates to a knowledge base and to itself. A program in KABUL is a set of *statements*, each statement being a type of condition-action rule that can be seen as encoding an agent behaviour. The epistemic effects of actions can be either an update to the knowledge base of the agent, represented by an MDLP program, or a self update to the KABUL program, thus changing the behaviour of the agent over time. Conditions range from external observations, epistemic state of the agent, as well as concurrent execution of other actions. This allows for a combination of reactive and proactive behaviour, in the sense that no external stimuli are needed to trigger the behaviour of the agent, while these can be combined with the rational features provided by the underlying MDLP knowledge representation framework and its formal and precise ASP-based semantics. More information regarding MDLP, KABUL, and MINERVA can be found in [32].

DALI [14] is an Active Logic Programming language designed for executable specification of logical agents. It uses plain Horn Clauses and its semantics is based on Least Herbrand Models. It intends to provide constructs to represent reactivity and proactivity in an agent by means of rules. A DALI agent is a logic program that contains reactive rules, events, and actions aimed at interacting with an external environment. The reactive and proactive behaviour of a DALI agent is triggered by several kinds of events: external, internal, present, and past events. All the events and actions are time stamped so as to record when they occurred. The new syntactic entities, i.e., predicates related to events and proactivity, are indicated with special postfixes. When an event occurs in the agent's "external world", the agent can perceive it and decide to react. The reaction is defined by a reactive rule which has in its head that external event. The internal events define the behaviour of a DALI agent, making it proactive independently of the environment and allowing it to manipulate and revise its knowledge.

ReSpecT [38] is a logic-based language, with a well-defined formal semantics, allowing for the definition of reactions, expressed in terms of rules. A rule in ReSpecT consists of a head specifying the communication event that triggers the reaction and a body specifying which actions (tuples from the tuple centre) are atomically executed when the reaction is triggered. When a basic action fails, the reaction atomically fails and all its effects on the tuple centre state are rolled back. The coordinating behaviour of

tuple centres can be changed and adapted at runtime by dynamically changing the reactions defined in ReSpecT. Such a feature is typically exploited to deal with dynamism and openness of MAS applications. The tuple centre programmed with these reactions acts as a basic scheduler, encapsulating the policy adopted to coordinate the various (autonomous) agent tasks. By changing the reactions, the overall coordinating behaviour of the system changes, without the need to change the agent's behaviour. This language is used within the TuCSoN framework (discussed below in Section 6).

3 Imperative Languages

Purely imperative approaches to agent-oriented programming are less common, mainly due to the fact that most abstractions related to agent-oriented design are, typically, declarative in nature. There are however many programmers who still use conventional, i.e. non-agent oriented, imperative languages for developing multi-agent systems; as a result, in practice agent notions are often implemented in an *ad-hoc* manner. An example of an agent-oriented language which is still essentially imperative, while incorporating agent-specific abstractions, is the language available with the development environment JACK [57, 26].

The **JACK Agent Language (JAL)** has been developed by a company called Agent Oriented Software. JAL is based on ideas of reactive planning systems resulting from the work on the BDI agent architecture and is, in this respect, similar to the hybrid languages Jason, 3APL, and Jadex (discussed below in Section 4). However, instead of providing a logic-based language, JAL is an extension of Java (implementing some features of logic languages such as logical variables). A number of syntactic constructs is added to Java, allowing programmers to create plans and belief bases, all in a graphical manner as JAL has a sophisticated IDE which provides a tool for such purpose. In JAL, plans can be composed of *reasoning methods* and grouped into *capabilities* which, together, compose a specific ability an agent is supposed to have, thus supporting a good degree of modularisation. Another structuring mechanism present in JAL is the ability to use *teams* of agents, or agent organisations, a notion that is increasingly important both in agent-oriented design [4] and because of recent developments in self-organising systems [47]. Although JAL has no formal semantics, as a commercial platform, JACK has extensive documentation and supporting tools. It has been used in a variety of industrial applications as well as for research. For evaluation purposes, a free trial license for JAL can be obtained; more information is available at `http://www.agent-software.com`.

4 Hybrid Approaches

Various well-known agent languages combine declarative and imperative features. In this section we describe agent programming languages which are declarative while at the same time providing some specific constructs allowing for the use of code implemented in some external imperative language. These constructs serve as a means for the use of legacy code. The languages chosen to illustrate the hybrid approach are: 3APL, *Jason*, IMPACT, Go!, and AF-APL.

3APL (**A**n **A**bstract **A**gent **P**rogramming **L**anguage "**triple-a-p-l**") is a programming language for implementing cognitive agents that have beliefs, goals, and plans as mental attitudes, can generate and revise their plans to achieve their goals, and are able to interact with each other and with the environment they share with other agents. The first version of 3APL was designed by Hindriks et al. at Utrecht University [28]. Since its initial design, the 3APL programming language has been subject to continuous development [17, 16].

One of the main features of 3APL consists of programming constructs to implement mental attitudes of an agent as well as the deliberation process which manipulates them [15]. In particular, 3APL allows direct specification of mental attitudes such as beliefs, goals, plans, actions and reasoning rules. Actions form the basic building blocks of plans and can be internal mental actions, external actions, or communication actions. The deliberation-related constructs allow the implementation of selection and execution of actions and plans through which an agent's belief base can be updated and through which the shared environment can be modified. It also allows the selection and application of reasoning rules through which the plan base can be modified.

The 3APL programming language is designed so as to respect a number of software engineering and programming principles such as separation of concerns, modularity, abstraction, and reusability. It also supports the integration of Prolog (declarative) and Java (imperative) programming languages. Interested readers will find in the 3APL user guide (`http://www.cs.uu.nl/3apl`) a number of illustrative toy-problem applications such as the "blocks world", Axelrod's tournament, an English auction system, and the Contract Net protocol. 3APL has also been applied to the implementation of the high-level control of mobile robots. In particular, 3APL is being used for controlling the behaviour of SONY AIBO robots and to implement small-device mobile applications.

Jason is an interpreter, implemented by R.Bordini and J.Hübner, for an extended version of AgentSpeak(L), a logic-based agent-oriented programming language introduced by A. Rao in [43]. The language is influenced by the work on the Beliefs-Desires-Intentions (BDI) architecture and BDI logics [44]. The semantics of the extended language (which we call simply AgentSpeak), given by Bordini and colleagues, was recently revised and appears in [55]. The core of the interpreter available with ***Jason*** is in fact an implementation of that operational semantics. ***Jason*** is available *Open Source* under GNU LGPL at `http://jason.sourceforge.net` [6]. Although the documentation is available at that URL, the best mate-

rial for an overview of the work on Jason is [7].

Some of the features available in ***Jason*** are: (i) speech-act based inter-agent communication (and belief annotation of information sources); (ii) annotations on plan labels, which can be used by elaborate (e.g., decision-theoretic) selection functions; (iii) fully customisable (in Java) selection functions, trust functions, and overall agent architecture (perception, belief-revision, inter-agent communication, and acting); (iv) straightforward extensibility (and use of legacy code) by means of user-defined "internal actions"; (v) a clear notion of a *multi-agent environment*, which is implemented in Java (this can be a simulation of a real environment, e.g., for testing purposes before the system is actually deployed). ***Jason*** has a simple IDE which is discussed in Section 5.

IMPACT is a system developed by Subrahmanian et al. [49], with the main purpose of providing a framework to build agents on top of heterogeneous sources of knowledge, i.e., to transform legacy code into agents that can communicate and act. To "agentise" such legacy code, IMPACT provides the notion of an agent program written over a language of so-called *code-calls*. A code-call can be seen as an encapsulation of whatever the legacy code is, represented logically through conditions and queries on the results produced by such code. These are used in clauses, that form agent programs, determining constraints on the actions that are to be taken by agents. Actions in IMPACT use some deontic notions such as agent actions being, at a certain time, "obligatory", "permitted", "forbidden", etc. Such agent programs and their semantics resemble logic programs extended with deontic modalities. The semantics is given by the notion of a *rational status sets*, which are generalisations of the notion of stable models in logic programming.

The IMPACT platform provides a number of features, including agent deployment over a network, registration of available agent services and yellow-page facilities. Information on the IMPACT platform is available at `http://www.cs.umd.edu/projects/impact/`. The framework has been extended to support also temporal or probabilistic reasoning [20]. A recent overview of the IMPACT language and platform can be found in [21].

Go! [12] is a multi-paradigm agent programming language, with a declarative subset of function and relation definitions, an imperative subset comprising action procedure definitions, and rich program structuring mechanism. Based on the symbolic programming language April [36], Go! extends it with knowledge representation features of logic programming, yielding a multi-threaded, strongly typed and higher order (in the functional-programming sense) language.

Inherited from April, threads primarily communicate through asynchronous message passing. Threads, executing action rules, react to received messages using pattern matching and pattern-based message reaction rules. A communication daemon enables threads in different Go! processes to communicate transparently over a network. Typically, each agent will comprise several threads, each of which can directly communicate with threads in other agents. Threads within a single Go! process, hence in the same agent, can also communicate by manipulating shared cell or dynamic relation objects. As in Linda tuple stores, these elements are used to coordinate the activities of different threads within an agent. Go! is strongly typed, which can often reduce the programmer's burden, and compile-time type checking improves code safety. New types can be declared and thereby new data constructors can be introduced.

The design of Go! took into consideration critical issues such as security, transparency, and integrity, in regards to the adoption of logic programming technology. Features of Prolog that lack a transparent semantics, such as the cut ('`!`') were left out. In Prolog the same clause syntax is used both for defining relations, with a declarative semantics, and for defining procedures which only have an operational semantics. In Go!, behaviour is described using action rules that have a specialised syntax.

Agent Factory Agent Programming Language (AF-APL) is the core programming language that resides at the heart of Agent Factory, which will be reviewed in Section 5. AF-APL is originally based on Agent-Oriented Programming as first put forward by Y.Shoham [48], but was revised and extended with BDI concepts, such as beliefs and plans. The syntax and semantics of the AF-APL language have been derived from a logical model of how an agent commits itself to a course of action. Details of this model can be found in [13, 46]. Specifically, the model defines the mental state of an agent to be comprised of two primary mental attitudes: beliefs and commitments. In AF-APL, the belief set is comprised of a set of declarations about the current state of the environment. Agents are *situated*, given that an AF-APL programmer can declare explicitly, for each agent, a set of sensors referred to as perceptors and a set of effectors known as actuators. Perceptors are realized as instances of Java classes that define how to convert raw sensor data into beliefs that may be added to the belief set of the agent. Similarly, an actuator is realized as an instance of a Java class, which has two responsibilities: (1) to define the action identifier that should be used when referring to the action that is realized by the actuator, and (2) to contain code that implements the action. Collectively, these declarations are termed the embodiment configuration of the agent, and they are specified within the agent program.

5 Integrated Development Environments

Integrated Development Environments (IDEs), focus on the programming language level and intend to enhance the productivity by automating tedious coding tasks. Looking at current IDEs from the object-oriented domain it can be

seen that such IDEs tend to provide functionalities that can be classified into five categories: *project management*, e.g. organising the project structure according to developers' needs; *creating and editing source files*, e.g. providing structure views for quick and easy navigation, online error detection, auto-completion, and so on; *refactoring* to enable fast and reliable code restructuring operations; *build and run process* allowing the execution of applications from within the IDE; and *testing*, e.g. supported by unit testing with test cases.

In the agent world, the situation differs from conventional programming in that there is no common ground with respect to agent programming languages and agent architectures. Hence, current agent IDEs exist only for agent languages of specific agent frameworks. Additionally, we found that only a small proportion of available agent frameworks offer IDE support at all, considering AgentLink (`http://www.agentlink.org`) as a representative selection of existing agent-related software. From this small number, we selected some representative examples: 3APL IDE, Jason IDE, JDE, CAFnE, Visual Soar, AgentBuilder, AgentFactory, and the Living Systems Developer.

The **3APL IDE** allows developers to load/edit 3APL programs that implement individual agents, execute one or more agent programs in either a step-by-step or continuous fashion, implement and configure the environment that is shared by the agents, monitor the internal state of individual agents through an agent property window, monitor the exchange of messages through the sniffer tool, send an external-user message to an individual agent, and read the system messages. The 3APL IDE is built on top of the 3APL multi-agent platform that consists of a directory facilitator called agent management system, a message transport system which delivers agent messages, and a plugin interface that allows agents to execute actions in the shared environment. The 3APL platform thus allows the implementation and concurrent execution of a set of 3APL agents. The 3APL development environment, its user guide, and further documentation can be found at `http://www.cs.uu.nl/3apl`.

Jason [6] is distributed with an IDE which provides a graphical interface for editing a multi-agent system configuration file, as well as AgentSpeak code for the individual agents. Through the IDE, it is also possible to run and control the execution mode of a multi-agent system, and to distribute agents over a network in a very simple way. The IDE also provides another tool, called "Mind Inspector", which allows the user to inspect agents' internal states when the system is running in debugging mode. This is very useful for debugging AgentSpeak MAS, as it allows the programmer to inspect agents' mental attitudes across a distributed system.

The JACK Development Environment (JDE) is a full-featured commercial IDE for the JACK BDI agent platform [57] developed by Agent Oriented Software Ltd. It is based on the JACK Agent Language (JAL) which was presented in Section 3. JDE allows agent developers to organise their files into projects offering a semantically organised tree view with respect to the different kinds of contained elements. The editing of agent code is supported by a rudimentary integrated editor that, for example, provides syntax highlighting for JAL. More advanced features such as auto-completion and error-detection are not available. However, the IDE provides a graphical plan editor that allows the construction of a plan from visual components similar to statecharts. Once the code base for a project is complete, it is possible to compile and run an application directly from within the IDE.

The **CAFnE** (**C**omponent **A**gent **F**ramework for **n**on-**E**xperts) tool [29] does not represent an IDE in the classical sense. Its objective is to enable domain experts to modify an existing agent application. CAFnE has been conceived to support the development of BDI agents based on a rather platform-independent BDI component language adapted from SMART [34]. The rationale of CAFnE is to hide the agent code layer and provide interactive dialogues for the development. Transformer modules can then be used to generate platform-dependent code from the internal representation.

Visual Soar is a freely available IDE for the Soar agent architecture [31]. It supports basic project management capabilities and mainly facilitates Soar agent programming through syntax highlighting and some consistency checking functionalities. Additionally, the IDE provides a connection to a Soar runtime environment allowing Soar agents to be executed from the IDE.

AgentBuilder is an agent platform directly based Agent-Oriented Programming (AOP), as originally defined by Shoham [48], developed by Acronymics Inc. It relies on the Reticular Agent Language which is an extension of Shoham's Agent0. As the used agent language is not intended for direct programming, an agent developer has to use the AgentBuilder IDE, which consists of a variety of different tools supporting all aspects of building agent applications. The IDE is conceived to hide agent code as much as possible and offers graphical wizards and tools whenever possible. It provides simple project management functionalities and integrates with a compiler tool. Successfully built agent applications can directly be executed from the IDE.

The **Agent Factory** [13] Development Environment offers support for basic project management, editing, and assembling the different agent constituents. It contains a cohesive layered framework for the development and deployment of agent-oriented applications. At the centre of this framework is the Agent Factory Agent Programming Language (AF-APL) described above in Section 4. The AF-APL interpreter is embedded within the distributed FIPA-compliant Run-Time Environment (RTE) which can be seen as a collection of agent platforms. Besides the IDE, a tool named VIPER [45] allows the composition of Agent UML Sequence Diagrams that sit at the heart of the Protocol Model. In addition to the tools that have been provided to support the development of AF-APL agents, the

Agent Factory Development Environment also includes a suite of tools that facilitate the testing and debugging of agent-oriented applications.

The **Living Systems Developer** is a commercial IDE for the Living Systems Technology Suite developed by Whitestein (`http://www.whitestein.com`). The underlying agent platform supports Java-based agents, rather than supporting a specialised agent language. The IDE is designed as an Eclipse (`http://www.eclipse.org`) plug-in, hence providing sophisticated editing and refactoring functionalities for Java code. In addition, several agent related aspects such as project management in accordance to the agent features used have been added. To facilitate the development process of agent-based applications, the IDE has been extended to fully support all phases of ADEM, the Agent Development Methodology also created at Whitestein.

6 Agent Platforms and Frameworks

Most languages described in this paper have some underlying platform which implements the semantics of the agent programming language. However, some implemented frameworks exist that are not so strongly tied to a particular programming language. Instead, these frameworks are more concerned with providing support for aspects such as agent communication and coordination. In this Section we focus on such frameworks, having chosen TuCSoN, JADE, and DESIRE as illustrative examples.

TuCSoN (**Tu**ple **C**entre **S**pread **o**ver the **N**etwork) is a framework for MAS coordination, based on a model and a related infrastructure providing general-purpose, programmable services for supporting agent communication and coordination [39]. The model is based on *tuple centres* as runtime programmable abstractions whose coordinating behaviour can be dynamically specified with a logic-based language called ReSpecT. Tuple centres are an example of coordination artifacts (see the survey on Environment modelling for MAS [56]), i.e, first-class entities (tools) populating the agent cooperative working environment, shared and used collectively by the agents to support their coordination. Such abstractions are also used in the SODA methodology (see the survey on Agent Oriented Software Engineering [4]) as basic building blocks for designing the social level and the environment in a MAS.

The TuCSoN technology is available as an open source project (`http://tucson.sourceforge.net`). It is completely based on Java, and is composed of: a runtime platform to be installed on hosts to turn them into nodes of the infrastructure; a set of libraries (APIs) to enable agents access to the services; and a set of tools mainly to support the runtime inspection and control (monitoring, debugging) of tuple-centres' state and coordinating behaviour. At the heart of the TuCSoN technology is the tuProlog technology, a Prolog engine fully integrated with the Java environment, available also as a standalone library and environment (the tuProlog technology is available at `http://tuprolog.sourceforge.net` [19]). Besides being adopted in research projects (e.g., for distributed workflow management, logistics, and e-learning), TuCSoN is currently used as one of the reference platforms for building agent-based systems in academic projects and thesis developed at the Engineering Faculties in Cesena and Bologna.

JADE (**J**ava **A**gent **DE**velopment Framework) [2] is a Java framework for the development of distributed multi-agent applications. It represents an agent middleware providing a set of available and easy-to-use services and several graphical tools for debugging and testing. One of the main objectives of the platform is to support interoperability by strictly adhering to the FIPA specifications concerning the platform architecture as well as the communication infrastructure. Moreover, JADE is very flexible and can be adapted to be used on devices with limited resources such as PDAs and mobile phones.

JADE has been widely used over the last years by many academic and industrial organisations (see [2]) ranging from tutorials for teaching support in agent-related University courses to Industrial prototyping. As an example, Whitestein has used JADE to construct an agent-based system for decision-making support in organ transplant centres [10].

The JADE platform is open source software, distributed by TILAB (Telecom Italia LABoratories) under the terms of the LGPL license and can be obtained at `http://jade.tilab.com`. Since May 2003, the International JADE Board has been responsible for supervising the management of the project. Currently, the JADE Board consists of five members: TILAB, Motorola, Whitestein Technologies AG, Profactor, and France Telecom.

Jadex [42] is a software framework for the creation of goal-oriented agents following the belief-desire-intention (BDI) model. The framework is realized as a rational agent layer that sits on top of a middleware agent infrastructure such as JADE [2], and supports agent development with well established technologies such as Java and XML. The Jadex reasoning engine addresses traditional limitations of BDI systems by introducing new concepts such as explicit goals and goal deliberation mechanisms (see, e.g., [8]), making results from goal-oriented analysis and design methods (e.g., KAOS and Tropos) more easily transferable to the implementation phase.

Jadex has been used to build applications in different domains such as simulation, scheduling, and mobile computing. For example, Jadex was used to develop a multi-agent application for negotiation of treatment schedules in hospitals [40]. Jadex has also been successfully used in several software engineering courses at the University of Hamburg.

The Jadex system, developed at the Distributed Systems and Information Systems group at the University of Hamburg, is freely available under the LGPL license and can be downloaded from `http://jadex.sourceforge.net`. Besides the framework and additional development

tools, the distribution contains an introductory tutorial, a user guide, and several illustrative example applications with source code.

DESIRE (**DE**sign and **S**pecification of **I**nteracting **RE**asoning components) is a compositional development method for multi-agent systems, based on a notion of compositional architecture, and developed by Treur et al. [9] at the Vrije Universiteit Amsterdam. In this approach, agent design is based on the following main aspects: process composition, knowledge composition, and relations between knowledge and process composition. In this component-based agent approach, an agent's complex reasoning process is built up as an interaction between the components representing the subprocesses of the overall reasoning process [9]. The reasoning process is structured according to a number of reasoning components that interact with each other. Components may or may not be composed of other components, where components that are not further decomposed are called primitive components. The functioning of the overall agent system is based on the functionality of these primitive components plus the composition relation that coordinates their interaction. Specification of a composition relation may involve, for example, the possibilities of information exchange among components and the control structure that activates the components. The DESIRE approach has been used for applications such as load balancing of electricity distribution and diagnosis systems. Further information and documentation of the tools supporting the development and implementation of multi-agent systems based on DESIRE is available at `http://www.few.vu.nl/~wai/demas/tools2.html`.

7 Further Reading

This paper should be complemented with related literature. Besides the references spread throughout the text, pointing to more detailed explanations of the systems described, we recommend the survey on agent programming languages by Mascardi et al. [35], which provides a detailed view of ConGolog, Agent-0, IMPACT, DyLog, Concurrent MetateM, and $\mathcal{E}_{hhf}$. A reference book on programming languages for Multi-Agent Systems has been published recently [5]. It contains detailed description of a selection of practical programming languages and tools which support MAS programming and implement key MAS concepts in a unified framework. Another extensive overview of agent technology is available in [3], which includes a comprehensive collection of papers on technologies, methodologies, and current research trends in the MAS domain.

As we have mentioned before, the criteria in which we based our choice of systems was, in part, the extent to which European researchers have contributed to their development. Of course there are various other agent languages, platforms, and tools besides those referred here. A good collection of agent-related software can be found in the AgentLink III website (`www.agentlink.org`).

Overall, the systems described here focus on the implementation phase. However, current research trends include the attempt to make implementation easier by bridging the analysis and design phase directly to implementation [4]. Examples of such research efforts are INGENIAS and its Development Kit [27] (`http://ingenias.sourceforge.net`), and MaSE and its AgenTool [18] (`http://macr.cis.ksu.edu/projects/agentTool/agentool.htm`).

8 Final Remarks

Programming Multi-Agent Systems is rapidly turning into a new discipline of its own. Throughout the paper, we have described several examples of languages and systems currently being developed in this area. We now draw some conclusions on the three main topics of this survey, namely languages, IDEs, and platforms.

Languages. Most research in agent-oriented programming languages is based on declarative approaches. There are many declarative solutions, most of them logic based. Purely imperative languages are unusual in the Agents literature, as in essence they are inappropriate for expressing the high-level abstractions associated with agent systems design. On the other hand, as we saw above, agent-oriented programming languages tend to allow for easy integration with (legacy) code written in imperative languages. Interestingly, the characteristics of the underlying agent architectures determine that it is often more appropriate to use interpreters rather than compilers.

IDEs. The existing IDEs provide basic support for project management, creating/editing files, and building/running the systems, but fail to support sophisticated features within all these categories. In addition, none of the agent IDEs covers aspects of refactoring and testing of agent applications. One reason for this is that, except for the Living Systems Developer, all IDEs have been developed from scratch and thus do not rely on existing reliable technology. In general, IDE support for developing agent-based systems is rather weak and the existing agent tools do not offer the same level of usability as state-of-the-art object-oriented IDEs. One of the main reason for this is the currently unavoidable tight coupling of agent IDEs and agent platforms, which results from the lack of agreement on an unified programming language for multi-agent systems. Another trend (observable in some of the IDEs), which is in contrast to object-oriented IDEs, is that they partly try to abstract away from the underlying programming language in favour of using graphical means of programming, such as wizards and statecharts.

Platforms. Closed frameworks such as DESIRE, strongly based on a platform, provide more complete solutions than others such as Jadex or TuCSoN. They usually offer an agent architecture and a system model, very useful for novel developers, together with the communication infras-

tructure and a range of robust services, such as directory facilitators, agent management services, and monitoring facilities. As a drawback, closed frameworks limit the development. For example, the design approach of the framework may not fit certain domain problems. Perhaps that is the reason why most researchers tend to use more open solutions. Currently, the most popular solution is to use JADE as underlying agent infrastructure combined with some other (higher-level) approach to program the agents' behaviour. When dealing with more general frameworks (rather than tied to a platform), their use (i.e., defining the agents that will run within it, together with the required services and resources) should be automated as much as possible, in part to free the developer from low-level details (e.g. location of the configuration files, their concrete syntax, etc.). Despite this, few existing frameworks have IDE support. Concerning the paradigm of communication used, there are several on offer, often being an important issue when choosing which framework to adopt. TuCSoN is representative of tuple-centred communication, JADE of message passing, and DESIRE of data flow among processes.

The various approaches mentioned along this survey indicate that there is still much work to be done. Among the major challenges faced by this research community are:

- The conception and development of specialised debugging tools, in particular for cognitive agent languages;
- The integration of agent tools into existing IDEs, rather than starting from scratch;
- The separation of MAS frameworks from agent platforms, so that each framework can be used for deploying systems on a variety of platforms.
- The dissemination of the MAS programming paradigm, so that programmers have a better understanding of its foundations as well as practical characteristics.

We believe that the recent developments surveyed here show a lively interest in this area of research. Despite the large number of open issues and challenges, we expect that the experience gathered in developing MAS with these tools will take us closer to a more mature programming paradigm. Arguably, this is one of the few concrete ways for allowing wider audiences to use in practice, and in a systematic way, the various techniques that the MAS research community has developed over the last two decades.

Acknowledgements

We gratefully acknowledge the help and support of AgentLink III, in particular its Technical Fora, which not only motivated the authors to work together in producing this joint survey, but also provided the conditions for much of the discussion that we used in this paper and indeed that will guide future work in this area of research. We also acknowledge the valuable comments and suggestions provided by the anonymous referees.

References

[1] J. J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Evolving logic programs. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, volume 2424 of *LNAI*, pages 50–61. Springer, 2002.

[2] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi. JADE — a java agent development framework. In Bordini et al. [5], chapter 5, pages 125–148.

[3] F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors. *Methodologies and Software Engineering for Agent Systems*. Kluwer, 2004.

[4] C. Bernon, M. Cossentino, and J. Pavon. An overview of current trends in european aose research. *Journal of Informatica*, 2005. In this volume.

[5] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer, 2005.

[6] R. H. Bordini, J. F. Hübner, et al. ***Jason***, manual, release 0.7 edition, Aug. 2005. `http://jason.sf.net/`.

[7] R. H. Bordini, J. F. Hübner, and R. Vieira. ***Jason*** and the Golden Fleece of agent-oriented programming. In Bordini et al. [5], chapter 1, pages 3–37.

[8] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf. Goal representation for BDI agent systems. In R. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Programming Multi-Agent Systems, second Int. Workshop (ProMAS'04)*, volume 3346 of *LNAI*, pages 44–65. Springer Verlag, 2005.

[9] F. Brazier, C. Jonker, and J. Treur. Principles of compositional multi-agent system development. In *Proceedings of Conference on Information Technology and Knowledge Systems*, pages 347–360. Austrian Computer Society, 1998.

[10] M. Calisti, P. Funk, S. Biellman, and T. Bugnon. A multi-agent system for organ transplant management. In A. Moreno and J. Nealon, editors, *Applications of Software Agent Technology in the HealthCare Domain*, pages 199–212. Birkhäuser Verlag, 2004.

[11] L. Cardelli and A. D. Gordon. Mobile ambients. In M. Nivat, editor, *Foundations of Software Science and Computational Structures*, volume 1378 of *LNCS*, pages 140–155. Springer, 1998.

[12] K. Clark and F. McCabe. Go! — a multi-paradigm programming language for implementing multi-threaded agents. *Annals of Mathematics and Artificial Intelligence*, 41(2–4):171–206, 2004.

[13] R. W. Collier. *Agent Factory: A Framework for the Engineering of Agent-Oriented Applications*. PhD thesis, University College Dublin, 2001.

[14] S. Costantini and A. Tocchio. A logic programming language for multi-agent systems. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, volume 2424 of *LNAI*, pages 1–13. Springer, 2002.

[15] M. Dastani, F. de Boer, F. Dignum, and J.-J. Meyer. Programming agent deliberation: An approach illustrated using the 3APL language. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 97–104. ACM, 2003.

[16] M. Dastani, M. B. van Riemsdijk, F. Dignum, and J.-J. C. Meyer. A programming language for cognitive agents: goal directed 3APL. In M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *Programming multiagent systems, first international workshop (ProMAS'03)*, volume 3067 of *LNCS*, pages 111–130, Berlin, 2004. Springer Verlag.

[17] M. Dastani, M. B. van Riemsdijk, and J.-J. C. Meyer. Programming multi-agent systems in 3APL. In Bordini et al. [5], chapter 2, pages 39–67.

[18] S. DeLoach. Analysis and design using MaSE and agenTool. In *Proceedings of Midwest Artificial Intelligence and Cognitive Science*. Miami University Press, 2001.

[19] E. Denti, A. Omicini, and A. Ricci. Multi-paradigm Java-Prolog integration in tuProlog. *Science of Computer Programming*, 2005. In press.

[20] J. Dix, S. Kraus, and V. Subrahmanian. Agents dealing with time and uncertainty. In M. Gini, T. Ishida, C. Castelfranchi, and W. L. Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 912–919. ACM Press, 2002.

[21] J. Dix and Y. Zhang. IMPACT: a multi-agent framework with declarative semantics. In Bordini et al. [5], chapter 3, pages 69–94.

[22] A. El Fallah Seghrouchni and A. Suna. An unified framework for programming autonomous, intelligent and mobile agents. In V. Marik, J. Müller, and M. Pechoucek, editors, *Proceedings of Third International Central and Eastern European Conference on Multi-Agent Systems*, volume 2691 of *LNAI*, pages 353–362. Springer Verlag, 2003.

[23] A. El Fallah Seghrouchni and A. Suna. CLAIM: A computational language for autonomous, intelligent and mobile agents. In M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Programming Multiagent Systems, first international workshop (ProMAS'03)*, volume 3067 of *LNCS*, pages 90–110. Springer Verlag, 2004.

[24] A. El Fallah Seghrouchni and A. Suna. Programming mobile intelligent agents:an operational semantics. In *Proceedings of the International Conference on Intelligent Agent Technology*, pages 65–71. IEEE Computer Society, 2004.

[25] A. El Fallah Seghrouchni and A. Suna. Himalaya Framework: Hierarchical Intelligent Mobile Agents for building Large-scale and Adaptive sYstems based on Ambients. In T. Ishida, L. Gasser, and H. Nakashima, editors, *Proceedings of Massive Multi-Agent Systems workshop*, number 3446 in LNAI, pages 202–216. Springer Verlag, 2005.

[26] R. Evertsz, M. Fletcher, R. Jones, J. Jarvis, J. Brusey, and S. Dance. Implementing industrial multi-agent systems using JACK^{TM}. In *Programming multiagent systems, first international workshop (ProMAS'03)*, volume 3067 of *LNAI*, pages 18–48. Springer Verlag, 2004.

[27] J. Gomez-Sanz and J. Pavon. Agent oriented software engineering with INGENIAS. In V. Marik, J. Müller, and M. Pechoucek, editors, *Proceedings of the Third International Central and Eastern European Conference on Multi-Agent Systems*, volume 2691 of *LNCS*, pages 394–403. Springer Verlag, 2003.

[28] K. Hindriks, F. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Int. J. of Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[29] G. Jayatilleke, L. Padgham, and M. Winikoff. Component agent framework for non-experts (CAFnE) toolkit. In R. Unland, M. Klusch, and M. Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*. Birkhäuser Publishing Company, 2005.

[30] G. Klein, A. Suna, and A. El Fallah Seghrouchni. Resource sharing and load balancing based on agent mobility. In *Proceedings of International Conference on Enterprise Information Systems*, pages 350–355. ICEIS Press, 2004.

[31] J. F. Lehman, J. E. Laird, and P. S. Rosenbloom. A gentle introduction to Soar, an architecture for human cognition. In S. Sternberg and D. Scarborough, editors, *An invitation to Cognitive Science, vol. 4*. MIT Press, 1996.

[32] J. A. Leite. *Evolving Knowledge Bases*, volume 81 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2003.

[33] J. A. Leite, J. J. Alferes, and L. M. Pereira. MINERVA — a dynamic logic programming agent architecture. In J.-J. Meyer and M. Tambe, editors, *Intelligent Agents VIII — Agent Theories, Architectures, and Languages*, volume 2333 of *LNAI*, pages 141–157. Springer, 2002.

[34] M. Luck and M. d'Inverno. *Understanding Agent Systems*. Springer Series on Agent Technology. Springer, 2nd edition, 2004.

[35] V. Mascardi, M. Martelli, and L. Sterling. Logic-based specification languages for intelligent software agents. *Theory and Practice of Logic Programming*, 4(4):429–494, 2004.

[36] F. McCabe and K. Clark. April — agent process interaction language. In M. Wooldridge and N. Jennings, editors, *Intelligent Agents, ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, volume 890 of *LNAI*, pages 324–340. Springer, 1995.

[37] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. MASIF, the OMG mobile agent system interoperability facility. In *Proceedings of Mobile Agents'98*, volume 1477 of *LNAI*, pages 50–67. Springer, 1998.

[38] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, Nov. 2001.

[39] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Int. J. of Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, 1999.

[40] T. O. Paulussen, A. Zöller, A. Heinzl, A. Pokahr, L. Braubach, and W. Lamersdorf. Dynamic patient scheduling in hospitals. In M. Bichler, C. Holtmann, S. Kirn, J. Müller, and C. Weinhardt, editors, *Coordination and Agent Technology in Value Networks*. GITO, Berlin, 2004.

[41] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN 19, Department of Computer Science, Arhaus University, 1981.

[42] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In Bordini et al. [5], chapter 6, pages 149–174.

[43] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of Modelling Autonomous Agents in a Multi-Agent World*, number 1038 in LNAI, pages 42–55. Springer Verlag, 1996.

[44] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *Proceedings of International Conference on Multi Agent Systems*, pages 312–319. AAAI Press / MIT Press, 1995.

[45] C. F. B. Rooney, R. W. Collier, and G. M. P. O'Hare. VIPER: A visual protocol editor. In R. D. Nicola, G. Ferrari, and G. Meredith, editors, *Proceedings of the International Conference on Coordination Models and Languages*, volume 2949 of *LNCS*, pages 279–293. Springer Verlag, 2004.

[46] R. Ross, R. Collier, and G. O'Hare. AF-APL: Bridging principles and practices in agent oriented languages. In R. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Programming Multi-Agent Systems, second Int. Workshop (ProMAS'04)*, volume 3346 of *LNCS*, pages 66–88. Springer Verlag, 2005.

[47] G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos. Self-organisation and emergence in mas: An overview. *Journal of Informatica*, 2005. In this volume.

[48] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[49] V. Subrahmanian, P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Özcan, and R. Ross. *Heterogenous Active Agents*. MIT-Press, 2000.

[50] A. Suna and A. El Fallah Seghrouchni. Adaptative mobile multi-agent systems. In *Proceedings of International Central and Eastern European Conference on Multi-Agent Systems*, LNAI, 2005. To appear.

[51] A. Suna and A. El Fallah Seghrouchni. A mobile agents platform: architecture, mobility and security elements. In R. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Programming Multi-Agent Systems, second Int. Workshop (ProMAS'04)*, volume 3346 of *LNAI*, pages 126–146, New-York, 2005. Springer Verlag.

[52] A. Suna, C. Lemaitre, and A. El Fallah Seghrouchni. E-commerce using an agent oriented approach. *Revista Iberoamericana de Inteligencia Artifical*, 9(25):89–98, 2005.

[53] M. Thielscher. FLUX: A logic programming method for reasoning agents. *Theory and Practice of Logic Programming*, 2005. To appear.

[54] M. Thielscher. *Reasoning Robots: The Art and Science of Programming Robotic Agents*, volume 33 of *Applied Logic Series*. Springer, 2005.

[55] R. Vieira, A. Moreira, M. Wooldridge, and R. H. Bordini. On the formal semantics of speech-act based communication in an agent-oriented programming language. *To appear*, 2005.

[56] D. Weyns and T. Holvoet. On the role of environments in multiagent systems. *Journal of Informatica*, 2005. In this volume.

[57] M. Winikoff. JACK^{TM} intelligent agents: An industrial strength platform. In Bordini et al. [5], chapter 7, pages 175–193.

Self-Organisation and Emergence in MAS: An Overview

Giovanna Di Marzo Serugendo
University of Geneva, Switzerland
E-mail: Giovanna.Dimarzo@cui.unige.ch

Marie-Pierre Gleizes
IRIT, Université Paul Sabatier, France
E-mail: Marie-Pierre.Gleizes@irit.fr

Anthony Karageorgos
University of Thessaly, Greece
E-mail: karageorgos@computer.org

Keywords: multi-agent systems, self-organisation, emergence

Received: June 5, 2005

The spread of the Internet and the evolution of mobile communication, have created new possibilities for software applications such as ubiquitous computing, dynamic supply chains and medical home care. Such systems need to operate in dynamic, heterogeneous environments and face the challenge of handling frequently changing requirements; therefore they must be flexible, robust and capable of adapting to the circumstances. It is widely believed that multi-agent systems coordinated by self-organisation and emergence mechanisms are an effective way to design these systems. This paper aims to define the concepts of self-organisation and emergence and to provide a state of the art survey about the different classes of self-organisation mechanisms applied in the multi-agent systems domain. Furthermore, the strengths and limits of these approaches are examined and research issues are provided.

Povzetek: Članek opisuje pregled samoorganizacije v MAS.

1 Introduction

Natural self-organising systems function without central control and operate based on contextual local interactions. The particularity of self-organised systems is their capacity to spontaneously (without external control) produce a new organisation in case of environmental changes. These systems are particularly robust, because they adapt to these changes, and are able to ensure their own survivability. In some cases, self-organisation is coupled with emergent behaviour, in the sense that although individual components carry out a simple task, as a whole they are able to carry out complex tasks emerging in a coherent way through the local interactions of the various components.

The complexity of today's applications is such, e.g. world scale, that no centralised or hierarchical control is possible. In other cases, it is the unforeseeable context, in which the application evolves or moves, which makes any supervision difficult. Therefore, we are witnessing an increased interest from both the academic community and the industry in naturally inspired (robust and simple) solutions for building modern applications favouring self-organisation and/or emergence of properties.

We can foresee that among the applications of tomorrow, a great many of them will be biologically inspired: self-organising sensors networks, allowing the control of aerospace vehicles, or of dangerous zones; self-organising traffic management, allowing re-routing of emergency vehicles, or individual cars; storage facilities, or self-managing operating systems facilities. Some others applications tackle with complex problem solving in which complexity is due to the great space search such as optimisation problems and non linear problems.

Software agents naturally play the role of autonomous entities subject to self-organise themselves. Usually agents are used for simulating self-organising systems, in order to better understand or establish models. The tendency is now to shift the role of agents from simulation to the development of distributed systems where components are software agents that once deployed in a given environment self-organise and work in a decentralised manner towards the realisation of a given (global) possibly emergent functionality.

Sections 2 and 3 review the notions of self-organisation and emergence respectively. Section 4 provides the description of several implementations in MAS. Section 5 discusses the strengths and limits of self-organising approaches. The main problems and challenges related to the software engineering of self-organising systems which exhibit emergent properties are discussed in Section 6. Finally, Section 7 concludes the paper.

2 Self-Organisation

2.1 History

By studying the social behaviour of insects (termites), Grassé [29] proposed in 1959 the theory of *stigmergy*, which can be summarised in "*the work excites the workers*". The consequence is that direct interactions are not necessary to coordinate a group, for example indirect communications through environment are enough. Coordination and regulation tasks are realised on the basis of information deposited into the environment, without central control. In the case of ants and termites, stigmergy is ensured by depositing a chemical substance in the environment, called *pheromone*.

In the 70es, the term self-organisation itself has been established by Nobel Prize Ilya Prigogine [26] and his colleagues through thermodynamics studies. Essentially the idea is that open systems decrease their entropy (order comes out of disorder) when an external energy is applied on the system. Matter organises itself under this external pressure to reach a new state where entropy has decreased. Compared to the stigmergy concept identified by Grassé, there is a fundamental difference here. Indeed, in the first case self-organisation results from a behaviour occurring from inside the system (from the ants or termites themselves). In the second case, self-organisation is the result of a pressure applied from the outside on the system.

In the 70es, through biological studies, Francisco Varela [61] established the notion of autopoiesis (meaning self-production) as being the process through which an organisation is able to produce itself. Autopoiesis applies to closed systems made of autonomous components whose interactions self-maintain the system through the generation of system's components, such as living systems (cells, or organisms).

Koestler [37] in the late 60es established the definition of holons and holarchies. Holons are at the same time whole systems and parts of larger systems. Holarchies are hierarchies of such holons. Koestler gives a hierarchical view of self-organisation, which applies to the universe or to enterprise organisations. The idea here is that, for complex organisations, order appears from disorder, due to simple relations that statistically evolve through complex relations progressively organising themselves.

During the last 20 years, research in artificial systems has been oriented towards introducing self-organisation mechanisms specifically for software applications. These different works take diverse inspiration: from stigmergy, to autopoiesis, or to the holon concept. Recently, in addition to reproducing natural system behaviour into artificial systems, recent research efforts have been oriented towards introducing self-organisation mechanisms specifically for software applications [20] . Section 4 describes such mechanisms in more details.

2.2 Examples

Natural self-organising systems include well-known examples concerning social insects, such as ants, termites and honey bees. Communication occurs through stigmergy by the means of pheromone deposited into their environment. Other collective behaviours of animals referred to as self-organising are flocks of birds, and schools of fish. By following simple rules, such as getting close to a similar bird (or fish) but not too much, getting away from dissimilar birds (or fishes), they are able to collectively avoid predators.

Social behaviour of humans is also self-organised and gives rise to emergent complex global behaviours. Human beings typically work with local information and through local direct or indirect interactions producing complex societies.

Biology provides a great source of self-organising systems as well. Examples include the immune system of mammalians, the regeneration of cells and brain behaviour.

Among artificial multi-agent based self-organising systems, we observe different trends ranging from application of naturally-inspired self-organising models, to the establishment of new mechanisms and whole infrastructures supporting self-organisation of artificial systems. Swarms provide a great source of inspiration, especially for fixed and mobile networks systems management [11] , such as routing, load balancing [43] , or security [25] . Holarchies as well have inspired researchers dealing with e-Government and e-Society issues [60] . At the level of whole infrastructures (middleware) supporting artificial self-organising systems, some works take their inspiration from magnetic fields [40] , or ants [2] .

2.3 Definition

Self-organisation essentially refers to a spontaneous, dynamically produced (re-)organisation. We present here several definitions corresponding to the different self-organisation behaviours identified in Section 2.1.

Swarm Intelligence. According to [Bonabeau, 1999] mechanisms identifying swarms behaviour are: 1. Multiple interactions among the individuals; 2. Retroactive positive feedback (increase of pheromone when food is detected); 3. Retroactive negative feedback (pheromone evaporation); 4. Increase of behaviour modification (increase of pheromone when new path is found).

Decrease of entropy. Prigogine and his colleagues have identified four necessary requirements for systems exhibiting a self-organising behaviour under external pressure [26] . "Mutual Causality: At least two components of the system have a circular relationship, each influencing the other. Autocatalysis: At least one of the components is causally influenced by another component, resulting in its own increase. Far-from equilibrium condition: the system imports a large amount of energy from outside the system, uses the energy to help renew its own structures (autopoiesis), and dissipates rather than accumulates, the accruing disorder

(entropy) back into the environment. Morphogenetic changes: At least one of the components of the system must be open to external random variations from outside the system. A system exhibits morphogenetic change when the components of the system are themselves changed [15] ."

Autopoiesis. "An autopoietic system is organised (defined as a unity) as a network of processes of production (transformation and destruction) of components that produces the components that: 1. Through their interactions and transformations continuously regenerate and realise the network of processes (relations) that produced them; and 2. Constitute it (the machine) as a concrete unity in the space which they [the components] exist by specifying the topological domain of its realisation as such a network" [61] .

Artificial Systems. Works of the Agentlink Technical Forum on Self-Organisation in MAS [21] have established two definitions of self-organising systems: 1. "Strong self-organising systems are systems that change their organisation without any explicit, internal or external, central control"; 2. "Weak self-organising systems are systems where reorganisation occurs as a result of internal central control or planning".

Furthermore, self-organisation implies organisation, which in turn implies some ordered structure and component behaviour. In this respect, the process of self-organisation changes the respective structure and behaviour and a new distinct organisation is self-produced.

When self-organisation meets emergence. Emergence is the fact that a structure, not explicitly represented at a lower level, appears at a higher level. In the case of dynamic self-organising systems, with decentralised control and local interactions, intimately linked with self-organisation is the notion of emergent properties. The ants actually establish the shortest path between the nest and the source of food. However in the general case, as pointed out by [18] self-organisation can be witnessed without emergence and vice-versa.

3 The Emergence Concept

3.1 History

The emergent phenomena are studied since the Greek antiquity and can be found in the writings of the Socrate periods with the notion of "the whole before the parts" or "the whole is more than all the parts". There were two different schools for studying the emergence: the proto-emergentism during the XIX century and the neo-emergentism during the XX century.

The proto-emergentists consider the emergent process as a black box (see Figure 1). Only the inputs and the outputs at the lowest level can be discerned. We don't know how the entries are transformed in outputs. Researchers such as: G.H. Lewes, C.L. Morgan, J.S. Mill, S. Alexander, D. Broad, W. Wheeler, and A.N. Whitehead try to explicit the characteristics of emergent phenomena.

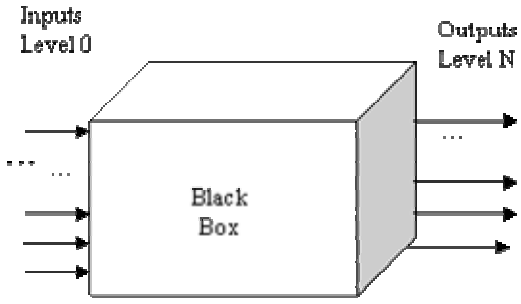


Figure 1: Proto-emergentist view

From 1930 until just now, a different perspective has been envisaged, by a movement called the neo-emergentism. It has its root in dynamic of systems in Physics, in Mathematics and in Computer Science with main examples being the work of Haken, Holland, Kauffman, Langton, Prigogine, and Thom. Its aim is to develop tools, methods and constructions which enable the expression of the emergent process as less dense and by consequence as less miraculous (Figure 2). This movement tries to understand and to reproduce the process which leads to emergence.

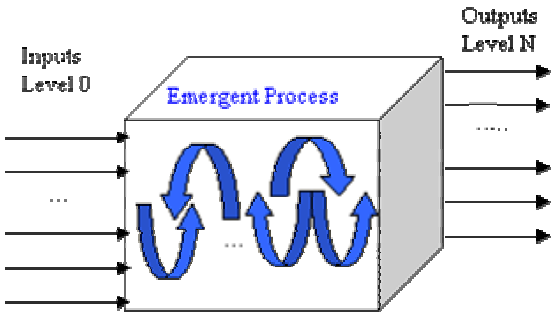


Figure 2: Neo-emergentist view

3.2 Examples

To illustrate the notion of emergent phenomena, this section presents examples of systems where emergent phenomena can be observed.

The first example is taken from natural systems and concerns foraging ants [16] . A foraging ant has the role to explore an environment to find food. When it finds food it comes back to the nest in tracing the path in the environment with pheromone. The shortest path to find food is the structure which emerges from the collective activity of the ants. This path has reality only for an observer of the system and an ant does not view it.

Another example concerns an application where robots have to transport boxes from always the same departure room to a destination one. Two corridors are available to go from one room to the other and two robots cannot cross in a corridor and there is no sense

associated to them. The robots have a local perception. In using cooperative attitude to embody robots, we can observe corridors dedication and traffic way [50] .

The apparition of conscience is an example of emergent phenomenon for humans. The conscience is viewed by Searle [57] as a property of the brain at the higher or global level. Biologically, the brain is a complex system composed of a set of neurons and interactions between them. These neurons are the lower or micro level. Nowadays, we cannot understand or explain the conscience in observing the neurons and their interactions.

3.3 Definition

The emergence is a captivating concept and we try to explain it in answering the following questions:

- What does emerge?
- What are the characteristics of an emergent phenomenon? These characteristics must enable to answer yes or no to the question: “is this phenomenon an emergent one?”
- What are the properties of a system producing emergent phenomena? These properties can guide designers to build systems which provide emergent phenomenon.
- What can be emergence in artificial systems? How can you decide if a program provides an emergent result or not?

The object of emergence is often called phenomenon and it can be a structure or a framework such as the Bénard’s cells, a behaviour such as the glider in the game of life [4] , or a function (not as mathematic function but as the functionality of a system) such as the building of a course schedule by several local entities [44] , [52] .

An emergent phenomenon requires at least two levels (a micro and a macro level), and needs to be observable at least at the macro level. Its main property is the irreducibility of the properties of a high level theory to properties of a lower level theory [1] . In general, there are interdependencies between the levels, the macro level constrains the micro level and the micro level causes the macro level. The phenomenon must show novelty: something new is produced that did not exist previously; must be ostensible; and must produce some coherence in the sense that it has its own identity but it is strongly linked to parts that produce it [28] . A chain of linear activities enables explanation and predictability of a collective phenomenon. On the opposite, an emergent one needs non linear activities at the micro-level. For a given phenomenon, if most of the previous properties can be observed then the phenomenon can be qualified as emergent.

Engineers should be provided with a guide including models, tools and methods to design systems having an emergent behaviour or presenting emergent results. Furthermore, the guide should list the main properties such systems should have. To provide emergent phenomena, a system or a mechanism must at least have two levels. The system must present a dynamic during its time life. Because an emergent phenomenon is observable during time, it needs a form of self-maintained equilibrium. Nevertheless it is not a homeostatic but dynamic equilibrium. Emergence occurs in a narrow possibility space lying between conditions that are too ordered and too disordered. This boundary or margin is the edge of chaos [36] , which is always far from equilibrium. Near these equilibriums, a system has the ability to self-organise allowing an emergent phenomenon.

The emergence in artificial system is conceptually close to emergent computation defined by Stephanie Forrest [24] as follows:

- a collection of interactive agents : the process;
- an epiphenomenon produced by this process at the macro level;
- a natural interpretation of this epiphenomenon as computation or computation results.

An operational definition is given by the SMAC team at IRIT [13] . This “technical” definition of emergence has strong computer science coloration and it is based on two points:

1. **The subject.** The goal of a computational system is to realise an adequate function, judged by a relevant user. It is this function, which may evolve during time, that has to emerge.
2. **The condition.** This function is emergent if the coding of the system does not depend in any way of the knowledge of this function. This coding has to contain the mechanisms allowing the adaptation of the system during its coupling with the environment, so as to tend anytime towards the adequate function.

Therefore, when we design an agent for a multi-agent system, the code of the agent doesn’t contain any knowledge of the collective function we want the MAS to compute. As a result, no agent controls the global system.

4 Implementation in MAS

Studies on self-organisation and emergence focus on naturally inspired approaches (bio-inspired approaches [41] , and socially-based approaches [30]) and non naturally inspired approaches. Researchers have been experimented with several mechanisms leading to self-organisation and often at the same time to emergent phenomenon on different kinds of applications [5] . The different approaches can be divided in five classes depending on the mechanisms they are based on:

- direct interactions between agents using basic principles such as broadcast and localisation;
- indirect interactions between agents and stigmergy;
- reinforcement of agent behaviours;
- cooperation behaviour of individual agents;
- choice of a generic architecture.

For a more general survey of languages and platforms for MAS implementations not directly related to self-

organising mechanisms, the interested reader can refer to [9] .

4.1 Mechanisms based on direct interactions

Zambonelli et al. [64] discuss different ways to engineer self-organisation. The approaches proposed consist in using few basic principles, such as localisation and broadcast, coupled with local interactions and local computations done by agents in order to provide a final coherent global state. These algorithms differ from traditional distributed algorithms in that they focus on ensuring that they eventually will converge to and maintain a desired stable state despite micro-level contingencies and any perturbations in the environment, for example changes in the network structure.

Typical examples of such mechanisms are those applied in the areas of self-assembly and distributed self-localisation where the formation of regular spatial patterns in mobile objects is required. An example is described in [40] where simple leader election algorithm determines the centre of gravity of the objects and propagates it to all objects which move until a specific distance from the centre is reached. The result eventually is a circular organisation of objects. The same mechanism is used in the system for modelling fluid dynamics [58] . Local interactions between drops and interactions with a physical environment enable the formation of rivers or ponds.

These mechanisms focus on changing the structural aspects of the agent organisation, such as topological placement of agents and agent communication lines.

4.2 Mechanisms based on stigmergy

The self-organisation mechanisms based on the stigmergy concept aim at achieving complex system behaviours resulting of indirect interactions between agents. These interactions are due to changes in the environment. This behaviour leads towards the desired global system behaviour.

Recently, several approaches to self-organisation relying on this idea of stigmergy have been proposed and their effectiveness in achieving difficult global coordination tasks has been demonstrated. For instance, this mechanism has been used for manufacturing control [35] , supply network management [55] , managing computer networks security [25] and coordination of unmanned vehicles [48] . Stigmergy has also been implemented with social spiders to detect regions in a scene [10] . This principle is also used to obtain the formation of non-symmetric patterns in self-assembly applications [40] which in some cases are not exactly known in advance but emerge during system execution [53] . An example of such non-symmetrical pattern formation using principles of biological formation of morphogenesis is given in [40] .

These mechanisms can be evaluated by experimentation, for example by simulation and prototyping [23] , [38] . In particular there is a tendency to integrate simulation experiments in the methodologies for engineering such systems, such as the one described in [47] . In such approaches, the design phase involves selecting an appropriate self-organising model and verifying its correctness via experimentation. Such a model may be relevant, but not necessarily the most suitable for the particular application scenario. Therefore, the model is calibrated via iterative refinement based on the experimentation results.

In these cases, due to the non-linearity and the complexity of the phenomena involved, neither it is possible to have direct control of the system behaviour nor can it be proven that the desired behaviour will be achieved. Furthermore, the resulting system state cannot be accurately known in advance and multiple solutions can be reached. One can only obtain some statistical confidence about the system convergence to the desired globally coordinated behaviour with experimentation.

4.3 Mechanisms based on reinforcement

In some approaches self-organisation is based on the capabilities of the agents to modify dynamically their behaviour according to some reinforcement. It consists in the following basic principles: rewards increase agent behaviour and punishments decrease agent behaviour. The consequence is that an individual agent can adapt its capabilities and we can observe specialisation of roles for example. In these approaches self-organisation is based on adaptive behaviour capabilities of individual agents which are dependent on particular agent architectures. In these approaches, agents dynamically select a new behaviour (or action) based on the calculation of a probability value which is dependent on the current agent state and the perceived state of the environment, as well as on the quality of the previous adaptation decisions, for example the ones discussed in [39] and [17] . Other early approaches to self-organisation that re-assign roles and responsibilities to different organisational nodes are detailed in [49] .

A typical example of this approach is the model of adaptive agents described in [62] . The model focuses on dynamically adapting logical relations between different behaviours, represented by roles, an agent can successively follow starting from its current state. These relations are used to select the new agent behaviour when adaptation of behaviour needs to be made. Agent behaviour is described as a graph termed *behaviour graph*. A behaviour graph includes two types of nodes corresponding to *roles* and *links*. Role nodes are connected to each other only via appropriate link nodes, which contain *conditions* specifying when the agent can switch between the respective roles. Adaptive role selection takes place on runtime based on *factors* associated with the links of the behaviour graph. Factors are parameters representing properties of agents and their perceived environment whose values can change dynamically during agent execution.

4.4 Mechanisms based on cooperation

The *Organisation Self-Design* (OSD) framework [33] uses the primitives of agents composition and decomposition. Decomposition involves division of an

agent into two and can be performed to respond to overwhelming environmental demands. Composition merges two agents into one and can be useful when communication overheads between the two agents are too high. The system tries to be cooperative with its environment in creating one agent or in merging two agents in order to improve the response time to the environment. The initial organisation starts with one agent containing all domain and organisational knowledge. Simulation results demonstrate the effectiveness of the approach in adapting to changing environmental demands.

Cooperation is also used in the AMAS theory [27] where the desired collective behaviour emerges, and can always occur as the result of cooperation [13] , [14] . This emergent outcome corresponds to the delivered system functionality (referred to as the *global function*), which is only modelled using emergence; in other words there is no agent having a global view of the system status or purpose and no centralised control. Each agent possesses the ability of self-organisation, for example the capability to locally rearrange its interactions with other agents and the environment depending on its knowledge, on its representation of the others and on the individual task it has to solve. This enables realising dynamic changes in the global system function without explicitly coding the modifications at the upper level of the system. Self-organisation is founded on the capability agents possess to be locally "cooperative". Cooperation capabilities do not imply that agents are always helpful or altruistic but they are able to recognise cooperation failures called *Non Cooperative Situations (NCS)* (which correspond to exceptions found in classical programs) and handle them. The local handling of NCS maximises the flexibility and adaptation capability of the system to unexpected situation occurring due to the dynamism of the agent interactions and the environment.

4.5 Mechanisms based on generic architecture

A particular class of self-organisation mechanisms is based on generic reference architectures or meta-models of the agents' organisation which are instantiated and subsequently dynamically modified as needed according to the requirements of the particular application.

Examples of reference architectures are the mediator architecture proposed by Maturana and Norrie [42] and the PROSA [8] architecture which are both based on the holonic hierarchy model. The holonic hierarchy model involves structural patterns that form nested hierarchies of self-replicating structures named *holarchies* [37] . The elements of holonic systems are referred to with the term *holon* which is a combination of the Greek word *holos,* meaning "whole", with the suffix "on" meaning part as in proton or neuron.

A common aspect in reference architectures is that they involve characteristic agent types from which the basic agents of a holonic organisation are derived. For example, the mediator reference architecture is based on the mediator agent type. In PROSA [8] the holonic organisation consists out of three types of basic holons — order holons, product holons, and resource holons. When agents are organised according to the holonic metaphor they participate in holons forming holonic structures. Self-organisation then refers to altering the holonic hierarchy following perturbations of the agent environment using a known decision making technique such as fuzzy-evolutionary reasoning [59] .

Examples of approaches based on meta-models and architectural reflection are presented in [22] and [54] . In such approaches, the current system architecture organisation is described as a particular configuration of a generic architectural meta-model which provides the architectural components and their features and also an associated set of architectural constraints that define how and when to safely reconfigure the software architecture.

The meta-model configuration can be inspected and modified at run-time. Modifications of the architecture meta-model result in modifications of the software architecture itself, and the architecture is therefore reflective. Such dynamic modifications can take place either automatically, as is the case in [22] or after user intervention as is done in [54] . The common technique for representing such architectural meta-models is as a typed, directed configuration graph.

5 Strengths and Limits

Mechanisms based on direct interactions have the significant advantage that they enable the design of specific robust self-organised behaviours with exactly known outcomes. However, as mentioned in [64] these approaches are needed only to a limited number of applications. The reason is that only simple global equilibrium states (or patterns of activity) that can be modelled in simple linear terms can be achieved. As a result when more complex behaviour involving non-linear interactions is needed then either too many restrictions for the system operation need to be made or direct mechanisms cannot be applied.

The mechanisms based on stigmergy have additional advantages. Firstly, they enable increased reusability since they make possible to reuse the strengths of known self-organisation mechanisms from biology to build self-organising software. Secondly, once modelling and experimentation for the purposes of calibration has been carried out, the simulation models can be the basis for the actual implementation, reducing thus development time and resources required and hence facilitating development. Furthermore, the simple local behaviours they are based on are quite easy to implement, resulting in increased ease of programming. Furthermore, the multi-solution capability of these mechanisms is one of their strengths since it increases their robustness. Furthermore, although suboptimal solutions are more likely to occur, the effectiveness of these mechanisms is relatively high compared to their low development cost.

The mechanisms based on cooperation behaviour, enable to treat applications with continuous or discontinuous global behaviour. The bottom-up design simplifies also the development and the resulting systems

are robust, because adaptive. For instance, the AMAS theory guarantees that the system only adapts its behaviour to be cooperative with its environment and to satisfy it. The difficulty lies in the exhaustive list of all the non cooperative situations an agent can be faced on. Nevertheless, this is always theoretically feasible because the number of non cooperative situations related to the agent skills is enumerable.

However there are also disadvantages which are essentially related with harnessing emergent behaviour. Firstly, it is currently not possible to effectively control the behaviour of such systems. As a result it is common for undesired emergence states to occur [47] . Furthermore, there can be cases where specific global states are required to emerge, such as the positions of robot players in a football game and hence the many possible solutions offered by such mechanisms can be a problem. A relevant case is when a global solution has emerged and then it is only desirable to maintain it via self-organisation and not converging to another one.

The rest of the mechanisms have similar limitations. An additional strength of mechanisms based on adaptive architectures and meta-mechanisms is that modelling is done using agent-oriented software engineering terms which increases ease of understanding by software designers – in contrast to applying a model from another discipline which would require them to obtain the necessary knowledge to master the terminology and the concepts involved. However this comes to the expense of increased difficulty in modelling global emergent behaviour.

6 Problems and Challenges

From a multi-agent systems development point of view, the central question is: how to program single agents so that, when taken as a whole, they self-organise. In the particular case of multi-agent systems, the interest and the difficulty lies in having both self-organisation and emergent properties, mainly emergent functionality that arises from individual simple tasks performed by the agents. Therefore, the engineering of self-organising applications needs means to define a global goal, and to design local behaviours so that the global behaviour emerges. This is difficult, because the global goal is *not* predictable as the sum or a function of the local goals. Consequently, the verification task turns out to be an arduous exercise, if not realised through simulation.

Traditional software engineering techniques are insufficient, since they are based on interfaces fixed at design time, or well established ontology. As for current methodologies, they only make it possible to define a global behaviour when it is a function of the behaviour of the various parts.

Traditional practices in multi-agent systems introduce basic techniques for autonomously interacting or retrieving information, such as agent coordination, service description, or ontology [6] . However, these techniques rely on pre-programmed interaction patterns, preventing adaptation to unexpected environmental changes. Current engineering practices, which directly address self-organisation, consist in designing distributed algorithms taking inspiration from natural mechanisms, both bio-inspired and socially-inspired. Some agent-oriented methodologies such as ADELFE [51] provide to designer means to design self-organising systems. More recently, specific electronic interaction mechanisms, non-naturally inspired, are being defined, and middleware technology developed, that will help the development of self-organising applications. However, verification and whole engineering methods remain open issues.

Currently, it is necessary to find means to “control” emergence to use it to solve problems. It is antinomic to speak about emergence and about control on the emergence. But, when designing artificial systems, it is necessary to have operational definition and tools to enable such systems to produce the wanted emergent phenomenon.

In addition the environment plays an important role both as a coordination media and as source of changes and adaptation for the agents. The environment, its engineering and its role in self-organising systems must be well understood and not be underestimated. For a deeper discussion on environments, the interested reader may refer to [63] .

A research axis will be to find new principles, theories, models, mechanisms and methodologies to engineer self-organising systems with or without emergent phenomena. In this perspective it is important to be aware of the differences, and to distinguish solutions that tackle self-organisation issues only (without intended causal emergence); emergent issues only (without self-organisation), and solutions that intend to consider both cases in the resulting system. However, in any cases, this is a delicate problem, in the sense that unintended emergent phenomenon that have a causal effect on the system may always arise.

The growing complexity of applications needs solutions that favour autonomous, robust and adaptive systems. Natural systems must be an inspiration sources but we have to devise really new techniques, mechanisms to design self-organisation and emergent phenomenon. This new wave of systems can be called neo-computation and will be useful for designing applications in the domains such as autonomic computing, pervasive and ubiquitous computing.

7 Conclusion

Self-organisation and emergence interest more and more the community of computer scientists and in particular the MAS developers. This craze is due to the fact that self-organisation enables to tackle a new field of applications and that multi-agent systems are well adapted to implement self-organisation.

The paper aims are twofold: it clarifies these two concepts and proposes operational definitions; it then gives an overview of researches on self-organising MAS and emergent phenomena produced by MAS. The different mechanisms studied can be grouped into five families: direct mechanisms, characterized by simple principle of functioning in the agents and direct

communication; mechanisms based on stigmergy, which use indirect interactions between agents and where the perceptions reinforce some agent actions; reinforcement mechanisms, which enable designing adaptive agents that change their roles or their behaviour in runtime; cooperative attitude of agents; and predefined architecture of the system. The paper ends in proposing some research axis such as finding new mechanisms, developing methods to design self-organising systems, providing means to control the global behaviour of the system, or proving convergence.

8 Acknowledgement

This work is partly supported by the Swiss NSF grant 200020-105476/1 and Agentlink III.

References

[1] S. Ali R. Zimmer and C. Elstob. The question concerning emergence: implication for artificiality. In D.M. Dubois (Ed.), *First Computing Anticipatory Systems Conference* (CASYS'97), CHAOS, Liège, Belgium, 1998.

[2] O. Babaoglu, H. Meling and A. Montresor. Anthill: a framework for the development of agent-based peer-to-peer systems. In *Proceedings of the 22th International Conference on Distributed Computing Systems (ICDCS '02),* pp. 15-22, IEEE Computer Society, Los Alamitos, CA, USA, 2002.

[3] Y. Bar-Yam. Dynamics of Complex Systems. *Perseus Books*, Cambridge, MA, 1997.

[4] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for your Mathematical Plays*, Volume 2, 2nd edition. AK Peters, Ltd., Wellesley, MA, 2001.

[5] C. Bernon, V. Chevrier, V. Hilaire, P. Marrow, Applications of self-organising multi-agent systems: an initial framework of comparison. *Informatica,* Ljubljana, Slovenia. In press, 2005.

[6] C. Bernon, M. Cossentino, J. Pavon. An Overview of Current Trends in European AOSE Research. *Informatica,* Ljubljana, Slovenia. In press, 2005.

[7] E. Bonabeau, M. Dorigo, and G. Théraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, New York, NY, USA, 1999.

[8] L Bongaerts. *Integration of Scheduling and Control in Holonic Manufacturing Systems*, PhD Thesis, Katholieke Universiteit Leuven, 1998

[9] R. Bordini et al. A survey on languages and platforms for MAS implementation. *Informatica,* Ljubljana, Slovenia. In press, 2005.

[10] C. Bourjot, V. Chevrier, and V. Thomas. A new swarm mechanism based on social spiders colonies: from web weaving to region detection. *Web Intelligence and Agent Systems,* 1(1): 47-64, IOS Press, Amsterdam, The Netherlands, 2003.

[11] S. Brueckner and H.V. Parunak. Self-organising MANET management. *Engineering Self-Organising Applications Systems*. G. Di Marzo Serugendo et al. (Eds), Lecture Notes in Artificial Intelligence, volume 2977, pp. 20-35. Springer-Verlag, Berlin, 2004.

[12] S. Camazine, J.-L. Deneubourg, R. F. Nigel, J. Sneyd, G. Téraulaz, and E. Bonabeau. *Self-Organisation in Biological System*. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ, USA, 2001.

[13] D. Capera, J-P. Georgé, M-P. Gleizes, P. Glize. Emergence of organisations, emergence of functions. *In AISB'03 symposium on Adaptive Agents and Multi-Agent Systems,* pp 103 – 108, D. Kudenko, D. Kazakov, and E. Alonso (Eds), University of Wales, Aberystwyth, 2003

[14] D. Capera, J. P. Georgé, M.-P. Gleizes, and P. Glize. The AMAS theory for complex problem solving based on self-organising cooperative agents. *International Workshop on Theory and Practice of Open Computational Systems (TAPOCS). Twelfth International IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2003)*, pp. 383-388. IEEE Computer Society Press, Los Alamitos, CA, 2003.

[15] N. S. Contractor and D. R. Seibold. Theoretical frameworks for the study of structuring processes in group decision support system - adaptive structuration theory and self-organising systems theory. *Human Communication Research*, 19(4):528-563, 1993.

[16] J-L., Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, L. Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots - Simulation of animal behaviour. *Proceedings of the first international conference on simulation of adaptive behaviour.* J.A. Meyer and S. Wilson (Eds), pp. 356-363, MIT Press, Cambridge, MA, USA, 1991.

[17] T. De Wolf and T. Holvoet. Adaptive behaviour based on evolving thresholds with feedback. *Proceedings of the AISB'03 Symposium on Adaptive Agents and Multiagent Systems*, pp. 91-96, D. Kudenko, D. Kazakov, and E. Alonso (Eds.), University of Wales, Aberystwyth, 2003.

[18] T. De Wolf and T. Holvoet. Emergence and self-organisation: a statement of similarities and differences. *Engineering Self-Organising Systems.* S. Brueckner et al. (Eds), Lecture Notes in Artificial Intelligence, volume 3464, pp. 1-15, Springer-Verlag, Berlin, 2005.

[19] G. Di Marzo Serugendo et al. Self-organising applications: paradigms and applications. *Engineering Self-Organising Systems*. G. Di Marzo Serugendo et al. (Eds), Lecture Notes in Artificial Intelligence, volume 2977, pp. 1-19. Springer-Verlag, Berlin, 2004.

[20] G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli (Eds). *Engineering Self-Organising Systems.* Lecture Notes in Artificial

Intelligence, volume 2977, Springer-Verlag, Berlin, 2004.

[21] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos. *AgentLink First Technical Forum Group Self-Organisation in Multi-Agent Systems*, AgentLink Newsletter, Issue 16, ISSN 1465-3842, pp. 23-24, 2004.

[22] J. Dowling and V. Cahill. The K-Component Architecture Meta-Model for Self-Adaptive Software. *Proceedings of Reflection 2001*. Lecture Notes in Computer Science, volume 2192, pp. 81-88. Springer-Verlag, Berlin, 2001.

[23] B. Edmonds. Using the experimental method to produce reliable self-organised systems. In *Engineering Self-Organising Systems.* S. Brueckner et al. (Eds), Lecture Notes in Artificial Intelligence, volume 3464, pp. 84-99. Springer-Verlag, Berlin, 2005.

[24] S. Forrest. Emergent computation: self-organising, collective, and cooperative phenomena in natural and artificial computing network. *Proceedings of the Ninth annual CLNS conference,* 1990.

[25] N. Foukia. IDReAM: Intrusion Detection and Response executed with Agent Mobility. *The International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05)*, pp 264-270, Utrecht, The Netherlands, 2005.

[26] P. Glansdorff and I. Prigogine. *Thermodynamic study of Structure, Stability and Fluctuations*. Wiley, 1971.

[27] M.-P. Gleizes, V. Camps, and P. Glize. A theory of emergent computation based on cooperative self-organisation for adaptive artificial systems. *Fourth European Congress of Systems Science*. Valencia, 1999.

[28] J. Goldstein. Emergence as a construct: history and issues. *Emergence* 1(1):49-72, ISCE Publishing, Mansfield, MA, USA 1999.

[29] P. Grassé. La reconstruction du nid et les interactions inter-individuelles chez les bellicositermes natalenis et cubitermes sp. La théorie de la stigmergie: essai d'interprétation des termites constructeurs. *Insectes Sociaux*, 6:41-83, 1959.

[30] S. Hassas, G. Di Marzo Serugendo, A. Karageorgos, C. Castelfranchi. Self-organising mechanisms from social and business/economics approaches, *Informatica,* Ljubljana, Slovenia. In press, 2005.

[31] J.H. Holland. *Emergence – from order to chaos*. Addison-Wesley, Boston, MA, 1997.

[32] O. Holland and C. Melhuis. Stigmergy, self-organisation, and sorting in collective robotics. *Artificial Life*, 5(2):173-202. MIT Press, Cambridge, MA, USA, 1999.

[33] T. Ishida, L. Gasser, and M. Yoko. Organisation self-design in distributed production systems. *IEEE Transactions on Knowledge and Data Engineering*, 4(2): 123-134, IEEE Computer Society, Los Alamitos, CA, USA, 1992.

[34] P. Jiang and Q. Mair. A self-organisational management network based adaptive resonance theory. *Agent Technologies, Infrastructures, Tools, and Applications for e-Services*. Kowalczyk et al. (Eds), *Lecture Notes in Artificial Intelligence*, volume 2592, pp. 211-225. Springer-Verlag, Berlin, 2003.

[35] H. Karuna P. Valckenaers, B. Saint-Germain, P. Verstraete, C. B. Zamfirescu, H. Van Brussels, Emergent Forecasting using a stigmergy approach in manufacturing coordination and control. *Engineering Self-Organising Systems*. S. Brueckner et al. (Eds), Lecture Notes in Artificial Intelligence, volume 3464, pp. 210-226, Springer-Verlag, Berlin, 2005.

[36] S. A. Kauffman, S.A. and S. Johnsen. Coevolution of the edge of chaos: coupled fitness landscapes, poised states, and coevolutionary avalanches. C.G. Langton et al. (eds.), *Artificial Life II,* Proceedings Volume X in the Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, Reading, MA, 1992.

[37] A. Koestler. *The Ghost in the Machine*, Reprint edition, Penguin, East Rutherford, NJ, USA,1990.

[38] J. Liu, X. Jin, and K. C. Tsui. Autonomy oriented computing (AOC): formulating computational systems with autonomous components. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, In press, IEEE Computer Society, Los Alamitos, CA, USA, 2005.

[39] P. Maes. Modeling Adaptive Autonomous Agents. C.G. Langton et al. (Eds), *Artificial Life 1(1-2):135-162,* MIT Press, Cambridge, MA, 1994.

[40] M. Mamei, M. Vasirani, and F. Zambonelli. Self-organising spatial shapes in mobile particles: the TOTA approach. *Engineering Self-Organising System.* S. Brueckner et al. (Eds), Lecture Notes in Artificial Intelligence, volume 3464, pp. 138-153. Springer-Verlag, Berlin, 2005.

[41] J-P. Mano, C. Bourjot, G. Leopardo, P. Glize. Bio-inspired mechanisms for artificial self-organised systems. *Informatica,* Ljubljana, Slovenia. In press, 2005.

[42] F. Maturana and D. H. Norrie. Multi-agent mediator architecture for distributed manufacturing. *Journal of Intelligent Manufacturing*, 7:257-270. Kluwer Academic Publishers, Amsterdam, The Netherlands. 1996.

[43] A. Montresor, H. Meling and O. Babaoglu. Messor: load-balancing through a swarm of autonomous agents. G. Moro and M. Koubarakis (Eds.) *Agents and Peer-to-Peer Computing,* Lecture Notes in Artificial Intelligence, volume 2530, pp. 125-137. Springer-Verlag, Berlin 2003.

[44] J.-P. Müller. Emergence of Collective Behaviour and Problem Solving. *Engineering Societies in the Agents World - 4th International Workshop (ESAW 2003),* A. Omicini, P. Petta, and J. Pitt (Eds), Lecture Notes in Artificial Intelligence, volume 3071, pp. 1-20. Springer-Verlag, Berlin, 2004.

[45] H. V. Parunak and R. S. Vanderbok. Managing emergent behaviour in distributed control systems. *Proceedings of ISA Tech '97*, Instrument Society of America, 1997.

[46] H. V. Parunak, J. Sauter, and S. Clark. Toward the specification and design of industrial synthetic ecosystems, *Intelligent Agents IV: Agent Theories, Architectures, and Languages*, Lecture Notes in Artificial Intelligence, volume 1365, pp. 45-59. Springer-Verlag, Berlin, 1998.

[47] H. V. Parunak and S. Brueckner, Entropy and Self-Organisation in Multi-Agent Systems. In *International Conference on Autonomous Agents (Agents'01)*, 124-130. ACM Press, New York, NY, USA, 2001.

[48] H. V. Parunak, S. Brueckner, J.A. Sauter: Digital pheromone mechanisms for coordination of unmanned vehicles. *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pp. 449-450, ACM Press, New York, NY, USA, 2002.

[49] H. E. Pattison, D. D. Corkill, and V. R. Lesser. Instantiating descriptions of organisational structures. *Distributed Artificial Intelligence*, pp. 59-96. M. N. Huhns (Ed.) Pitman, London, 1987

[50] G. Picard, M-P. Gleizes. An agent architecture to design self-organising collectives: principles and application. *Proceedings of the AISB'02 Symposium on Adaptive Agents and Multi-Agent Systems*, Lecture Notes in Artificial Intelligence, volume 2636, pp. 141-158. Springer-Verlag, Berlin, 2002.

[51] G. Picard, M-P. Gleizes. The ADELFE methodology - designing adaptive cooperative multi-agent systems. F. Bergenti, M-P. Gleizes, and F. Zambonelli (Eds). *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook*, pp. 57-176, Kluwer Publishing, Amsterdam, The Netherlands, 2004.

[52] G. Picard, C. Bernon, M-P. Gleizes. ETTO: emergent timetabling by cooperative self-organisation. *Proceedings of the third International Workshop on Engineering Self-Organising Applications (ESOA'05), pp. 31-45. Utrecht, The Netherlands, 2005*

[53] G. Poulton, Y. Guo, G. James, P. Valencia, V. Gerasimov, and J. Li. Directed self-assembly of 2-dimensional mesoblocks using top-down/bottom-up design. *Engineering Self-Organising Systems.* S. Brueckner et al. (Eds), Lecture Notes in Artificial Intelligence, Volume 3464, pp. 154-166. Springer-Verlag, Berlin, 2005.

[54] R. Razavi, J. F. Perrot, and N. Guelfi. Adaptive modeling: an approach and a method for implementing adaptive agents. *Massively Multi-Agent Systems I.* T. Ishida, L. Gasser, H. Nakashima (Eds). Lecture Notes in Artificial Intelligence, volume 3446, pp. 136-148. Springer-Verlag, Berlin, 2005.

[55] A. Reitbauer, A. Battino, A. Karageorgos, N. Mehandjiev, P. Valckenaers, and B. Saint-Germain. The MaBE middleware: extending multi-agent systems to enable open business collaboration. In *6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS'04),* 2004.

[56] M. Schillo, B. Fley, M. Florian, F. Hillebrandt, and D. Hinck, Self-Organisation in Multiagent Systems: From Agent Interaction to Agent Organisation. *Third International Workshop on Modelling Artificial Societies and Hybrid Organisations (MASHO)*, pp. 37-46, 2002.

[57] J. R. Searle. *The rediscovery of the mind*. MIT Press, Cambridge, MA, USA, 1992.

[58] D. Servat, J. Leonard, E. Perrier, and J. P. Treuil. The Rivage project: a new approach for simulating runoff dynamics. J.Feyen and K.Wiyo, (Eds.) *Modelling of transport processes in soils*, pp. 592-601. Wageningen Press, Leuven, 1999.

[59] M. Ulieru, Emergence of holonic enterprises from multi-agent systems: a fuzzy evolutionary approach. *Frontiers in Artificial Intelligence and Applications - Soft Computing Agents*, 83: 187-215. IOS Press - Frontiers in AI and Applications Series, Amsterdam, The Netherlands, 2002.

[60] M. Ulieru. Adaptive Information Infrastructures for the e-Society. *Engineering Self-Organising Systems* S. Brueckner et al. (Eds), Lecture Notes in Artificial Intelligence, volume 3464 pp. 32-51. Springer-Verlag, Berlin, 2005.

[61] F. Varela. *Principles of Biological Autonomy*. *Elsevier*, New York, NY, USA, 1979.

[62] D. Weyns, K. Schelfthout, T. Holvoet, and O. Glorieux. Role based model for adaptive Agents. *Fourth Symposium on Adaptive Agents and Multi-agent Systems at the AISB '04 Convention,* 2004.

[63] D. Weyns, T. Holvoet. On the role of environments in multiagent systems. *Informatica*, Ljubljana, Slovenia. In press. 2005.

[64] F. Zambonelli, M.-P. Gleizes, M. Mamei, and R. Tolksdorf. Spray computers: frontiers of self-organisation for pervasive computing. *Second International Workshop on Theory and Practice of Open Computational Systems (TAPOCS 2004) in 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04),* pp. 397-402. IEEE Computer Society, Los Alamitos, CA, USA 2004.

Bio-inspired Mechanisms for Artificial Self-organised Systems

Mano Jean-Pierre
Irit - Université Paul Sabatier - 118 Route de Narbonne - 31062 Toulouse Cedex – France
E-mail : mano@irit.fr, http://www.irit.fr/SMAC

Bourjot Christine
Loria - 615 rue du Jardin Botanique - 54600 Villers les Nancy - France
E-mail: bourjot@loria.fr, http://webloria.loria.fr/~bourjot/,

Lopardo Gabriel
Agents Research Lab, University of Girona - Campus Montilivi - 17071 Girona – Spain
E-mail: glopardo@eia.udg.es, http://eia.udg.es/arl

Glize Pierre
Irit - Université Paul Sabatier - 118 Route de Narbonne - 31062 Toulouse Cedex – France
E-mail: glize@irit.fr, http://www.irit.fr/SMAC

Keywords: self-organization, stigmergy, reinforcement, cooperation

Received: May 7, 2005

Self-organization is a growing interdisciplinary field of research about a phenomenon that can be observed in the Universe, in Nature and in social contexts. Research on self-organization tries to describe and explain forms, complex patterns and behaviours that arise from a collection of entities without an external organizer. As researchers in artificial systems, our aim is not to mimic self-organizing phenomena arising in Nature, but to understand and to control underlying mechanisms allowing desired emergence of forms, complex patterns and behaviours. Rather than attempting to eliminate such self-organization in artificial systems, we think that this might be deliberately harnessed in order to reach desirable global properties. In this paper we analyze three forms of self-organization: stigmergy, reinforcement mechanisms and cooperation. The amplification phenomena founded in stigmergic process or in reinforcement process are different forms of positive feedbacks that play a major role in building group activity or social organization. Cooperation is a functional form for self-organization because of its ability to guide local behaviours in order to obtain a relevant collective one. For each forms of self-organisation, we present a case study to show how we transposed it to some artificial systems and then analyse the strengths and weaknesses of such an approach.

Povzetek: Biološke osnove umetnih samo-organizirajočih se sistemov.

1 Introduction

Self-Organization refers to a broad range of pattern-formation processes in both physical and biological systems, such as sand grains assembling into rippled dunes, chemical reactants forming swirling spirals, cells making up highly structured tissues, and fish joining together in schools. Concepts and mechanisms relatives to self-organization in biological systems have been largely defined and explained in [1]: basic modes of nonlinear interaction among components as well as information acquisition and process. In most self-organised systems in biology nonlinear interactions involve amplification or cooperation. Complex behaviours may emerge even though the system is composed of similar units that follow local rules and without intervention from external guiding influences.

Computer science is interested in understanding the underlying principles of self-organization because -like in nature- the rules specifying interactions among many artificial system's components are executed using only local information, without reference to the global pattern, which is not easily accessible or possible to be found. For more developments on self-organization and emergence, see the overview in [17].

The following three parts concern different mechanisms of self-organization in either ethology or cellular biology: stigmergy, reinforcement and cooperation. In each part, after description of the general principles of the mechanism, we develop the understanding of a particular instance, especially for the quite new instances of stigmergy and reinforcement mechanisms that come from agent-based simulation models we undertook with biologists.

Then we present a case study to show how we transposed it to some artificial systems.

More indeed, in the first part we analyse the stigmergy mechanism allowing indirect task coordination and regulation in insects societies or social spiders. This

principle is replicated with some changes to be used in some artificial applications like region detection.

In the second part we present a reinforcement mechanism together with direct interactions studied in ethology and how it leads to specialization in groups of animals. The transposition of these mechanisms concerns dynamic task allocation in a network.

In the third part we study a cooperation mechanism observed between cells as well as in animal societies. This phenomenon is applied in artificial neural networks in order to produce plasticity and adaptation

In the last part we discuss about strengths and weaknesses of self-organizing principles in order to engineer artificial systems.

2 Self-organization Patterns from Stigmergy Mechanisms

2.1 Stigmergy Mechanisms in Biology

Stigmergy has been defined by the biologist Grassé [2] to refer to the mechanism by which the termites coordinate their nest building activities. In stigmergic labour it is the product of work previously accomplished, rather than direct communication among nest mates, that induces the insects to perform additional labour [3]. It explained how individual builders could act independently on a structure without direct interactions or sophisticated communications. The state of the building is the stimulus, its response is the construction activity.

So a stigmergic process is a sequence of indirect stimulus/responses behaviours and contributes to the coordination between insects through the environment. Another illustration of how stimergy and self-organization can be combined into more subtle adaptive behaviours is recruitment in social insects. Self-organised trail laying by individual ants is a way of modifying the environment to communicate with nest mates that follow such trails. It appears that task performance by some workers decreases the need for more task performance: for instance, nest cleaning by some workers reduces the need for nest cleaning [4], [5]. Therefore, nest mates communicate to other nest mates by modifying the environment (cleaning the nest), and nest mates respond to the modified environment (by not engaging in nest cleaning); that is stigmergy. Division of labor is another paradigmatic phenomenon of stigmergy. But by far more crucial, is how ants form piles of items such as dead bodies (corpses), larvae, or grains of sand. There again, stigmergy is at work: ants deposit items at initially random locations. When other ants perceive deposited items, they are stimulated to deposit items next to them, being this type of symmetry clustering organization and brood sorting a type of self-organization and adaptive behaviour. There are other types of examples (e.g. prey collectively transport), yet stigmergy is also present: ants change the perceived environment of other ants (their cognitive map, according to Chialvo [6]), and in every example, the environment serves as a medium of communication. Finally, stigmergy is often associated with flexibility: when the environment changes because of an external perturbation, the insects respond appropriately to that perturbation, as if it were a modification of the environment caused by the colony's activities. In other words, the colony can collectively respond to the perturbation with individuals exhibiting the same behaviour.

What all these examples have in common is that they show how stigmergy can easily be made operational because of the simplicity of the behaviours involved.

2.2 Stigmergy Mechanism Under-standing

Dorigo [5] replicated stigmergic principle from ants colony, including the pheromone trails, to derive algorithms applied either to static or dynamic combinatorial optimization problems with applications on many problems like the traveling salesman problem. The brood sorting behaviour can be reproduced with robots, for example, to achieve collective sort [7]. We replicate another kind of stigmergic mechanism to perform region detection. We analyse the stigmergy process involved during the building of web in a species of social spiders through an agent-based model. This simulation shows that the mechanism that underlies the movement of spiders can be expressed as a stigmergic one where silk and silk attraction play the major role.

The web weaving activity needs two behavioural items: movement and silk fixing. Items are independent i.e., a spider-agent (SA) can make these two action types at the same time: to move to a close stake and to fix a silk dragline. Furthermore, items are fired stochastically according to a constant or contextual probability. When fixed, the dragline provides a new path (the shortest one) between the current stake and the last on which silk was fixed (whatever the spider moves were). The probability to fix the silk is constant over time. When a SA moves, it can be from the current stake to an adjacent one (the 8 accessible neighbours). Since silk draglines are fixed between stakes, they offer new directions of movement. When facing such a situation, the SA has to choose whether to follow a dragline or to move to an adjacent stake. The probability for a SA to move to a given stake depends on the type of access. In the neighborhood, the probability is constant.

When following a silk dragline, the probability is proportional to the number of silk draglines and to the silk attraction. This mechanism that underlies the movement can be expressed as a stigmergic process. Studies [18] demonstrated the key role of the silk attraction: when too low, no web is built and all available space is used; when too strong, SA were trapped in their own silk and no collective weaving occurred; when well chosen, we showed that the web size is related to the attraction: the more the attraction is, the smaller the covered surface is.

The behaviour of the colony of spider-agents can be interpreted as much as:

- A stigmergy pattern a collective mechanism for space exploration which is characterized by limited

perception and indirect interaction, the environment (the web woven by spiders) being the medium of interaction,

- as a self-organised pattern with some regulation performed without explicit coordination, the size of the explored space (the size of the web) being related to the silk attraction factor.

2.3 Region Detection by Stigmergy

This stigmergy process has been transposed to region detection. The problem is to extract a region from an image. A region must be a connected set of pixels with homogeneous radiometric characteristics. In our case, all the pixels of a region should have the same grey level, more or less a given tolerance. From a given picture, our model produces an intermediate structure constituted by the woven collective web, interpreted later to deduce region by considering the pixels on which the web is fixed. It requires an exploration of a space that has to be restricted to a subset of its elements (the pixels of the region).

A grey level image is the environment in which agents will evolve; stakes correspond to pixels and the height to their grey level. The behavioural items of agents are similar to SA. The movement remains unchanged and silk fixing now depends on the context and, thus, is related to the grey level of the region to detect. The interaction principle is based on stigmergy. To avoid different regions of the same grey level being woven on a unique web a third behavioural item was added to make an agent probabilistically return back to the web [19]. All agents have the same features determined by four parameters. Two parameters govern the movements of the agents and thus the exploration process. The two last ones are related to the selection of pixels, thus determining the relevance of the extracted region. Because the process is based on the stigmergy ensured by the silk draglines laid down in the environment, selection and movement are tied. But we could initially specify the influence of silk attraction factor as shown in figure 1: when it is high (the left picture) the five agents construct five different webs and do not explore the entire region. When it is low (the right picture) the region covered is bigger and corresponds to a collective web. Thus, when well chosen the parameters for stigmergic process allow decentralised coordination.

Fig.1. Influence of silk-attraction factor on webs for detection region

3 Self-Organization from Reinforcement Mechanisms

3.1 Reinforcement Mechanisms in Biology

Reinforcement has been discussed as a mechanism that shapes the differentiation between specialists and the remaining work force. The concept of reinforcement proposes that the impact of a single worker on stimulus intensity increases with experience. This can be achieved in one of two ways: first, the efficiency of a worker may increase with experience, e.g. because individuals learn to perform a task. Second, response threshold for task associated stimuli may decrease with experience in performing the task [8]. Learning and increase in task efficiency have often been considered as the main reason for the efficiency of division of labor [9], [10]. Then, reinforcement may play an important role in specializations [11].

Reinforcement learning is a synonym of learning by interaction. During learning, the adaptive system tries some actions (i.e., output values) on its environment, then, it is reinforced by receiving a scalar evaluation (the reward) of its actions. The reinforcement learning algorithms selectively retain the outputs that maximize the received reward over time.

Reinforcement mechanisms like increase in task efficiency assiociated with direct interactions in biology conducts to social organization, specifically dominances hierarchies [12]. For example, dominances hierarchies are obtained by simple model based on positive feedback. Two individuals enter in a contest. An individual that wins or loses the contest is more likely to win or lose subsequent contests. The reinforcement mechanism amplifies small initial differences between individuals.

3.2 Reinforcement Mechanism Understanding

The problem of reinforcement learning is knowing what to reinforce. Motivation cannot rely on a blind mechanism that strengthens or weakens connections based on their temporal proximity to pain or pleasure stimuli. While temporal difference reinforcement may work well enough in small systems, it becomes prohibitive in large systems.

The second reinforcement mechanism is relevant when we want to realize a distributed collective task implying a lot of agents. As an example, an elaborate self-organized phenomenon is observed in rats' groups in the diving-for-food situation. This situation is a complex social task in which, for a group of six rats, the food accessibility decreases by progressive immersion of its only path. This experimental schedule leads to the emergence of a specialization in the group of rats, in two profiles: supplier and non-carrier rat. The non-carrier animals never dive, but get food only by stealing it from the suppliers after fight. The supplier rats dive, bring the

food back to the cage and cannot defend the food they carried.

An agent based simulation shows that this social differentiation is possible from a set of interacting individuals without any social cognition. It implements two reinforcement mechanisms: when the action of diving is performed the anxiety of the rat is reduced according adaptive response thresholds models [8]. Whether the action of fighting is successful or not, the strength of the winner is reinforced whereas the strength of the looser is decreased. Alterations of strength are computed according to dominance formula presented in [12]. This specialization is stable, robust and presents adaptive properties like adaptation to the number of agents or adaptation to external conditions [13]. For example, the ratio between carrier agents and supplier agents' number evolves according to the energetic supply coefficient of a pellet.

3.3 Task Allocation in a Computer Network by Reinforcement Mechanism

The general framework to transpose these mechanisms consists in a dynamic task allocation problem among machines, connected together in a network. Initially the tasks are available on a central server. The machines can acquire the data by accessing directly the server or by 'attacking' each other. As some policies are put on the server in order to avoid crash, some agents can easily access the server while other not. The aim of the self organized process is to reduce the exploitation of the network between the machines and the server by means of specialization among machines. It corresponds to dynamically (and efficiently) allocate tasks on an unknown set of machines by making some of them accessing directly the server (because it is easier for them) while others acquire data indirectly (as shown in figure 2).

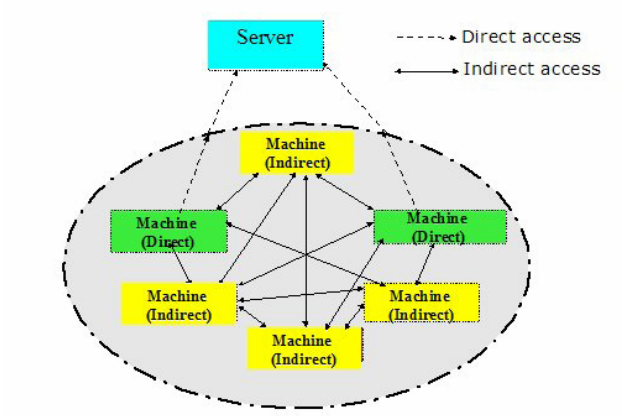


Figure 2. Expected organization for task allocation problem in network

We developed a prototype at applicative layer of the network that assumes existing communication architecture and that only deals with the data processing including execution and redirection. The first tests have been performed with a simulator including the server, the machines and the network. We got some encouraging results like specialization appearance and some improvements in processing time. These results are today obtained with only specific instances of the problem and with hand tuned parameters.

4 Functional Self-Organization from Cooperative Mechanisms

4.1 Cooperative Mechanisms in Biology

We analyse cooperation in complex biological systems trough four main kinds of mechanisms: parallelism, coordination, specialization and recruitment. Those mechanisms are presented according to the interaction complexity between parts of the biological system.

- Parallelism defines the more basic level of cooperation. When different elements of the system are independent in their activities, but share a common goal, they make up a parallel system. Polistes Wasps' nest construction [20] is a good example of parallel activity: wasp workers are interchangeable; they share a same goal that is building the biggest nets as quickly as possible. Efficiency of such a system depends mainly on the number of constitutive elements.
- Coordination is observable when at least two elements of the biological system have to act together or simultaneously in order to perform a particular task impossible to achieve by an alone agent. A demonstrative experience is provided when ants have to act collectively to take a straw off the entrance of their nest. Theoretically, at least two ants are required: when the first ant has lifted up the straw of a few millimetres, the second ant catch the straw lower than the first ant and lift it up on her turn. Because of the lack of intentionality in ants, this example can be discussed as a singularity of parallel system. So, another kind of coordination appears when army ants make bridges with they bodies to smooth the trail between sources of food and their nest.
- Specialization increases the heterogeneity of the biological system, addressing a particular task or function to some elements of this system. In fact system's activity is improved thanks to some elements that either become more efficient in a subset of activities that was already performed or become able to perform new kinds of tasks. The best example in cellular streams [21] is the specialisation of cells which at the simplest level favours a branch of their metabolic activity to high rate to store and produce metabolites for other cells or for the whole organism. At a more complex level, cells can specialize in modifying their structure and develop particular abilities like production of antibodies in immunity cells, gas transportation in blood red corpuscles, chemical energy storage in liver cells, or production and propagation of spikes in neurons.
- Recruitment and mass effect occurring in foraging or in colony aggregation, present a real interest

when collective behaviours improve single ones and beyond trigger events that a few elements wouldn't have produced. This part of cooperative mechanisms clearly includes reinforcement mechanisms discussed in part 3.1. Many examples in nature illustrate mass effect like temperature regulation in penguin colonies, improvement of predator detection in sheep flock, or reaching locally a threshold concentration in trophic factor during embryogenesis that will trigger specialization of cells exposed to this threshold. Regulation is the inherent counterpart of recruitment, and prevents the biological system from being trapped in a single activity before its exhaustion.

4.2 Cooperation Mechanism Understanding

Self-organization concerns always several entities that act, from an external point of view, as a coherent collective. Beyond self-organization, cooperative self-organization constrains more precisely the behaviour of these entities in order to make their interaction reach most of the time, a state of cooperation. From the point of view of the entity the three phases of her functioning are concerned by cooperation:

- At the perception phase the signal received by an entity from its environment or from a second entity (social environment) must not be misunderstood or ambiguous .
- At the processing phase, the information contained in the signal must not be unproductiveness or inability.
- At the action phase, the decision of the entity (transformation of the world or even message sending) must not be useless, or implying some concurrency or conflict in its environment.

This is the basic cooperation principle inspiring the AMAS theory [14], which will be used, in many applications such as the following adaptive neural network.

4.3 Cooperative Artificial Neural Network

Biological neural structures may be considered as the combined result of self-organizing cellular activities and of the following of many strong planned processes. Such a system is the result of the permanent reorganization of its parts upon among others, the pressure of its environment.

The concept of cooperative neuro-agent (CNA) can be detailed in three functional subsets that justify the neuro-agent term [15]. CNAs have the usual transfer function of an artificial neuron, have also a vegetative behaviour and have moreover a set of cooperative social behaviours according to the laws of the AMAS theory. The role of vegetative and social behaviours accounts mainly for balancing the lack of an initial topology in the network.

Cooperative behaviour when addressing to CNAs, means that CNAs help each other to find not only their right place in the network but also their right function in the network. Back propagation is cooperative as it helps CNAs to find their function but is not sufficient to position them in the network. So we can distinguish two other sets of cooperative behaviours. The first includes pro-cooperation, for example when a CNA informs one of its neighbours that it is searching accountancies, with the rest of its neighborhood (including virtual links). The last set regroups the behaviours appearing to resolve some particular potential and well defined troubles: the non cooperative situations.

CNA Coordination. The objective of a CNA is to be useful to the others by having a coherent activity and supplying them with relevant information. So, learning consists in reinforcing weights according to correlated temporal activities of inputs. A CNA estimates the rightness of its activation by interpreting messages from its outputs. Following the mean error, a CNA adjusts the weights of the concerned inputs. As in a back propagation mechanism, a CNA informs in turn its inputs of the error it has detected.

That means that a CNA modifies its functioning to fulfil other CNAs it is working with. So at a given time the behaviour of a CNA is the result of its code expression under the regulation of its local environment.

CNA Specialization. A CNA realizes a positive integration of the information carried through incoming links, and then this weighted sum is transformed using a non-linear transfer function into a positive integer value. A CNA can also use an inhibitory input that nullifies the transferred value.

When the coordination process adapts insufficiently its output, a CNA modifies its transfer function. Thus we can observe, at the collective level, clusters in which neuro-agents have a similar transfer function. Moreover, a CNA can be used as activator or inhibitor to others. Without any predefined role, some CNAs tend to be used preferably (but not exclusively) as inhibitors.

CNA Recruitment. If a CNA keeps on receiving error messages that it cannot satisfy, it triggers an adaptation process of the network structure. We call this process vegetative behaviour, as the CNA can determine by itself whether it has to proliferate or search for new inputs, or if finally it has to disappear in an apoptosis-like mechanism.

This vegetative behaviour grants the dynamics and the self-organization of the network. That is why the learning stage begins with a not connected network where only inputs and outputs of the future network are created. The mother CNA provides all the required instances, which are an exact copy of it. Nevertheless, the basic transfer function of the mother cell is adjusted in each individual CNA in order to find the best cooperative behaviour in accordance with its neighbourhood.

Emergent Collective Behaviour. A CNA network is initialized with only not connected CNAs located at the interfaces (input and output of the network). The behaviour of CNAs only depends on the local perception CNAs get about the system, and finally there is no imposed pattern which supervises the organization of the

system. Based on local non-cooperative criteria of neuro-agents, the system adjusts its function by reorganizing its parts. So the learning of the system globally results from population growth and from neuro-agents adaptation (weight adjustments, transfer function regulation, search and disappearance of connections).

The simple test case of learning a XOR logical function illustrates perfectly the different aspects of cooperation in a neural network. The initial step requires two inputs and one output as shell of the future network; that means 3 CNAs. Obviously they are not sufficient for computing a XOR function, so they have to recruit at least a fourth CNA which will inhibit the output of the network when both inputs are activated. In the figure 3 we can distinguish a first period of proliferations that do not improve the global learning performance, but give the network with the ability (in terms of neural population) of realizing the right function. In a second period, each CNA specialises itself in an integrator and coordinates the information flow between them by adjusting weights until outputs of the network do not produce errors anymore. Useless CNAs are eliminated.

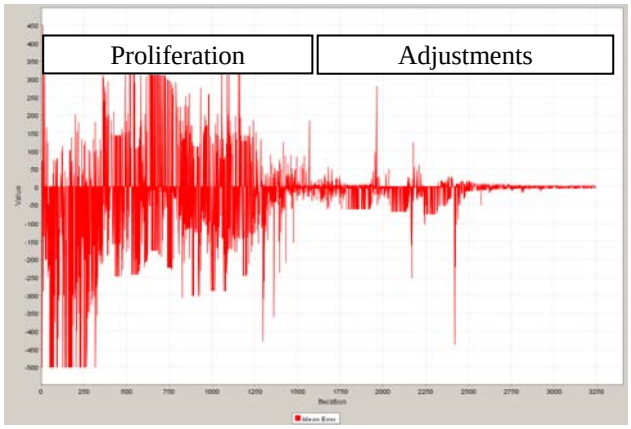


Figure 3 : Graph of the global error of a network learning a XOR function

5 Strenghts and Weaknesses of Self-organizing Mechanisms

5.1 Analysis of the Case Studies

The first experiments we undertook in region detection demonstrate the potential of the transposition of spiders-inspired self-organised mechanisms. As mentioned by Bonabeau [4], stigmergy is a promising first step to design groups of artificial agents which solve problems: replacing coordination (and possibly some hierarchy) through direct communications by indirect interactions is appealing if one wishes to design simple agents and reduce communication among agents. Flexibility to perturbation is priceless: it means that the agents can respond to a perturbation without being reprogrammed to deal with that particular instability.

However a major drawback has to be solved in order to produce a real application of detection region: parameters are until now empirically adjusted and we also have to determine initial conditions: the numbers of agents and their initial position. The right number of agents could be automatically adjusted by using for example a stimergic mechanism from adaptive recruitment behaviours in social insects.

Some results for dynamic task allocation in a network have been obtained by transposing the model of specialization in rats group. We show that apparition of some social pattern is possible from a set of interacting individuals without any social cognition and no direct communication. But these results can only be obtained by trial-error experiments in an iterative process since exact behaviour of these systems could only be known a posteriori . In order to understand influences of either the parameters or the combinations of parameters, differential analysis is required and a lot of experiments are carried out. One experiment must be proceed a lot of times in order to be statistically valid. So it is useful to store a full and detailed review of preceding experiments and often to analyse data from previous experiment with multiple other views.

Cooperative neuro-agent network is today evaluated on logical functions, but is also applied to model the migration of leatherback turtles. Even working rightly on these cases, tuning the cooperative local behaviour in each entity of a system was difficult in order to obtain good specialization, coordination and recruitment behaviours. The result is mainly a very generic approach for artificial neural networks and an efficient search solution in the global space problem avoiding experimentally local minima.

5.2 Tools for the Self-organization Process

Usual learning techniques (Q-learning, reinforcement learning, genetic algorithms...) try to find a solution by the way of an individual even its learning is improved by the relationships with others. On the opposite, all self-organizing systems -including ants algorithms or swarm particle algorithms- share the ability to solve a global problem at the collective level, where micro-level components discover only a small part of the solution. This is the case for the mechanisms showed in the paper:

1. Spiders work together to create a web corresponding to an image region individually without knowing what is the collective result.
2. Machine specialization in a network is obtained from local reinforcement mechanisms without any centralized control.
3. Adequate neural structures come from local cooperative behaviour without any learning strategy derived from the global function to obtain.

The main advantage for all these self-organizing problem solving approaches is the complexity reduction, because they are only concerned by specifying agent behaviour, even the solved problem is related to the collective complexity. We can exemplify that by expliciting the parameters used in the case studies:

1. In stigmergy mechanisms, the two behavioural items of an agent, movement and selection, are defined by four parameters where silk attraction factor plays a key role.

2. In reinforcement mechanisms the three behavioural items, diving, fighting and eating are triggered according to parameters characterizing the internal state of an agent. These are hunger, strength and anxiety. The reinforcement parameters concern strength and anxiety.
3. In cooperation mechanisms, the local actions are associated with each non cooperative action an agent may encounter. For a cooperative neuro-agent these actions are proliferation and apoptosis of a neuron, regulation (increasing or decreasing the weight of an input) between its current inputs or specialization (improving or not the sensibility of the inputs) of its own transfer function.

Self-organized systems are characterized mainly by non linear dynamics, by sensibility to initial conditions and parameter sensitivity. Thus the overall properties cannot be understood simply by examining separately the components. With agent-based modelling, a lot of work remains to precisely identify the link between the local parameters and the global results obtained. In order to obtain dynamic equilibrium due to unexpected changes in the environment and non linearities inside the system, all self-organizing agents must manage a given action and also its opposite one. This is the actual weaknesses of self-organizing mechanisms, because a lot of time must be spent by engineers in order to find from experimentations the right decision criteria firing all these actions.

6 Conclusion and Prospects

In this paper, three approaches of self-organization inspired from biological systems were analysed and case studies applying these mechanisms were presented. The bio-inspired mechanisms showed have the main descriptive criteria as defined in [22]. There is no external control and no internal entity centralize information or decision. The solution is built dynamically and consequently unpredictable, due to the set of interdependent individuals working in parallel and able to react relevantly to their reciprocal activities. These applications have also the anytime property, because they are able to give a more or less good solution according to the time given to the processes.

Even if these approaches are able to solve difficult problems, the study of such complex systems needs experiments to explore their behaviours as Zambonelli claims [16]. Thus, a very useful perspective for these mechanisms will be to define theories allowing automatic tuning of their parameters.

Self-organization mechanisms guide the behaviour of the local entities of a collective. Consequently these approaches allow a drastic reduction of the solution search space compared to global search algorithms. Though this is experimentally observed, a lack of demonstrations by formal proofs still remains today.

Working on self-organization implies the creation of disorders inside a collective in order to obtain later a more relevant response of the system faced with unexpected events. From an engineering point of view it could be interesting to propose global systems gauges able to link disorder and relevance behaviour at the system macro-level. Some tools are today available on MAS platforms as described on the AOSE overview [23]. They must be completed by new works on entropy measure in artificial systems in order to have a more relevant observables on their dynamics.

Acknowledgment

The authors wish to thank Aurélien Saint-Dizier for the implementation of the network simulator.

References

[1] Camazine S., Deneubourg J.L., Franks N.R., Sneyd J., Theraulaz G., Bonabeau E. (2001) *Self-Organization in Biological Systems*, Princeton University Press.

[2] Grassé P.P.(1959) La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp., La théorie de la stigmergie : essais d'interprétation du comportement des termites constructeurs, *Insectes Sociaux.*, 6, pp. 41-84.

[3] Wilson E. O. (1971), *The insect societies*. Cambridge : Harvard University Press.

[4] Bonabeau, E.; Dorigo, M. and Théraulaz, G. (1999) *Swarm Intelligence: From Natural to Artificial Systems*, Santa Fe Institute in the Sciences of the Complexity, Oxford Univ. Press, New York.

[5] Dorigo, M.; Bonabeau, E. & Theraulaz G. (2000) *Ant algorithms and stigmergy*. Future Generation Computer Systems 16: pp. 851–871

[6] Chialvo, D.R. and Millonas M.M. (1995) How Swarms Build Cognitive Maps. In Steels, L. (Ed.): *The Biology and Technology of Intelligent Autonomous Agents*, 144, NATO ASI Series, 439-450.

[7] Holland O., Melhuish C. (1999) Stigmergy, Self-Organization, and Sorting, *in Collective Robotics*, In Artificial Life 5: 173-202.

[8] Theraulaz, G., Bonabeau, E. & Deneubourg, J. (1998) Response threshold reinforcement and division of labour in insect societies *Proceedings of the Royal Society of London Series.* **265:** 327-332.

[9] Seeley, T. D. (1982) Adaptive significance of the age polyethism schedule in honeybee colonies. *Behavioral Ecology and Sociobiology* **11:** 287-293.

[10] Jeanne, R. L. (1986) The organization of work in *Polybia occidentalis*: costs and benefits of specialization in a social wasp. . *Behavioral Ecology and Sociobiology* **19:** 333-341.

[11] O'Donnel, S. & Jeanne, R. L. (1992) Forager success increases with experience in *Polybia occidentalis* (Hymenoptera, Vespidae*). Insectes Sociaux,* Birkhäuser Verlag, **39:** 451-454.

[12] Hemelrijk, C. K. (1996) Dominance interactions, spatial dynamics and emergent reciprocity in a virtual world. In Proceedings of the fourth international conference on simulation of adaptive

behavior, vol. 4 (ed. P. Maes, M. J. Mataric, J-A Meyer, J Pollack & S. W. Wilson), pp. 545-552. Cambridge, MA: The MIT Press.

[13] Thomas V., Bourjot C., Chevrier V., Desor D. (2004) Hamelin : A model for collective adaptation based on internal stimuli. In Stefan Schaal, Auke Ijspeert, Aude Billard, Sethu Vijayakumar, John Hallam, and Jean-Arcady Meyer, editors, *From animal to animats 8 - Eighth International Conference on the Simulation of Adaptive Behaviour 2004 - SAB'04*, Los Angeles, USA, pages 425–434.

[14] Georgé J.P., Edmonds B. and Glize P. (2004) Making Self-Organising Adaptive Multiagent Systems Work. In: *Methodologies and Software Engineering for Agent Systems.* Federico Bergenti, Marie-Pierre Gleizes, Franco Zambonelli (Eds.), Kluwer Academic Publishers, pp. 319-338

[15] Mano J.P. and Glize P. (2004) Self-adaptive Network of Cooperative Neuro-agents, S*ymposium on Adaptive Agents and Multi-Agent Systems*, Leeds, UK, pp. 129-134.

[16] Zambonelli, F. and Omicini, A. (2004) Challenges and Research Directions in Agent-Oriented Software Engineering, *Autonomous Agents and Multi-Agent Systems* . New-York, USA, 9(3), pp. 253-283.

[17] Di Marzo, Gleizes, Kargeorgos (2005) Self-organization and emergence in MAS: an overview, *Informatica*, this issue.

[18] Bourjot C., Chevrier V. (2001) Multi-agent simulation in biology: application to social spiders case, *In Proc. of Agent Based Simulation Workshop II*, Passau, pp18-23.

[19] Bourjot C., Chevrier V., Thomas V.(2003) A new swarm mechanism based on social spiders colonies : From web weaving to region detection, in Web Intelligence and Agent Systems: An International Journal , IOS Press, Vol 1, N.1, pp. 47-64.

[20] Theraulaz, G., Bonabeau, E. & Deneubourg, J.L. (1999) The mechanisms and rules of coordinated building in social insects,. In: Detrain, C., Deneubourg, J.L. & Pasteels, J. Edits. *Information Processing in Social Insects*. Birkhauser Verlag. pp. 309-330

[21] Chandebois R. (1999) *Comment les cellules construisent l'animal,* Phénix éditions, Paris

[22] Bernon C., Chevrier V., Hilaire V., Marrow P. (2005) Applications of self-organizing MAS, *Informatica* , this issue.

[23] Bernon C., Cossentino M., Pavon J. (2005) An overview of current trends in European AOSE research, *Informatica* , this issue.

On Self-Organising Mechanisms from Social, Business and Economic Domains

Salima Hassas
LIRIS-CNRS, University of Lyon, France
E-mail: hassas@liris.cnrs.fr
http://www710.univ-lyon1.fr/~hassas

Giovanna Di Marzo-Serugendo
University of Geneva, Switzerland
E-mail: Giovanna.Dimarzo@cui.unige.ch
http://cui.unige.ch/~dimarzo/

Anthony Karageorgos
University of Thessaly, Greece
E-mail: karageorgos@computer.org
http://inf-server.inf.uth.gr/~karageorgos/

Cristiano Castelfranchi
Unit of AI, Cognitive Modelling and Interaction, CNR, Italy
E-mail: cristiano.castelfranchi@istc.cnr.it
http://www.istc.cnr.it/

Keywords: self-organisation, networks, social functions, business networks, social learning

Received: April 25, 2005

This paper discusses examples of socially inspired self-organisation approaches and their use to build socially-aware, self-organising computing systems. The paper presents different mechanisms originating from existing social systems, such as stigmergy from social insects behaviours, epidemic spreading, gossiping, trust and reputation inspired by human social behaviours, as well as other approaches from social science related to business and economics. It also elaborates on issues related to social network dynamics, social network patterns, social networks analysis, and their relation to the process of self-organisation. The applicability of socially inspired approaches in the engineering of self-organising computing systems is then illustrated with applications concerning WWW, computer networks and business communities.

Povzetek: Podani so primeri mehanizmov samoorganizacije.

1 Introduction

Nowadays computing systems are open systems evolving in a dynamic complex environment. They are designed as sets of interacting components, highly distributed both conceptually and physically. The growing complexity of these systems and their large scale distribution make the use of traditional approaches based on hierarchical functional decomposition and centralised control no more applicable. Increasingly, a real need for new paradigms, mechanisms and techniques allowing endowing these systems with the capacity to autonomously manage their functioning and evolution, is expressed. Existing social systems, for example large scale, decentralised and autonomic human, insect or business and economic systems, are well known to exhibit interesting characteristics, such as robustness, capacity of self-management and self-adaptation, survivability in uncertain and dynamic environments. They can provide a great inspiration for busiding self-organising computing systems.

Socially inspired computing gathers computing techniques that make use of metaphors inspired by social behaviours, exhibiting self-organisation, self-adaptation and self-maintainance of the society organisation. These social behaviours range from those observed in biological entities such as bacteria, cells and social insects to animals and human societies. One important characteristic of these societies is their emergence as patterns developed from relatively simple interactions in a network of individuals. These patterns, are supposed to be driven by self-organising processes that are governed by simple but generic laws [19][5]. This paper is focused on self-organising mechanisms observed in natural social systems and in business and economic ones, and the illus-

tration of their use for building self-organising computing systems. We distinguish natural systems from business and economic systems, since generic laws guiding self-organisation in the first kind of systems is dictated by nature whereas in the others, self-organisation is governed by business and market laws.

From a natural systems perspective, species survival is the ultimate goal. This goal is not expressed explicitly at the individual level, but seems to guide the collective behaviour towards the emergence of *social functions* and dynamics allowing the maintainance of the system organisation. In business and economic systems, individual behaviours are goal-oriented and their primary goal is to increase their profit. In this case, the system's dynamics is handled by the activity developed to face business and economic constraints to reach a global equilibrium through which the system can survive. In both systems, one important issue is their capacity to globally maintain a sufficiently good level of information allowing them to deploy the effective global behaviour that permits the realisation of their intentional or non intentional goals.

In the following, we first present examples of socially inspired self-organising mechanisms in natural business and economic systems. Before concluding, we present example applications of such mechanisms in WWW, computer networks and business communities.

2 The Social Human Behaviour Inspiration

2.1 Social Functions

Human collective behaviour occurs without central control, and through self-organisation. In this case, intimately linked with the notion of self-organisation is the notion of *"emergence"* in the sense that *"social functions"* arise out from (self-interested) human collective behaviour. In social sciences different interpretations of the notion of social functions have been expressed, essentially considering that even if social functions are not intentional and possibly unknown they constitute the ultimate end of the society and explain its existence.

The social functions concept has also been explained as the *"invisible hand"* which would manage forms of unplanned coordination (like market) in which human interest increases [31] through the apparently *"spontaneous emergence of an unintentional social order and institutions"*. As pointed out by [13], the problem with this view is: *"how an unintentional effect can be an end"* for the society; and *"how is it possible that we pursue something that is not an intention of ours"*. An alternative could be avoiding the concept of social functions because of the problems and questions that they provoke. However, this is not satisfactory too, because nevertheless social emergence happens and has the form of a goal-oriented process.

Therefore, it is important to distinguish two kinds of social emergence: 1. the emergent phenomenon is perceived by an observer, but has no effect on the society; 2. the emergent phenomenon has an effect on the society by self-reproducing and enforcing the social phenomenon.

Given the considerations above, Castelfranchi considers that *"in order to have a function, a behaviour or trait or entity must be replicated and shaped by its effects"*.

The principal argument is that *"the invisible hand"* is not necessarily a good thing for society (especially in the case of self-interested agents). The optimum order for the society can actually be bad for individuals or for everybody. For instance, prisons generate criminals that in turn feed prisons. This is a function not a social objective.

The important thing is that *"re-organisation simply maintains the system, but not necessariy the optimal value"*.

2.2 Social Activities Based on Social Networks and Their Inspiration for Computing

Propagation of information or knowledge allowing social activities in social systems lays on the social network formed by the the interaction held between the society individual components during social activities. Social behaviour both shapes and is shaped by such social networks.

2.2.1 Social Learning and Propagation of Knowledge

In social science, it is now established that social interactions play a fundamental role in learning dynamics, and lead to cognitive development. This phenomenon is known as *"Zone of Proximal Development"* which Vygotsky describes it as *"the distance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers"* [51] [15]. The effect of socialisation has also been proven to benefit to the propagation of knowledge inside an interconnected population. In [14] the authors considered social learning in a population of myopic, memoryless agents. They have made some experiments to study how technology diffuses in a population based on individual or collective evaluation of the technology. The authors have shown that under a learning rule where an agent changes his technology only if he has had a failure (a bad outcome), the society converges with probability 1 to the better technology. In contrast, when agents switch on the basis of the neighbourhood averages, convergence occurs if the better technology is sufficiently better. These experiments show how a better technology spreads in a population through a mechanism of imitation and thanks to neighbourhood connections. In another work [3], the authors develop a general framework to study the relationship between the structure of these neighborhoods and the process of social learning. They show that, in a

connected society, local learning ensures that all agents obtain the same payoffs in the long run. Thus, if actions have different payoffs, then all agents choose the same action.

2.2.2 Epidemic Spreading and Gossiping Metaphors

As cited in [34] Gossip is one of the most usual social activities. This mechanism allows for the aggregation of a global information inside a population, through a periodic exchange and update of individual information among members of a group. The neighbourhood as well as the level of precision of the exchanged information play an important role on the nature of social learning which occurs by this way. This mechanism provides a powerful abstraction metaphor for information spreading, knowledge exchange and group organisation in large scale distributed systems. In peer-to-peer (P2P) systems, a class of protocols categorised as epidemic protocol has been proposed [50]. These protocols are characterised by their high robustness and large scalability. This metaphor has been also used for routing in sensor networks. For example in [8], a rumour routing algorithm for sensors networks is proposed. This algorithm is based on the idea of creating paths leading to each event and spreading events in the wide-network through the creation of an event flooding gradient field. A random walk exploration permits to find event paths when needed.

2.2.3 Trust and Reputation

Uncertainty and partial knowledge are a key characteristic of the natural world. Despite this uncertainty human beings make choices, take decisions, learn by experience, and adapt their behaviour.

Trust management systems deal with security policies, credentials and trust relationships, for example issuers of credentials. Most trust-based management systems combine higher-order logic with a proof brought by a requester that is checked at run-time. These systems are essentially based on delegation, and serve to authenticate and give access control to a requester [53]. Usually the requester brings the proof that a trusted third entity asserts that it is trustable or it can be granted access. These techniques have been designed for static systems, where an untrusted client performs some access control request to some trusted server [1, 6]. Similar systems for open distributed environment have also been realised, for instance [38] proposes a delegation logic including negative evidence, and delegation depth, as well as a proof of compliance for both parties involved in an interaction. The PolicyMaker system is a decentralised trust management systems [4] based on proof checking of credentials allowing entities to locally decide whether or not to accept credentials (without relying to a centralised certifying authority). Eigentrust [36] is a trust calculation algotrithm that allows to calculate a global emergent reputation from locally maintained trust values. Recently, more dynamic and adaptive schemas have been defined, which allow trust to *evolve* with time as a result of evidence, and allows to adapt the behaviour of principals consequently. We report here the results of the European funded SECURE [11] project, which has established an operational model for trust-based access control. Systems considered by the SECURE project are composed of a set of autonomous components, called principals, able to take decisions and initiatives, and are meaningful to trust or distrust. Principals maintain local *trust values* about other principals. A principal that receives a request for collaboration from another principal decides to actually interact with that principal or not on the basis of the current trust value it has on that principal for that particular action, and on the risk it may imply for performing it. If the trust value is too low, or the associated risk too high, a principal may reject the request. After each interaction, participants update the trust value they have in the partner, based on the evaluated outcome (good or bad) of the interaction. A principal may also ask or receive *recommendations* (in the form of trust values) about other principals. These recommendations are evaluated (they depend on the trust in the recommender), and serve for updating current trust values. Artificial systems built on the human notion trust as exposed above have the particularity to exhibit a self-organising behaviour [16], as identified by Nobel prize Ilya Prigogine and his colleagues [24]. Additional trust and reputation systems are surveyed in [25], and for the particular case of multi-agent systems they are reviewed in [41].

3 The Social Insects Behaviour Metaphor

Social insects societies such as ants, bees, wasps and termites exhibit many interesting complex behaviours such as emergent properties from local interactions between elementary behaviours achieved individually. The emergent collective behaviour is the outcome of a process of self-organisation, in which insects are engaged through their repeated actions and interactions with their evolving environment [32]. Self-organisation in social insects relies on an underlying mechanism : Stigmergy, originally introduced by Grassé in 1959 [26]. Grassé studied the behaviour of a kind of termites during the construction of their nests and noticed that the behavior of workers during the construction process is influenced by the structure of the constructions themselves. This mechanism is a powerful principle of cooperation in insect societies. It has been observed within many insect societies such as wasps, bees and ants. It is based on the use of the environment as a medium of inscription of past behaviours effects, to influence future behaviours. This mechanisms defines what is called auto-catalytic process, that is the more a process occurs, the more it has a chance to occur in the future. More generally, this mechanism shows how simple systems can produce a wide range of more complex coordinated behaviors, simply by exploiting the influence of the environment. Many behaviours in social insects, such as foraging or col-

lective sorting are rooted on the stigmergy mechanism.

Foraging is the collective behaviour through which ants collect food. During the foraging process, ants leave their nest and explore their environment following a random path. When an ant finds a source of food, it carries a piece of food and returns back to the nest by laying a trail of a hormone called pheromone along its route. This chemical substance persists in the environment for a particular amount of time before it evaporates. When other ants encounter a trail of pheromone, while exploring their environment, they are influenced to follow the trail until the food source, and while coming back to the nest they enforce the initial trail by depositing additional amounts of pheromone. The more the trail is followed, the more it is enforced and has a chance to followed by other ants in the future. Ants foraging behaviour have inspired many works in computing domains, ranging from "Ant Colony Optimisation" (ACO) metaheuristic for optimisation problems [18], to the design of ant-like systems using mobile agents with applications in several domains such as computers network routing and load-balancing [42][17][21], computers network security [20][23], information sharing in peer to peer systems [2], etc.

Collective clustering and sorting is a collective behaviour through which some social insects sort eggs, larvae and cocoons. As mentioned in [7], an ordering phenomenon is observed in some species of ants when bodies are collected and later dropped in some area. The probability of picking up an item is correlated with the density of items in the region where the operation occurs. This behaviour has been studied in robotics through simulations and real implementations [32]. Robots with primitive behaviour are able to achieve a spatial environment structuring by forming clusters of similar objects via the mechanism of stigmergy described above. Moreover, these kind of social insect behaviours have inspired many mechanisms for building artificial self-organised systems [7][32] [30] [39].

4 Business and Economics Approaches

4.1 Market-based Mechanisms

Market-based mechanisms are built along the lines of economic markets. In this approach, systems are modelled along the lines of some economic model in which participating entities act towards increasing their personal profit or utility. System wide parameters are modelled in a manner similar to macroeconomic variables such as economic growth. The parameters of the individual entities correspond to microeconomic parameters. The key point in such systems is to select suitable micro level parameter values and market interaction rules so that desired system goals, both local and global, are achieved.

Market-based approaches contrast the traditional way of modelling self-organisation and emergence in economic systems, which is primarily based on analytic general equilibrium models, for example as is done in [22]. The main problem with analytic approaches is that they cannot represent all possible situations due to the non-linearity of economic phenomena [10], which is due to the fact that economies are complex dynamic systems [48]. Instead, market-based approaches view macroeconomic phenomena as *emergent results* of local interactions of the economic entities [10, 33, 48]. An example is economic growth which can be *described* at the macro level but it can never be *explained* at that level [12]. The reason is that economic growth results from the interaction of a variety of economic actors, who create and use technology, and demanding customers.

There are numerous variations of market-based self-organisation mechanisms. An exemplar such mechanism which is based on the creative destruction principle is described in the following section.

4.1.1 Creative Destruction

Creative destruction is a term coined by Schumpeter [43] to denote a *"process of industrial mutation that incessantly revolutionizes the economic structure from within, incessantly destroying the old one, incessantly creating a new one."* In other words, creative destruction occurs when a new setting eliminates an old one leading to economic development. According to this view an economic system must destroy less efficient firms in order to make room for new, possibly more efficient entrants. A representative example of creative destruction is the evolution of personal computer industry which under the lead of Microsoft and Intel destroyed many mainframe computer companies; however, at the same time one of the most important technological achievements of this century was created.

The main roles that economic actors play in a market-based economy are those of producer, worker and consumer. Producers produce goods or provide services that consumers demand. Consumers consume the goods and use the services in exchange of some monetary or utility value. When there is high demand producers tend to hire workers to assist them in goods production or service provision in exchange of some wage. Since producers cannot sell their production beforehand, they must hold enough money to pay the workers in order to start up production and they can only get the necessary money by entering debt. According to the creative destruction principle, if producers are not able to pay the worker wages then they go bankrupt and they are removed from the system, for example they are reduced to simple workers, opening the way to other economic entities to try to become successful producers and satisfy the consumer demand.

The creative destruction process is better illustrated in a credit economy. In contrast to a monetary economy where producers can only borrow existing money from lenders, credit economy allows producers to obtain credit up to a

certain level from creditors in order to pay for production of new products. In this way, producers can more easily force their way into the market but the danger of becoming bankrupt is increased. To explain economic development in this framework one only needs to explain why entrepreneurs would want to introduce new products to the market. Effective entrepreneurs survive the battle and increase their profit. Failed entrepreneurs cannot repay their debt and therefore they go bankrupt and they are eliminated. As initially stated by Schumpeter [43] and later evaluated experimentally, for example [9], economic growth in this model is generated in cycles that emerge from the disturbance caused by entrepreneurs entering the market introducing new products.

In such a model there is particular interest from both the global, macro economic perspective and the local microeconomic one. Individual producers can decide on their entrepreneur policy so that to increase their profit and avoid the risk of getting bankrupt. On the other hand the economic system regulators can decide on the self-organisation rules so that to increase overall system production and growth.

4.2 Business Related Mechanisms

Business related mechanisms are based on business models and theories which use self-organisation. In an increasingly complex global economy, businesses are faced with unpredictable behaviours and fast pace of change. As a result, the emphasis in contemporary business models has shifted from efficiency to flexibility and the speed of adaptation.

More recent approaches, for example the one described in [46], increasingly introduce business models originating from the study of complex adaptive systems. Adaptive business organisations are guided and tied together by ideas, by their knowledge of themselves, and by what they do and can accomplish. Therefore, the focus in such models is on the complex relationships between different business components and the effects that a change into some part of the system or its environment, however distant, might have on the behaviour of the entire system.

As examples of self-organising business models we discuss personalised marketing and activity-based management.

4.2.1 Personalised Marketing

Personalised marketing refers to following a personalised market strategy for each individual customer which is evolving according to customer reactions [52]. A typical example of this approach is the *one-to-one variable pricing model* [29], which refers to providing an individual offer to each customer using Internet technologies. The model uses self-organisation in the marketing policies by changing customers targeted and the prices quoted based on market dynamics, customer characteristics and the business goals.

A shift towards to personalised marketing models is viewed as being driven by *syndication* [54]. Syndication involves the sale of the same good to many customers, who then integrate it with other offerings and redistribute it, as is the case in redistributing popular TV programs. An example of a company using syndication is FedEx which syndicates its tracking system in several ways [54]. The company allows customers to access computer systems via its Web site and monitor the status of their packages. For corporate customers FedEx provide software tools that enable the organisation to automate shipping and track packages using their own computing resource. Each customer is offered different prices depending on a variety of parameters. Many websites, such as eBay, also apply variable pricing for their offers.

4.2.2 Activity-based Management

Another example from the area of management is the *theory of activity* described in [49]. In this view a company consists of networks of working groups that can change their structure, links and behaviour in response to business requirements. The aim is to capture the self-organisation decisions that need to be taken during the business operations both by managers and by interactions between employees. The emphasis is on solving potential conflicts of interests in both the inner and the external co-operative activity of the company.

In this approach the structure of the company is virtual. There is no clear hierarchy and control; instead control effects can be initiated both vertically and horizontally via *"round table meetings"*, which are organised along the lines of assessment meetings normally held in companies to assess results and handle exceptions. In these virtual round tables suitable participants soon emerge as de facto leaders due to their knowledge and experience. Subsequently, leaders tend to participate in each newly formed "round table". The view expressed in [49] is that to model the interactions of participants in a "round table", it is necessary to simulate the whole activity of each of them including their reasoning and communication.

5 Socially Inspired Computing Applications: An Illustration

5.1 E-mails and WWW Oriented Applications

Based on the SECURE trust and risk security framework, an anti-spam tool has been developed which allows collaboration among e-mail users by exchanging recommendations about e-mail's senders. An authentication scheme has been combined to the SECURE framework in order to increase the level of sender authentication [44].

On the WWW, a plethora of systems have been developed for content retrieval, filtering or organisation using socially inspired computing. As an illustration, we present here a pioneering work [40], in information retrieval field which combined inspiration of social human behaviours, and economic markets to propose an interesting system for information retrieval on the web. In this work, documents are represented by keyword vectors, representing individuals (agents) of an artificial ecosystem. This population evolves through an evolutionary process of natural selection using a genetic algorithm to find documents which best fit the user request. The user feedback is used to reward (resp. to punish) the fittest individual (the less fitting individual) by giving it a credit value. These credits are then used by agents in a market based metaphor to estimate the cost of inhabiting the artificial ecosystem. The fittest agents have enough credits to continue living in the ecosystem and the less fitting agents will die. Another system called WACO has been proposed in [30]. The WACO system is composed of a population of agents deployed on the web to form clusters of semantically similar documents and dynamically organise the web content. These agent behaviours, take inspiration of social insect behaviours. They combine foraging ant behaviour and the collective sorting behaviour.

5.2 Computer Network Applications

T-Man is a generic protocol based on a gossip communication model and serves to solve the topology management problem [35]. Each node of the network maintains its local (logical) view of neighbours. A ranking function (e.g. a distance function between nodes) serves to reorganise the set of neighbours (e.g. increasing distance). Through local gossip messages, neighbour nodes exchange or combine their respective views. Gradually, in a bottom-up way, through gossiping and ranking, nodes adapt their list of neighbours, and consequently change and re-organise the network topology. The T-Man protocol is particularly suited for building robust overlay networks supporting P2P systems, especially in the presence of a high proportion of nodes joining and leaving the network.

The SLAC (Selfish Link and behaviour Adaptation to produce Cooperation) algorithm [28] favours self-organisation of P2P network's nodes into *tribes* (i.e. into specialised groups of nodes). The SLAC algorithm is a selfish re-wiring protocol, where by updating its links with other nodes in order to increase its utility function, a specific node leaves its current tribe, and joins a new one.

In addition to P2P systems, the SLAC algorithm has many potential applications, for instance to organise collaborative spam / virus filtering in which tribes of trusted peers share meta-information such as virus and spam signatures. This would elimite the need for trusted third parties with central servers.

5.3 Applications in Business and Economics

5.3.1 Business Community Networks

Typical applications of market-based self-organisation mechanisms can be found in the domains of business community networks [37]. An example of such approach is the self-organising semantic network of document indexing agents described in [45].

In such a network, agents maintain indices to actual documents and to other agents as well, treating both in a similar manner - based on the semantics of their content. The key feature in this approach is content dependent query redirection, based on semantic indexing. If an agent is unable find a document on a given topic, it re-directs the received query to the agents which believes are most likely to find it. The connections between the agents adapt themselves based on the history of successfully served queries, forming a distributed self-organising search engine which is capable of executing on heterogeneous servers over the internet and dynamically indexing all available documents. The important aspect of such a search engine is that each node, though possessing only limited amount of local information, can handle global queries.

Each piece of information received from an agent corrects the coordinates of its representation in the semantic index of the recipient. Furthermore, each link to an agent has also its own utility based rating. Those ratings are used for the selction of the right candidates for redirecting queries.

Rating adaptation is done using a *free market* approach. According to this approach agents provide chargeable search services to each other. Each query has some limited amount of network currency, termed *neuro*, which dissipates in the course of query processing in the network. Neuros circulating through the network are used by the agents to update their connections with the other agents, based on their utility, in a similar manner that money flow in a real economy determines the structure of business relationships.

The semantic network economy is based on the following simple rules:

- The cost of each delegated query processing is one neuro;
- The cost of each document (query) transaction is one neuro;
- Agents aim to minimize their expenditures.

According to these rules each agent keeps track of the balance of transactions of all other agents it is linked with. Agents are considered economically rational and aiming to maximise their profit they tend to delegate queries to experts in the query topic, thus minimizing effective cost of search in the network.

Similar market-based techniques are applied in trade networks where the aim is to select trade partners based on continually updated expected payoffs [27, 47].

6 Conclusion

In this paper we have surveyed some self-organising social approaches and presented their use as metaphors for distributed computing systems. These socially inspired computing techniques have shown their effectiveness for systems and applications evolving in distributed and highly dynamic environments, such like current complex networks. Social behaviours ranging from those observed in biological entities such as bacteria, cells and social insects, to animals and human societies, are rooted in the dynamics of their underlying social network. Social behaviour both shapes and is shaped by such social networks. One important characteristic of societies is their emergence as patterns developed from relatively simple interactions in a network of individuals. The obtained patterns are then enforced through dynamics underlying the so obtained social network. These systems are well known to exhibit interesting characteristics such as robustness, capacity of self-adaptation and survivability in uncertain and dynamic environment and tolerance to randomness. We have presented different mechanisms of social behaviours and showed their use in computing environments through some illustrative applications. Socially inspired computing metaphors, provide a real new paradigm for programming highly distributed and dynamic computing systems. However, proposed approaches are still developped in an ad hoc manner, and a real theory for socially inspired computing needs to be provided.

References

[1] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *ACM Conference on Computer and Communications Security*, pages 52–62, New York, NY, USA, 1999. ACM Press.

[2] O. Babaoglu, H. Meling, and A. Montresor. Anthill: A framework for the development of agent-based paeer to peer systems. In *Proceedings of the ICDCS'02*, Vienna, A., July 2002.

[3] Venkatesh Bala and Sanjeev Goyal. Learning from neighbours. *Review of Economic Studies*, 65(3):595–621, 1998.

[4] M. Balze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, pages 164–173, Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[5] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[6] L. Bauer, M. A. Schneider, and E. W. Felten. A proof-carrying authorization system. Technical report, Princeton University Computer Science, 2001.

[7] E. Bonabeau, G. Théraulaz, V. Fourcassié, and J-L. Deneubourg. The phase-ordering kinetics of cemetery organization in ants. Technical Report 98-01-008, Santa Fe Institute, 1998.

[8] D. Braginsky and D. Estrin. Rumour routing algorithm for sensor networks. In *Proceedings of the Fisrt Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, USA, September 2002.

[9] Charlotte Bruun. The economy as an agent-based whole simulating schumpeterian dynamics. *Industry and Innovation*, 10(4):475–491, 2003.

[10] Charlotte Bruun. Introduction to agent-based computational economics. Aalbog, Danemark, Aalborg University, Department of Economics, Politics and Public Administration, 2004. Available at: http://www.socsci.auc.dk/~cbruun/aceintro.pdf.

[11] V. Cahill and al. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing Magazine, special issue Dealing with Uncertainty*, 2(3):52–61, 2003.

[12] Bo Carlsson and Eliasson Gunnar. Industrial dynamics and endogenous growth. In *Industry and Innovation*, pages 1–25. Taylor & Francis, Abingdon, UK, 2003.

[13] C. Castelfranchi. The theory of social functions: challenges for computational social science and multi-agent learning. *Journal of Cognitive Systems Research*, 2(1):5–38, 2001.

[14] Kalyan Chatterjee and Susan H. Xu. Technology diffusion by learning from neighbours. *Advances in Applied Probability*, 36(2):355–376, 2004.

[15] Nada Dabbagh. Lev vygotsky's social development theory. Available at http://www.balancedreading.com/vygotsky.html.

[16] G. Di Marzo Serugendo. Trust as an interaction mechanism for self-organising systems. In *International Conference on Complex Systems (ICCS'04)*, 2004. Available at http://cui.unige.ch/~dimarzo/papers/iccs04.pdf.

[17] M. Dorigo and G. Di Caro. Ants colonies for adaptive routing in packet-switched communication networks. *Lecture Notes in Computer Science*, page 673, 1998.

[18] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 26(1), 1996.

[19] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the Special Interest Group on Data Communications (ACM SIGCOMM'99)*, pages 251–262, New York, NY, USA, 1999. ACM Press.

[20] S. Fenet and S. Hassas. A distributed intrusion detection and response system based on mobile autonomous agents using social insects communication. *Electronic Notes in Theoretical Computer Science*, 63:21–31, 2002.

[21] S. Fenet and S. Hassas. An ant based system for dynamic multiple criteria balancing. In *Proceedings of the Fisrt Workshop on ANT Systems*, Brussels, Belgium, September 1998.

[22] Sergio Focardi, Silvano Cincotti, and Michele Marchesi. Self-organization and market crashes. *Journal of Economic Behavior and Organization*, 49(2):241–267, 2002.

[23] N. Foukia, S. Hassas, S. Fenet, and P. Albuquerque. Combining immune systems and social insect metaphors: a paradigm for distributed intrusion detection and response systems. In *Proceedings of the 5th International Workshop on Mobile Agents for Telecommunication Applications, MATA'03*, Marrakech, Morocco, October 2003. Lecture Notes in Computer Science -Springer Verlag.

[24] P. Glansdorff and I. Prigogine. *Thermodynamic study of structure, stability and fluctuations*. Wiley, 1971.

[25] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[26] P.P Grassé. La reconstruction du nid et les interactions inter-individuelles chez les bellicoitermes natalenis et cubitermes, la théorie de la stigmergie - essai d'interprétation des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

[27] Sergei Guriev, Igor Pospelov, and Margarita Shakhova. Self-organization of trade networks in an economy with imperfect infrastructure. In *Second International Conference on Computing in Economics and Finance*, volume 22. Society for Computational Economics, Geneva, Switzerland, 1996.

[28] D. Hales. Choose your tribe! evolution at the next level in a peer-to-peer network. In S. Brueckner, G. Di Marzo Serugendo, D. Hales, and F. Zambonelli, editors, *Engineering Self-Organising Applications (ESOA'05)*, Utrecht, The Netherlands, July 2005. (to appear).

[29] Glenn Hardaker and Gary Graham. Energizing your e-commerce through self-organising collaborative marketing networks. Technical report, School of Business, University of Salford, UK, 2002.

[30] S. Hassas. Using swarm intelligence for dynamic web content organization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 19–25, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[31] F. A. Hayek. *Studies in philosophy, politics and economics*. Routledge & Kegan, London, 1967.

[32] O. Holland and C. Melhuis. Stigmergy, self-organization and sorting in collective robotics. *Artificial Life*, 5(2):173–202, 1999.

[33] Peter Howitt and Robert Clower. The emergence of economic organization. *Journal of Economic Behavior and Organization*, 41(1):55–84, 2000.

[34] M. Jelasity. Engineering emergence through gossip. In Bruce Edmonds, Nigel Gilbert, Steven Gustafson, David Hales, and Natalio Krasnogor, editors, *Proceedings of the Joint Symposium on Socially-Inspired Computing*, pages 123–126, Hatfield, UK, 2005. University of Hertfordshire.

[35] M. Jelasity and O. Babaoglu. T-man: Gossip-based overlay topology management. In S. Brueckner, G. Di Marzo Serugendo, D. Hales, and F. Zambonelli, editors, *Engineering Self-Organising Applications (ESOA'05)*, Utrecht, The Netherlands, July 2005. (to appear).

[36] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *12th International World Wide Web Conference : WWW2003*, pages 640–651, Budapest, Hungary, May 20-24 2003.

[37] Ulrike Lechner and Beat F. Schmid. Communities - business models and system architectures: The blueprint of mp3.com, napster and gnutella revisited. In E. Sprague, editor, *the 34th Hawaii International Conference on System Sciences*, 2001.

[38] N. Li, J. Feigenbaum, and B. N. Grosof. A logic-based knowledge representation for authorization with delegation. In *12th IEEE Computer Security Foundations Workshop*, page 162, 1999.

[39] J.-P. Mano, C. Bourjot, G. Lopardo, and P. Glize. Bio-inspired mechanisms for artificial self-organised systems. *Informatica*, In press, 2006.

[40] A. Moukas. Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. *Applied Artificial Intelligence*, 11(5):437–457, 1997.

[41] S. D. Ramchurn, D. Huynh, and N. Jennings. Trust in multi-agent systems. *Knowledge Engineering Review*, 19(1):1–25, 2004.

[42] R.Schoonderwoerd, O. Holland, and J.Bruten. Ant-like agents for load balancing in telecommunications networks. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 209–216, February 5-8 1997.

[43] J. A. Schumpeter. The economy as a whole - seventh chapter of the theory of economic development. *Industry and Innovation*, 9(1/2), 2002.

[44] Jean-Marc Seigneur, Nathan Dimmock, Ciarn Bryce, and Christian Damsgaard Jensen. Combating Spam with TEA, Trustworthy Email Addresses. In *Proceedings of the Second Annual Conference on Privacy, Security and Trust (PST'04)*, pages 47–58, Fredericton, New Brunswick, Canada, October 2004.

[45] Sergey Shumsky. Self-organizing internet semantic network. White paper, NeurOK LLC, 2001.

[46] Max Stewart. *The Coevolving Organization*. Decomplexity Associates LtD, Rutland, UK, 2001.

[47] Leigh Tesfatsion. A trade network game with endogenous partner selection. In H. Amman, B. Rustem, and A. B. Whinston, editors, *Computational Approaches to Economic Problems*. Kluwer Academic Publishers, 1997.

[48] Leigh Tesfatsion. Agent-based computational economics: A constructive approach to economic theory. In K. L. Judd and Leigh Tesfatsion, editors, *Handbook of Computational Economics, Volume 2: Agent-Based Computational Economics*, Handbooks in Economics Series. North-Holland, 2005.

[49] V. A. Vittikh and P. O. Skobelev. Multi-agent systems for modelling of self-organization and cooperation processes. In *XIII Intern. Conference on the Application of Artificial Intelligence in Engineering*, pages 91–96, Galway, Ireland, 2002.

[50] S. Voulgaris, M. Jelasity, and M. van Steen. A robust and scalable peer-to-peer gossiping protocol. In G. Moro, C. Sartori, and M.P. Singh, editors, *Proceedings of Agents and Peer-to-Peer Computing: Second International Workshop, AP2PC03*, volume 2872 of *Lecture Notes in Artificial Intelligence*, pages 47–58, Berlin, 2003. Springer-Verlag.

[51] L.S. Vygotsky. *Mind and society: The development of higher mental processes*. Harvard University Press, Cambridge, MA, USA, 1978.

[52] S.C. Wang, K.Q. Yan, and C.H. Wei. Mobile target advertising by combining self-organization map and decision tree. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEEi04)*, pages 249–252, 2004.

[53] S. Weeks. Understanding trust management systems. In *IEEE Symposium on Security and Privacy*, pages 94–105, 2001.

[54] K. Werbach. Syndication: The emerging model for business in the internet era. *Harvard Business Review*, 85:85–93, 2000.

Applications of Self-Organising Multi-Agent Systems: An Initial Framework for Comparison

Carole Bernon
IRIT, University Paul Sabatier
31062 Toulouse Cedex 09, France
E-mail: bernon@irit.fr, http://www.irit.fr/SMAC

Vincent Chevrier
LORIA
BP 239
54506 Vandoeuvre Les Nancy Cedex, France
E-mail: chevrier@loria.fr, http://www.loria.fr/~chevrier/

Vincent Hilaire
SeT, UTBM
90010 Belfort Cedex, France
E-mail: Vincent.Hilaire@utbm.fr, http://set.utbm.fr/membres/hilaire/index.php

Paul Marrow
Pervasive ICT Research Centre, BT Group plc
Orion 1 PP 12, Adastral Park, Ipswich IP5 3RE, United Kingdom
E-mail: paul.marrow@bt.com, http://www.btplc.com/

Keywords: multi-agent system, self-organisation, applications, software

Received: April 18, 2005

A lot of work is devoted to formalizing and devising architectures for agents' cooperative behaviour, for coordinating the behaviour of individual agents within groups, as well as to designing agent societies using social laws. However, providing agents with abilities to automatically devise societies so as to form coherent emergent groups that coordinate their behaviour via social laws, is highly challenging. These systems are called self-organised. We are beginning to understand some of the ways in which self-organised agent systems can be devised. In this perspective, this paper provides several examples of multi-agent systems in which self-organisation, based on different mechanisms, is used to solve complex problems. Several criteria for comparison of self-organisation between the different applications are provided.

Povzetek: Članek opisuje primere in kriterije samoorgarnizacije v agentnih sistemih.

1 Introduction

Multi-Agent Systems (MAS) have attracted much attention as means of developing applications where it is beneficial to define function through many autonomous elements. As multi-agent systems get more complex, questions arise about the best way to control agent activity, and thus application performance. Centralised control of MAS is one approach, but is of limited use because of the risk of dependency on the controlling element, and the consequential lack of robustness. This also makes little sense when agents have capabilities of autonomy that can provide useful benefits in applications. Partially or completely decentralised control is an alternative, but means of implementing this without disrupting agent performance in support of applications are important. Mechanisms of self-organisation [7] are useful because agents can be organised into configurations for useful application without imposing external centralised controls.

This paper discusses several different mechanisms for generating self-organisation in multi-agent systems [8]. Reactive multi-agent systems [29] provide the basis for self-organisation in several examples as the interaction between the agents and their environment provides the flexibility for dynamic change. Cooperation drives self-organisation in the AMAS agent modelling theory [2][8]. The holon concept can also be used to define and analyse self-organising agent systems [26]. In this introductory part, these approaches are now discussed in a more detailed way.

1.1 Self-organisation by Reactive Multi-agent Systems

Reactive multi-agent systems [29] are systems made up of simply behaving units with decentralized control. Agents are situated in a dynamic environment through which they interact. They are characterized by limited (possibly no) representation of themselves, of others and

of the environment. Their behaviours are based on stimulus-response rules. Decision-making is based on limited information about the environment and on limited internal states and does not refer to explicit deliberation. The individuals do not have an explicit representation of the collective task to be achieved because of their simplicity. Therefore, the solution of the problem is a consequence of successive interactions between agents and the environment. Their characteristics enable them to adapt dynamically their function or structure to changing conditions without external intervention.

Using such a model to solve a given problem requires designing a system as three components: the environment, the agent behaviours and the dynamics of the whole such that the agent society is able to fulfil its requirements with a reasonable efficiency.

1.2 Self-organisation using Cooperative Information Agents

A specific example of a platform for self-organisation using reactive multi-agents is provided by the DIET Agents platform. This platform [14][6] is a suitable basis for self-organising applications using cooperative information agents. This platform was developed as part of the EU DIET project, inspired by the way that complexity emerges in natural ecosystems.

The DIET Agents platform [14][6] is designed as a three layer architecture: (1) core layer; (2) application reusable component layer; (3) application layer. The core layer provides the minimal software needed to implement multi-agent functionality, through the DIET platform kernel. It also provides basic support for debugging and visualisation. The basic classes and elements in the DIET platform kernel are arranged around an element hierarchy: worlds, environments; agents; connections, and messages.

Agents are located in environments, and can form connections with each other through which messages can be passed. Multiple environments can be situated in worlds. Agents are initially created with only four possible behaviours: creation (of other agents); destruction (of itself); communication (with other agents); migration (between environments). But they are designed so that their properties can be extended.

The other two layers of the platform, the application reusable component (ARC) layer, and application layer support this extension. The ARC layer provides functionality that can be shared between applications, but is not essential for the DIET kernel, while the application layer provides application-specific functionality. Software for applications can be developed in this layer without having to disrupt the core layer.

This platform is appropriate for applications involving cooperative information exchange because individual agents can take on cooperative behaviour by extension of their autonomous capability. The platform is designed that agent action is resource-constrained, so that actions will stop if they start consuming too much system resources. Actions are also fail-fast; they will fail if they are not executed immediately. In this way applications requiring the interaction of many agents can be supported within realistic resource constraints.

This platform is also suitable for applications involving self-organisation because no decisions have been made about how agents should be organised, and they are free to rearrange within and between environments according to application requirements.

1.3 Self-organisation by Cooperation in AMAS

For several years the SMAC (for Cooperative MAS) team has studied self-organisation as a means to get rid of the complexity and openness of computing applications [2]. A theory has been proposed (called AMAS for Adaptive Multi-Agent Systems) in which cooperation is the engine thanks to which the system self-organises for adapting to changes coming from its environment (see [2] and section 4.4. in [8]). Cooperation in this context is defined by three meta-rules: (1) perceived signals are understood without ambiguity, (2) received information is useful for the agent's reasoning, and (3) reasoning leads to useful actions toward other agents. Interactions between agents of the system depend only on the local view they have and their ability to cooperate with each other. These modifications make the organisation of the system also change and therefore make the global behaviour of the system emerge. At the agent level, cooperation is described in a proscriptive way: an agent knows how to detect situations it judges being non cooperative, from its point of view, and acts for always trying to remain cooperative toward others but also toward itself.

1.4 Self-organisation by Holons

According to Koestler, a holon is a self-similar structure that may consist of several holons as sub-structures [17]. The hierarchical structure composed of holons is called a holarchy. Holarchies allow the description of systems as recursive self-similar entities which constitute the holons.

We have chosen to describe the behaviour of the members of a holon and their interactions in terms of roles and organisation. These roles represent the "status" of the holon inside a specific holon. In our approach each holon may play four roles: StandAlone, Head, Part and Multi-Part. As a holon joins a HMAS (Holonic Multi-Agent System) Organisation, it has no special bindings and does not collaborate with any other holon.

This situation represents a *Stand Alone* Behaviour. In this state, the agent's decisions are not attached to any restriction but its own goals and objectives. The holon will remain in this state as long as it is satisfied. The Stand-Alone represents how "non-members" are seen by an existing holon. Following the Holonic Paradigm, the holon seen as a Stand-Alone can actually be the Representative of a holon.

As the representative, the holon plays the *Head* role. According to the objective and rules of the holon, the Head responsibilities and rights may range from merely administrative tasks to be able to take decisions concerning all members. The head is not necessarily a unique holon. After an holon starts performing the Head Role, it will be the representative of the members of his Holon at this level and therefore, able to engage the holon in new tasks.

Members not playing the *Head* role are considered as *Parts* of the holon. Once a holon is accepted in a Holon, its autonomy is reduced because of its obligations with the Holon. The degree of this autonomy lost may vary according to the holon's purpose.

The *MultiPart* Role is a special case of the Part Role. This role is played by holons belonging to more than one Holon. Interesting possibilities are available when a holon is shared.

In order to enable holons to dynamically change their roles, we define a notion of satisfaction. Each holon tries to be self-satisfied. If it cannot reach a satisfaction threshold it tries to change its role. Eventually the last concept of the framework is affinity. The affinity enables one StandAlone holon to choose with which holon to merge. It measures the compatibility of the holon's goals and services.

Self-organisation by holons uses direct interactions and cooperation (see section 4.5 in [8]).

1.5 Overview

Given the existence of multiple mechanisms for generating self-organisation in multi-agent systems, what can self-organised systems be used for? Section 2 reviews a variety of examples of MAS applications drawing upon self-organisation. Section 3 seeks to compare these applications, by identifying some criteria that are general to multi-agent systems. Section 4 provides a conclusion.

2 Applications

The diversity of approaches for stimulating self-organisation within multi-agent systems means that MAS have the potential to support a variety of applications. This section describes some example applications using MAS that draw upon self-organisation to make the applications more effective.

Two applications address problems in information retrieval, using middle agents (section 2.1) and evolutionary algorithms (section 2.2). Further applications are considered in the areas of timetabling (section 2.3), flood forecasting (section 2.4), land use allocation (section 2.5), localisation and tracking (section 2.6), adaptive meshing in wireless networks (section 2.7) and traffic simulation (section 2.8). Other examples of application can be found in [18].This wide range of examples gives an indication of the usefulness of self-organisation in achieving the complex behaviour required for real-world applications.

2.1 Self-organisation of User Communities using Middle Agents

Multi-agent systems can be used to support information exchange within user communities by providing each user with a user agent that represents their interests. But how does a user agent make contact with other user agents that represent users with common interests? Assuming that not all users know each other, which is probably realistic, a pure peer-to-peer network could be used. This would involve flooding a network with queries. But this is inefficient, and risks overloading the system with queries. Middle agents or brokers are an alternative - user agents communicate with middle agents [5]. Multi-agent systems for information exchange using middle agents have been proposed which are centralised (e.g. [22]) - all queries go to one broker - but there is a risk that they will not be so robust, if the middle agent does not perform well. In this example we consider an application where middle agents are used in a decentralised configuration.

The self-organising communities application [31] assigns each user a user agent. There are also multiple middle agents in the system. User agents do not retain a profile for their user, but they forward queries to the middle agents. The user agents carry and seek to acquire information for their users. Each user agent registers with at least one middle agent. Once they are registered with the middle agent, the middle agent can access information that the user agent holds that may be of interest to other user agents. Each middle agent receives queries from multiple user agents. Given these queries the middle agent carries out a search of the pool of information it holds from user agents already registered with it. If it can respond to the query using this information then that information is dispatched to the user agent that issued the original query and the search is rapidly completed. If not, the middle agent can interact with other middle agents to try and obtain information from them. Once the middle agent has carried out this search, it relays the result to the user agent.

The middle agent then examines whether the search was successful, and if so provides a positive mark to the two user agents, both the requestor and the provider. If the search was unsuccessful, the requestor gets a negative mark, to indicate load on the system. After marks have been assigned the middle agent checks the location of the requestor and provider user agents. If the search has been successful, and the requestor and provider agents are not already registered with it, the middle agent requests the middle agent with which the provider agent is currently registered to transfer the provider agent to the group of the requestor agent. Movement of agents between groups is regulated by the awards given to user agents following responses to queries, and designed to get user agents into the same groups around middle agents where they often have queries covering common areas. In this way user communities can be built up using user agents and middle agents, without any central control on agent behaviour. This is a highly scalable process that continues to operate highly efficiently even as the number of users increases substantially.

2.2 Self-organisation through Evolving Agent Populations

We can use a MAS to represent user interests through user agents, but given that there may be many different users in different locations there may be problem in finding other users to interact with. The evolving preferences application [18] considers an application scenario where many users interact with each other via a DIET Agents platform supporting user agents. Each user deploys a user agent in a DIET environment, but because users may interact with the system in different contexts,

there will be a lot of different environments. The different environments are connected in a peer network. User agents stay in their own environment, but create a population of scout agents that they send through the peer network to search out other user agents representing users with common interests.

Each scout agent carries information representing the interests of the user that it represents. It also has a preference for environments determined by a bitstring genome created when it is generated. Based on this genome it will search out other environments and interact with other scout agents in them. Scout agents then return to their home environment and report back information that they have gathered in other environments; both about different environments and about their success in interacting with other scout agents representing similar interests. On return to their home environment scout agents are destroyed, but their genome is used in an evolutionary algorithm where the selection criteria is defined by the success of the scout agent in locating environments where there are other scout agents to interact with that have similar interests. Over time the evolutionary algorithm converges to a situation where scout agents will converge in environments according to their users' preferences, so that different environments hold different user agents that can interact on behalf of their users. This shows how an evolutionary algorithm can be combined with agent interaction in a distributed network to stimulate self-organisation of agents into different environments, and thus to stimulate information exchange between users in different environments.

2.3 Self-organisation for the School Timetabling Problem

An example of a classical constraint-satisfaction problem (CSP) is the school timetabling problem in which a timetable for a certain duration must be found while respecting the explicit constraints (availability, specialisation, equipment needed…) of different stakeholders (teachers, student groups and possibly rooms) as well as their implicit constraints (for example, impossibility to be in two places at the same time). The inherent distributed aspect of the timetabling problem explains a processing by a MAS. Unlike most of the approaches using MAS (for instance [4]), agents do not use negotiation to find a solution in ETTO (Emergent Time Table Organisation), the problem solver presented here [23].

The *environment* is made of a three-dimensional virtual grid composed of cells. Each cell represents a time slot for a given day, for a given hour, and for a given lecture room.

Two kinds of cooperative *agents* were identified: a Representative Agents (RA) and Booking Agents (BA). A RA is associated with every human stakeholder, manages its constraints and represents an interface with the real world. RAs delegate time slot and room search to Bas which are the actual self-organizing agents. A BA explores the grid to find free cells and meet potential partners in order to fulfil its aim: booking a time slot for a given lecture to give (for a teacher) or to take (for a student group) in accordance with constraints used by its proxy RA.

The *behavioural model* is based on the AMAS theory, the engine of self-organisation is cooperation. Five different situations for reorganisation are identified based on the three meta-rules ensuring cooperation (see section 1.3) For instance, if a BA ba1 encounters, in a given cell, another BA ba2 with which it cannot partner (for example, two teachers meet), ba1 judges this situation as incompetence and changes its location to find a more relevant partner. Furthermore to enable a more efficient exploration of partnership possibilities, ba1 will memorise the location and the BAs it may know via ba2, to exchange them during further encounters. In a cooperative situation, a BA books the cell in which it is situated, and partners with another BA.

The positive results obtained by now show that the approach used is suited for this kind of problem. BAs are able to relax constraints to find a solution. A solution is found when constraints or stakeholders vary (added or removed) in a dynamical way. Furthermore, the ability to insert agents has enabled us to show that adding supernumerary agents helps finding a solution and gives better results. This can be explained by the fact that the added agents can disrupt others which are satisfied with a solution that could be optimised. However ETTO has weaknesses. By nature, cooperative agents in AMAS have only a limited knowledge about their environment and do not know the global goal to achieve as well as the global cost of the solution they may found. Thus they go on exploring the grid to find a more relevant solution even if the best solution is already found. An external observer has to stop the solving process when the organisation fits his requirements.

Many approaches have been used to try to solve such a problem (see for example, the “Practice and Theory of Automated Timetabling“ at the URL: http://mat.gsia.cmu.edu/PATAT04/). Most of them are (distributed) CSP-based solvers, some are agent-based solutions, some use evolutionary approaches and others ant algorithms [28]. Timetabling problems in the real world are dynamic problems, restarting from scratch each time a constraint is modified (added, removed) would not be efficient and few works are interested in this problem. Usually, the main objective is to have the smallest impact possible on the current solution as in [21] in which this is done by introducing a new search algorithm that limits the number of additional perturbations. In ETTO, self-organisation enables the system to adapt to perturbations and changes in its environment because modifying a stakeholder’s constraint makes the corresponding BA question its bookings and its possible partnership. If it judges that they are inconsistent with its new state, it tries to find new ones by roaming the grid and applying its usual behaviour.

2.4 Self-organisation for Flood Forecasting

Flood forecasting is a complex dynamic problem, parameters that can explain this phenomenon are numerous and heterogeneous: including hygrometry, declivity, surface, nature and permeability of the ground,

rain heights, stations topologies, … Current forecasting systems have a physical approach of this phenomenon: the better these parameters are known, the better results are. Tuning these parameters for a forecast station take several months and they have to be adjusted when environmental conditions evolve.

The STAFF real-time simulator uses an adaptive model for flood forecasting, which is composed of two levels of self-organizing multi-agent systems [13].

The *environment* is made of the sensors of the Garonne river basin.

Agents of the lower level represent each physical sensor. Such an agent has to encapsulate its datum for determining its influence on the forecast that the system has to model. The goal of each upper level agent is to compute the water level variation during a unitary period (typically an hour); for that, it uses a weighted sum of agents in the lower level.

The *behavioural model* is based on the AMAS theory. The hydrological model's adaptive nature is obtained by adjustment of these weights, decided from cooperation between the agents. Agents do not know the objective of the global system, the self-organisation by cooperation between agents defines how the model has to be adjusted according to the input data, the results coming from other agents and the error made on the forecast. This makes the model generic and improves its performances.

Positive results were obtained showing that the model correctly followed the real evolution of the flood even in limit use cases (such as noisy and missing data, totally upstream stations or real-time learning) in which usual hydrological models are inadequate. The model does not need any predefined parameters because it is adjusted just once, when installed, using historical flood data. For example, one week was sufficient to adjust the 24 models currently used for the stations making flood forecasting in the Garonne river basin.

Classical physic-hydrological forecasting models are mathematical approaches that consist in generic formula which parameters are tuned from measures on ground and from historical account of flood. Neural networks have been used in flood forecasting, for instance in [30] or used with self-organising feature map [15]. In the former approach, relevant stations must be selected by hand and the learning algorithm is not a generic one. In both approaches, contrary to STAFF, there is no real-time learning.

2.5 Self-organisation for Land Use Allocation

Based on a real-world problem, we applied a self-organising approach to simulate the assignment of land-use categories in a farming territory, in the north-east of France [9][8]. This problem exhibits a function to optimise, while respecting a set of constraints, both local (compatibility of grounds and land-use categories) and global (ratio of production between land-use categories). This problem is one instance of quadratic assignment problem.

The *environment* is the set of available zones in the farming territory, each zone is featured by its surface, its distance to the village, the kind of soil, etc.

Agents are gathered into groups, each being associated to a land-use category. A group has a goal to satisfy by conquering spatial zones in the environment while respecting some constraints.

The *behavioural model* is based on a few principles inspired by the eco-problem solving approach [11]. An agent can conquer a zone in the environment and then contribute to the satisfaction of its group; the zones are more or less attractive for an agent; when searching for a zone, an agent chooses the most attractive one. If the zone is free the agent occupies it; if it is already occupied, the two agents have to fight, and the outcome is determined by the respective strengths of their groups. Finally, the strength of a group decreases while its satisfaction increases, in order to ensure that groups farther from their objectives gain an advantage over those closer.

The problem-solving process exhibits interesting properties. The system produces results that fit the expert's requirements and that are comparable with results obtained by simulated annealing.

The dynamic of the problem-solving process is convergent to a stable state (which is a solution to the problem); is an anytime process (the system can be stopped at any step and is able to produce a solution the quality of which is dependent of the number of steps). Furthermore, the model exhibits self-adaptation properties: at runtime we can add or remove zones or land-use category and the system stabilises again to a solution. We obtained the same properties when modifying a group's goal.

A lot of works on optimisation problems exist (e.g. [3]), however most of them are not self-organizing; two exceptions are built on reactive agents that self-organize. The first (Ant Colony Optimisation) is inspired by the foraging behaviour of ants [1] and the second (Particle Swarm Optimisation) by flocking [10]. Both provide results comparable to more conventional optimisation methods.

In our case, we compared our system with simulated annealing. Simulated annealing provided better mean results even if the best solutions were found by the MAS approach.

2.6 Localisation and Tracking using Self-organisation

The localisation task can be defined as finding the position of an object (or more than one), mobile or not, in a well defined referential location. The tracking problem is to provide a succession of positions that are spatially and temporally coherent. We proposed a reactive model to tackle this issue [12].

The *environment* of the agents is a representation of the real world. It is a square grid in which each state represents a target's possible position and is featured by an altitude that represents the possibility of presence of a target at this position and is provided by sensors. Environment dynamics is determined by accumulation and evaporation principles. The altitudes are refreshed

(accumulation) continuously as soon as sensors can furnish data. In the absence of data, altitude is decreasing (evaporation).

Agents are equivalent to weighted particles evolving in an environment of force field.

The *behavioural model* used is inspired by a model of flocking [24] but is expressed through a formulation taken from Newtonian physics (i.e. all behaviours are expressed as a combination of classical forces). Agents are attracted by position according to their altitude and are mutually repulse each other. Agents' movements are the consequence of these forces.

We designed these antagonist behaviours to obtain a focusing of the agents on the position of highest altitude and a homogeneous spatial distribution of them in the rest of the environment (where there is a null altitude). Focusing is an emergent phenomenon, and is the solution of the problem: a group corresponds to the detection of a target.

We compare our proposition with the Kalman filter in case of real robots' localisation [27]. The Kalman filter is better than the agent-based method when there is no noise. This advantage decreases when noise is introduced. Furthermore, the agent-based approach requires less knowledge about the problem than the Kalman one.

The approach is able, at runtime, to deal with a variable number of targets and it is possible to add or remove sensors which is very difficult to take into account with classical algorithms. As far as we know there is no self-organized approach for localization.

2.7 Self-organisation for Adaptive Meshing in Cellular Radio Networks

A distinguishing feature of cellular radio mobile networks is the rapid increase of the consumer demand and the ensuing complexity in their design and management. Responding to this demand requires the space to be partitioned between a large amount of service units or cells. The adaptive meshing problem for dimensioning considers traffic statistics as a predefined resource that must be attributed to many adaptive low power Base Transceiver Stations. The environment is discretized in meshes which contains a number of resources according to traffic statistics. Each mesh will be assigned an agent whose main and unique goal is to cover the traffic in that mesh [26]. This goal must be accomplished respecting certain constraints like geometry and the maximal traffic that an antenna can cover. The problem solving is done by building up holons which cover a resource.

In the adaptive mesh problem, a stand-alone holon must ensure the coverage of its resource, then it will try to join a mesh immediately. The only situation where it remains in a stand-alone role is when its resource can get an antenna for it alone. A holon that performs the head role will be responsible for respecting the constraints of a mesh. It will be representing a possible mesh in the system, and will accept or refuse other holon's requests to fusion according the constraints. Although all heads represent possible meshes in the system, a Holon Head can decide to leave its role if, after trying to improve the Holon's satisfaction, the satisfaction is insufficient to remain as a Holon. In order to improve the Holon's satisfaction, will accept new holons to increase the Holon covered resource, or will command member holons to leave the Holon if they don't respect the geometrical constraints or if the covered resource has exceeded the maximum.

A holon gets the Part role if negotiations with a Holon succeed. It will remain in the Holon if its satisfaction level is raising. However, it is also possible that the agent receives a command to leave the holon, in that case, it must return to Stand-Alone and restart the merging process.

The holon's identifier should give the position of the holon's resource (X, Y coordinates) and the traffic it contains.

Using these values, a holon can determine whether or not to merge. As explained before, the affinity should give a measure of the compatibility of the holon's goal and services. In this particular case, both holons will have the same goal, to ensure the coverage of their resources. Therefore, the main problem is to ensure that the geometrical constraints are respected. The affinity could be decomposed in two main parts: the distance affinity will provide a geometry dependent value used to ensure that the geometrical constraints are respected. As we need square meshes, we will use two parameters to test the distance affinity. First, we will check if the holon trying to merge is inside the acceptance distance. The Resource affinity is used to ensure that the limits of an antenna are not exceeded.

2.8 Self-organisation for Traffic Simulation

Multi-agent Systems operate within an environment and therefore, in an Agent Based Simulation (ABS) special attention must go to the analysis, model and implementation of the environment [20].

We propose the use of holarchies for the modelling of environments [25]. We simulate traffic within an industrial plant. The environment of this simulation is defined by the topology and road network of the plant. The concept of road is divided into links. A link represents a one-way lane of a road. A segment is composed of two exchange points, called input and output exchange points, and a link. Exchange points let vehicles pass from one link to the other. An exchange point is always shared by at least two segments and thus plays the multi-part role. The industrial plant is composed of a set of zones, that in turn contain Buildings and Segments. Buildings and Segments can also communicate through shared exchange points. Usually an exchange point represents a crossroad, but in can also represent an entrance used by trucks to access buildings. The agents will be the different vehicles driving through the plant. Each holon of the holarchy represents a specific context. For this simulation (HTS in the sequel) it's a specific place in the plant. These places have different granularity levels according to their level in the holarchy. During the simulation vehicle agents move from one holon to another and the granularity is chosen by execution or simulation constraints such as which

features can be observed. The dynamic choice of the environment granularity level during the simulation is transparent for the agents. The problem here is the simulation of traffic and the solving process is again based upon building and re-organisation of holarchies as vehicles drive through the plant and change the holon they belong to.

This holarchy defines the organisational and topological structure in which agents will evolve. Each environmental holon will enforce contextual physical laws and represent a specific granularity level of the real plant topology. This holarchy is predefined as it represents the real plant environment. Indeed, the latter can't evolve and the physical laws we need to enforce are known *a priori*. All necessary information to simulate the traffic inside a link is local (other vehicles, road signs, etc). This makes the model easier to distribute in a network and leaves the door open to Real-time applications as well as Virtual Reality implementations.

This approach has many advantages for the simulation. Indeed, such a definition of the environment allows the progressive decomposition of the environment complexity and enables to assign environmental laws to the pertinent holon.

3 Comparison and Discussion

This part is an attempt to compare self-organised systems presented above using a list of criteria inspired by the work done in the AgentLinkIII "Self-organisation in MAS" Technical Forum Group. The first paragraph of this section lists all the criteria we use along a classification based on the level at which they can be expressed. In the next section, each criterion is discussed in more detail with regard to our different approaches.

3.1 Criteria Used

Some of the criteria we use can be considered as descriptive/static criteria of the approach whilst others are related to the dynamical aspects of the problem solving process. Criteria belonging to the first group can be stated without running the system but by "simply" looking at its description:

- Absence of external or of centralised control: no entity, external or internal to the system, is explicitly responsible for the actions of agents or for centralising information flow.
- Dynamic operation: the solution is built in a dynamic way and not by applying a predefined plan or by instantiating a predefined solution.
- Emergent properties: properties that emerge from local interactions within the system and that cannot be deduced by simply observing individual behaviours (see section 3 in [8]).
- Simple local rules: do simple individual behavioural rules lead to complex patterns?
- Reusability: is the solution (or part of it) reusable in other contexts?

On the contrary, criteria found in the second group need experiments to be tested:

- Anytime property: the system can be stopped at any step and is able to produce a solution the quality of which is dependent of the number of steps.
- Instability: is the system non-linear, is it sensitive to parameters variations?
- Adaptation: how does the system react to changes coming from the environment of the system?

3.2 Discussion

In some applications, the absence of ***external control*** may not exist and some entities may centralise information or decision. Therefore, applications can be classified from fully decentralised to partially centralised.

For example, localisation application is fully decentralised, as well as timetabling or flood forecasting.

On the contrary, in the holonic approach, Head role refers to a partial centralisation of decisions. This loss of autonomy corresponds to the holarchy handling. Moreover, the Head role may be played by the entire holon as a group.

The self-organising communities application, using middle agents, is decentralised in that information retrieval is distributed across multiple middle agents. But the evolving preferences application, which evolves environmental preferences for information exchange, includes an element of centralisation through the use of selection on scout agent populations; although this is only partial as multiple populations are selected in parallel.

In all the applications presented above, the solution is built ***dynamically.***

Emergent properties refer to ***simplicity*** of individuals, in terms of local rules or behaviours, despite their collective ability to produce a complex pattern. In other words the concepts needed to explain the global properties are not present at agents' level.
For example, in the land use allocation problem, the global constraint (ratio of production between land-use categories) is not explicitly represented at the agent level but implicitly formulated through the groups' goal and the strength of groups that affect the conflict's outcome. In the localisation problem, we need to interpret the spatial positions of the agents in order to detect groups and then obtaining the target position. In the examples of allocation and localisation, the agents' behaviour is equivalent to stimulus-response rules and therefore is simple (at least simpler than the collective patterns that emerge).

In all problems solved with self-organisation by cooperation, and following the AMAS theory, the function of the system is not known by the agents which only know their own local and simple function led by cooperation rules. By not being able to have a global knowledge, agents do not know when a solution is found, and an external observer is needed to make this detection.

For the holonic approach since holons are recursive structures the behaviour of a holon may be the result of interaction of sub-holons. Indeed, a holon which is

unable to accomplish its goal will try to merge with a holon with complementary capabilities.

Emergent properties are also apparent in the self-organising communities application, which converges to a situation where groups of user agents with common interests can interact despite an arbitrary configuration of agents initially.

Following the ***anytime property***, experiments have shown that the main feature of the timetabling application is that modifications can be done without stopping the search for a solution (the schedule) while this latter is in progress, unexpected events are processed while actors are changing their constraints. The schedule is constantly changing as agents are searching for better bookings and partnerships and it becomes better as the solving makes progress. In the flood forecasting, a current solution (model) is given also at anytime, it becomes better as the learning process progresses. However, if disturbances appear, in both cases, the current solution may be questioned by agents for which unpredictable changes create non cooperative situations. Therefore the solution may be totally changed, may become totally "false" before converging again towards a good solution in a more or less great time.

In the AM there is a first step which is the construction of holarchies. After then the solution improves as the time allocated grows.

All applications based on reactive agents experimentally show their anytime ability.

In the self-organising communities application, user agents and middle agents rearrange user communities dynamically depending upon queries, and this can be stopped and resumed at any time, and thus support the anytime property. The evolving preferences application uses an evolutionary algorithm to stimulate preferences in populations of scout agents; the algorithm can be stopped and restarted but it will not continue changing for ever as an optimum will be reached.

Some solutions are ***reusable*** like the application independent framework of the holonic approach which can be applied to problem solving or simulation in different contexts. The general framework of the solution built for the timetabling problem can be also reused in other constraint satisfaction problems (such as supply chain management, for example) but rules enabling agents to detect and solve non cooperative situations are specifically suited to the problem.

Reaction to perturbations (sensitivity, robustness, adaptation or instability). Given a system that stabilizes on a state (solution), when a perturbation occurs the system can i) escape from this state and potentially reach another stable one (this is adaptation); ii) temporarily change its state and come back to the initial stable state (this is robustness), iii) change from state to state without stabilizing (instability), or iv) change of state even in case of small perturbations (sensitivity). To assess these criteria, we need to perturb the system at runtime. A perturbation can be viewed as an external event on the system in relation with the unpredictable and dynamic features of the application domain. In the timetabling application, perturbations come from the stakeholders that may change their constraints in a dynamic way, new actors can also vary at runtime. Failures of sensors can also be viewed as unexpected events for the flood forecasting application.

In land-use assignment, we did successful experiments where at runtime we changed the number of zones, the number of land use categories or by changing the goal of groups. In localization application, the system can successfully deal with situations where the number of targets and of sensors can vary at runtime. In all these situations, the systems were robust or adaptive according to the degree of the perturbation.

The timetabling application is a good example of a non-linear complex system in which simple and small variations (personal constraints, for example) may imply major changes in the problem solution. Partnerships an agent makes can disturb other agents, thus a little modification in the timetable can question the current solution which may vary greatly. The system is sensitive to perturbations, but is also able to adapt to these changes. Indeed, the very essence of systems built by applying the AMAS theory is adaptation which is obtained by enabling agents to locally decide to change their interactions with each others using cooperation as a local criterion.

In the AM and HTS the adaptation is done by the reorganisation of holarchies and is the basis of the approach.

In the self-organising communities application, the behaviour of middle agents is designed to respond to perturbations from users introduced via their user agents. In this example a perturbation is in the form of a novel query. This may provide information about a change of interests of the user and hence of the user agent, and as a result the user agent may move from one community around a middle agent to another. The use of rewards for successful responses to queries provides a mechanism to react to perturbations. In the evolving preferences application such response to perturbation is reflected in changes in selection pressure on the scout agent population, and hence changes in the outcome of the evolutionary algorithm.

This framework is still a tentative way to compare self-organised applications and it can be improved in two directions.

The first is the criteria list itself: it is still subject to discussion as the definition of criteria can be questioned and other criteria can be added to refine this comparison framework.

The second is related to the kind of answer for each criterion; it is currently subjective according to the interpretation of the definition. It would be of interest, when criteria are well established, to provide measurements of them. For example, measuring the decentralisation degree by the percentage of agents in the system that are directly involved by the decision of another one (this can be measured by the number of agent in a holon when the head agent takes a decision).

4 Conclusion

Multi-agent systems can be developed in many different ways. The autonomous nature of individual agents means that complex properties can emerge at the multi-agent system level. Self-organisation can be a useful way of controlling and regulating this complexity, especially when seeking to support an application. In a general way, applications that are too complex to give an a priori algorithm, that are plunged into open and real environments (the Internet, for instance) and for which a perfect design cannot be guaranteed can benefit from self-organisation.

This paper has presented several examples of applications based on multi-agent systems that use self-organising behaviour among the agents to facilitate application properties. The AMAS theory which uses self-organisation by cooperation has been successfully applied to various application domains: simulation, e-commerce, network management, collective robotics, mechanical design in avionics, flood forecasting, and biological modelling. Reactive multi-agent systems have proved useful in diverse application areas such as localisation in mobile robotics They have also provided the basis for cooperative information agents for information retrieval. Agents based on self-organisation through holons have been useful in meshing for cellular networks, and in traffic simulation, for example.

Self-organising multi-agent systems are at an early stage of development, with many different mechanisms of self-organisation to be explored. Despite this a number of examples have been outlined here, and already a diversity of application areas are being explored. We can anticipate self-organisation being of further relevance for applications of multi-agent systems in the future.

The framework of comparison we have provided has proven its usefulness to understand these approaches even if it has to be considered as a first and tentative approach that needs to be improved. It is obvious that the choice of one approach is application-dependent and these criteria may be of some help in this choice.

The authors would like to thank the participants of the two meetings of the AgentLinkIII “Self-organisation in MAS” TFG for their fruitful discussions.

References

[1] Bonabeau E., Dorigo M., and Théraulaz G. (1999) *Swarm Intelligence: From Natural to Artificial Systems.* Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, New York, NY, USA.

[2] Gleizes M.-P., Camp, V. and Glize P. (1999) A Theory of Emergent Computation Based on Cooperative Self-Organisation for Adaptive Artificial Systems, *4th European Congress of Systems Science,* Valencia.

[3] Corne D., Dorigo M., and Glover F. (1999) *New Ideas in Optimization*, Mac Graw Hill.

[4] De Causmaecker P., Ouelhadj D., and Vanden Berghe G. (2003) Agents in Timetabling Problems. Proc. of the *1st Multidisciplinary International Conference on Scheduling Theory and Applications*, pp. 67-71, UK.

[5] Decker K., Sycara K. and Williamson M. (1997) Middle-Agents for the Internet. Proc. of *International Joint Conference on Artificial Intelligence (IJCAI-97)*, Japan, pp. 172-175.

[6] DIET Agents platform: http://diet-agents.sourceforge.net/index.html

[7] Di Marzo Serugendo G., Foukia N., Hassas S., Karageorgos A., Kouadri Mostéfaoui S., Rana O. F., Ulieru M., Valckenaers P., and Van Aart C., (2004) Self-Organising Applications: Paradigms and Applications. *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering*, G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, F. Zambonelli (Eds), Lecture Notes in Artificial Intelligence 2977, Springer-Verlag, Berlin, pp. 1-19.

[8] Di Marzo Serugendo G., Gleizes M-P., and Karageorgos A., (2005) Self-Organisation and Emergence in MAS: An Overview, *Informatica*, this issue, Ljubljana, Slovenia.

[9] Dury A., Le Ber F., and Chevrier V. (1998) A Reactive Approach for Solving Constraint Satisfaction Problems: Assigning Land Use to Farming Territories, In Proc. of *Agents Theories, Architectures and Languages 98 (ATAL’98)*, Lecture Notes in Artificial Intelligence 1555 “Intelligent Agents V”, J.P. Muller, M.P. Singh et A. S. Rao (eds), Springer-Verlag, pp. 397-412.

[10] Eberhart R., Kennedy J., and Shi Y. (2001) *Swarm Intelligence*, Morgan Kaufmann Publishers.

[11] Ferber J., and Jacopin E. (1990) The Framework of Eco-problem Solving. In *Proceedings of the European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'90)*, pp. 181-193.

[12] Gechter F., Chevrier V., and Charpillet F. (2004) Localizing and Tracking Targets with a Reactive Multi-Agent System, In *Second European Workshop on Multi-Agent Systems (EUMAS'04)*, pp. 255-262.

[13] Georgé J.-P., Gleizes M.-P. Glize P., and Régis C. (2003) Real-time Simulation for Flood Forecast: an Adaptive Multi-Agent System STAFF, *Proceedings of the AISB'03 Symposium on Adaptive Agents and Multi-Agent Systems*, University of Wales, Aberystwyth, pp. 7-11.

[14] Hoile C., Wang F., Bonsma E., and Marrow P. (2002) Core Specification and Experiments in DIET: a Decentralised Ecosystem-Inspired Mobile Agent System, Proc. *1st Intl. Conf. Autonomous Agents and Multi-Agent Systems (AAMAS’02)*, pp. 623-630.

[15] Hsu K., Sorooshian S., Gupta H. Y., Gao X., and Imam B. (2002) Hydrologic Modeling and Analysis Using a Self-Organizing Linear Output Network, In Rizzoli A.E. and Jakeman A.J. (eds), Integrated Assessment and Decision Support, Proceedings of the *First Biennial Meeting of the International Environmental Modelling and Software Society* (iEMSs’0), Manno, Switzerland, pp. 172-177.

[16] Kaplansky E., Kendall G., Meisels A., and Hussin N. (2004) Distributed Examination Timetabling, In Proc. of the *5th International Conference of the Practice and Theory of Automated Timetabling (PATAT)*, Pittsburg, USA, pp. 511-516.

[17] Koestler A. (1990) *The Ghost in the Machine*, Reprint edition, Penguin, East Rutherford, NJ, USA.

[18] Mano J.-P., Bourjot C., Lopardo G., and Glize P. (2005) Bio-inspired Mechanisms for Artificial Self-organised Systems, *Informatica*, this issue, Ljubljana, Slovenia.

[19] Marrow P., Hoile C., Wang F., and Bonsma E. (2003) Evolving Preferences among Emergent Groups of Agents, In *Adaptive Agents and Multi-Agent Systems*, E. Alonso, D. Kudenko & D. Kazakov (eds.), Lecture Notes in Artificial Intelligence 2636, Springer-Verlag, Berlin, pp. 157-173.

[20] Michel F., Gouaich A., and Ferber J. (2003) Weak Interaction and Strong Interaction in Agent Based Simulations. *Multi-Agent Based Simulation III.* D. Hales et al. (Eds), Lecture Notes in Artificial Intelligence 2927, Springer-Verlag, Berlin, pp. 43-56.

[21] Müller T., and Rudova H. (2004) Minimal Perturbation Problem in Course Timetabling, In *Proc. of the 5th International Conference of the Practice and Theory of Automated Timetabling (PATAT'04)*, Pittsburg, USA, pp. 283-304.

[22] Paolucci, M., Niu Z., Sycara C., Domashev S., Owens S. and Van Velsen, M. (2000) Matchmaking to Support Intelligent Agents for Portfolio Management, In *Proc. of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, Calif., AAAI Press, pp. 1125-1126.

[23] Picard G., Bernon C., and Gleizes M.-P., (2005) ETTO: Emergent Timetabling by Cooperative Self-Organisation, *Third International Workshop on Engineering Self-Organising Applications (ESOA'05)*, Utrecht, The Netherlands, pp. 31-45.

[24] Reynolds C. W. (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model, In *Computer Graphics, SIGGRAPH Conference Proceedings*, pp. 25–34.

[25] Rodriguez S., Hilaire V., and Koukam (2005) A. Holonic Modelling of Environments for Situated Multi-Agent Systems, Submitted to *E4MAS'05*.

[26] Rodriguez S., Hilaire V., and Koukam A. (2003) Towards a Methodological Framework for Holonic Multi-agent Systems, In *Proceedings of the Fourth Workshop on Engineering Societies in the Agents World (ESAW'03), pp. 31-45*.

[27] Roumeliotis S.I., Sukhatme G.S., and Bekey G. (1999) Circumventing Dynamic Modeling: Evaluation of the Error-State Kalman Filter applied to Mobile Robot Localization, *IEEE International Conference on Robotics and Automation*, IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 1656-1663.

[28] Socha K., Sampels M., and Manfrin M. (2003) Ant Algorithms for the University Timetabling Problem with Regard to The-State-of-the-Art, In *Proceedings of the 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP'03)*, Essex, UK, pp. 334-345.

[29] Van Parunak H. (1997) Go to the Ant: Engineering Principles from Natural Multi-Agent Systems, *Annals of Operations Research 75*, pp. 69-101.

[30] Vergnes J.-C. (1995) Etude de modèles de prévision à la station de Nant - Exploitation de l'utilisation des réseaux neuronaux en prévision de crues, *Report ENSEEIHT/DIREN*.

[31] Wang F. (2002) Self-organising Communities Formed by Middle Agents, Proc. *1st Intl. Conf. Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pp. 1333-1339.

[32] Yokoo M., Durfee E., Ishida Y., and Kubawara K. (1998) The Distributed Constraint Satisfaction Problem: Formalization and Algorithms, *IEEE Transactions on Knowledge and Data Engineering*, 10, pp. 673-685.

Eye-Tracking Adaptable e-Learning and Content Authoring Support

Maja Pivec, Christian Trummer and Juergen Pripfl
Department of Information Design,
Alte Poststrasse 152, A- 8020 Graz, Austria, Europe
University of Applied Sciences FH JOANNEUM, Graz, Austria

Keywords: adaptive e-learning, real time eye tracking, learning research, cognition, authoring

Received: May 28, 2005

In this paper we describe our ongoing research project called AdeLE, a framework for adaptive e-learning utilising both eye tracking and content tracking technology. Possible areas of application are described, such as using the information about the position of the eye for providing additional context specific information to the learner. We report more in detail about current research challenges where we observe users' learning behaviour in real time by monitoring characteristics such as objects and areas of interest, time spent on objects, frequency of visits, and sequences in which content is consumed. This research is focused on analysing eye-movement patterns during learning and linking these patterns with cognitive processes. The concept of the appropriate authoring tool is outlined as one of the challenges for the future work.

Povzetek: V članku je opisan raziskovalni projekt AdeLE, katerega cilj je postaviti okvirno rešitev za adaptivno učno okolje, ki temelji tako na tehnologiji sledenja pogledu in spremljanja učnih vsebin.

1 Introduction

Past research and projects at FH JOANNEUM, Department of Information Design, have been among others in the area of adaptive hypermedia systems [SCALEX] and application of eye tracking for web usability evaluation in the Web Usability Center. Gained competences and experiences from the previous research in the area of adaptivity and personalisation, as well as user centered applications, created the motivation to merge adaptive hypermedia systems and eye-tracking technology with the goal of making learning more efficient and effective. The AdeLE prototype offers a solution framework in the direction of eye tracking based real-time adaptive hypermedia systems.

What is AdeLE? The presented research of the AdeLE (Adaptive e-Learning with Eye tracking) project is focused on a new generation of adaptable knowledge transfer in e-learning environments [AdeLE]. This new and innovative approach strives to capture dynamically user behaviour based on a real-time eye-tracking system (see also [Pivec et. al 2004], [Pivec et al. 2005a]). We apply eye-tracking for a more profound learning research and improvement of cognitive processes understanding to be able to support adaptive teaching and learning in a technology-based e-learning environment in the future.

In our research we are concentrating on how information from eye-movements could be used to support the learning process. In most e-learning environments information is mainly provided by means of written text. Thus, reading this information is essential for learning. By means of real-time tracking of the user behaviour, unseen sections of content units provided to the learner are identified by the system and the system might intervene in an appropriate way. Of course, this method can not reflect information about pre-knowledge. Based on the information about content sections skipped by the learner, adaptable and context specific assessment tests can be compiled to check the learner's knowledge about these particular concepts.

The aim of this paper is to present the research endeavours of the AdeLE project. The paper is based on two conference publications: (i) Look Into My Eyes And I Will Tell You How To Learn, by [Pripfl et al. 2005], and (ii) Adaptable Features and needed Content Authoring Support, by [Pivec et al. 2005b]. The paper gives inside in the eye-tracking technology and outlines application scenarios of adaptable e-learning by means of eye tracking, as well as ongoing research and further developments in the field of adaptive authoring.

2 Adaptable Learning Based on Eye-Tracking

Real-time Eye-tracking (User Tracking)

Defining a reliable set of parameters is one of the emerging research issues in the AdeLE project. Eye movements, scanning patterns and pupil diameter are indicators of thought and mental processing involved during visual information extraction [Rayner 1998], [Kahneman and Beatty 1966]. Thus, real-time information of the precise position of gaze and of pupil diameter could be used for supporting and guiding learners through their learning journey.

How does eye-tracking work? Very roughly, eye movements can be divided into two components: fixations, i.e. periods of time with relatively stable eye

movements, where visual information is processed, and saccades, which are defined as rapid eye movements that bring a new part of the visual scene into focus. However, more important indicators can be gained by analysing both components together with other derived parameters. Gaze duration (i.e. time spent on an object) and fixations are not indicative of attention per se, because one can also pay attention to objects, which do not lie in the centre of the focused region. Nevertheless, by considering other indicators, such as saccadic velocity, blink velocity and rate as well as eyelid's degree of openness, a better and more meaningful approximation can be gained. Saccadic velocity, for example, is said to decrease with increasing tiredness and to increase with increasing task difficulty [Fritz et al. 1992]. Further, blink rate, decreasing blink velocity and decreasing degree of openness may be indicators for increasing tiredness [Galley 2001]. Thus, if tiredness is identified, it should be possible through adaptive e-learning mechanisms to suggest optimised strategies such as the best time to take a break.

At the present there basically exist two types of eye-tracking systems on the market: remote systems and head-mounted systems. Remote systems are characterised by the fact that one or more cameras record the eye of the participant and trace the gaze in a scene through imaging algorithms. The cameras are positioned in front of the participant. One of the advantages of these systems is given by the fact that the camera can be integrated into the monitor, and therefore remains basically invisible (i.e. a relatively non-intrusive monitoring is possible). Head-mounted systems are characterised by a special device that the participant has to wear on the head like a helmet. More characteristics of both systems along with advantages and disadvantages related to the requirements of the AdeLE project are outlined in detail in [Pivec et al. 2004].

The AdeLE project team decided to utilize the remote eye-tracking system "Tobii 1750" with the infrared diode lamps and camera are integrated into a 17" TFT monitor. This system is easy to operate and all tracking processes run automatically. Thus, it can be used for all forms of eye-tracking studies with stimuli that can be presented on a monitor, such as Web sites, slide shows, videos and text documents. With real-time eye tracking user gaze data are gathered. The evaluation of the users' eye gaze data gives information about what the user is doing, e.g. learning or reading, looking at the pictures and illustrations or eventually struggling with the system's navigation.

Eye Movement Parameters and Their Influence on Adaptation

In our research we are concentrating on how information from eye-movements could be used to support the learning process. The study focuses on finding eye-movement patterns which distinguish between skimming through, reading of and learning facts from written text. Based on various research findings different models of eye-movement control during reading have been developed. Their application is in general not feasible for the contents of a real life e-learning environment. Guided by the research literature but also considering the practical usage of our system, we designed an eye-tracking study in which subjects have to deal with texts of three difficulty levels under four different conditions (1. skimming through text, 2. single reading of text, 3. learning the content of the text and 4. searching for a specific information within the text). The study was carried out with 40 test persons. The gathered data of this study are applied for the definition of eye-movement parameters which can reliably distinguish between the four conditions listed above. In a further step these parameters will be taken to identify user behaviour within an e-learning environment in real-time (e.g. if the user really learns a text or just reads it).

By merging eye-tracking technology with proper content presentation the goal of the research is to identify, evaluate and develop methods of adaptive instruction for personalised e-learning. From *real-time eye tracking* data six different user behaviour parameters are reported to the AdeLE prototype: (i) learning, (ii) reading, (iii) skimming through text, (iv) searching in text, (v) observing a picture or reading a text and (vi) looking on the navigational elements. User parameters can be in the range between 0 and 1 expressing the probability of a certain user behaviour. Reported user parameters trigger further various reactions of the system in terms of adaptation of the content and additional information offered to the user. Application of SCORM run time environment enables dynamic content sequencing, which calculates the next steps and provides personalised content for the individual user (see Garcia-Barrios et al. 2004 for detailed description of the AdeLE framework architecture). For example in an e-learning course about the handling of dangerous chemicals our system will react as follows: The user is faced with a text he should learn and know for understanding the following chapters – the system detects that the user is just skimming through this text – the system consults the user if he just wants to get quick information about the lecture content or structure, or if he already knows the details of the content - – according the answers the system interacts in one of the following ways: it suggests to show just the abstracts of each chapter or to go on without any changes; it provides some content specific questions or suggests to repeat the lecture.

Real time user observation can be applied also for the enhancement of a user profile that has influence on adequately personalised course content presentation for each individual learner. For example, the prototype can distinguish between (a) learning style of the user (e.g. text based, acoustic) and (b) cognitive style i.e. holist or analyst, meaning that to the analyst the entire content is presented consequently in contrast to the presentation of the content to the holist, where an overview of chapters and subchapters along with summaries is optionally offered.

Possible Application Scenarios

Currently, the research efforts of the AdeLE team concentrate on three issues, which are discussed in the following sections. The first issue is to develop methods to extract individual learning strategies from the learner's gaze behaviour and adapt against the identified learning style. Comprehensive reviews of cognitive psychology research indicate that people exhibit significant individual differences in how they learn [Schmeck 1988], [Glaser 1984], [Honey 1986]. A simple example being individuals who have a strong visual memory but weaker verbal processing will find text based material harder to process than individuals who have stronger verbal skills. In the traditional classroom environment a teacher has the chance to adapt or explain material to suit individual's needs. In e-learning environments where a teacher is frequently not present, pedagogical material is nowadays more uniformly presented. In an e-learning environment information about the learner's gaze behaviour would be a great opportunity to optimise material to an individual's needs. For example, if somebody prefers text and ignores pictures the amount of pictures presented could be reduced, and vice versa.

The second issue addresses the usage of information about the specific content accessed by the user (specific words, paragraphs, areas of pictures, tables, and the like) to provide additional context specific information. For example, an animated picture could accompany textual information, whereas the integration of the picture proceeds in relation to the words or paragraphs accessed by the user, as illustrated in the following example. In an e-learning course concerned with Alexander the Great's Conquest of Persia, a map of Alexander's advance in the region is shown parallel to the text. The map content is updated in correspondence to the paragraph currently read by the learner. When the second paragraph about Granikos is being read, the map shows in animated form the journey of Alexander from Macedonia to Granikos. When the reader has advanced to the fourth and fifth paragraph about Alexander's journey to Gaugamela, the map is automatically updated with a corresponding illustration containing the passage from Issos to Gaugamela. Further research questions related to this topic are "Does such an eye-triggered animation really help a student to learn?" or "How should such an animation be integrated into the text to support cognitive processes?" among others.

The third research challenge is based on developing and testing appropriate intervention strategies when the learner is found to have problems. The e-learning environment might react in an appropriate way when a learner is not focused on a relevant part of the computer screen, or is focused completely outside the task area for a certain period of time, or the eye gaze is sufficiently quick for a given period of time. Just to give one example, in case of knowledge acquisition problems for a particular content section more detailed content or background information can be provided to the learner.

3 Adaptable Content Authoring Tool Concept

To provide content structured to support adaptable and personalised content presentation, adequate authoring tools for content authors have to be provided. The basic requirements of such a tool are as follows: to make the authoring a structured process; to keep this process as simple as possible and not to require from the authors to think explicitly about adaptivity and various content presentations.

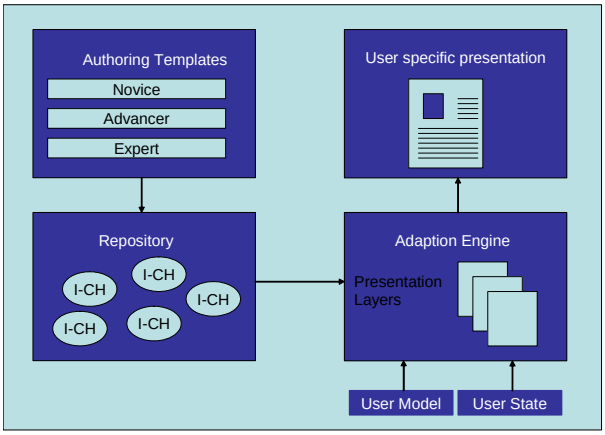


Figure 1: Authoring of adaptable contents

Within the further development of the AdeLE framework a template based authoring tool that supports stepwise authoring and separation of presentation layers and content chunks will be provided. The structure of the authoring tool is depicted on Figure 1. Based on *Templates* the author is guided to provide different chunks of content appropriate for different knowledge levels of the learner (e.g. novice, advancer, expert: note that the levels can be broken down in various manners). Templates enable stepwise and structured authoring. The information and knowledge chunks are then semi-automatically tagged with meta-data and saved in the *Content Repository*. The authoring tool has to support the re-use of the content i.e. the application of the content chunks from various repositories. The function of the *Adoption Engine* is to dynamically create personalised content from the available information chunks and different presentation layers. The presentation layers can be of different kinds e.g. role specific (learner, teacher, author, etc.), goal or context specific (introduction to the topic, basic definitions, application scenarios, etc.), respective to the output device. The creation of a user specific presentation is also influenced by the user model and the user state.

4 Conclusions

The AdeLE framework with the assets of extreme adaptation and personalisation to each individual user on various levels (e.g. macro level in terms of general adaptations of the course and micro level, where each page can be different, considering also the pace and momentary user performance) is the first step to

innovative human centered technology enhanced learning and knowledge management solutions.

The ultimate goal of our approach is to interpret various users' parameters in form of input data for an adaptable e-learning system that assists users to improve their learning behaviour thus achieving better learning results. In the context of user behaviour interpretation, it is very important not to rely exclusively on eye tracking data, but to supplement it also with constant user feedback. It is possible to suggest optimised strategies such as the best time to take a break, the best time for repeating specific learning content considering the forgetting curve [Davis and Palladino 2002] or suggesting better sequencing of the learning objects. However, the user will always retain the final decision on whether to accept or reject the system's suggestions.

The AdeLE framework can be integrated into different applications e.g. content management systems, e-learning environments, knowledge management systems etc., thus providing new highly user sensitive personalised adaptive solutions. The proposed authoring tool will support appropriate adaptive content authoring.

Evidently, the price of an advanced eye-tracking system plays a decisive role in the application possibilities of the AdeLE solution approach. Nevertheless, existing systems show that the eye-tracking device can be integrated into a standard monitor. Due to the continuing trend of rapid technical progress, we expect that in the next few years it will be possible to build a low-cost but high-quality eye-tracking system based on standard hardware components, which will be suitable for real-time analysis of eye-tracking information as described in this paper. This will make it possible to provide applications related to attentive workplaces for broad populations.

Acknowledgements

The AdeLE project is partially funded by the Austrian ministries BMVIT and BMBWK, through the FHplus impulse programme. The support of the following institutions and individuals is gratefully acknowledged: Department of Information Design, Graz University of Applied Sciences (FH JOANNEUM); Institute for Information Systems and Computer Media (IICM), Faculty of Computer Science at Graz University of Technology; especially Karl Stocker and Hermann Maurer.

References

[AdeLE] AdeLE "Adaptive e-Learning with Eye Tracking" project page http://adele.fh-joanneum.at, accessed on 26.01.2005

[Davis and Palladino 2002] Davis, S. F. & Palladino, J. J.: *Psychology* (3rd ed.), Prentice-Hall: Upper Saddle River, NJ, 2002.

[Fritz et al. 1992] Fritz, A., Galley, N., Groetzner, Ch.: Zum Zusammenhang von Leistung, Aktivierung und Motivation bei Kindern mit unterschiedlichen Hirnfunktionsstörungen, *Zeitschrift für Neuropsychologie*, 1(1), pp. 79-92, 1992.

[Galley 2001] Galley, N.: "Physiologische Grundlagen, Meßmethoden und Indikatorfunktion der okulomotorischen Aktivität"; In Frank Rösler (ed.): Enzyklopädie der Psychologie, 4, Grundlagen und Methoden der Psychophysiologie, 2001, 237-315.

[Glaser 1984] Glaser, R.: Education and Thinking: The Role of Knowledge, *American Psychologist*, 39, pp. 93-104, 1984.

[Honey 1986] Honey, P.: *The Manual of Learning Styles*, Peter Honey: Maidenhead, Berks, 1986.

[Kahneman and Beatty 1966] Kahneman, D. & Beatty, J.: Pupil diameter and load on memory, *Science*, 154, pp. 1583-1585, 1966.

[Pivec et al. 2004] Pivec M., Preis M.A., Garcia V., Guetl C., Mueller H., Trummer C., Moedritscher F.: Adaptive Knowledge Transfer in E-Learning Settings on the Basis of Eye Tracking and Dynamic Background Library, in Proceedings of EDEN 2004 Annual Conference, Budapest, Hungary.

[Pivec et al 2005a] Pivec, M., Pripfl, J., Gütl, C., Garcia-Barrios, V.M., Mödritscher, F. & Trummer, C.: AdeLE First Prototype: Experiences Made. In Proceedings of I-KNOW'05, 2005.

[Pivec et al.2005b]] Pivec, M., Pripfl, J., & Trummer, C.: Adaptable Features and needed Content Authoring Support. In Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education (E-Learn 2005), October 24-28, Vancouver, Canada.

[Pripfl et al. 2005] Pripfl, J., Pivec, M., Trummer, C. & Umgeher, M.: Look into my eyes and I will tell you how to learn. Proceedings of the European Distance and E-Learning Network 2005 Annual Conference, June 21-23, Helsinki, Finland.

[Rayner 1998] Rayner, K.: Eye movements in reading and information processing: 20 years of research, *Psychol Bull*, 124, pp. 372-422, 1998.

[SCALEX] SCALEX home page http://www.scalex.info/, accessed on 20. 08. 2005

[Schmeck 1988] Schmeck, R.R.: *Learning Strategies and Learning Styles*, New York, Plenum Press, 1988.

Integration of Access Control in Information Systems: From Role Engineering to Implementation

Thion Romuald and Coulondre Stéphane
LIRIS / INSA University of Lyon
20 Av. Albert Einstein
69621 Villeurbanne Cedex, France
romuald.thion@insa-lyon.fr and stephane.coulondre@insa-lyon.fr

Keywords: role-based access control, object-oriented models, role engineering, security

Received: November 11, 2005

Pervasive computing and proliferation of smart gadgets lead organizations to open their information systems, especially by extensive use of mobile technology: information systems must be available any-time, any-where, on any media. This cannot be done reasonably without thorough access control policies. Such access control must be able to deal with user profile, time and even with more complex contexts including geographical position. This paper shows that it is possible to take into account confidentiality constraints straight into the logical data model in a homogeneous way, for various aspects generally treated independently (user profile, time, geographical position, etc.). We propose a language called RAPOOL which allows the expression of authorizations at the class level. We first present the syntactical aspects, then the semantics of the language, based on the object-oriented paradigm.

Povzetek: Članek opisuje mobilne informacijske sisteme.

1 Introduction

Companies and general public interest for new technologies keeps growing, either for mobile use (laptops, Wi-fi, pocket-PC, GPS, UMTS, Java technology in GSM, etc..) or for legacy use. Information Systems (IS) now become open and online. Therefore, security is in great demand, particularly for IS containing confidential data. New mobile and pervasive technologies introduce the concept of context. The need to include it in access control mechanisms arises [16, 13]. Thus, from now on, access control tends towards integration of user profile, time, state of the computing environment and even geographical position [8].

1.1 Motivations

Security is considered as a non-functional requirement in software engineering. Contrary to other non-functional requirements, such as efficiency, modularity or usability, confidentiality has been unconsidered for long. Thus, access control management is often postponed until the end of the design cycle and is implemented at the very end of the design process. The software is therefore developed without taking confidentiality constraints into account. This usual approach often leads to serious design challenges (e.g. integration of roles) and problems (e.g. software vulnerabilities, information leakage) [17].

In this paper, we show how to take into account general context data (user roles, spatio-temporal environment, etc.) in a homogeneous way, straight in the object data model (and more generally in Information Systems, Objects, Web Services, etc) to provide an object-oriented language which allows the expression of authorizations at the class level. We do think that security must be present throughout the whole development cycle. Our proposal describes a logical data model in which contextual role-based access control is integrated. We thus provide a support to upstream design methods [9, 14] which can rely on it.

1.2 Our approach

We do think that security concerns must be considered all the development cycle long. As user interface is incorporated in software architecture (e.g. the model-view-controller architecture separates an application's data model, user interface and its control logic), we argue that access control, and in a broader sense security, must be considered in the software development cycle, and not neglected until the very end of this process.

To bridge this gap between the security will and its implementation, our approach is to provide a consistent logical data model including role-based access control policies. Since Role-Based Access Control (RBAC) is currently one of the most seducing security models and since extending Object-Oriented (OO) systems with roles has been amply studied in the literature, we chose to integrate role-based authorization policies at the class level. With such embedded authorizations in an OO language, developers can now integrate their security policies in their code in a declarative manner. As inheritance is used by programmers without

worrying about how polymorphism or dynamic linking are implemented, authorization policies can be used without worrying about the mechanisms involved in the authorization decision.

The rest of the paper is organized as follows. Section 2 presents the original Role-Based Access Control on which our proposal relies for describing privileges organization within an IS. Section 3 details syntactical aspects of the RAPOOL (for Role-based Authorizations Policies Object-Oriented Language) language we propose. Section 4 details functional aspects with an illustrative example in the medical area. This section also describes how to implement RAPOOL. Section 5 surveys attempts in integrating the role concept in object data models and compare our approach to related work on security integration within object-oriented models. Section 6 finally concludes the paper and discusses some perspectives on security integration within the software design process.

2 Modelling authorizations with roles

Role modelling has been introduced into many computer sciences areas: databases, programming languages, ontologies or agent oriented modelling. For security purposes, roles have been introduced to make access control policies administration easier: this is the main idea of the *Role-Based Access Control* (RBAC) model.

2.1 An Access Control model

Access control denotes the fact of determining whether a *user* (not necessarily an human user: process, computer, etc.) is able to perform an *operation* (read, write, execute, delete, search, etc.) on an *object* (more generally: a tuple in a database, a table, an object, a file, etc.). An operation right on an object is called *permission*. An access control model define how to organize the permissions of users.

The RBAC Model [21] was defined in the 90's and has been extended in many ways (temporal, geographical extensions, etc. [8, 13]). It was introduced in order to tackle the weaknesses of DAC (Discretionary Access Control) and MAC (Mandatory Access Control) models: the former is difficult to implement with a large number of users, and the latter is too rigid for modern applications. We focused on RBAC rather than other recent access control models because it is currently the most seducing access control paradigm, as shown by its use in major databases management systems such as Oracle Enterprise Server v.8 or Sybase Adaptive Server v.11.5. Even for legacy systems which are not role- based, the use of RBAC may simplify management [18].

The basic RBAC philosophy is based on the observation that most of the access permissions are determined by a person authority or function, inside an organization. This defines the central concept of role. The introduction of role concept in access control policies as an intermediate layer between users and permissions, really facilitates and simplifies the system administration task. The RBAC definition of a role is "a job function within the organization with some semantics regarding the authority and responsibility conferred on the member of the role"([21]).

The RBAC model family is based on the identification of a certain number of roles [20], each of them representing a set of actions and responsibilities within the system (roles can be seen as a collection of permissions). Thus in the RBAC model (figure 1):

- no permission is granted directly to the user, permissions are only granted to roles,
- the users endorse the roles which are given by the administrator (it is only possible to specify positive authorizations, no prohibitions),
- roles are defined and organized in a hierarchy: a child role has the permissions granted to his/her parents.

In OO systems, the concept of permission is related to objects methods. A permission is an access privilege on a ressource. In OO systems, ressources are objects (or attributes of objects) and access privileges are objects methods (e.g. `getAttribute()` is a read-access, `setAttribute()` is a write-access). Thus, granting an access privilege to an object consists in authorizing the object method call. The rules defining permission assignments to roles are *access control policies*.

2.2 Access control policies

Access control policies define the users rights on objects, in order to enforce the security of an organization. In the RBAC model, policies define which permissions are granted to roles (*permission-role assignment* in figure 1). Thus users are granted permissions through role-assignment (*user-role assignment* in figure 1).

An example of role-oriented access control policies in health sector would be:

- a nurse can only read the patient prescriptions. But she can write the last care date and time, provided it takes place during her working time,
- a doctor can only prescribe if he/she is geographically located in the hospital. He has access to the whole medical record, but he/she cannot write the last care date and time,
- a head nurse has read access to prescriptions and cares history without conditions of time.

Permissions associated to roles allow the expression of access authorizations in a generic way. Therefore we do not specify that *Dr. Johnson* has access to *Mr. Rabot* records. Instead we only specify that doctors have write access to

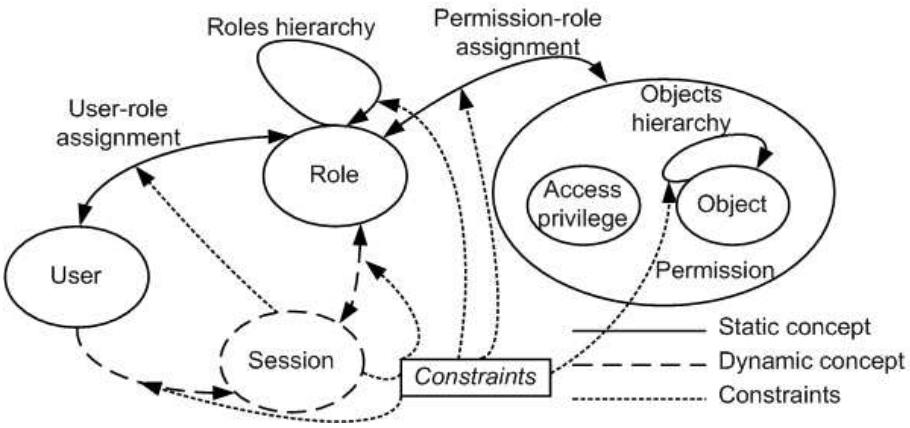


Figure 1: The RBAC Model

patient records. According to RBAC principle "permissions are only granted to roles", our proposal do not include policies related to individuals. Thus is is not possible to specify that only *Dr. Johnson* has access to *Mr. Rabot* record. The RBAC roles, their hierarchical organization and the associated permissions make up the organization confidentiality policy.

The language we chose to express access control policies has been heavily influenced by [8] which formalize authorization policies, including temporal aspects, in first-order logic (FOL). Thus we use a tractable fragment of FOL (no function symbol, no negation, only conjunctive and disjunctive connectors) suitable to express role-based authorization policies at the class level.

2.3 A suitable subset of RBAC

Our approach intends to include role-based authorizations into class models. Thus, authorizations policies are common to every objects instantiated from a class. As classes are most-of-the-time static in OO systems, we are not able to express dynamic aspects of RBAC. The session concept for example, cannot be included into the class model: each session is related to exactly one user, and it represents the roles its user is actually endorsing.

Moreover, our proposal is an OO language designed for secured software. Its goal is not to integrate the whole RBAC model into class models, but only a subset describing authorizations policies related to the application which is to be developed. Thus user-role assignments, sessions or delegations are outside the scope of this work: we only model the authorization policies related to the application. For example, let us suppose that a hospital is developing an intranet web-portal. User-role assignments are stored in a dedicated directory (which is used by other applications), not in web-portal itself. Only the policies describing "which role can access a given method of a given class of the web-portal" are included straight into the code.

Thus, the concepts of user, user-role assignment, sessions and context retrieval are not included in our proposal and will be referred as *user profile* (section 4).

3 The RAPOOL Language

In order to tackle the problems of RBAC integration within object data models, we propose a generic language RAPOOL allowing the expression of RBAC authorizations and integrating an access control mechanism. The declarative part of the language is composed of:

- *the body*, which relies on C++ syntax (on a purely illustrative basis, as any class-based language could have been used: Java, Python, etc.) while adding access authorizations formulae to methods,
- *the header*, which defines the roles which are to be used within the definition of access authorizations.

3.1 The Header

The header is used to specify (BNF grammar is given in annex):

- various categories of roles to be taken into account. In the example we included the categories of [3] which are adapted to organizations: *functional*, *seniority* and *context*. These categories, freely chosen by the developer, form groups of roles. These groups represent transverse role aspects, which are combined to form complex roles. It would be possible to add some other groups such as *ward* (ex: cardiology, radiology, etc. which remains static), or *sensitivity* (ex: white, grey, black information according to the sensitivity of data) which can be used to simulate a MAC access control,
- hierarchical relations between roles [15]. For example *head* $<<$ *assistant* means that the head has (at least) all the privileges of the assistant. Thus, the conjunction of seniority roles with the functional role *doctor*

makes it possible to specify complex roles, for example *head doctor*, who would have more privileges than a doctor, but fewer privileges than the *manager doctor*,

– the various contexts in which the access authorizations are defined. These contexts can be geographical (by using the predicate *position*) or temporal (with the predicate *hour*). We suppose that the position of the user is obtained by reliable mechanisms which are not in the scope of this paper. We suppose we can get an absolute reference as a couple of (X, Y) co-ordinates, indicating the user position from where he/she invokes the service. In practice, space modelling by mean of linear constraints is sufficient for many cases [7]. Within the header we can for example restrict access only if the user is located within the hospital or the building.

All simple roles defined in the header are combinable via conjunctions and disjunctions, in order to create complex roles, modelling access control constraints based on the transverse aspects of the profile, time and space at the same time.

```
Functional Roles {
Roles : nurse, doctor, day_nurse, night_nurse;
Hierarchy : day_nurse << nurse, night_nurse << nurse;
}

Seniority Roles {
Roles : manager, head, assistant;
Hierarchy : manager << head << assistant;
}

Contextual Roles {
Hospital_enclosure = (position(X,Y)
   and X>10 and X<50 and Y<10 and Y>30);
First_shift = (hour(H) and H>=4 and H<12);
Second_shift = (hour (H) and H>=12 and H<20);
Third_shift = ((hour (H) and H>=20)
   or (hour(H) and H<4));
}
```

3.2 The Body

In RAPOOL, the body part allows the expression of access authorizations at the method level. This is made possible using the *auth* keyword, followed by an appropriate logical formula. The authorization logical formulae are used to condition access to each method, according to the roles defined in the header. These access authorizations model access control rules defined in the confidentiality policy (section 2.2).

```
Class CElectronicPatientRecord {
Public:
 contact getPatientContact()
    auth (doctor or nurse);
 string getLastPrescription()
    auth (doctor or nurse);
 string getPrescriptionHistory()
    auth (doctor or (nurse and head));
 string getCareHistory()
    auth (doctor or (nurse and head));
 void setPrescription(string prescription)
    auth (doctor and Hospital_enclosure);
 void setLastCare(hour h, string care)
    auth ((day_nurse and first_shift)
    or (day_nurse and second_shift)
    or (night_nurse and third_shift));
 /* This authorization prevents a day nurse from
 filling the LastCare field of the e-Patient
 record during night, and a night nurse during the day */
}
```

4 Functional aspects

As the access control we propose is defined at the class level, the following statements hold:

– for confidentiality-critical applications, access control authorizations should be taken into account from the very beginning of an information system design cycle [17]. We do think that it does not have to be postponed until the end of the cycle,

– roles must be defined as soon as the requirement engineering stage [20, 11],

– roles and authorizations can only be static [3], as the class structure is modified, therefore recompiling is necessary. We consider that this is not necessarily a major problem, as the set of information defined in the header and authorizations are very static (ex: hierarchical levels, internal organization, administrative responsibilities, etc.). However, no recompiling is necessary for dynamic user role assignment or revocation. Moreover, privilege delegation is possible between users. In the case of developing a wrapper (for accessing legacy application through web services for instance), recompilation does only involve the wrapper, not the wrapped applications.

4.1 The authorization decision

The principle of access control decision is as follows: when a method call is detected, the RAPOOL engine checks if the dynamic user profile fulfills the method authorization policy. As described in section 2.3, our proposal does not include the management of user profiles: we suppose that a system storing role assignments, running sessions and providing contextual information (e.g. time) exists. Once this information is retrieved (a cache mechanism can be used to improve retrieval efficiency), the RAPOOL engine can check if the requested access is granted. An architecture for such a context repository is described in [16].

The basic idea of the access control decision is based on logical implication. The user profile and the authorization policy of the requested method needs to be translated into first-order logic formulae:

1. each role is replaced by itself and the conjunction of all its parents roles. If two roles are set to be mutually inherited, they are considered as a same role,

2. each category c act as a predicate symbol. Each role r defined within c is replaced by atom $c(r)$,

3. if a role is equivalent to a formula, then it is replaced by this formula,

Once theses transformations are applied to both user profile and requested authorization formula, we need to add contextual information to the user profile. This information is obtained by mean of software/hardware tools such as LDAP, GPS, time clock, etc. and are also translated into atoms. E.g. $hour(18)$ or $position(10, 23)$. Then if the user profile (plus context) implies the authorization formula of the requested method, the method is invoked, otherwise a catchable exception is raised.

4.2 Example of authorization decision

Let us suppose that a user, John, wants to access the *setLastCare()* method from his mobile device. John, who has previously identified himself on the information system, has the following profile: $functional(nurse) \wedge functional(night\_nurse) \wedge position(150, 45) \wedge hour(23)$ The functional part can be extracted from a LDAP directory for example, and the spatio-temporal part can be added by a time and position server.

The authorization policy associated with the *setLastCare()* method is specified within the RAPOOL body, as $((day\_nurse \wedge first\_shift) \vee (day\_nurse \wedge second\_shift) \vee (night\_nurse \wedge third\_shift))$ The RAPOOL engine replaces these role names by logical predicates, as defined in the header:

- day_nurse is replaced by $functional(nurse) \wedge functional(day\_nurse)$. Indeed, day_nurse has at least all the privileges of $nurse$. The same hold for $night\_nurse$,
- $first\_shift$ is replaced by $hour(H) \wedge H \geq 4 \wedge H < 12$. The same holds for $second\_shift$ and $third\_shift$.

The resulting formula (under disjunctive form) is: $(functional(nurse) \wedge functional(night\_nurse) \wedge hour(H) \wedge H < 4) \vee (functional(nurse) \wedge functional(night\_nurse) \wedge hour(H) \wedge H \geq 20) \vee (functional(nurse) \wedge functional(day\_nurse) \wedge hour(H) \wedge H \geq 4 \wedge H < 12) \vee (functional(nurse) \wedge functional(day\_nurse) \wedge hour(H) \wedge H \geq 12 \wedge H < 20)$

The RAPOOL engine checks if the dynamic user profile logical formula implies this formula. As the user profile is $functional(nurse) \wedge functional(night\_nurse) \wedge position(150, 45) \wedge hour(23)$. Implication holds, therefore access is granted.

4.3 Implementation in an object-oriented framework

The first approach to implement RAPOOL is to add a layer over an existing object-oriented language. Such a layer has to retrieve user profile and contextual information from role-assignment database. This layer needs to implement the profile and authorization policy transformation into logical formulae. This framework allows software designers to integrate access control in a declarative manner, without worrying about the mechanisms involved in authorization decision.

A proof-of-concept pre-processor RAPOOL to C++ has been implemented. For a developer, the RAPOOL to C++ pre-processor is a black-box transforming his code into C++. The basic steps of the pre-processor are:

- the input is a RAPOOL source file, as written by developers according to RAPOOL grammar,
- the pre-processor includes the C++ framework files to the source code,
- the pre-processor parse the header of RAPOOL (according to its grammar) and suppress it from the source file,
- the pre-processor analyse the body of RAPOOL and transform authorization policies into logical formulae (section 4.1) according to the previously parsed header,
- the pre-processor add a call to the authorization decision mechanism (included from C++ framework files) at the beginning of each method,
- the output is a C++ source file, obtained from RAPOOL source file once header and authorization policies are translated into calls to the framework.

4.4 Implementation in a role-based object-oriented model

In many class-based object-oriented systems the association between an instance and a class is exclusive and permanent. Therefore these systems have serious difficulties in representing dynamic evolution of objects over time. The problem is the most severe for OO databases in which objects are stored over long periods during which the entities evolve. Object-role oriented models intend to fill this shortcomings of object-oriented models by adding an orthogonal concept to classes: roles.

The authors of [23] describe an RBAC framework organized into 7 layers, as the OSI (Open Systems Interconnect) network stack is (from *physical* layer #1 to abstract *application* layer #7). The least abstract layer is the *object* layer, used by the directly higher one: *objects handles*. This second layer is used to keep the association between objects and roles. An object-role oriented model integrate directly such a layer: handles are no more needed, associations between roles and objects are handled in declarative manner in the object-role oriented paradigm.

The implementation of RAPOOL in a role-based object-oriented model is possible if:

- the model integrates a role hierarchy,
- the role hierarchy is independent of the class hierarchy,
- the model respects the Object Data Management Group (ODMG) standard (e.g. encapsulation, inheritance, polymorphism, etc.) to be compliant with an existing object-oriented language.

Section 5.2 survey previous works on integration of role mechanisms into object-oriented models. According to previous surveys [2, 10], Samovar is the most suitable model for hosting the RAPOOL language. This model respects the ODMG standard and integrate roles into object-oriented model. In this model, class can be seen as *abstract role containers*: no method is directly associated to a class. Attributes are only associated to roles or combinations of roles (including conjunction and disjunction). These combinations are expressed in first-order logic formulae (the same fragment of FOL used to express authorization policies in RAPOOL).

Thus, to implement RAPOOL in a role-based object-oriented model, we must carry a prior transformation step on access control policies. In RAPOOL, each authorization is associated to an object method, roughly said as "policies are organized by permissions". In order to implement our language, we must infer on these policies to organize them by roles, rather than by permissions. Note that this process can be performed automatically without human intervention.

For example, assuming we are working with the example from section 3.2. Policies of the `CElectronicPatientRecord` class are organized by permissions:

- permission `getPatientContact()` is granted to roles `(doctor or nurse)`,
- permission `getLastPrescription()` is granted to roles `(doctor or nurse)`,
- permission `getPrescriptionHistory()` is granted to roles `(doctor or (nurse and head))`.

In order to be implemented in a role-based object-oriented model, policies must be organized by roles. Thus, the above example will be organized as follows:

- role `(doctor)` is granted access to `getPatientContact()`, `getLastPrescription()` and `getPrescriptionHistory()`,
- role `(nurse)` is granted access to `getPatientContact()` and `getLastPrescription()`
- role `(nurse and head)` is granted access to `getPrescriptionHistory()`.

Everybody who is assigned to several roles is granted all permissions assigned to each role, as permitted by role hierarchy. Thus, `(nurse and head)` is also granted access which `(nurse)` is granted.

5 Related work

Very few work focused on integrating of access control models within logical data models. This section survey related work on introduction of security in object-oriented models and on the role-object oriented paradigm.

5.1 Integration of access control in object-oriented systems

Many papers have described how to implement security mechanisms involving roles and contexts (e.g. [16, 3]) in a role-oriented system. Our goal is not to describe how role-based access control can be implemented *with* classes, but is to describe how (and which subset of) role-based access control mechanisms can be implemented *in* classes.

Integration of access control into OO systems has already been studied. The authors of [24] describe how to implement Mandatory Access Control (MAC) in OO database systems. Roughly stated, MAC is a military-oriented model, in which users and resources are associated to labels. Access is granted if and only if the user label is as least as high as the requested resource label. Commonly used labels are *unclassified*, *confidential*, *secret* and *top-secret*. However, MAC has been shown to be too rigid for current applications, particularly when multiple users with different profiles are working on the system.

A more recent approach in [5] integrate MAC to UML diagrams. Their framework bridges the gap between software engineers and an organization security. A very interesting contribution which is not limited to class diagram and intends to integrate MAC in use cases and sequence diagrams. This paper describes a logic data model, but we are working on a conceptual model integrating role-based authorizations (section 6).

5.2 Extending object-oriented systems with roles

The object paradigm is a very expressive framework, largely used. According to [22], implementing object roles is a difficult task. Indeed, the multiplicity of roles and their lifecycle (creation, deletion) is incompatible with the hard constraints of class-based models: object identity, strong typing, etc.

This problem could be partly solved with multiple inheritance (figure 2a) in an object programming language. But each combination of role must lead to create a new class, which leads to an explosion of the number of necessary classes. Moreover, their existence is only motivated by technical reasons and not by a modelling need. Another

solution is to create a structure of *handles* [23] (figure 2b) which corresponds to the desired multiple-role instances. The handle references several OIDs, each of them corresponding to a role played by this instance. This leads to a referencing problem and involves the use of message delegation. Moreover, *Jacques* would be only a *handle*, loosing its encapsulation, and therefore not an object anymore.

The implementation of RBAC models in OO systems clearly points out that maintaining association between roles and classes can be a tough design challenge, particularly when dealing with role hierarchies. For example, [3, 4] describes a, UML class-diagram to implement RBAC. Their framework includes *role* and *role instance* classes. Thus, software designers have to implement a mechanism to ensure that an object instance of *role* is linked with another object instance of *role instance*.

A review of role-based object models in the programming and database areas can be found in [10, 2, 6]. However, these models are intended mainly to take into account the dynamic part of the objects during their life, but either they do not propose in general any access control primitive or they do not totally respect the standard paradigms of object programming (e.g. [25]).

5.3 Integration of security in role-oriented systems

To the best of our knowledge, the closest paper to our is [25]. This approach intends to integrate a subset of RBAC into a role-oriented system: DOOR. This model permits modelling an owner relationship between roles and objects, but this approach does not entirely respect the standard paradigms of object programming. Section 5.2 explains how RAPOOL can be implemented in role-object oriented systems, nevertheless in such implementation, roles can still be used for non role-based authorization purposes. The Samovar model [2] is well suited to include a basic form of RBAC because it includes role definition in a logical manner. These role formulae can easily capture permission-role assignments of RBAC.

The aspect-oriented paradigm can be seen as an alternative to the role-oriented one. Both of them aim at a more flexible use of objects in OO systems. The authors of [19] describe their approach based on an “aspect-oriented modelling (AOM) technique that allows system developers to isolate cross-cutting design structures in aspects to support controlled evolution of the structures". In this approach, design structures that represent access control policies are treated as aspects. Policies are integrated into system by merging aspects with the system design model in which acces control concerns are not adresses. This composition results in a *woven model*. This approach is interesting for its dynamic perspectives on access control modelling but do not consider role hierarchy. We chosed to exclude dynamic concerns in our approach to provide a less flexible but easier to use language. In our approach the basic subset of RBAC is incorporated directly, thus do not impose the composition of two models. We are investigating whether RAPOOL can be as easily implemented in an aspect-based model as it is in a role-based model.

6 Discussion

Our proposal makes possible to take into account RBAC access control straight into the logical object data model. We presented the generic RAPOOL language, which contains two parts. The header allows specification of roles categories and hierarchies. The body part allows specification of authorizations at the method level, by mean of logical connectors in order to build more complex ones. We also presented the functional part of RAPOOL, which relies on a first-order logic engine.

Security is often divided into confidentiality, reliability and integrity. Confidentiality is the least considered non-functional requirements of security. Access control models, and nowadays role-based ones, are designed to enhance confidentiality. Integration of Mandatory Access Control [5] in UML diagrams and security extensions for UML [12, 14] are promising. They will bridge the gap between a security will and its implementation. These conceptual and logical propositions can be core models for methodologies considering security as in integral part of the whole software design process such as [17]. We are currently working on automatic translation into RAPOOL of UML diagrams expressed in specific security models. RAPOOL can indeed be used as a target language for a CASE supporting a RBAC-based design method, such as SecureUML. We currently plan to validate this approach using our prototype, a RAPOOL to C++ pre-processor, with the Foundstone SecureUML Visio template [1].

References

[1] R. Araujo and S. Gupta (2005) Design authorisation systems using SecureUML, *Foundstone*, Technical report.

[2] S. Coulondre and T. Libourel (2002) An integrated object-role oriented database model, *Data Knowl. Eng.*, 42(1):113–141.

[3] R. Crook, D. C. Ince, and B. Nuseibeh (2003) Modelling access policies using roles in requirements engineering, *Information & Software Technology*, 45(14):979–991.

[4] J. P. Davis and R. D. Bonnell (1999) Role-playing: A mechanism for bridging the object-oriented design-level gap, *OOPSLA-97: Workshop on Object Technology, Architectures and Domain Analysis*.

[5] T. Doan, S. Demurjian, T. C. Ting, and A. Ketterl (2004) Mac and uml for secure software design, *FMSE ’04: Proceedings of the 2004 ACM workshop*

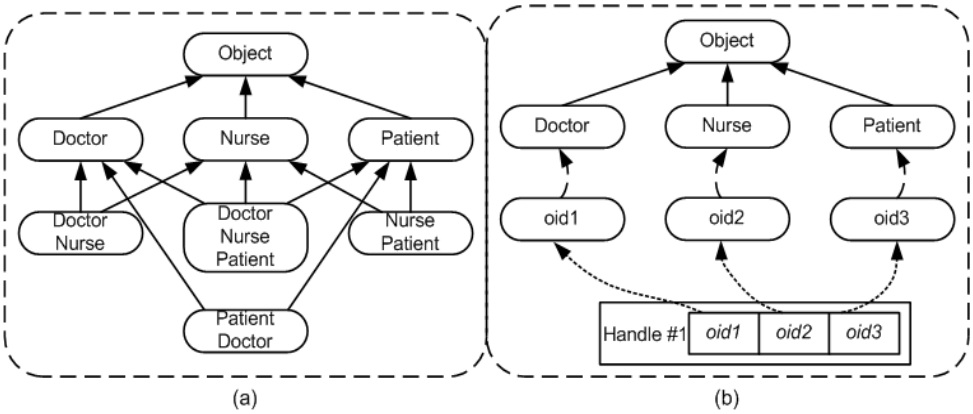


Figure 2: Empirical solutions for role implementation

on Formal methods in security engineering, ACM Press, 75–85.

[6] G. Gottlob, M. Schrefl, and B. Rock (1996) Extending object-oriented systems with roles, *ACM Trans. Inf. Syst.*, 14(3):268–296.

[7] S. Grumbach, P. Rigaux, and L. Segoufin (2001) Spatio-temporal data handling with constraints, *GeoInformatica*, 5(1):95–115.

[8] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor (2005) A generalized temporal role-based access control model, *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23.

[9] J. Jürjens (2002) UMLsec: Extending UML for secure systems development, *UML 2002 - The Unified Modeling Language. Model Engineering, Languages, Concepts, and Tools. 5th International Conference*, Springer, Dresden Germany, 412–425.

[10] G. Kappel, W. Retschitzegger, and W. Schwinger (1998) A comparison of role mechanisms in object-oriented modeling, *Modellierung CEUR Workshop Proceedings*, CEUR-WS.org.

[11] A. Kern, M. Kuhlmann, A. Schaad, and J. D. Moffett (2002) Observations on the role life-cycle in the context of enterprise security management, *SACMAT*, 43–51.

[12] D.-K. Kim, I. Ray, R. B. France, and N. Li (2004) Modeling role-based access control using parameterized uml models, *FASE*, Lecture Notes in Computer Science, 180–193.

[13] A. Kumar, N. Karnik, and G. Chafle (2002) Context sensitivity in role-based access control, *SIGOPS Oper. Syst. Rev.*, 36(3):53–66.

[14] T. Lodderstedt, D. A. Basin, and J. Doser (2002) Secureuml: A uml-based modeling language for model-driven security, *UML '02: Proceedings of the 5th International Conference on UML*, Springer-Verlag, London UK, 426–441.

[15] J. D. Moffett and E. Lupu (1999) The uses of role hierarchies in access control, *ACM Workshop on Role-Based Access Control*, 153–160.

[16] G. K. Mostéfaoui and J. Pasquier-Rocha (2003) Deterministic context-based security policies: An object-oriented approach, *SNPD*, ACIS, 160–165.

[17] H. Mouratidis, P. Giorgini, and G. A. Manson (2003) Integrating security and systems engineering: Towards the modelling of secure information systems, *CAiSE*, Lecture Notes in Computer Science, 63–78.

[18] S. L. Osborn, Y. Han, and J. Liu, (2003) A methodology for managing roles in legacy systems, *SACMAT*, ACM, 33–40.

[19] I. Ray, R. B. France, N. Li, and G. Georg (2004) An aspect-based approach to modeling access control concerns, *Information & Software Technology*, 46(9):575–587.

[20] H. Roeckle, G. Schimpf, and R. Weidinger (2000) Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization, *ACM Workshop on Role-Based Access Control*, 103–110.

[21] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman (1996) Role-based access control models, *IEEE Computer*, 29(2):38–47.

[22] F. Steimann (2000) On the representation of roles in object-oriented and conceptual modelling, *Data Knowl. Eng.*, 35(1):83–106.

[23] D. Thomsen, D. O'Brien, and J. Bogle (1998) Role-based access control framework for network enterprises, *ACSAC '98: Proceedings of the 14th Annual Computer Security Applications Conference*, IEEE Computer Society, Washington DC USA.

[24] M. B. Thuraisingham (1989) Mandatory security in object-oriented database systems, *OOPSLA '89: Conference proceedings on Object-oriented programming systems, languages and applications*, ACM Press, New York USA, 203–210.

[25] R. K. Wong (1997) Rbac support in object-oriented role databases, *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, ACM Press, New York USA, 109–120.

Annex: BNF Grammar of RAPOOL header

Note: grammar of non–terminal symbol *logical_formula* is not included.

RAPOOL_header := groups_list
groups_list := groups_list group
|
group := group_identifier 'Roles' '{' definitions_list '}'
group_identifier := IDENTIFIER

definitions_list := definitions_list definition
|
definition := role_definition
| hierarchy_definition
| equivalency_definition

role_definition := 'Roles' ':' roles_list ';'
roles_list := role_identifier roles_list_suite
roles_list_suite := ',' role_identifier roles_list_next
|
role_identifier := IDENTIFIER

hierarchy_definition := 'Hierarchy' ':' hierarchical_relations_list ';'
hierarchical_relations_list := hierarchical_relation hierarchical_relations_list_suite
hierarchical_relations_list_suite := ',' hierarchical_relation hierarchical_relations_list_suite
|
hierarchical_relation := role_identifier '<<' role_identifier

equivalency_definition := equivalency_identifier ':=' logical_formula ';'
equivalency_identifier := IDENTIFIER

A Formal Framework Supporting the Specification of the Interactions between Agents

Farid Mokhati
Département d'Informatique
Université d'Oum El Bouaghi - Algérie
E-mail: Mokhati@yahoo.fr

Mourad Badri and Linda Badri
Département de Mathématiques et d'Informatique
Université du Québec à Trois-Rivières - Canada
E-mail: Mourad.Badri@uqtr.ca, Linda.Badri@uqtr.ca

Keywords: multi-agent systems, RCA, Maude, translation, behavior, interactions, formal specification, verification and validation.

Received: May 24, 2005

In this paper we present a formal framework supporting the translation of interactions between agents (the interactions are described with the help of the RCA formalism) in a Maude specification. Based on rewriting logic, the formal and object-oriented language Maude supports formal specification and programming for a wide range of applications. The main motivations of our work are essentially: (1) to formally specify the behavior of multi-agent systems and (2) to provide a solid basis for their verification and validation. The translation process is illustrated by means of a real case study.

Povzetek: Opisano je prevajanje interakcij med agenti.

1 Introduction

In Multi-Agent Systems (MAS), agents interact in order to exchange information, to cooperate and to coordinate their tasks [24]. The usual approach to the description of interactions between agents consists in using protocols [8, 26]. Several agents' interaction protocols (AIP) have been proposed in the literature [7]. They constitute an important part of MAS's infrastructures. However, most of the protocols published in the literature are semi-formal, vague or contain errors as mentioned in [23]. Knowing that AIP play a crucial role in MAS development [30], their formal specification as well as their verification constitute essential tasks [11]. In the field of agents' behavior specification, three major approaches emerge in the literature: state-charts based approaches [27, 22], Petri Nets based approaches [5, 1], and approaches representing an adaptation of object-oriented specification methods [19, 20].

Among the agents' interaction protocols proposed in the literature, we can mention the RCA formalism (Représentation des Comportements d'Agents) [27], which is based on strongly typed states-transitions graphs. The RCA formalism allows describing agents' behaviors graphically. This formalism has been used in the design of several Cooperative Information Systems (CIS) based on informational agents. We can mention, for example, the NetMan project based on the coordination of several agents [4], a project related to the reactive reorganization of production shops and treating the cooperation between agents having to solve a problem in a distributed and cooperative way [28], as well as a project on the hydraulic management of the Camargue ecosystem and based on a negotiation process between agents (Project SIMFONHYC) [18].

One of the strong points of the RCA formalism [18, 28] resides in the modular design of agents' behaviors. Indeed, the use of composite action states makes it possible the overlapping of behavioral plans and therefore a description by successive refinements of agents' behavior. This characteristic comes directly from the notion of composite state of RCA graphs. Nevertheless, some critiques on RCA graphs can be formulated, notably on their formalization and on the sequential aspect of the execution cycle of behavioral plans [28]. Furthermore, this formalism allows the visualization of the synchronization points between dual protocols thanks to the complementarity between communication states and external transition. It is then easy to recognize the coordination points between dual protocols [28]. However, RCA graphs as well as the existing formalisms in the literature describing agents' interaction protocols are not endowed again with a formal semantics [28]. They only offer a semi-formal specification [23] of interactions between agents. These weaknesses can generate several problems in MAS development and verification.

Using formal notations for the description of MAS' behavior offers several advantages. It essentially allows producing rigorous and precise descriptions supporting efficiently their verification and validation process. The Maude language, based on the rewriting logic, seems to

us to be an interesting candidate. It offers, through its rich notation, an interesting way for concurrent systems formal specification and programming. Furthermore, it also supports the description of multi-agent interactions [21, 16]. In this paper, we present a formal framework supporting the translation of multi-agent interactions, specified using the RCA formalism, in a Maude specification. The main motivations of our approach are essentially: (1) to specify formally the behavior of multi-agent systems, in particular, the interactions between agents, and (2) to provide a solid basis for their verification and validation process. The Maude specifications, generated in the context of the developed framework, have been validated using the platform supporting the Maude language. The remainder of the paper is organized as follows: Section 2 gives a brief survey on the main related works. We present summarily the RCA formalism in section 3. In section 4, we give the basic concepts related to the rewriting logic as well as the Maude language. Section 5 presents the translation process. The proposed approach is illustrated using a concrete case study in section 6. Finally, section 7 gives some conclusions and future work directions.

2 Related Work

We present briefly in this section three formalisms (AUML, CATN and RCA) supporting the description of agents' interaction protocols. AUML [19, 9] is an extension of the UML language allowing describing interactions between agents. To represent multi-agent interaction protocols, AUML adopts in fact an approach in three layers. It uses, in the first level, packages and templates to represent the protocol in whole. Sequence diagrams, collaboration diagrams, activity diagrams, and states-transitions diagrams are used to represent interactions between agents. Activity diagrams and states-transitions diagrams are also used to capture agents' internal behavior (for more details see [19]). However, AUML only offers a semi-formal specification of the interactions between agents.

The CATN formalism (Coupled Augmented Transition NetWork) [10] is a states-transitions machine, to which a particular goal (or significance) is associated. A CATN can be decomposed in sub-CATNs. Each of these components is a CATN, having its own goal. The components of a CATN are joined together by ad-hoc transitions named "interactions transitions". Among these, we distinguish the non-terminal interactions transitions of those that are terminal. These last correspond to language acts (between agents) or to private actions of agents. This recursive aspect of the CATN allows a top-down design approach, from the most abstract behavior of a group of agents until their most concrete actions (individual terminal actions and communications through the interactions transitions). Each agent can execute in a concurrent way several CATNs depending on the tasks that it has to achieve [10, 25].

The RCA formalism [27, 28], supporting the description of role protocols, is used to describe agents' behavior. It is based on states-transitions diagrams introducing seven types of states and two types of transitions. The seven states are: the initial state, the final state, the elementary action state, the composite action state, the communication state and the waiting states (limited and unlimited). The two types of transitions are the internal transition and the external transition. Using this formalism, it is easy to recognize the coordination points between dual protocols. The RCA formalism is not limited to the description of the exchanges of messages between agents (as the case in the other formalisms). It also allows clarifying the actions that they undertake. In addition, the RCA graphs describe the working of the agents and help thus the design of their interactions. The links that exist between the macro level (i.e. the system's behavior) and the micro level (i.e. the agent's behavior) may be considered in an integrated way [28, 29].

These different approaches certainly offer some elements of answer to some problems related MAS development. However, they only allow a partial formalization of MAS. Furthermore, some authors [6, 5] opposed to the use of formalisms based on state-transition graphs two major arguments: 1) the impossibility to be able to verify the consistency of the protocols thus specified; and 2) the absence of taking into account the concurrent aspects of protocols [28]. In spite of the advantages that it offers relatively to the other formalisms, the RCA formalism only offers a graphic semi-formal description [18]. Furthermore, it is not endowed again with a formal semantics. These weaknesses combined to the complexity of MAS can generate several problems in their development and verification processes. The use of an appropriate formal notation for the description of MAS' behavior offers several advantages. It essentially allows the production of rigorous and precise descriptions supporting efficiently their verification and validation process. Our approach is similar, in terms of objectives, to the previously quoted approaches. It consists, essentially, to support the important stage of the specification of agents' behaviors. However, we preferred to adopt a more formal approach in the specification of agents' behaviors in terms of interactions allowing, among others, to support the verification of consistency (internal and global) in the behavior. Our approach allows translating the interaction protocols described using the RCA formalism in the Maude language. The Maude system consists in a high-level language of programming, specification and modeling based on rewriting logic [2, 15, 21]. It is also endowed with a high performance interpreter. It allows describing concurrent systems and supports the formal specification of distributed systems [14, 29, 12].

3 RCA Formalism

RCA (Représentation des Comportements d'Agents) [27, 28] is a formalism allowing describing an agent's behavior graphically. It is based on a strongly typed graph: seven types of states and two types of transitions (figure 1). The seven states are the initial state (to show

the beginning of the graph), the final state (to mark the end of the graph), the elementary action state (that corresponds to the agent's simple action), the composite action state (it is in fact about the call to another protocol), the communication state (sending of message), and the limited and unlimited waiting states (waiting of treatments done by other agents). The two types of transitions are the internal transition (it corresponds to the end of an activity and provokes the passage to another state) and the external transition (it is in fact a reception of a message that provokes, like an internal transition, the change of the agent's activity). An external transition is triggered by a communication state at another agent.

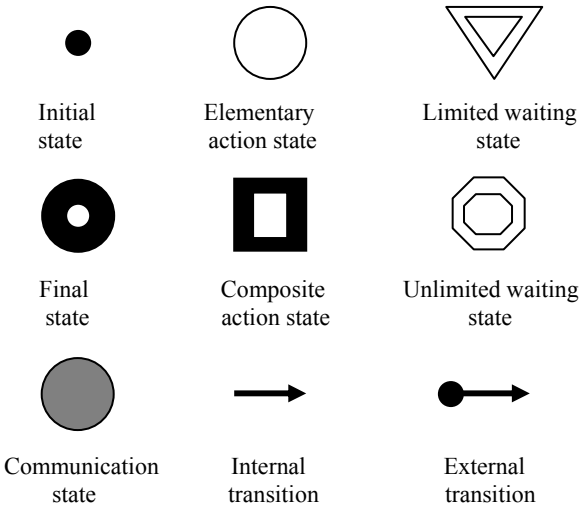


Figure 1 : Convention of representation of the RCA formalism.

The number of internal and external transitions depends on the type of the starting state and its transitions. It can be either null, limited or unlimited (figure 2).

Type of transition's departure state	Authorized internal transitions number	Authorized external transitions number
	[Min..Max]	[Min..Max]
Initial state	[0 .. 1]	[0 .. 1]
Elementary action state	[1 .. ∞]	[0 .. 0]
Composite action state	[1 .. ∞]	[0 .. 0]
Communication state	[1 .. 2]	[0 .. 0]
Limited waiting state	[1 .. 1]	[1 .. ∞]
Unlimited waiting state	[0 .. 0]	[1 .. ∞]
Final state	[0 .. 0]	[0 .. 0]

Figure 2 : Authorized transitions number according to the starting state.

Each states graph starts with a unique initial state and finishes by a unique final state. The internal events are the consequence of the agent's actions represented by action states (elementary or composite). They trigger the internal transitions. The external events result from communication activities of the agents, i.e. a reception of message constitutes an external event and provokes the crossing of an external transition. Of this fact, the type of allowed transition at a precise place of the graph depends exclusively of the origin state type of this transition:

- Initial state : only one transition (internal or external) may quit this state.
- Action state (simple or composite) : the internal transitions are in any number not null after action states.
- Communication state : one or two internal transitions may quit the communication state.
- Limited waiting state : the waiting may stop after the reception of a message (external transition), or if no message has been received beyond the waiting delay (internal transition). Furthermore, only one internal transition may quit a limited waiting.
- Unlimited waiting state : this waiting type remains while that it doesn't occur an external event (reception of message). It is therefore about a blocking state.

4 Rewriting Logic and Maude Language

4.1 Rewriting Logic

The rewriting logic, having a sound and complete semantics, was introduced by Meseguer [14]. It allows describing concurrent systems. This logic unifies all the formal models that express concurrence [13, 15]. In rewriting logic, the logic formulas are called rewriting rules. They have the following form: R:[t] → [t'] if C. Rule R indicates that term t becomes (is transformed into) t' if a certain condition C if verified. Term t represents a partial state of a global state S of the described system. The modification of the global state S of the system to another state S' is realized by the parallel rewriting of one or more terms that express the partial states. The distributed state of a concurrent system is represented as a term whose sub-terms represent the different components of the concurrent state. The concurrent state's structure can have a variety of equivalent representations because it satisfies certain structural laws (equivalence class).

```
1. sort Configuration .
2. sort Object .
3. sort Msg .
4. subsort Object < Configuration .
5. subsort Msg < Configuration .
6. op null : -> Configuration .
7. op_ _ : Configuration Configuration ->
          Configuration [assoc comm id : null] .
```

Figure 3 : Example of a portion of the Maude program.

For example, in an object-oriented system the concurrent state that is usually called configuration has the structure of a multi-set of objects and messages. Therefore, we can have configurations constructed by a binary operator applied to binary sets as illustrated in figure 3.

The portion of program illustrated in figure 3 gives a definition of three types: *Configuration*, *Object* and *Msg*. In lines 4 and 5, *Object* and *Msg* are sub-types of *Configuration*. Objects and messages are in fact multi-set configuration singletons. More complex configurations are generated from the application of the union on these multi-set singletons (objects and messages). Where there is neither floating messages nor live objects, we have in this case an empty configuration (line 6). The construction of a new configuration in terms of other configurations is done with line 7's operation. We can note that this operation has no name and that the two sub lines indicate the positions of two parameters of configuration type. This operation, which is the multi-set union, satisfies the structural laws of association and of commutation. It also possesses a neutral element null. For example, if we have a message *M1* that represents a configuration, and an object <*O : C|atts* > (please note that *O* is an object's identifier, *C* its class and *atts* the list of its attributes) that represents in itself another configuration, then we can construct another configuration in terms of those two configurations: *M1 < O : C | atts >*. This one is equivalent to the configuration *< O : C | atts > M1* because the __ operation is commutative.

4.2 Maude

Maude is a specification and programming language based on the rewriting logic [14, 3]. Three types of modules are defined in Maude. Functional modules allow defining data types and their functions through equations theory. Figure 4.a represents the functional module *Nat* specifying natural numbers. Such a module is imported in the module *FACT* (figure 4.b) to calculate the factorial of natural numbers. System modules define the dynamic behavior of a system. This type of modules extends functional modules by introducing rewriting rules. A maximal degree of concurrence is offered by this type of module. Finally, there are the object-oriented modules that can be reduced to system modules. In relation to system modules, object-oriented modules offer a more appropriate syntax to describe the basic entities of the object paradigm as, among others: objects, messages and configuration. Only one rewriting rule allows expressing the consumption of certain floating messages, the sending of new messages, the destruction of objects, the creation of new objects, state change of certain objects, etc.

Figure 5.a illustrates the use of a system module *BANK-ACCOUNT* to define an object counts banking *A* and the two operations capable to affect its content *credit* and *debit* while executing the rewriting rules defined in this module. Figure 5.b represents the same *BANK-ACCOUNT* module with a more appropriate object-oriented syntax.

```
fmod NAT is
sorts Zero NzNat Nat .
subsort Zero NzNat < Nat .
***constructors
op 0 : -> Zero .
op s_ : Nat -> NzNat .
….
endfm
```

(a)

```
fmod FACT is
Including NAT .
op _! : Nat -> NzNat .

var N : Nat .
eq 0 ! = 1 .
eq (s N) ! = (s N) * N !.
endfm
```

(b)

Figure 4 : Functional Modules *Nat* and *FACT*.

We note, that after the execution of the unconditional rule [credit], the message *credit(A, M)* is consumed and the content of the account is increased. In the same way, the execution of the conditional rule [debit] requires that the condition (N>=M) be verified. The execution of such rule generates the consumption of the message *debit(A,M)* and the reduction of the content of the account.

```
mod BANK-ACCOUNT is
protecting INT .
 including CONFIGURATION .
op Account : -> Cid.
op bal :_ : Int -> Attribute .
ops credit debit :    Oid Nat -> Msg .
var A  : Oid . vars M N : Int .

rl [credit] :    < A : Account | bal : N >   credit(A, M)
          =>   < A : Account | bal : N + M  > .

crl [debit] :   < A : Account | bal : N >   debit(A, M)
          =>  < A : Account | bal : N - M  >
              If N >= M .
endm
```

(a)

```
(omod BANK-ACCOUNT is
protecting MACHINE-INT .
class Account | bal : MachineInt .
msgs credit debit :   Oid MachineInt -> Msg .
var A : Oid .
vars M N : MachineInt .

rl [credit] :  < A : Account | bal : N >  credit(A, M)
          => < A : Account | bal : (N + M)  > .

crl [debit] :   < A : Account | bal : N >   debit(A, M)
          =>  < A : Account | bal : (N – M)  >
              If N >= M .
endom)
```

(b)

Figure 5 : The same *BANK-ACCOUNT* module in system module and O.O module forms.

5 Translating RCA Descriptions in Maude

We developed a formal framework allowing the formal specification of role protocols described using RCA formalism. The framework is composed, as illustrated by figure 6, of several modules: an object-oriented module (*ROLE-PROTOCOLE*) and several functional modules (the remainder of modules).

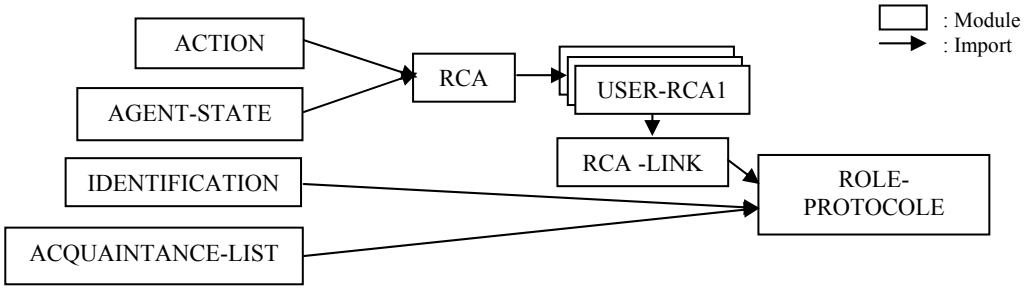


Figure 6 : *RCA-Maude* frameworks' architecture.

```
(fmod AGENT-STATE is
sorts AgentState KindAgentState NameAgentState .                    ***[1]

ops initial final communication elementary composite
                    limitedWaiting UnlimitedWaiting : -> KindAgentState .   ***[2]

op AgentState : NameAgentState KindAgentState -> AgentState .      ***[3]
op IsInitial : AgentState -> Bool .                                ***[4]
op IsFinal : AgentState -> Bool .                                  ***[5]
op IsOfCommunication : AgentState -> Bool .                        ***[6]
op IsElementary : AgentState -> Bool .                             ***[7]
op IsComposite : AgentState -> Bool .                              ***[8]
op IslimitedWaiting : AgentState -> Bool .                         ***[9]
op IsUnlimitedWaiting : AgentState -> Bool .                       ***[10]

var k : KindAgentState . var ns : NameAgentState .

eq  IsInitial(AgentState(ns, k)) = if  k == initial then true                    ***[11]
            else false fi .
eq  IsFinal(AgentState(ns, k)) = if  k == final then true                        ***[12]
            else false fi .
eq  IsOfCommunication(AgentState(ns, k)) = if  k == communication then true      * **[13]
            else false fi .
eq  IsElementary(AgentState(ns, k)) = if  k == elementary then true              ***[14]
            else false fi .
eq  IsComposite(AgentState(ns, k)) = if  k == composite then true                ***[15]
            else false fi .
eq  IslimitedWaiting(AgentState(ns, k)) = if  k == limitedWaiting then true      ***[16]
            else false fi .
eq  IsUnlimitedWaiting(AgentState(ns, k)) = if  k == UnlimitedWaiting then true  ***[17]
            else false fi .
endfm)
```

Figure 7 : The functional module *AGENT-STATE*.

The functional module *AGENT-STATE* (figure 7) contains the different necessary type declarations for the definition of a state (line [1]) and, on the other hand, the definition of operations used for the construction and the manipulation of a state (lines [2, 3, 4, 5, 6, 7, 8, 9, 10]), as well as equations implementing these operations (lines [11, 12, 13, 14, 15, 16, 17]).

In the *ACTION* module (figure 8), in addition to the type Action, we define the two functions *IsSendingToOnlyOne* and *IsSendingToAll.* The first

function determines if an action is destined to only one agent's acquaintance, on the other hand the second function indicates if it is necessary to send a message to all agent's acquaintances. To describe the identification mechanism of agents, we define the functional module *IDENTIFICATION* (figure 9). Furthermore, an agent must be endowed with a list of its acquaintances allowing it to exchange messages with the other agents. We define for it the functional module *ACQUAINTANCE-LIST* to manage the lists of the agents' acquaintances . Due to imitation of space and a considerable size of this last module, we don't present it in this paper.

```
(fmod ACTION is
protecting BOOL .
sort Action .
op IsSendingToAll : Action -> Bool .
op IsSendingToOnlyOne : Action -> Bool .
endfm)
```

Figure 8 : The functional module *ACTION*.

```
(fmod IDENTIFICATION is
 sort AgentIdentifier .
 subsort AgentIdentifier < Oid .
 endfm)
```

Figure 9 : The functional module *IDENTIFICATION*.

To define an RCA diagram, we propose the *RCA* module (figure 10). This module reuses the *AGENT-STATE* and *ACTION* modules. It includes the definition of two operations: *TargetState* that determines the target state according to a state source and an action, and the *FeedBack* operation used in the case where the treatment accomplished by the agent takes place while toppling between two states during a limited length. To each event coming from a state source, such a function determines the appropriate action that should be executed from the target state as a feedback.

```
(fmod RCA is
protecting ACTION .
protecting AGENT-STATE .
op TargetState : AgentState Action -> AgentState .
op FeedBack : Action -> Action .
endfm)
```

Figure 10 : The functional module *RCA*.

For the construction of an RCA diagram for an application, we propose to extend the *RCA* module in another *USER-RCA* module (figure 11). In this module, the user must: mention all states constituting the RCA diagram, define all possible actions, attach the actions in the states using the *TargetState* function, determine the actions constituting feedbacks using the *Feedback* function, and finally specify for every communication action whether it is sent to all (using the *IsSendingActionToAll* function) or to only one (using *IsSendingActionToOnlyOne*). An *USER-RCA* module (figure 11) is associated with every category of agents (playing the same role).

```
(fmod USER-RCA is
extending RCA .

***User part***
endfm)
```

Figure 11 : The functional Module *USER-RCA*.

To respect the interaction protocol used between agents, we propose to realize a sort of link between the RCA diagrams of the different agents. Basing on the synchronization points, main characteristic of this formalism, such a link consists in guaranteeing that at the moment of the reception of a message, an agent can't consume such a message except if it is in the corresponding state of the state of the sender agent. An agent that is in a communication state generates an external event that causes an external transition at the agent receiver. To receive such an event, this last must be in a waiting state (limited or unlimited). Indeed, the sending actions accomplished by a sender agent represent events for receiver agent. Thus, there is a correspondence between the sending actions of the sender and the events received by the receiver. For it, the user must develop the *RCA-LINK* module (figure 12) that contains the correspondence on the one hand, between the different states of agents and, on the other hand, between the events generated by the sender and the events received by the receiver.

```
(fmod RCA-LINK is
protecting USER-RCA .
…
op CorrespondingState : AgentState -> AgentState .
op CorrespondingAction : Action -> Action .

***User part***************
…
endfm)
```

Figure 12 : The functional module *RCA-LINK*.

The object-oriented module *ROLE-PROTOCOL* (figure 13) represents the main module. It imports the *RCA-LINK*, *IDENTIFICATION,* and *ACQUAINTANCE-LIST* modules. For the formal description of agents, we propose the class *Agent* (line 2).

The definition of this class has as attributes *PlayRole*, *State,* and *AcqList,* to contain in this order, the agent's actual role, the current state of the agent, and the list of its acquaintances. In addition to different types of states defined in figure 7, we define in this module (figure 13)

the type *EventType* (line 1) relative to the two types of events used in this formalism (Internal and External). The appearance of an event is expressed by message *Event* (line 3) having as parameters an agent, a role, the type of the event, the agent's state, and an action.

In the RCA formalism, an agent changes state while doing either an internal transition or an external one. Figure 13 illustrates the necessary rewriting rules we developed modeling the possible cases of transitions (internal and external), while respecting the constraints of this formalism described by the table given in figure 2.

```
(omod ROLE-PROTOCOLE is
protecting RCA-LINK .
protecting IDENTIFICATION .
protecting ACQUAINTANCE-LIST .
sorts Agent Role EventType .

ops Internal External : -> EventType .                                    ***[1]
class  Agent | PlayRole : Role, State : AgentState, AcqList : acquaintanceList  .    ***[2]
Msg Event : Oid Role EventType AgentState Action -> Msg .                 ***[3]

*******************************************************************************
vars A A1 : Oid . var S : AgentState . vars R R1 : Role .
var Act : Action . var ACL : acquaintanceList .

*****************************Possible cases of internal transition**************************
*****************************************First case***********************************
crl[InternalTransitionCase1] :                                            ***[4]
        Event(A, R, Internal, S, Act)
        < A : Agent | PlayRole : R, State : S, AcqList : ACL >
       =>
        < A : Agent | PlayRole : R, State : TargetState(S, Act), AcqList : ACL >
        if (IsInitial(S) or IsElementary(S) or IsComposite(S) or IslimitedWaiting(S)) .

*****************************************Second case**********************************
crl[InternalTransitionCase2] :                                            ***[5]
       Event(A, R, Internal, S, Act)
       < A : Agent | PlayRole : R, State : S, AcqList : ACL >
      =>
       < A : Agent | PlayRole : R, State : TargetState(S, Act), AcqList : TailA(ACL) >
       Event(HeadA(ACL), R1, External, CorrespondingState(S), CorrespondingAction (Act))
       if IsOfCommunication(S) and IsSendingToOnlyOne(Act) .

*****************************************Third case***********************************
crl[InternalTransitionCase3] :                                            ***[6]
        Event(A, R, Internal, S, Act)
        < A : Agent | PlayRole : R, State : S, AcqList : ACL >
      =>
        < A : Agent | PlayRole : R, State : S, AcqList : TailA(ACL) >
     Event(A, R, Internal, S, Act)
       Event(HeadA(ACL), R1, External, CorrespondingState(S), CorrespondingAction(Act))
       if IsOfCommunication(S) and IsSendingToAll(Act) and ACL =/= EmptyacquaintanceList .

*********************Possible case of External transition*****************************
crl[ExternalTransition] :                                                 ***[7]
        Event(A, R, External, S, Act)
        < A : Agent | PlayRole : Initiator, State : S, AcqList : ACL >
       =>
        < A : Agent | PlayRole : Initiator, State : TargetState(S, Act), AcqList : ACL >
        if  IsInitial(S) or IslimitedWaiting(S)  or  IsUnlimitedWaiting(S) .

*******************************************************************************
…
endom)
```

Figure 13 : The object-oriented module *ROLE-PROTOCOLE*.

An agent doesn't do an internal transition except if it is in one of the following states: initial, elementary, composite, limited waiting or communication (see figure 2). In the first four states, an internal transition is described by the rewriting rule (line 4) of figure 13. Such a rule expresses that at the moment of the appearance of an internal event, the agent consumes the message and changes its state using the *TargetState* function defined in the *RCA* module (figure 10). We treated separately the case of a communication state, knowing that from this state the agent generates an external event (sending of message) allowing its acquaintances that are in waiting to change their states. A message can be sent by an agent to only one agent belonging to its acquaintance list or to all its acquaintances.

The first case is described by the rule of the line 5. Such a rule expresses, on the one hand, the consumption of an internal event, on the other hand, the generation of an external event sent to only one agent (here we adopt the strategy choosing the agent that is at the head of the acquaintances list using the *HeadA* function), if the agent sender is in a communication state. The second case is described by the rule of the line 6. Such a rule presents the sending of a message by the agent *A* to all its acquaintances. It presents a conditional loop. Indeed, it allows browsing the acquaintance list (*ACL*) of the agent, while using the two operations *HeadA* (determines the head of the list) and *TailA* (determines the rest of the list). Such a loop stops when the list is browsed completely. An agent doesn't do an external transition except if it is in a waiting state (limited either unlimited) or sometimes in its initial state (see figure 2). This is expressed by the rewriting rule of the line 7. When it occurs an external event to an agent, this last changes its state while doing an external transition, but the agent must be in an initial or waiting state (limited either unlimited).

6 Case Study : Auction Application

This section illustrates the application of our approach on a concrete example. It is about a simple example of an auction.

We have two kinds of agents: *Auctioneer* and *Bidder*. Each auction involves one Auctioneer and several Bidders.

The Auctioneer has a catalog of products. Before beginning the auction, the Auctioneer sends the catalog to all participants. Then, it begins the auction for all products. The products are proposed sequentially to the participants. Figures 14.a and 14.b describe the representation of the Auctioneer and Bidder roles respectively using the RCA formalism.

6.1 Application of the Translation Process

The formal description of the behaviors of the agents whose roles are described using the RCA formalism implies all defined modules previously with the definition of the *USER-RCA* and *RCA-LINK* modules. Figures 15 and 16 illustrate the defined modules corresponding to the Auctioneer and Bidder roles respectively. The correspondence between these roles is presented in figure 17. Indeed, the two modules *USER-RCA1* (figure 15) and *USER-RCA2* (figure 16) describe the Auctioneer and Bidder roles respectively in the same way. We limit ourselves to detail the *USER-RCA1* module only.

In figure 15, we define the different states of the Auctioneer agent (lines 1 and 2). For example, the state *AgentState(CommitmentDecision, communication)* means that the state named *CommitmentDecision* is a communication state (see figure 14.a). The actions given in figure 14.a are described by line 3. To determine the target state (line 4) according to a source state and a given action, we used the operation *TargetState* defined in figure 10. If the Auctionner agent is in its *CommitmentDecision* state, and the action to execute is *AcceptProposalSent*, the target state of this transition must be the final state *EndI*. To select the conditional rule to execute when the agent is in a communication state (see figure 13, lines 5 and 6), it is necessary to know the type of the action. For example, the line 5 of figure 15 indicates that the *CFP-Sent* action must be sent by the Auctioneer to all Bidders.

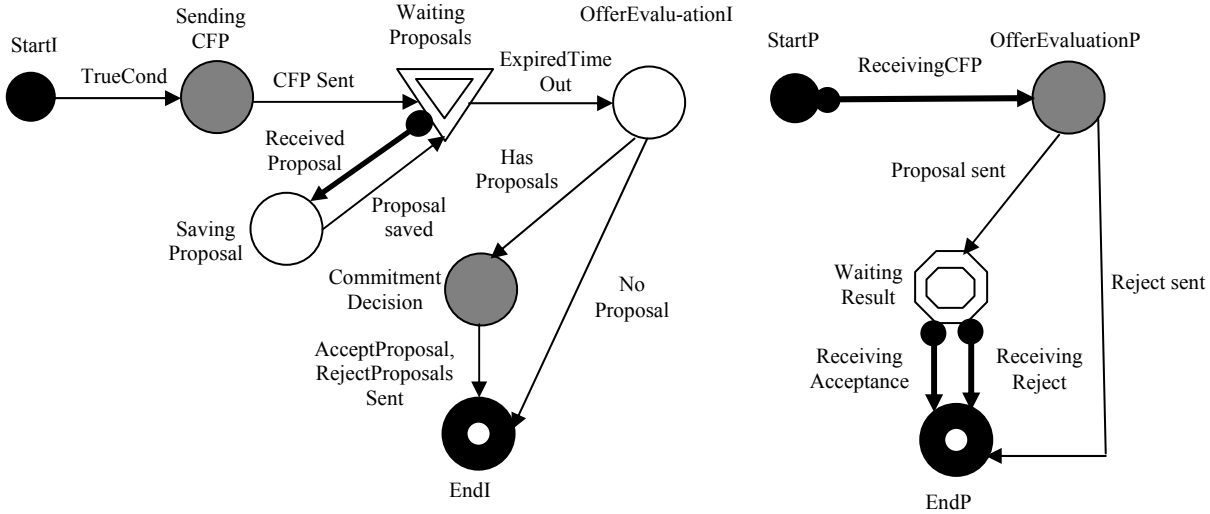


Figure 14 : Representation of the roles, *Auctioneer* and *Bidder* using RCA formalism.

```
fmod USER-RCA 1 is
extending RCA .

****************States of an Auctioneer************************************************
ops StartI SendingCFP WaitingProposals OfferEvaluationI SavingProposal
                                        CommitmentDecision EndI : ->  NameAgentState .      ***[1]

ops AgentState(StartI, initial) AgentState(SendingCFP, communication)
    AgentState(WaitingProposals, limitedWaiting) AgentState(OfferEvaluationI, elementary)
    AgentState(SavingProposal, elementary) AgentState(CommitmentDecision, communication)
                                                AgentState(EndI, final) : -> AgentState .      ***[2]

***************Actions to accomplish by an Auctioneer**********************************
ops TrueCondition CFP-Sent ExpiredTimeOut NoProposal HasProposal  ReceivedProposal
                          ProposalSaved AcceptProposalSent RejectProposalSent : -> Action .      ***[3]

***************Determination of the target state according to a state source and an action *********
eq TargetState(AgentState(StartI, initial), TrueCondition) = AgentState(SendingCFP, communication) .
…
eq TargetState(AgentState(CommitmentDecision, communication), AcceptProposalSent) =
                                                                AgentState(EndI, final) .      ***[4]
eq TargetState(AgentState(CommitmentDecision, communication), RejectProposalSent) =
                                                                AgentState(EndI, final) .

************* Determination of the type of an action ***************************************
eq IsSendingToAll(CFP-Sent) = true .                                                          ***[5]
eq IsSendingToOnlyOne(AcceptProposalSent) = true .

endfm
```

Figure 15 : The module *USE-RCA1* corresponding to the Auctioneer agent.

```
fmod USER-RCA2 is
extending RCA .

****************States of a Bidder*****************************************************
ops StartP OfferEvaluationP WaitingResult EndP : -> NameAgentState .

ops AgentState(StartP, initial) AgentState(OfferEvaluationP, communication)
          AgentState(WaitingResult, UnlimitedWaiting) AgentState(EndP, final) : -> AgentState .

***************Action to accomplish by a Bidder**************************************
ops ReceivingCFP ProposalSent RejectSent ReceivingAcceptance  ReceivingReject : -> Action .

****************Determination of the target state according to a state source and an action *****
eq TargetState(AgentState(StartP, initial), ReceivingCFP) = AgentState(OfferEvaluationP, communication) .
…
eq TargetState(AgentState(WaitingResult, UnlimitedWaiting), ReceivingAcceptance ) = AgentState(EndP, final) .
eq TargetState(AgentState(WaitingResult, UnlimitedWaiting), ReceivingReject ) = AgentState(EndP, final) .

*************** Determination of the type of an action**********************************
eq IsSendingToOnlyOne(ProposalSent) = true .
eq IsSendingToOnlyOne(RejectSent) = true .

endfm
```

Figure 16 : The module *USER-RCA2* corresponding to the Bidder agent.

The *RCA-LINK* module of figure 17, presents a correspondence on the one hand, between the different states of the agents Auctioneer and Bidder and, on the other hand, between the events they exchange. For example, if the Auctioneer agent is in its communication state *SendCFP*, the Bidder must be in its initial state *StartP* (line 1). In the same way, if the Bidder is in its communication state *OfferEvaluationP* (line 3), the Auctioneer must wait its decision. Indeed, an external event for an agent receiver corresponds to a message sent by a sender agent. For example, when the Auctioneer throws a call-for-proposal (*CFP-Sent*), the Bidder agent receives the call-for-proposal event (*ReceivingCFP*). This is expressed by the rule of the line 2. Also, when the Bidder accepts to propose, it sends its proposition (*ProposalSent*), and of the other side, the Auctionner receives its proposition (*ReceivedProposal*) (line 4).

```
fmod RCA-LINK is
protecting RCA1 .
protecting RCA2 .
sort EventType .

ops Internal External : -> EventType .
op CorrespondingState : AgentState -> AgentState .
op CorrespondingAction : Action -> Action .

**********************************Auctioneer Part**********************************

eq CorrespondingState(AgentState(SendingCFP, communication)) = AgentState(StartP, initial) .          ***[1]
eq CorrespondingState(AgentState(CommitmentDecision, communication)) =
                                                AgentState(WaitingResult, UnlimitedWaiting) .
…
eq CorrespondingAction(CFP-Sent) =  ReceivingCFP  .                                                  ***[2]
eq CorrespondingAction(AcceptProposalSent) = ReceivingAcceptance .

**********************************Bidder Part**************************************

eq CorrespondingState(AgentState(OfferEvaluationP, communication)) =
                                                        AgentState(WaitingProposals, limitedWaiting) .   ***[3]
…
eq CorrespondingAction(ProposalSent) = ReceivedProposal .                                            ***[4]

endfm
```

Figure 17 : The module *RCA-LINK*.

6.2 Validation of the Generated Description

The rewriting logic offers a great flexibility in terms of simulation of a specification, in particular, concerning the choice of the initial configuration. This choice plays a primordial role in the validation of the description of a system. Using all the system's description, we can validate a part of the system without involving the rest. For a validation of the AIP given by figure 14, we consider two essential cases: the case where there are Bidders that accept to propose and others do not, and the case where all Bidders refuse to propose. For the first case, we propose the following initial configuration :

```
< "Auctioneer" : Agent | PlayRole : Initiator, State : AgentState(StartI, initial), AcqList : ("Bidder1" :
                                                                                ("Bidder2" : "Bidder3")) >
< "Bidder1" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
< "Bidder2" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
< "Bidder3" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
Event("Auctioneer", Initiator, Internal, AgentState(StartI, initial), TrueCondition)
Event("Auctioneer", Initiator, Internal, AgentState(SendingCFP, communication), CFP-Sent)
Event("Bidder1", Participant, Internal, AgentState(OfferEvaluationP, communication), ProposalSent)
Event("Bidder2", Participant, Internal, AgentState(OfferEvaluationP, communication), ProposalSent)
Event("Bidder3", Participant, Internal, AgentState(OfferEvaluationP, communication), RejectSent)  .
```

Figure 18 : Initial configuration.

We define an initial configuration including an agent initiator " Auctionneer ", and three agents participants ("Bidder1", "Bidder2", "Bidder 3"). In the beginning, every agent is in its initial state. From its *OfferEvaluationP* state a Bidder agent can send a proposition as it can refuse to propose. In the configuration of figure 18, Bidder1 and Bidder2 send their propositions whereas Bidder3 refuses to propose while sending a reject. The unlimited rewriting (without indicating the number of the rewriting steps) of this configuration gives the result illustrated by figure 19.

```
< "Auctioneer" : Agent | PlayRole : Initiator, State : AgentState(EndI, final), AcqList :
                                                        ("Bidder1" :  ("Bidder2" :  "Bidder3") >
< "Bidder1" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
< "Bidder2" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
< "Bidder3" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
```

Figure 19: Auctioneer and Bidders in their final states.

After it sends a call for proposal to all Bidders, the agent Auctioneer begins to receive the proposal from Bidders agents. Once the considered deadline is expired (internal event) the initiator throws its evaluation process while choosing the most appropriate proposition (here we adopt the strategy based on the first proposing).

So, the Auctioneer sends to the chosen Bidder (here "Bidder1") an acceptance, and to the other (here "Biddert2") a reject. Bidder3 is not concerned because it refused to propose and therefore passes to its final state (see figure 14.b). For the second case, we propose the initial configuration of the following figure:

```
< "Auctioneer" : Agent | PlayRole : Initiator, State : AgentState(StartI, initial), AcqList :
                                                        ("Bidder1" : ("Bidder2" : “Bidder3”)) >
< "Bidder1" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
< "Bidder2" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
< "Bidder3" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
 Event("Auctioneer", Initiator, Internal, AgentState(StartI, initial), TrueCondition)
 Event("Auctioneer", Initiator, Internal, AgentState(SendingCFP, communication), CFP-Sent)
 Event("Bidder1", Participant, Internal, AgentState(OfferEvaluationP, communication), RejectSent)
 Event("Bidder2", Participant, Internal, AgentState(OfferEvaluationP, communication), RejectSent)
 Event("Bidder3", Participant, Internal, AgentState(OfferEvaluationP, communication), RejectSent)  .
```

Figure 20 : Initial configuration.

The configuration of figure 20 looks like the one of figure 18 except that the Bidders refuse to propose. The unlimited rewriting (without indicating the number of the rewriting steps) of this configuration gives the result illustrated by figure 21.

```
< "Auctioneer" : Agent | PlayRole : Initiator, State : AgentState(EndI, final), AcqList :
                                                        ("Bidder1" : ("Bidder2" :  "Bidder3") >
< "Bidder1" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
< "Bidder2" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
< "Bidder3" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
```

Figure 21: Auctioneer and Bidders in their final states.

Every participant who refuses to propose passes to the *EndP* state (see figure 14.b). In the same way, the initiator waits for the expiration of the deadline and as it doesn't receive any proposition during this interval of time, it passes on its turn in the *EndP* state (see figure 14.a). Indeed, the configuration of figure 21 seems to be the same that the one of figure 19. It is due to the fact that in the RCA formalism an agent can have only one final state. However, such configurations are different (for example, the *EndP* state of agent Bidder1 in figure 19 is a success state, but in figure 21 such a state presents a failure).

6.3 Implementation

Figure 22 illustrates a part of the code we developed. It visualizes the rewriting rule that describes the reception of an external event by the agent *A1* who plays the *Participant* role and exists in the state *S*. This rule also expresses the transition from the state *S* of the agent *A1* to another target state determined by the function

TargetState(S, Act). The triggering of such a transition only takes place if the agent *A1* is in one of waiting (limited or unlimited) or initial states. This is expressed in this conditional rule by the boolean functions *IsUnlimitedWaiting(S)*, *IslimitedWaiting(S)* and *IsInitial(S)* respectively.

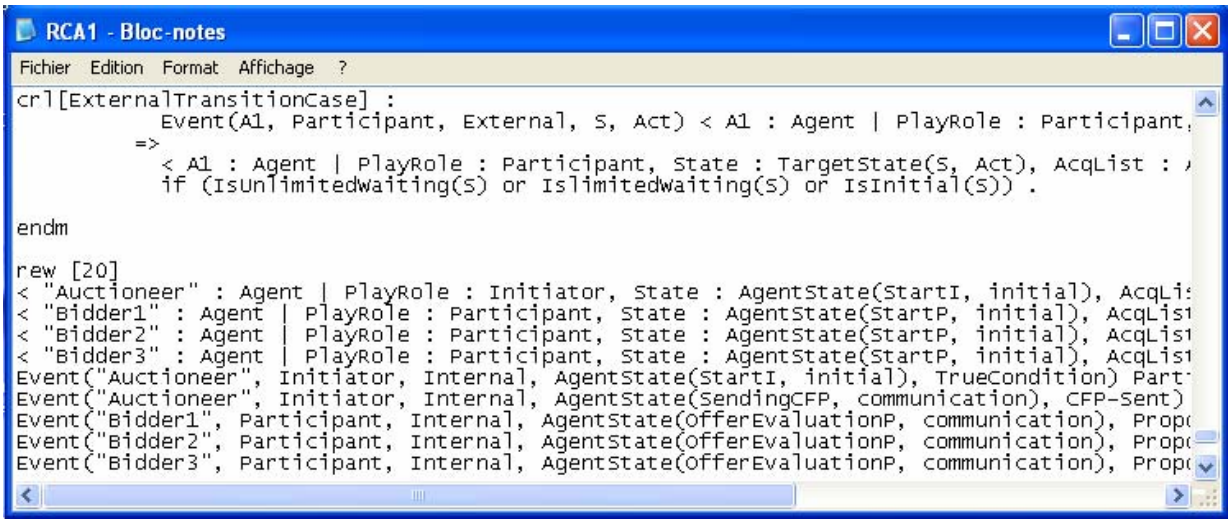


Figure 22 : Part of the developed code.

Furthermore, figure 22 shows the limited rewriting (after 20 rewriting steps) of an initial configuration. In this configuration, we have the agent " Auctioneer " playing the *Initiator* role, and the three agents " Bidder1 ", " Bidder2 " and " Bidder3 " each playing the *Participant* role. All agents are in the departure in their initial states (*StartI* for agent Auctioneer and *StartP* for the Bidders). We suppose, in this initial configuration, that after the sending of the call for proposal by the Auctionner to all Bidders, these last send propositions in the case where they are in state of evaluation of proposal *OfferEvaluationP*. This state is a communication state (see figure 14).

The result of rewriting of such an initial configuration is illustrated by figure 23. The Auctioneer throws its decision process, and all Bidders wait for an answer from it. The agent Auctioneer is in its elementary state *OfferEvaluationI* and all Bidders are in their unlimited waiting states *WaitingResult*.

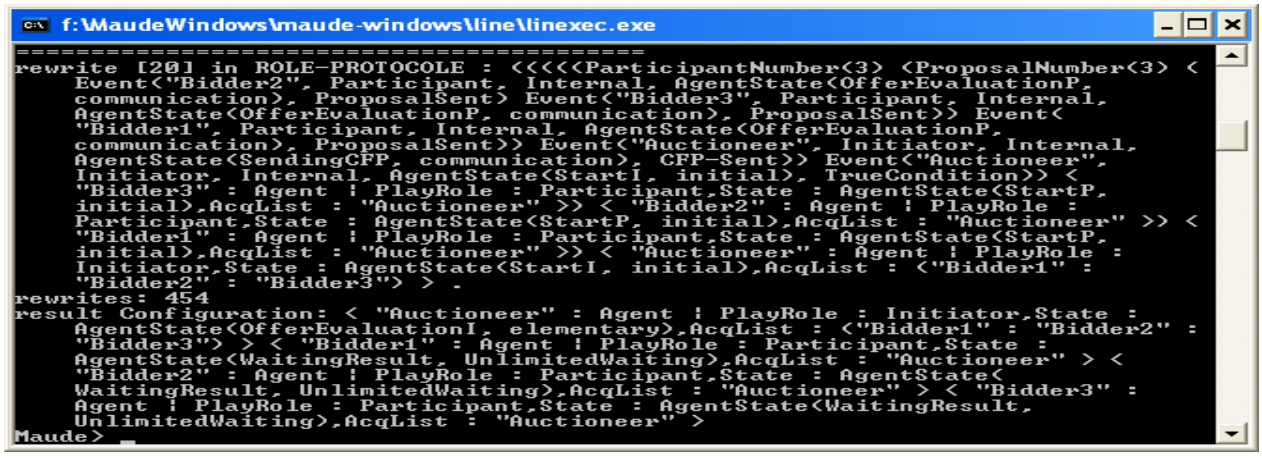


Figure 23 : Result of limited rewriting (after 20 steps) of the initial configuration.

7 Conclusions and Future Work

The RCA formalism allows specifying the roles protocols and is used to describe agents' behavior. Compared to others formalisms, RCA allows recognizing the synchronization points between dual protocols. As for the other existing formalisms, RCA is not endowed yet with a formal semantics [28]. Furthermore, it only allows a partial formalization of MAS [17, 22].

In this article, we proposed a formal framework supporting the translation of interactions between agents, specified using the RCA formalism, in a Maude specification. The translation process is based on the RCA graphs. All the concepts used by the RCA

formalism are supported by Maude. Based on rewriting logic, the formal and object-oriented language Maude supports formal specification and programming for a wide range of applications. The result of the translation procures a formal description of the interactions between agents preserving the consistency in their behavior. It offers a solid basis for their verification and validation process. The generated Maude specifications are flexible and remain open to extension.

Maude is supported by a tool. This allowed us, as a first experiment, in addition to the modeling, to perform a validation (based on a simulation) of our approach. Furthermore, we work on the extension of our approach in order to integrate the possibilities offered by the Maude language (model-checker) to verify some properties of the interactions between agents described using RCA graphs and translated in Maude.

References

[1] Bakam I., Kordon F., Le Page C., Bousquet F. « Formalization of a Spatialized Multiagent Model Using Coloured Petri Nets for the Study of a Hunting Management System ». First International Workshop, FAABS 2000, Greenbelt, MD, USA, April 2000. FAABS 2000.

[2] Bruni R., and Meseguer J., « Generalized rewrite theories ». In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP 2003), volume 2719 of Lecture Notes in Computer Science, pages 252-266. Springer, 2003.

[3] Clavel M., and al. "Maude : Specification and Programming in Rewriting Logic". Internal report, SRI International, 1999.

[4] Cloutier L. «Une approche multi-agents par conventions et contrats pour la coordination de l'entreprise manufacturière réseau », Université de Droit d'Economie et des Sciences d'Aix-Marseille III, DIAM-IUSPIM, Marseille, 1999.

[5] Cost R., and al. «Modeling Agent Conversations with colored Petri Nets», dans Working Notes of the Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents'99, Seattle, Washington, mai 1999.

[6] El Fallah-Seghrouchni A., et Mazouzi H., «Une démarche méthodologique pour l'ingénierie des protocoles d'interaction», in. Actes Ingénierie des systèmes multi-agents, JFIADSMA'99, 8-10 novembre 1999, Saint-Gilles, Ile de la Réunion.

[7] Guessoum Z. «Modèles et Architectures d'Agents et de Systèmes Multi-Agents Adaptatifs ». Dossier d'habilitation à diriger des recherches de l'Université Pierre et Marie Curie. Décembre 2003.

[8] Huget M.P., «Model Checking Agent UML Protocol Diagrams ». Technical report ULCS-02-012 from the department of computer science, University of Liverpool. Version 2002/04/16.

[9] Huget M.P., and Odell J. «Representing Agent Interaction Protocols with Agent UML » AAMAS'04, July 19-23, 2004, New York, New York, USA.

[10] Lemaître C., Prat, X., Magnin, L. et Dury A. «Description, programmation et validation d'interactions par Coupled Augmented Transition Network(CATNs) ». In Actes des Secondes Journées Francophones sur les Modèles Formels d'Interactions (MFI'03). Lille, France, 20-23 mai 2003.

[11] Mazouzi H., El fallah Seghrouchni A.,, and Haddad S., «Open protocol design for complex interaction in multi-agent systems ». In Proceedings of the first international joint conference on Autonomous agent and multi-agent systems, pages 517-526. ACM Press, 2002.

[12] McCombs T., «Maude 2.0 Primer, Version 1.0». Internal report, SRI International, 2003.

[13] Meseguer J., «Rewriting as a unified model of concurrency» In Proceedings of the Concur'90 Conference, Amsterdam, Pg 384-400, Springer LNCS Vol. 458, 1990.

[14] Meseguer J., «Logical Theory of Concurrent Objects and its Realization in the Maude Language» In G. Agha, P. Wegner, and A. Yonezawa, Editors, Research Directions in Object-Based Concurrency. MIT Press, 1992.

[15] Meseguer J., «Rewriting Logic and Maude : a Wide-Spectrum Semantic Framework for Object-Based Distributed Systems» In S. Smith and C. L. Talcott, editors, Formal Methods for Open Object-Based Distributed Systems, FMOODS2000, 2000.

[16] Mokhati F., Boudiaf N., Badri L., & Badri M., «Generating Maude Specification from AUML Diagrams: Toward A Systematic Approach». In Proc of CSITeA-04 conference. Cairo, Egypt. December 27-29, 2004.

[17] Mokhati F., Boudiaf N., Badri M., & Badri L., «DIMA-Maude: Toward a Formal Framework for Specifying and Validating DIMA Agents». In Proc of the MOCA'04 conference. Arrhus, Denmark, October 11-13, 2004. pp. 169-187.

[18] Nathalie F., «Modélisation et simulation multi-agents d'écosystèmes antropisés : une application à la gestion hydraulique en grande Camargue», Université de Droit d'Economie et des Sciences d'Aix-Marseille III, IUSPIM-DIAM, Marseille, 2001.

[19] Odell J., Parunak H.V.D., Bauer B., «Representing agent Interaction protocol In UML» conférence AAAI Agents 2000, Barcelone, 3-7 juin 2000.

[20] Odell J., Parunak H. V. D., Bauer B., «Representing agent Interaction protocol In UML», Agent Oriented Software Enginering, Paolo Ciancarini and Michael Wooldridge (eds.), Springer-Verlag, Berlin, 2001, pp. 121-140.

[21] Olveczky P.C., «Modeling and Analyzing Protocols in Maude» 8th Brazilian Symposium on Programming Languages (SBLP'04). May 26-28, 2004.

[22] Paurobally S, Cunningham J., «Achieving Common Interaction Protocols in Open Agent Environments», 2nd international workshop on Challenges in Open

Agent Environments, AAMAS 2003, Melbourne, Australia 14-18th July 2003.

[23]Paurobally S, Cunningham J, and Jennings N R., «Developing Agent Interaction Protocols Using Graphical and Logical Methodologies» in Proc. AAMAS03 PROMAS Workshop on Programming Multi-Agent Systems , 2003.

[24]Paurobally S., Cunningham J., and Jennings, N. R., «Verifying the contract net protocol: a case study in interaction protocol and agent communication semantics». In Proceedings of 2nd International Workshop on Logic and Communication in Multi-Agent Systems, Nancy, France 2004, pp. 98-117.

[25]Pham V. T., Laurent M., Houari S., «Adaptation dynamique des systèmes multi-agents basée sur le concept de méta-CATN». In Actes de la Deuxième Conférence Internationale Associant Chercheurs Vietnamiens et Francophones en Informatique, Hanoï Vietnam, 2-5 Février 2004.

[26]Toivonen S. and al. «Using Interaction Protocols in Distributed Construction Processes». In Seruca, I., Filipe, J., Hammoudi, S., and Cordeiro, J. (Eds.): Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS'04), Porto, Portugal, April 2004, pp. 344—349

[27]Tranvouez E., Espinasse B., «Protocoles de coopération pour le réordonnancement d'atelier». In Actes des journées francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA'99) à Saint-Gilles, île de la Réunion, novembre 1999, Gleizes J.-P., Marcenac P., Ed. Hermès, 1999.

[28]Tranvouez E., «IAD et ordonnancement : une approche coopérative du réordonnancement par systèmes mulit-agents». Thèse de doctorat. Université de Droit, d'Economie et des Sciences d'Aix-Marseille III. 2001

[29]Wooldridge M., and al., «A Methodology for Agent-Oriented Analysis and Design». Proc. 3rd Int. Conf. On Autonomous Agents (Agents99), Seatle, WA, 1999.

[30]Wooldridge M., and al., «The gaia methodology for agent-oriented analysis and design». Autonomous Agent and Multi-agent Systems, 3(3):285-312, 2000.

A Review of Modular Multiplication Methods and Respective Hardware Implementations

Nadia Nedjah
Department of Electronics Engineering and Telecommunications, Engineering Faculty,
State University of Rio de Janeiro, Rio de Janeiro, Brazil
nadia@eng.uerj.br, http://www.eng.uerj.br/~nadia

Luiza de Macedo Mourelle
Department of System Engineering and Computation, Engineering Faculty,
State University of Rio de Janeiro, Rio de Janeiro, Brazil
ldmm@eng.uerj.br, http://www.eng.uerj.br/~ldmm

Keywords: cryptography, encryption, modular multiplication, modular reduction.

Received: April 18, 2005

Generally speaking, public-key cryptographic systems consist of raising elements of some group such as GF(2n), Z/NZ or elliptic curves, to large powers and reducing the result modulo some given element. Such operation is often called modular exponentiation and is performed using modular multiplications repeatedly. The practicality of a given cryptographic system depends heavily on how fast modular exponentiations are performed. Consequently, it also depends on how efficiently modular multiplications are done as these are at the base of the computation. This problem has received much attention over the years. Software as well as hardware efficient implementation were proposed. However, the results are scattered through the literature. In this paper we survey most known and recent methods for efficient modular multiplication, investigating and examining their strengths and weaknesses. For each method presented, we provide an adequate hardware implementation.

Povzetek: Podan je pregled modernih metod kriptografije.

1 Introduction

Electronic communication is growing exponentially so should be the care for information security issues [10]. Data exchanged over public computer networks must be authenticated, kept confidential and its integrity protected against alteration. In order to run successfully, electronic businesses require secure payment channels and digital valid signatures. Cryptography provides a solution to all these problems and many others [17].

One of the main objectives of cryptography consists of providing *confidentiality*, which is a service used to keep secret publicly available information from all but those authorized to access it. There exist many ways to providing secrecy. They range from physical protection to mathematical solutions, which render the data unintelligible. The latter uses *encryption/decryption* methods [10], [17], [30], [31].

The modular exponentiation is a common operation for scrambling and is used by several public-key cryptosystems, such Deffie and Hellman [8], [9] and the Rivest, Shamir and Adleman encryption schemes [34], as encryption/decryption method. RSA cryptosystem consists of a set of three items: a *modulus* M of around 1024 bits and two integers D and E called *private* and *public* keys that satisfy the property $T^{DE} \equiv T \mod M$. Plain text T obeying $0 \leq T < M$. Messages are encrypted using the public key as $C = T^E \mod M$ and uniquely decrypted as $T = C^D \mod M$. So the same operation is used to perform both processes: encryption and decryption. The modulus M is chosen to be the product of two large prime numbers, say P and Q. The public key E is generally small and contains only few bits set (i.e. bits = 1), so that the encryption step is relatively fast. The private key D has as many bits as the modulus M and is chosen so that $DE = 1 \mod (P-1)(Q-1)$. The system is secure as it is computationally hard to discover P and Q. It has been proved that it is impossible to break an RSA cryptosystem with a modulus of 1024-bit or more.

The modular exponentiation applies modular multiplication repeatedly. So the performance of public-key cryptosystems is primarily determined by the implementation efficiency of the modular multiplication and exponentiation. As the operands (the plaintext or the cipher text or possibly a partially ciphered text) are usually large (i.e. 1024 bits or more), and in order to improve time requirements of the encryption/decryption operations, it is essential to attempt to minimize the number of modular multiplications performed and to reduce the time required by a single modular multiplication.

Modular multiplication $A \times B \mod M$ can be performed in two different ways: multiplying, i.e. computing $P = A \times B$; then reducing, i.e. $R = P \mod M$ or

interleave the multiplication and the reduction steps. There are various algorithms that implement modular multiplication. The most prominent are Karatsuba-Ofman's [12] and Booth's [3] methods for multiplying, Barrett's [2], [6], [7] method for reducing, and Montgomery's algorithms [18], and Brickell's method [4], [37] for interleaving multiplication and reduction.

Throughout this paper, we will consider each one of the methods cited in the previous paragraph. The review will be organised as follows: First we describe, in Section 2, Karatsuba-Ofman's and Booth's methods for multiplying. Later, in Section 3, we present Barrett's method for reducing an operand modulo a given modulus. Then we detail Montgomery's algorithms for interleaving multiplication and reduction, in Section 4.

2 Efficient Multiplication Methods

The multiply-then-reduce methods consist of first computing the product then reducing it with respect to the given modulus. This method is generally preferred as there are very fast on-the-shelf multiplication algorithms as they were over studied [3], [12], [33]. The nowadays most popular multiplication methods that are suitable for hardware implementation are Karatsuba-Ofman's method and Booth's method.

2.1 Karatsuba-Ofman Method

Karatsuba-Ofman's algorithm is considered one of the fastest ways to multiply long integers. Generalizations of this algorithm were shown to be even faster than Schönhage-Strassen's FFT method [35], [36]. Karatsuba-Ofman's algorithm is based on a divide-and-conquer strategy. A multiplication of a $2n$-digit integer is reduced to two n-digits multiplications, one $(n+1)$-digits multiplication, two n-digits subtractions, two left-shift operations, two n-digits additions and two $2n$-digits additions.

Even though this algorithm was proposed long ago and as far as we know, there is no published hardware implementation for this algorithm. In contrast with the work presented in this paper, and after an extensive paper research, we only found publications on hardware implementations of Karatsuba-Ofman's algorithm adapted to multiplication in the Galois fields [13], [32]. Unlike in our implementation, the addition (mod 2) of two bits in these implementations delivers a single bit using a XOR gate In contrast with these, our implementation cares about the carryout bit, as it is necessary to obtaining the product. It is unnecessary to emphasize that this makes the designer face a completely different problem as explained later on.

The hardware specification is expressed using the most popular hardware description language VHDL [20]. Note that VHDL does not provide a recursive feature to implement recursive computation [1], [27], [28]. The proposed model exploits the *generate* feature to yield the recursive hardware model.

This subsection is organized as follows: First, we describe the Karatsuba-Ofman's algorithm and sketch its complexity. Then, we adapt the algorithm so that it can be implemented efficiently. Subsequently, we propose a recursive and efficient architecture of the hardware multiplier for Karatsuba-Ofman's algorithm. After that, we implement the proposed hardware using the *Xilinx*™ project manager and present some figures concerning time and space requirements of the obtained multiplier. We then compare our hardware with a *Synopsis*™ library multiplier and two other multipliers that implement Booth's multiplication algorithm.

2.1.1 Karatsuba-Ofman's Algorithm

We now describe the details of Karatsuba-Ofman's multiplication algorithm [12], [27], [36]. Let X and Y be the binary representation of two long integers:

$$X = \sum_{i=0}^{k-1} x_i 2^i \quad \text{and} \quad Y = \sum_{i=0}^{k-1} y_i 2^i$$

We wish to compute the product XY. The operands X and Y can be decomposed into to equal-size parts X_H and X_L, Y_H and Y_L respectively, which represent the n higher order bits and lower order bits of X and Y. Let $k = 2n$. If k is odd, it can be right-padded with a zero.

$$X = 2^n\left(\sum_{i=0}^{n-1} x_{i+n} 2^i\right) + \sum_{i=0}^{n-1} x_i 2^i = X_H 2^n + X_L$$

$$Y = 2^n\left(\sum_{i=0}^{n-1} y_{i+n} 2^i\right) + \sum_{i=0}^{n-1} y_i 2^i = Y_H 2^n + Y_L$$

So the product $P = XY$ can be computed as follows:

$$\begin{aligned} P &= XY \\ &= (X_H 2^n + X_L)(Y_H 2^n + Y_L) \\ &= 2^{2n}(X_H Y_H) + 2^n(X_H Y_L + X_L Y_H) + X_L Y_L \end{aligned}$$

Using the equation above, it needs 4 n-bits multiplications to compute the product P. The standard multiplication algorithm is based on that equation. So assuming that a multiplication of k-bits operands is performed using $T(k)$ one-bit operations, we can formulate that $T(k) = T(n) + \delta k$, wherein δk is a number of one-bit operations to compute all the additions and shift operations. Considering that $T(1) = 1$, we find that the standard multiplication algorithm requires:

$$T(k) = \left(k^{\log_2 4}\right) = \left(k^2\right)$$

The computation of P can be improved by noticing the following:

$$X_H Y_L + X_L Y_H = (X_H + X_L)(Y_H + Y_L) - X_H Y_H - X_L Y_L$$

The Karatsuba-Ofman's algorithm is based on the above observation and so the $2n$-bits multiplication can be reduced to three n-bits multiplications, namely $X_H Y_H$, $X_L Y_L$ and $(X_H + X_L)(Y_H + Y_L)$. The Karatsuba-Ofman's multiplication method can then be expressed as in the algorithm in Figure 1. wherein function *Size*(X) returns the number of bits of X, function *High*(X) returns the higher half part of X, function *Low*(X) returns the lower half of X, *RightShift*(X, n) returns $X2^n$ and

```
Algorithm KaratsubaOfman(X, Y)
   If (Size(X) = 1) Then KaratsubaOfman= OneBitMultiplier(X, Y)
   Else Product1 := KaratsubaOfman(High(X), High(Y));
        Product2 := KaratsubaOfman(Low(X), Low(Y));
        Product3 := KaratsubaOfman(High(X)+Low(X), High(Y)+Low(Y));
        KaratsubaOfman := RightShift(Product1, Size(X)) +
                          RightShift(Product3-Product1-Product2,     Size(X)/2)     +
Product2;
End KaratsubaOfman.
```

Figure 1: Karatsuba-Ofman recursive multiplication algorithm

OneBitMultiplication(*X*, *Y*) returns *XY* when both X and Y are formed by a single bit. If *Size*(*X*) is odd, then *High*(*X*) and *Low*(*X*) right-pad *X* with a zero before extracting the high and the low half respectively.
The algorithm above requires 3 *n*-bits multiplications to compute the product *P*. So we can stipulate that:

$$T(k) = 2T(n) + T(n+1) + \delta' k \approx 3T(n) + \delta' k$$

wherein $\delta' n$ is a number of one-bit operations to compute all the additions, subtractions and shift operations. Considering that $T(1) = 1$, we find that the Karatsuba-Ofman's algorithm requires:

$$T(k) \approx \left(k^{\log_2 3}\right) = \left(k^{1.58}\right),$$

and so is asymptotically faster than the standard multiplication algorithm.

2.1.1 Adapted Karatsuba's Algorithm

We now modify Karatsuba-Ofman's algorithm of Figure 1 so that the third multiplication is performed efficiently.

For this, consider the arguments of the third recursive call, which computes $Product_3$. They have *Size*(*X*)/2+1 bits. Let *Z* and *U* be these arguments left-padded with *Size*(*X*)/2-1 0-bits. So now *Z* and *U* have *Size*(*X*) bits. So we can write the product $Product_3$ as follows, wherein *Size*(*X*) = 2*n*, Z_H and U_H are the high parts of *Z* and *U* respectively and Z_L and U_L are the low parts of *Z* and *U* respectively. Note that Z_H and U_H may be equal to 0 or 1.

$$\begin{aligned} \text{Product}_3 &= ZU \\ &= (Z_H 2^n + Z_L)(U_H 2^n + U_L) \\ &= 2^{2n}(Z_H U_H) + 2^n(Z_H U_L + Z_L U_H) + Z_L Y_L \end{aligned}$$

Depending on the value of Z_H and U_H, the above expression can be obtained using one of the alternatives of Table 1.
As it is clear from Table 1, computing the third product requires one multiplication of size *n* and some extra adding, shifting and multiplexing operations. So we adapt Karatsuba-Ofman's algorithm of Figure 1 to this modification as shown in the algorithm of Figure 2.

Z_H	U_H	$Product_3$
0	0	$Z_L Y_L$
0	1	$2^n Z_L + Z_L Y_L$
1	0	$2^n U_L + Z_L Y_L$
1	1	$2^{2n} + 2^n(U_L + Z_L) + Z_L Y_L$

Table 1: computing the third product2.1.3 Recursive Hardware Architecture

In this section, we concentrate on explaining the proposed architecture of the hardware.

The component *KaratsubaOfman* implements the

```
Algorithm AdaptedKaratsubaOfman(X, Y)
   If (Size(X) = 1) Then KaratsubaOfman := OneBitMultiplier(X, Y)
   Else Product1 :=  KaratsubaOfman(High(X), High(Y));
        Product2 := KaratsubaOfman(Low(X), Low(Y));
        P := KaratsubaOfman(Low(High(X)+Low(X)), Low(High(Y)+Low(Y)));
        If Msb(High(X)+Low(X)) = 1 Then A :=  Low(High(Y)+Low(Y)) Else A := 0;
        If Msb(High(Y)+Low(Y)) = 1 Then B :=  Low(High(X)+Low(X)) Else B := 0;
        Product3 :=  LeftShift(Msb(High(X)+Low(X))•Msb(High(X)+Low(X)), Size(X)) +
                     LeftShift(A + B, Size(X)/2) + P;
        KaratsubaOfman =  LeftShift(Product1, Size(X)) +
                             LeftShift(Product3-Product1-Product2,     Size(X)/2)     +
Product2;
End AdaptedKaratsubaOfman.
```

Figure 2: Adapted Karatsuba-Ofman's algorithm

algorithm of Figure 2. Its interface is given in Figure 3. The input ports are the multiplier *X* and the multiplicand *Y* and the single output port is the product *XY*. It is clear that the multiplication of 2 *n*-bit operands yields a product of 2*n*-bits product.

The VHDL recursive specification of the component architecture is given in the concise code of Figure 4. The architecture details of the component *KaratsubaOfman* are given in Figure 5.

```
Entity KaratsubaOfman is
   Generic(
        n: positive
   );
   Port(
        X:  In  bit_vector (Size-1 To 0);
        Y:  In  bit_vector (Size-1 To 0);
       XY: Out  bit_vector(2*Size-1 To 0)
   );
End KaratsubaOfman;
```

Figure 3: Interface of component KaratsubaOfman

The signals SX_L and SY_L are the two *n*-bits results of the additions $X_H + X_Y$ and $Y_H + Y_L$ respectively. The two one-bit carryout of these additions are represented in Figure 5 by *CX* and *CY* respectively.

The component *ShiftnAdd* (in Figure 5) first computes the sum *S* as $SX_L + SY_L$, SX_L, SY_L, or 0 depending on the values of *CX* and *CY* (see also Table 1). Then computes $Product_3$ as depicted in Figure 6, wherein *T* represents $CX \times CY$.

The computation implemented by component *ShiftSubnAdd* (in Figure 5) i.e. the computation specified in the last line of the Karatsuba-Ofman algorithm in Figure 1 and Figure 2 can be performed efficiently if the execution order of the operations constituting it is chosen carefully. This is shown in the architecture of Figure 7.

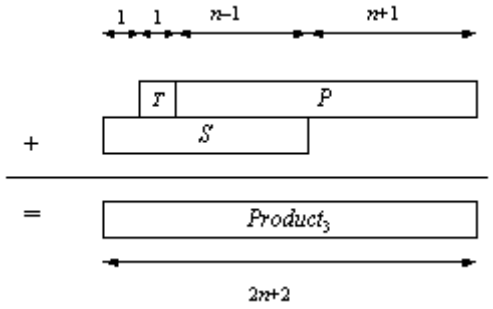


Figure 6: Operation performed by the $ShiftnAdder_{2n}$

Component *ShiftSubnAdd* proceeds as follows: first computes $R = Product_1 + Product_2$; then obtains *2CR*, which is the two's complement of *R*; subsequently, computes $U = Product_3 + 2CR$; finally, as the bits of $Product_1$ and *U* must be shifted to the left 2*n* times and *n* times respectively, the component reduces the first and last additions as well as the shift operations in the last line computation of Karatsuba-Ofman's algorithm (see Figure 1 and Figure 2) to a unique addition that is depicted in Figure 8.

```
Architecture RecursiveArchitecture of KaratsubaOfman is
   -- declaration part including components and temporary signals
Begin
   Termination: If k = 1 Generate
      TCell: OneBitMultiplier Generic Map(n) Port Map(X(0), Y(0), XY(0) );
   End Generate Termination;
   Recursion: If k /= 1 Generate
     ADD1: Adder Generic Map(k/2) Port Map(X(k/2-1 Downto 0), X(k-1 Downto k/2), SumX
);
     ADD2: Adder Generic Map(k/2) Port Map(Y(k/2-1 Downto 0), Y(k-1 Downto k/2), SumY
);
     KO1: KaratsubaOfman Generic Map(k/2)
                  Port Map(X(k-1 Downto k/2),Y(k-1 Downto k/2),Product1);
     KO2: KaratsubaOfman Generic  Map(k/2)
                  Port Map(X(k/2-1 Downto 0),Y(k/2-1 Downto 0),Product2);
     KO3: KaratsubaOfman Generic  Map(k/2)
                  Port Map(SumX(k/2-1 Downto 0), SumY(k/2-1 Downto 0), P);
     SA:  ShiftnAdder Generic Map(k)
          Port  Map(SumX(k/2),SY(n/2),  SX(k/2-1  Downto  0),  SY(k/2-1  Downto  0),
P,Product3);
     SSA: ShifterSubnAdder Generic Map(k) Port Map(  Product1, Product2, Product3, XY
);
   End Generate Recursion;
End RecursiveArchitecture;
```

Figure 4: Recursive architecture of the component KaratsubaOfman of size *n*

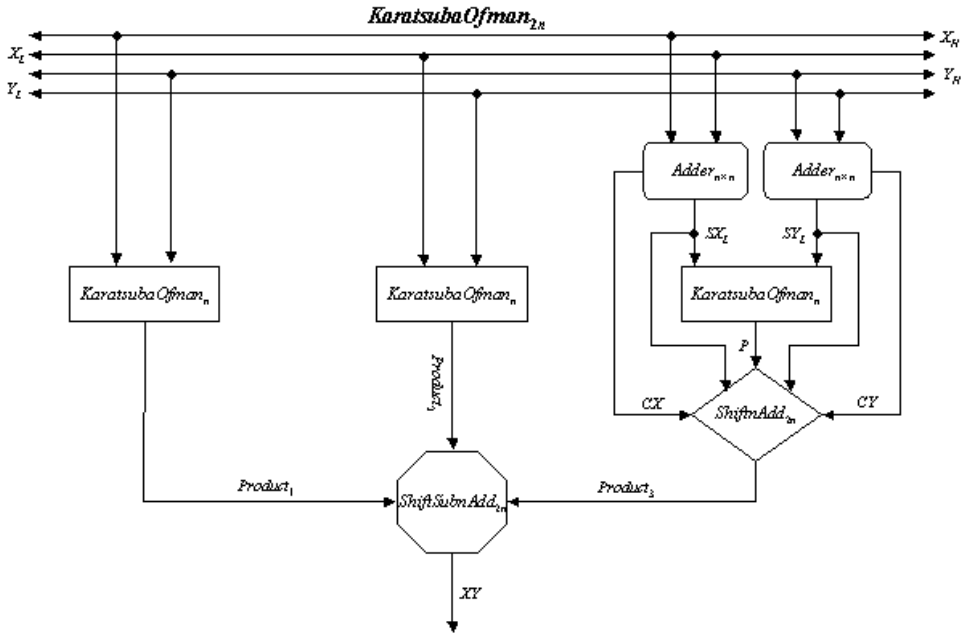


Figure 5: Macro view of KaratsubaOfman$_{2n}$ in terms of KaratsubaOfman of size 2n

2.2 Booth's Multiplication Method

Algorithms that formalize the operation of multiplication generally consist of two steps: one generates a partial product and the other accumulates it with the previous partial products. The most basic algorithm for multiplication is based on the add-and-shift method: the shift operation generates the partial products while the add step sums them up [3].

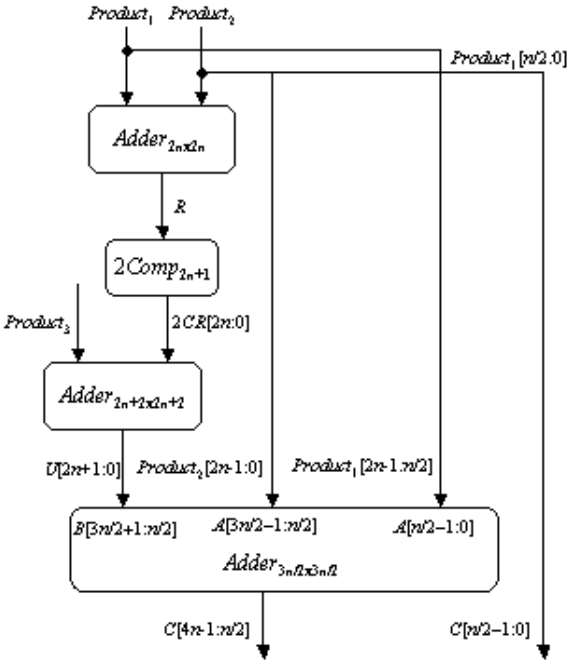


Figure 7: Architecture of ShiftSubnAdder$_{2n}$

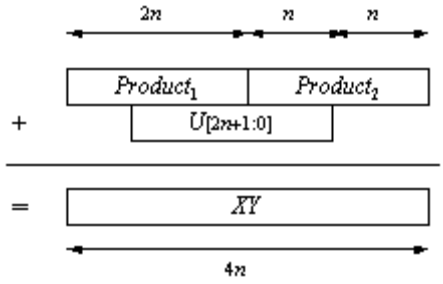


Figure 8: Last addition performed by ShiftSubnAdder$_2$

The straightforward way to implement a multiplication is based on an iterative adder-accumulator for the generated partial products as depicted in Figure 9. However, this solution is quite slow as the final result is only available after *n* clock cycles, *n* is the size of the operands.

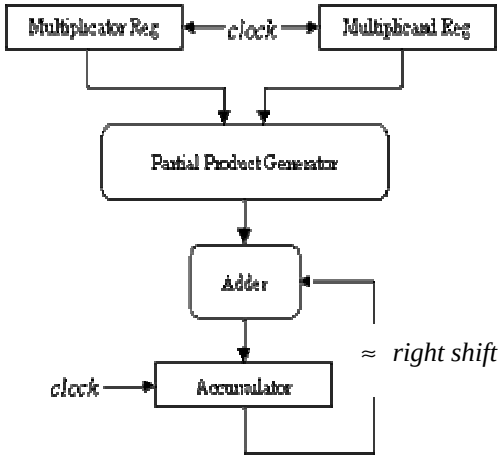


Figure 9: Iterative multiplier

A faster version of the iterative multiplier should add several partial products at once. This could be achieved

by *unfolding* the iterative multiplier and yielding a combinatorial circuit that consists of several partial product generators together with several adders that operate in parallel. In this paper, we use such a parallel multiplier as described in Figure 10. Now, we detail the algorithms used to compute the partial products and to sum them up.

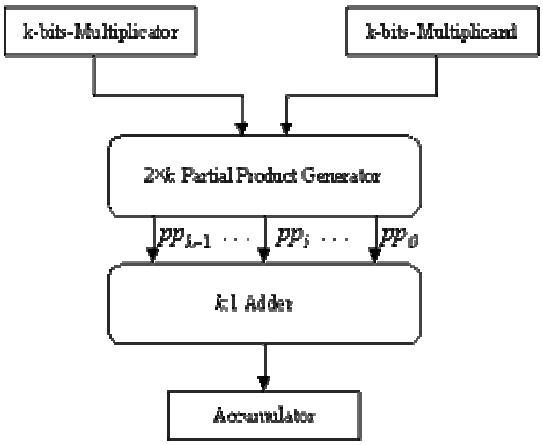


Figure 10: Parallel multiplier.

2.2.1 Booth's Algorithm

Now, we concentrate on the algorithm used to compute partial products as well as reducing the corresponding number without deteriorating the space and time requirement of the multiplier.

Let X and Y be the multiplicand and multiplicator respectively and let n and m be their respective sizes. So, we denote X and Y as follows:

$$X = \sum_{i=0}^{n} x_i \times 2^i \quad \text{and} \quad Y = \sum_{i=0}^{m} y_i \times 2^i$$

$$\Rightarrow X \times Y = \sum_{i=0}^{n} x_i \times Y \times 2^i$$

Inspired by the above notation of X, Y and that of $X \times Y$, the add-and-shift method [2], [3] generates n partial products: $x_i \times Y$, $0 \le i < n$. Each partial product obtained is shifted left or right depending on whether the starting bit was the less or the most significant and added up. The number of partial products generated is bound above by the size (i.e. number of bits) of the multiplier operand. In cryptosystems, operands are quite large as they represent blocks of text (i.e. ≥ 1024 bits).

Another notation of X and Y allows to halve the number of partial products without much increase in space requirements. Consider the following notation of X and $X \times Y$:

$$X = \sum_{i=0}^{\lceil (n+1)/2 \rceil - 1} \widetilde{x}_i \times 2^{2 \times i}, \text{ where } \widetilde{x}_i = x_{2\times i-1} + x_{2\times i} - 2 \times x_{2\times i+1}$$

$$\text{and } \widetilde{x}_{-1} = \widetilde{x}_n = \widetilde{x}_{n+1} = 0$$

$$X \times Y = \sum_{i=0}^{\lceil (n+1)/2 \rceil - 1} \widetilde{x}_i \times Y \times 2^{2 \times i}$$

The possible values of $\widetilde{x}_i$ with the respective values of $x_{2\times i+1}$, $x_{2\times i}$, and $x_{2\times i-1}$ are −2 (100), −1 (101, 110), 0 (000, 111), 1 (001, 010) and 2(011). Using this recoding will generate $\lceil (n+1)/2 \rceil - 1$ partial products.

Inspired by the above notation, the modified Booth algorithm [3], [12] generates the partial products $\widetilde{x}_i \times Y$. These partial products can be computed very efficiently due to the digits of the new representation $\widetilde{x}_i$. The hardware implementation will be detailed in Section 3.

In the algorithm of Figure 11, the terms $4 \times 2^{n+1}$ and $3 \times 2^{n+1}$ are supplied to avoid working with negative numbers. The sum of these additional terms is congruent to zero modulo $2^{n+\lceil (n+1) \rceil - 1}$. So, once the sum of the partial products is obtained, the rest of this sum in the division by $2^{n+\lceil (n+1) \rceil - 1}$ is finally the result of the multiplication $X \times Y$.

The partial product generator is composed of k *Booth recoders* [3], [6]. They communicate directly with k partial product generators as shown in Figure 12.

```
Algorithm Booth(x_{2xi-1}, x_{2xi}, x_{2xi+1}, Y)
   Int product := 0;
   Int pp[⌈(n+1)/2⌉ −1];
   pp[0] := (x̃_0 × Y + 4×2^{n+1}) × 2^{2×i} ;
   For i = 0 To ⌈(n+1)/2⌉ −1 Do
      pp[i] := (x̃_i × Y + 3×2^{n+1}) × 2^{2×i} ;
      product := product + pp[i];
   Return product mod 2^{n+⌈(n+1)⌉ −1};
End Booth
```

Figure 11: Multiplication algorithm

The required partial products, i.e. $\widetilde{x}_i \times Y$ are *easy* multiple. They can be obtained by a simple shift. The negative multiples in 2's complement form, can be obtained form the positive corresponding number using a bit by bit complement with a 1 added at the least significant bit of the partial product. The additional terms introduced in the previous section can be included into the partial product generated as three/two/one most significant bits computed as follows, whereby, ++ is the bits *concatenation* operation, $\langle A \rangle$ is the binary notation of integer A, 0^i is a run of i zeros and $B_{[n:0]}$ is the selection of the n less significant bits of the binary representation B.

$$\langle pp_0 \rangle = \overline{s_0} s_0 s_0 ++ \langle |\widetilde{x}_0| \times Y \oplus s_0 \rangle + s_0$$

$$\langle pp_{2\times j} \rangle = (1 \overline{s_{2\times j}} ++ \langle |\widetilde{x}_{2\times j}| \times Y \oplus s_{2\times j} \rangle + s_{2\times j}) ++ 0^{2\times j}$$

for $1 \le j < k-1$ and for $j = k - 1 = k'$, we have:

$$\langle pp_{2\times k'} \rangle = (\overline{s_{2\times k'}} ++ \langle |\widetilde{x}_{2\times k'}| \times Y \oplus s_{2\times k'} \rangle + s_{2\times k'}) ++ 0^{2\times k'}$$

$$\langle pp_{2\times k} \rangle = \langle |\widetilde{x}_{2\times k}| \times Y \rangle_{[n:0]} ++ 0^{2\times k}$$

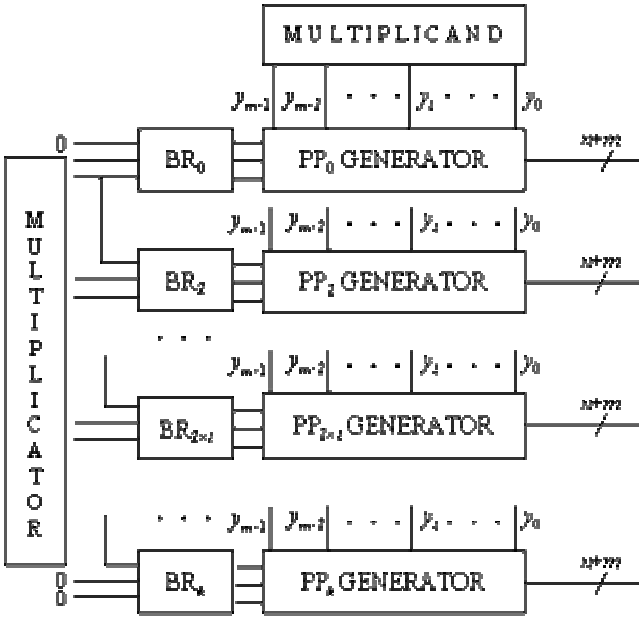


Figure 12: The partial product generator architecture.

The Booth selection logic circuitry used, denoted by BR_i for $0 \le i \le k$ in Figure 12, is very simple. The cell is described in Figure 13. The inputs are the three bits forming the Booth digit and outputs are three bits: the first one *SY* is set when the partial product to be generated is Y or $-Y$, the second one *S2Y* is set when the partial product to be generated is $2\times Y$ or $-2\times Y$, the last bit is the simply the last bit of the Booth digit given as input. It allows us to complement the bits of the partial products when a negative multiple is needed.

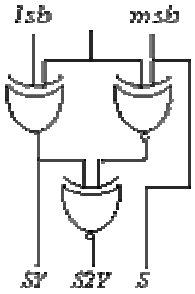


Figure 13: Booth recoder selection logic.

The circuitry of the partial generator denoted by PP_i *Generator*, is given in Figure 14.

In order to implement the adder of the generated partial products, we use a hybrid new kind of adder. It consists cascade of intercalated stages of *carry save adders* and *delayed carry adders*.

2.3 Multipliers Area/Time Requirements

The entire design was done using the *Xilinx*™ Project Manager (version Build 6.00.09) [40] through the steps of the Xilinx design cycle shown in Figure 15.

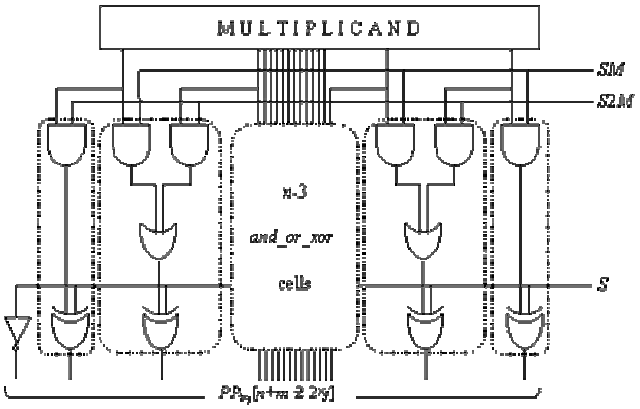


Figure 14: The partial product generator.

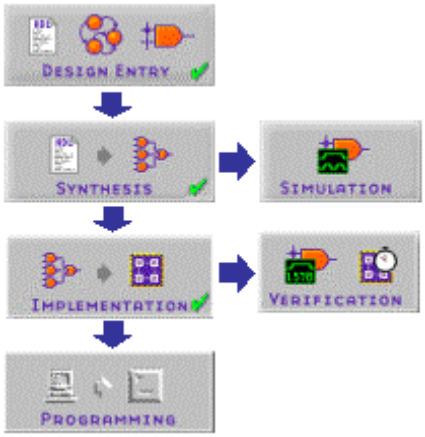


Figure 15: Design cycle.

The design was elaborated using VHDL [20]. The *synthesis* step generates an optimized netlist that is the mapping of the gate-level design into the Xilinx format: *XNF*. Then, the *simulation* step consists of verifying the functionality of the elaborated design. The *implementation* step consists of partitioning the design into logic blocks, then finding a near optimal placement of each block and finally selecting the interconnect routing for a specific device family. This step generates a logic PE array file from which a bit stream can be obtained. The implementation step provides also the number of configurable logic blocks (CLBs). The *verification* step allows us to verify once again the functionality of the design and determine the response time of the design including all the delays of the physical net and padding. The *programming* step consists of loading the generated bit stream into the physical device.

The design was implemented into logic blocks using a specific device family, namely SPARTAN S05PC84-4.

As explained before, the Karatsuba's multiplier reduces to an ensemble of adders. These adders are implemented using ripple-carry adders, which can be very efficiently implemented into FPGAs as the carryout signal uses dedicated interconnects in the CLB and so there is no routing delays in the data path. An n-bit ripple-carry adder is implemented using $n/2+2$ CLBs and has a total fixed delay of $4.5+0.35n$ nanoseconds.

Table 2 shows the delay introduced and area required by the Karatsuba-Ofman multiplier (*KO*) together with those for a hardware implementation of the Booth multiplier which uses a Wallace tree for adding up the partial products (*BW*), another hardware implementation of Booth's algorithm that uses a redundant binary Booth encoding (*PRB*) and the *Synopsys*™ library multiplier (*DW02*) [11]. This is given for three different operand sizes. The delays are expressed in ns. These delays are represented graphically in Figure 16.

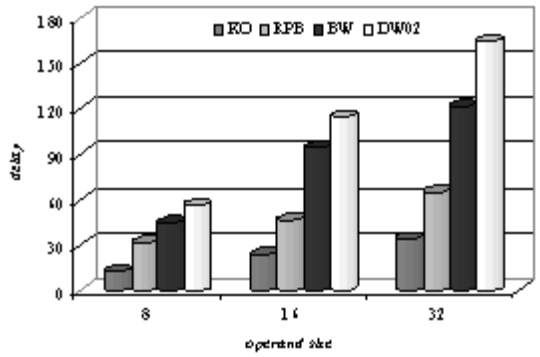


Figure 16: Representing time requirement

size	*KO*		*BW*		*PRB*		*DW02*	
	delay	*area*	*delay*	*area*	*delay*	*area*	*delay*	*area*
8	12.6	1297	44.6	1092	31.8	862	56.2	633
16	22.8	6300	93.9	5093	46.6	3955	114.9	2760
32	29.1	31740	121.5	20097	64.9	17151	164.5	11647

Table 2: Delays and areas for different multipliers

Table 2 also shows the area required by our multiplier compared with those needed for the implementation of *BW*, *PBR* and *DW02*. The areas are given in terms of total number of gates necessary for the implementation. These results are represented graphically in Figure 17.

It is clear from Figure 16 and Figure 17 that the engineered Karatsuba-Ofman multiplier works much faster than the other three multipliers. However, it consumes more hardware area. Nevertheless, the histogram of Figure 18, which represents the *area×time* factor for the four compared multipliers implementations, shows that proposed multiplier improves this product.

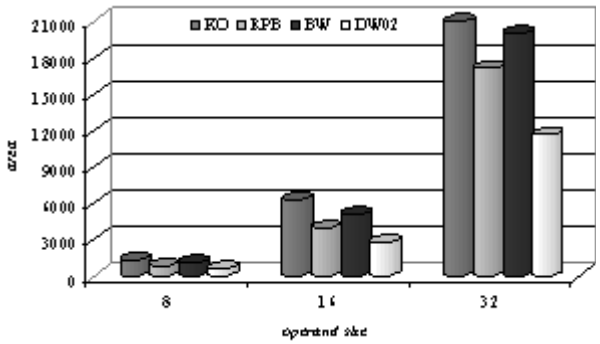


Figure 17: Representing space requirement

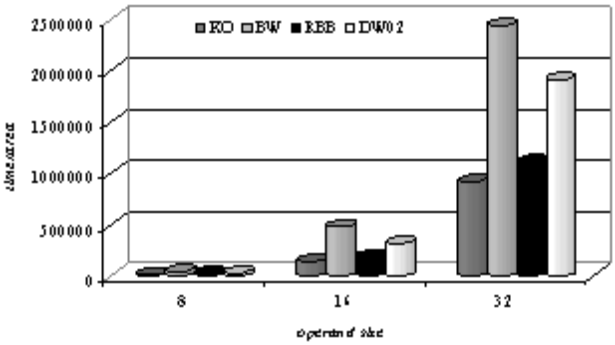


Figure 18: Representing area×time factor

So, our multiplier improves the *area×time* factor as well as time requirement while the other three improve area at the expense of both time requirement and the *area×time* factor. Moreover, we strongly think that for larger operands, the Karatsuba-Ofman multiplier will yield very much better characteristics, i.e. time and area requirements as it is clear from Figure 16, Figure 17 and Figure 18.

3 Barrett's Reduction Method

A modular reduction is simply the computation of the remainder of an integer division. It can be denoted by:

$$X \bmod M = X - \left\lfloor \frac{X}{M} \right\rfloor \times M$$

However, a division is very expensive even compared with a multiplication.

The naive sequential division algorithm successively shifts and subtracts the modulus until the remainder that is non-negative and smaller than the modulus is found. Note that after a subtraction, a negative remainder may be obtained. So in that case, the last non-negative remainder needs to be restored and so will be the expected remainder. This computation is described in the algorithm of Figure 19.

```
Algorithm NaiveReduction(P, M)
   Int R := P;
   Do R := R - M;
   While R > 0;
   If R ≠ 0 Then R := R + M;
   Return R;
End NaiveReduction
```

Figure 19: Naive reduction algorithm

In the context of this paper, P is the result of a product so it has at most $2n$ bits assuming that the operands have both n bits.

The computation performed in the naïve algorithm above is very inefficient as it may require 2^n-1 subtractions, 2^n comparisons and an extra addition. Instead of subtracting a single M one can subtract a

multiple of it at once. However, in order to yield multiples of M further computations, namely multiplications, need be performed, except for power of two multiples, i.e. 2^kM. These are simply M left-shifted k times, which can very cheaply implemented on hardware. This idea is described in the restoring division algorithm given in Figure 20. It attempts to subtract the biggest possible power of two multiple of M from the actual remainder. Whenever the result of that operation is negative it restores the previous remainder and repeats the computation for all possible power of two multiples of M, i.e. 2^nM, $2^{n-1}M$, …, $2M$, M.

```
Algorithm RestoringReduction(P, M)
   Int R0 := P;
   Int N := LeftShift(M, n);
   For i = 1 To n Do
      Ri := Ri-1 – N;
      If R < 0 Then Ri := Ri-1;
      N := RightShift(N);
   Return Ri;
End RestoringReduction
```

Figure 20: Restoring reduction algorithm

The computation performed in the restoring reduction algorithm requires n subtractions, n comparisons and some $2n$ shifting as well as some restoring operations. This is very much more efficient than the computation of the algorithm in Figure 19.

An alternative to the restoring reduction algorithm is the non-restoring one. The non-restoring reduction algorithm is given in Figure 21.

```
Algorithm NonRestoringReduction(P, M)
   Int R0 := P;
   Int N := LeftShift(M, n);
   For i = 1 To n Do
      If R > 0 Then Ri := Ri-1 – N;
      Else Ri := Ri-1 + N;
      N := RightShift(N);
   If Ri < 0 Then Ri := Ri-1 + N;
   Return Ri;
End RestoringReduction
```

Figure 21: Non-restoring reduction algorithm

It allows negative remainder. When the remainder is non-negative it sums it up with the actual power of two multiple of M. Otherwise, it subtracts that multiple of M from it. It keeps doing so repeatedly for all possible power of two multiples of M, i.e. 2^nM, $2^{n-1}M$, …, $2M$, M. The non-restoring reduction computation requires a final restoration that adds M to the obtained remainder when the latter is negative.

Using Barrett's method [2], [6], we can estimate the remainder using two simple multiplications. The approximation of the quotient is calculated as follows:

$$\left\lfloor \frac{X}{M} \right\rfloor = \left\lfloor \frac{\left\lfloor \frac{X}{2^{n-1}} \right\rfloor \times \left\lfloor \frac{2^{n-1} \times 2^{n+1}}{M} \right\rfloor}{2^{n+1}} \right\rfloor = \left\lfloor \frac{\left\lfloor \frac{X}{2^{n-1}} \right\rfloor \times \left\lfloor \frac{2^{2\times n}}{M} \right\rfloor}{2^{n+1}} \right\rfloor$$

The equation above can be calculated very efficiently as division by a power of two 2^x are simply a truncation of the operand' x-least significant digits. The term $\lfloor 2^{2\times n}/M \rfloor$ depends only on M and so is constant for a given modulus. So, it can be pre-computed and saved in an extra register. Hence the approximation of the remainder using Barrett's method [2], [6] is a positive integer smaller than $2\times(M-1)$. So, one or two subtractions of M might be required to yield the exact remainder (see Figure 22).

4 Booth-Barrett's Method

In this section, we outline the architecture of the multiplier, which is depicted in Figure 4. Later on in this section and for each of the main parts of this architecture, we give the detailed circuitry, i.e. that of the *partial product generator*, *adder* and *reducer*.

The multiplier of Figure 4 performs the modular multiplication $X\times Y$ mod M in three main steps:

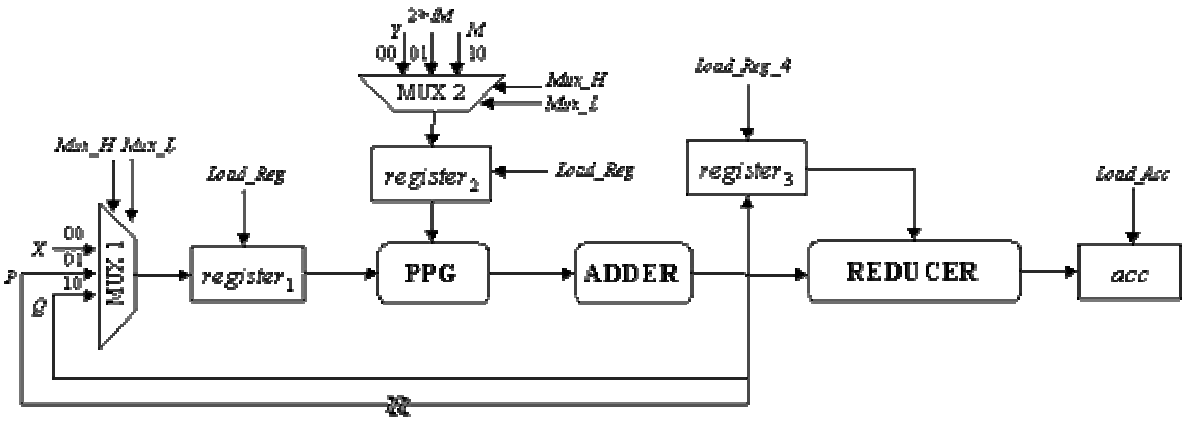


Figure 22: The modular multiplier architecture

1. Computing the product $P = X \times Y$;
2. Computing the estimate quotient $Q = P/M$
   $\Rightarrow \quad Q \cong P/2^{n-1} \times \lfloor 2^{2\times n}/M \rfloor$;
3. Computing the final result $P - Q \times M$.

During the first step, the modular multiplier first loads $register_1$ and $register_2$ with X and Y respectively; then waits for *PPG* to yield the partial products and finally waits for the *ADDER* to sum all of them. During the second step, the modular multiplier loads $register_1$, $register_2$ and $register_3$ with the obtained product P, the pre-computed constant $\lfloor 2^{2\times n}/M \rfloor$ and P respectively; then waits for *PPG* to yield the partial products and finally waits for the *ADDER* to sum all of them. During the third step, the modular multiplier first loads $register_1$ and $register_2$ with the obtained product Q and the modulus M respectively; then awaits for *PPG* to generate the partial products, then waits for the *ADDER* to provide the sum of these partial products and finally waits for the *REDUCER* to calculate the final result $P–Q\times M$, which is subsequently loaded in the accumulator *acc*.

4.1 The Montgomery Algorithm

Algorithms that formalize the operation of modular multiplication generally consist of two steps: one generates the product $P = A\times B$ and the other reduces this product P modulo M.

The straightforward way to implement a multiplication is based on an iterative adder-accumulator for the generated partial products. However, this solution is quite slow as the final result is only available after n clock cycles, n is the size of the operands [19].

A faster version of the iterative multiplier should add several partial products at once. This could be achieved by *unfolding* the iterative multiplier and yielding a combinatorial circuit that consists of several partial product generators together with several adders that operate in parallel [15], [16].

One of the widely used algorithms for efficient modular multiplication is the Montgomery's algorithm [18]. This algorithm computes the product of two integers modulo a third one without performing division by M. It yields the reduced product using a series of additions

Let A, B and M be the multiplicand and multiplier and the modulus respectively and let n be the number of digit in their binary representation, i.e. the radix is 2. So, we denote A, B and M as follows:

$$A=\sum_{i=0}^{n-1} a_i \times 2^i, \quad B=\sum_{i=0}^{n-1} b_i \times 2^i \text{ and } M=\sum_{i=0}^{n-1} m_i \times 2^i$$

The pre-conditions of the Montgomery algorithm are as follows:

The modulus M needs to be relatively prime to the *radix*, i.e. there exists no common divisor for M and the radix;

The multiplicand and the multiplicator need to be smaller than M.

As we use the binary representation of the operands, then the modulus M needs to be odd to satisfy the first pre-condition.

The Montgomery algorithm uses the least significant digit of the accumulating *modular partial product* to determine the multiple of M to subtract. The usual multiplication order is reversed by choosing multiplier digits from least to most significant and shifting down. If R is the current modular partial product, then q is chosen so that $R+q\times M$ is a multiple of the radix r, and this is right-shifted by r positions, i.e. divided by r for use in the next iteration. So, after n iterations, the result obtained is $R = A\times B\times r^{-n} \bmod M$ [14]. A modified version of Montgomery algorithm is given in Figure 23.

```
algorithm Montgomery(A, B, M)
      int R = 0;
      1: for i= 0 to n-1
      2:    R = R + a_i×B;
      3:    if r_0 = 0 then
      4:       R = R div 2
      5:    else
      6:       R = (R + M) div 2;
      return R;
end Montgomery.
```

Figure 23: Montgomery modular algorithm.

In order to yield the right result, we need an extra Montgomery modular multiplication by the constant $2^n \bmod M$. However as the main objective of the use of Montgomery modular multiplication algorithm is to compute exponentiations, it is preferable to Montgomery pre-multiply the operands by 2^{2n} and Montgomery post-multiply the result by 1 to get rid of the 2^{-n} factor. Here we concentrate on the implementation of the Montgomery multiplication algorithm of Figure 23.

In order to yield the right result, we need an extra Montgomery modular multiplication by the constant $r^{2n} \bmod M$. As we use binary representation of numbers, we compute the final result using the algorithm of Figure 24.

```
algorithm ModularMult(A, B, M, n)
   const C := 2^n mod M;
   int R := 0;
   R := Montgomery(A, B, M);
   return Montgomery(R, C, M);
end ModularMult.
```

Figure 24: Modular multiplication algorithm

4.2 Iterative Montgomery Architecture

In this section, we outline the architecture of the Montgomery modular multiplier. The interface of the Montgomery modular multiplier is given in Figure 25. It expects the operands A, B and M and it computes $R = (A\times B\times 2^{-n}) \bmod M$.

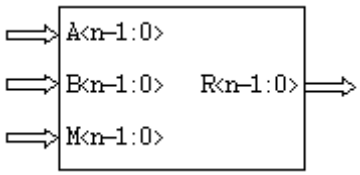


Figure 25: Montgomery multiplier interface

The detailed architecture of the Montgomery modular multiplier is given in Figure 26. It uses two multiplexers, two adders, two shift registers, three registers and a controller. The latter will be described in the next section.

The first multiplexer of the proposed architecture, i.e. MUX2_1 passes 0 or the content of register B depending on whether bit a_0 indicates 0 or 1 respectively. The second multiplexer, i.e. MUX2_2 passes 0 or the content of register M depending on whether bit r_0 indicates 0 or 1 respectively. The first adder, i.e. ADDER$_1$, delivers the sum $R + a_i \times B$ (line 2 of algorithm of Fig. 1), and the second adder, i.e. ADDER$_2$, yields the sum $R + M$ (line 6 of the same algorithm). The shift register SHIFT REGISTER$_1$ provides the bit a_i. In each iteration i of the multiplier, this shift register is right-shifted once so that a_0 contains a_i.

The role of the controller consists of synchronizing the shifting and loading operations of the SHIFT REGISTER$_1$ and SHIFT REGISTER$_2$. It also controls the number of iterations that have to be performed by the multiplier. For this end, the controller uses a simple down counter. The counter is inherent to the controller. The interface of the controller is given in Figure 27.

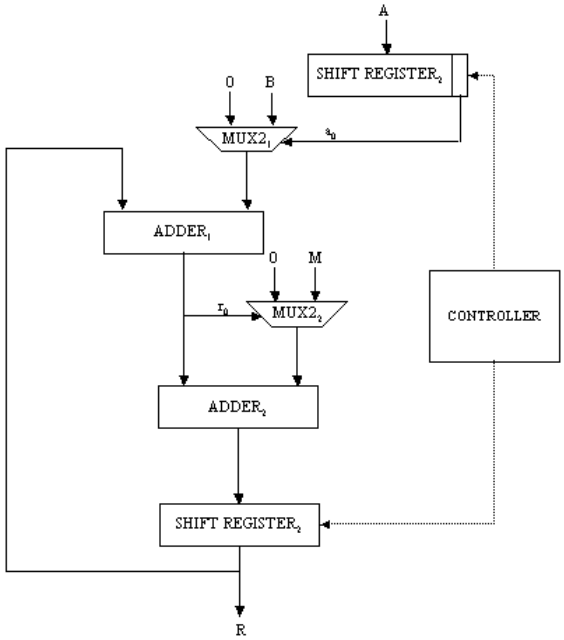


Figure 26: Montgomery multiplier architecture

Figure 27: Interface of the Montgomery controller

In order to synchronize the work of the components of the architecture, the controller consists of a state machine, which has 6 states defined as follows:

- S_0: Initialize of the state machine;
  Go to S_1;
- S_1: Load multiplicand and modulus into the corresponding registers;
  Load multiplier into *shift register*$_1$;
  Go to S_2;
- S_2: Wait for *ADDER*$_1$;
  Wait for *ADDER*$_2$;
  Load multiplier into *shift register*$_2$;
  Increment counter;
  Go to S_3;
- S_3: Enable shift register$_2$;
  Enable shift register$_1$;
- S_4: Check the counter;
  If 0 then go to S_5 else go to S_2;
- S_5: Halt;

4.3 Modular Multiplier Architecture

The modular multiplier yields the actual value of $A \times B \bmod M$. It first computes $R = A \times B \times 2^{-n} \bmod M$ using the Montgomery modular multiplier. Then, it computes $R \times C \bmod M$, where $C = 2^n \bmod M$. The modular multiplier interface is shown in Figure 28.

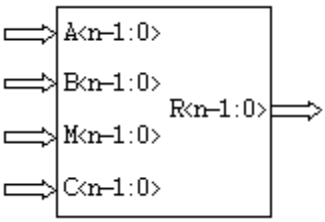


Figure 28: The modular multiplier interface

The modular multiplier uses a 4-to-1 multiplexer MUX4 and a register REGISTER.

- Step 0: Multiplexer MUX4 passes 0 or B. MUX2 passes A. It yields $R_1 = A \times B \times 2^{-n} \bmod M$. The register denoted by REGISTER contains 0.
- Step 1: Multiplexer MUX4 passes 0 or R. MUX2 passes C. It yields $R = R_1 \times C \bmod M$. The register denoted by REGISTER contains the result of the first step computation, i.e. $R = A \times B \times 2^{-n} \bmod M$.

The modular multiplier architecture is given in Figure 29. In order to synchronize the actions of the components of the modular multiplier, the architecture uses a controller, which consists of a state machine of 10 states. The interface of CONTROLLER is that of Figure 30.

The modular multiplier controller does all the control that the Montgomery modular multiplier needs as described in the previous section. Furthermore, it controls the changing from step 0 to step 1, the loading of the register denoted by REGISTER. The state machine is depicted in Figure 31.

- S_0: Initialize of the state machine;
  Set step to 0; Go to S_1;
- S_1: Load multiplicand and modulus; Load multiplier
  into SHIFT REGISTER$_1$; Go to S_2;

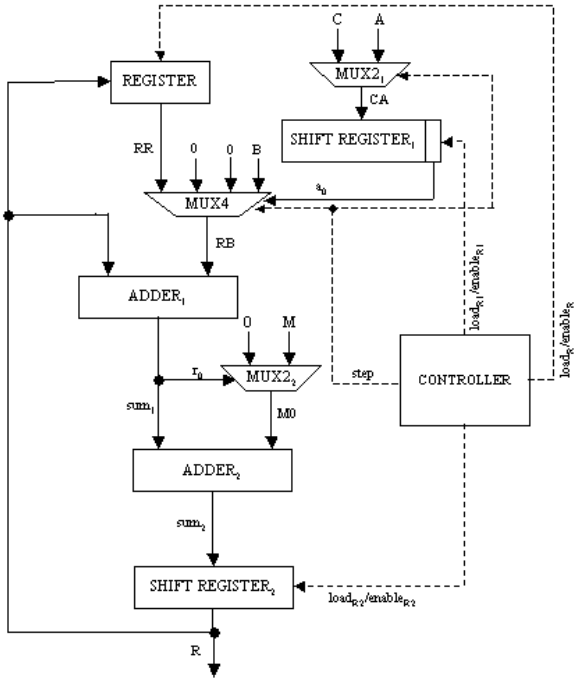


Figure 29: The modular multiplier architecture

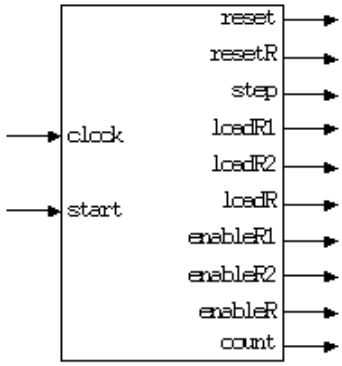


Figure 30. The interface of the multiplier controller

- S_2: Wait for adder1; Wait for ADDER$_2$;
  Load partial result into SHIFT REGISTER$_2$;
  Increment counter; Go to S_3;
- S_3: Enable SHIFT REGISTER$_2$;
  Enable SHIFT REGISTER$_1$; Go to S_4;
- S_4: Load the partial result of step 0 into REGISTER;
  Check the counter;
  If 0 then go to S_5 else go to S_2;
- S_5: Load constant into SHIFT REGISTER$_1$;
  Reset REGISTER;
  Set step to 1; Go to S_6;
- S_6: Wait for ADDER$_1$; Wait for ADDER$_2$;
  Load partial result into SHIFT REGISTER$_2$;
  Increment counter; Go to S_7;
- S_7: Enable SHIFT REGISTER$_2$;
  Enable SHIFT REGISTER$_1$; Go to S_8;
- S_8: Check the counter;
  If 0 then go to S_9 else go to S_6;
- S_9: Halt.

4.4 Simulation Results

The project of the modular multiplier described throughout this section was specified in Very High Speed Integrated Circuit Description Language - VHDL [20], and simulated using the *Xilinx*^{TM} *Project Manager* [40]. It allows the user to design and simulate the functionality

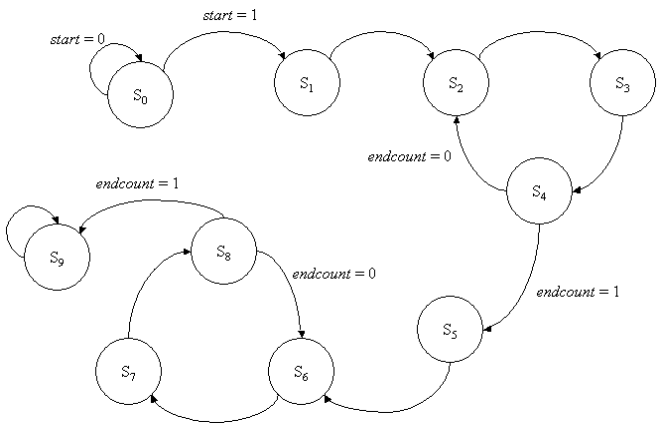


Figure 31: The state machine of the multiplier controller

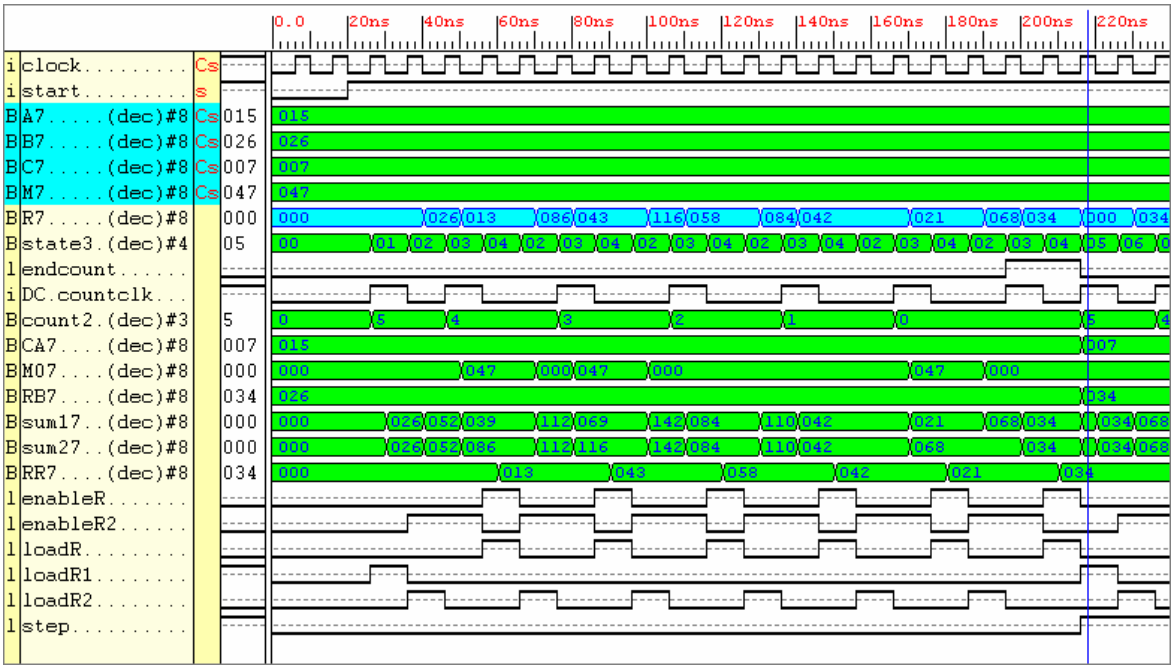


Figure 32: The modular multiplier behavior during the first multiplication: Montgomery(15, 26, 47) = 34

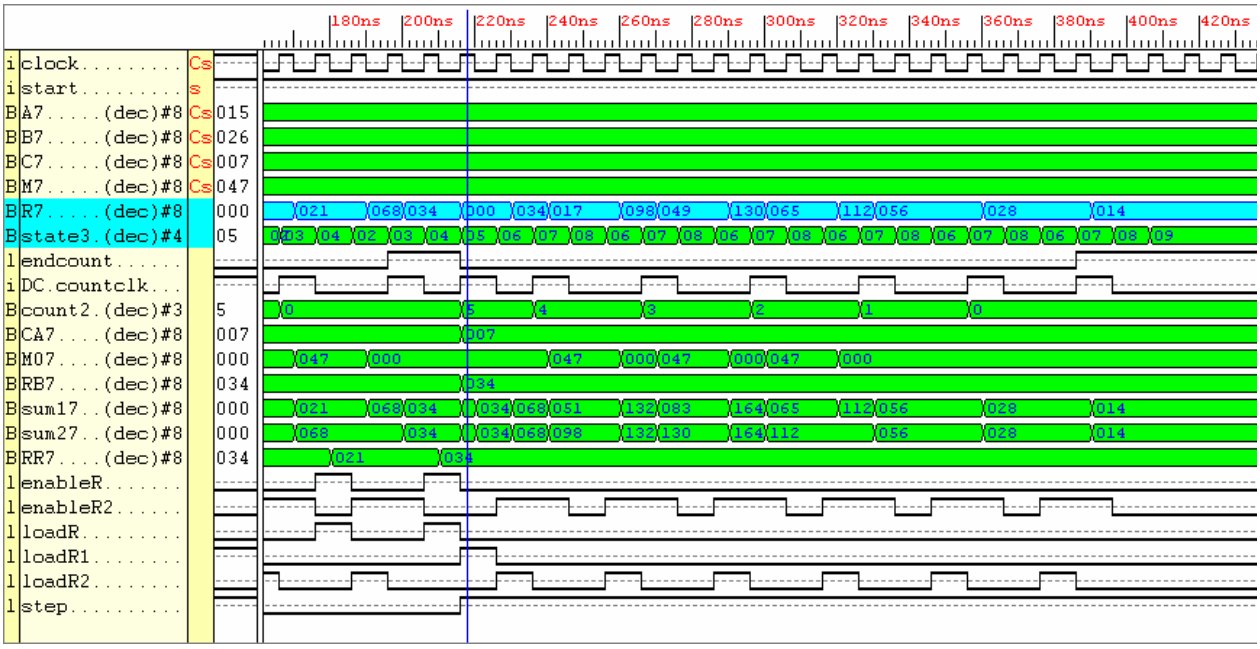


Figure 33: The modular multiplier behavior during the second multiplication: Montgomery(7,34,47) = 14

of his/her design. Moreover, it allows the synthesis of a correct design as well as its download on a specific FPGA.

First, we functionally simulated the Montgomery modular multiplier prototype for operands A = 15, B = 26, M = 47 and so the constant $C = 2^{2x6} \bmod 47$, which is C = 7. The signal values are shown in Figure 32 and Figure 33. The result is shown by signal R.

Figure 32 shows the behavior of the multiplier during the first modular multiplication (note that signal *step* is not set). Figure 33 shows the results of the second modular multiplication (note that signal *step* is set).

Also, we simulated the Montgomery modular multiplier prototype for bigger operand size, i.e. 16 bits. The operands are A = 120, B = 103, M = 143 and so the constant $C = 2^{2x8} \bmod 143$, which is C = 42. The result of the simulation is shown in Figure 34 and Figure 35.

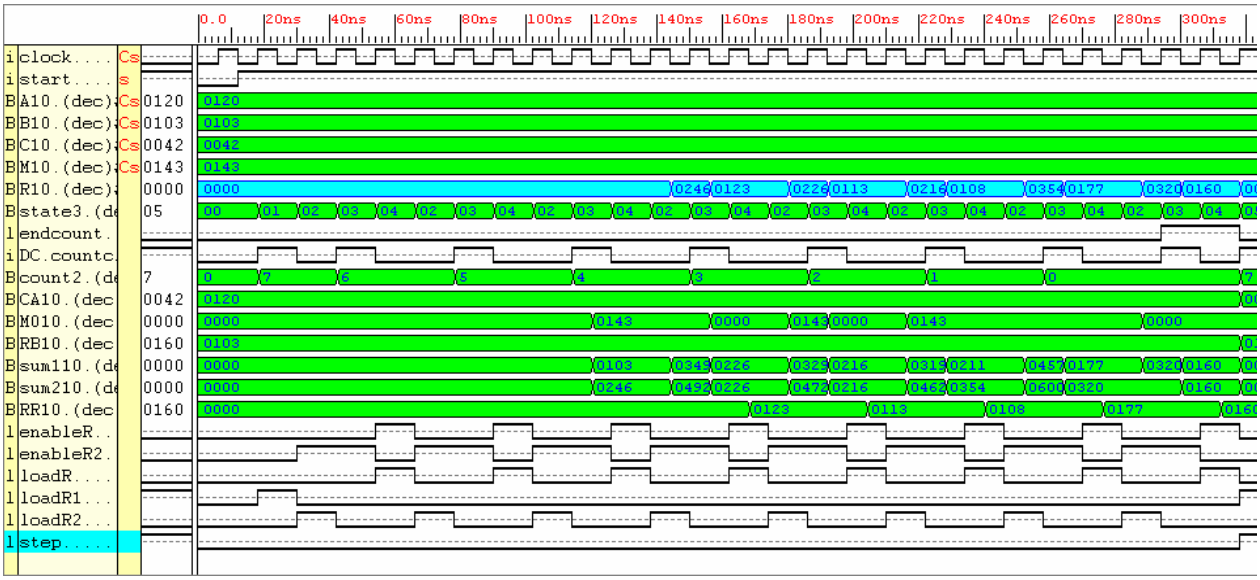


Figure 34: The multiplier behavior during the first multiplication: Montgomery(120, 103, 143) = 160

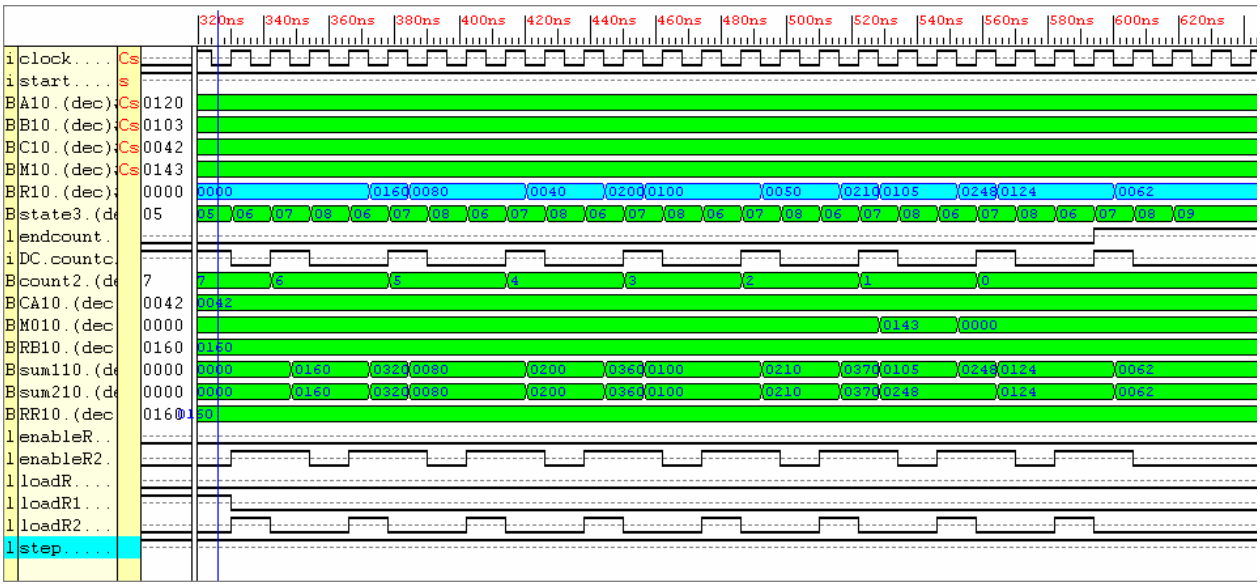


Figure 35: The modular multiplier behavior during the second multiplication: Montgomery(42, 160, 142) = 62

As before, Figure 34 shows the behavior of the multiplier during the first modular multiplication and Figure 35 shows the results of the second modular multiplication (note that signal *step* is set).

4.5 Systolic Montgomery Algorithm

A modified version of Montgomery algorithm [29] is that of Figure 36. The least significant bit of $R + a_i \times B$ is the least significant bit of the sum of the least significant bits of R and B if a_i is 1 and the least significant bit of R otherwise. Furthermore, new values of R are either the old ones summed up with $a_i \times B$ or with $a_i \times B + q_i \times M$ depending on whether q_i is 0 or 1.

```
algorithm ModifiedMontgomery(A, B, M)
      int R := 0;
      1: for i := 0 to n-1
      2:    qi := (r0 + ai×b0) mod 2;
      3:    R  := (R + ai×B + qi×M) div 2;
      return R;
end ModifiedMontgomery.
```

Figure 36: Modified Montgomery algorithm

Consider the expression $R + a_i \times B + q_i \times M$ of line 2 in the algorithm of Figure 36. It can be computed as indicated in the last column of Table 3 depending on the value of the bits a_i and q_i.

a_i	q_i	$R + a_i \times B + q_i \times M$
1	1	$R + MB$
1	0	$R + B$
0	1	$R + M$
0	0	R

Table 3: Computation of $R + a_i \times B + q_i \times M$

A bit-wise version of the algorithm of Fig. 4, which is at the basis of our systolic implementation, is described in Figure 37. All algorithms, i.e. those of Figure 23, Figure 24 and Figure 37 are equivalent. They yield the same result. In the algorithm of Figure 37, *MB* represents the result of $M + B$, which has at most has $n + 1$ bits.

4.6 Systolic Hardware Multiplier

Assuming the algorithm of Figure 37 as basis, the main processing element (PE) of the systolic architecture of the Montgomery modular multiplier computes a bit r_j of residue *R*. This represents the computation of line 8. The left-border PEs of the systolic arrays perform the same computation but beside that, they have to compute bit q_i as well. This is related to the computation of line 1. The duplication of the PEs in a systolic form implements the iteration of line 0. The systolic architecture of the systolic Montgomery multiplier is shown in Figure 38.

```
algorithm SystolicMontgomery(A,B,M,MB)
   int  R := 0;
   bit  carry := 0, x;
   0: for i := 0 to n
   1:    q_i := r_0^(i) ⊕ a_i.b_0;
   2:    for j := 0 to n
   3:       switch a_i, q_i
   4:          1,1: x := mb_i;
   5:          1,0: x := b_i;
   6:          0,1: x := m_i;
   7:          0,0: x := 0;
   8:       r_j^(i+1) := r_(j+1)^(i) ⊕ x_i ⊕ carry;
   9:    carry:=r_(j+1)^(i).x_i+r_(j+1)^(i).carry+x_i.carry;
   return R;
end SystolicMontgomery.
```

Figure 37: Systolic Montgomery algorithm

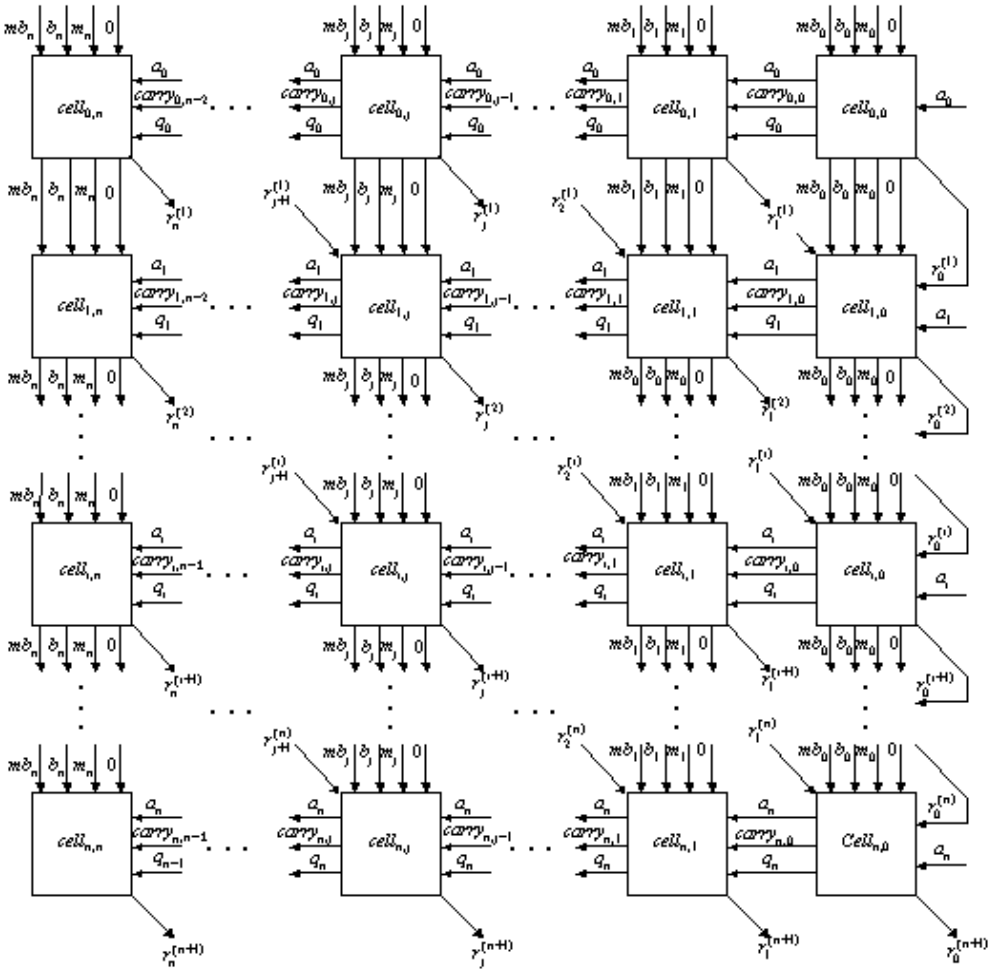


Figure 38: Systolic architecture of Montgomery multiplier

The architecture of the basic PE, i.e. $cell_{i,j}$ $1 \le i \le n-1$ and $1 \le i \le n-1$, is shown in Figure 39. It implements the instructions of lines 2-9 in systolic Montgomery algorithm of Figure 37. The architecture of the right-most top-most PE, i.e. $cell_{0,0}$, is given in Figure 40. Besides the computation of lines 2-9, it implements the computation indicated in line 1. However as $r_0^{(0)}$ is zero, the computation of q_0 is reduced to $a_0 . b_0$. Besides, the full-adder is not necessary as carry in signal is also 0 so $r_1^{(0)} \oplus x_i \oplus carry$ and $r_1^{(0)} . x_i + r_1^{(0)} . carry + x_i . carry$ are reduced to x_i and 0.

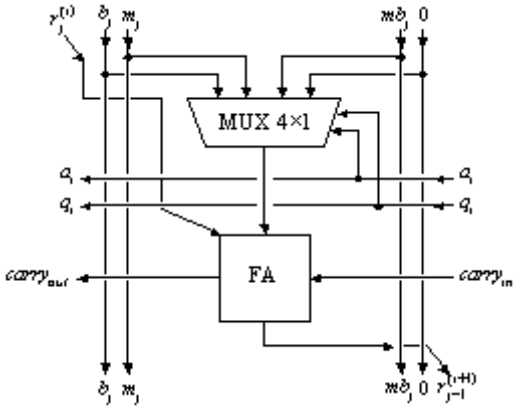


Figure 39: Basic PE architecture

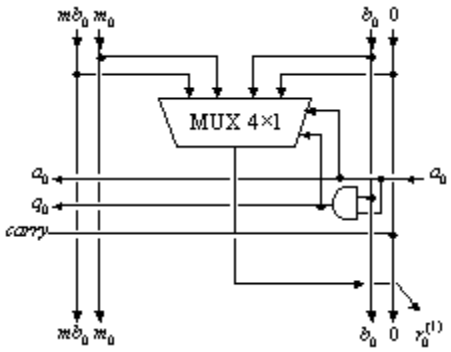


Figure 40: Right-most top-most PE – $cell_{0,0}$

The architecture of the rest of the PEs of the first column is shown in Figure 41. It computes q_0 in the more general case, i.e. when $r_0^{(i)}$ is not null. Moreover, the full-adder is substituted by a half-adder as the carry in signals are zero for these PEs.

The architecture of the architecture of the left border PEs, i.e. $cell_{0,j}$, is given in Figure 42. As $r_n^{(i)} = 0$, the full-adder is unnecessary and so it is substituted by a half-adder.

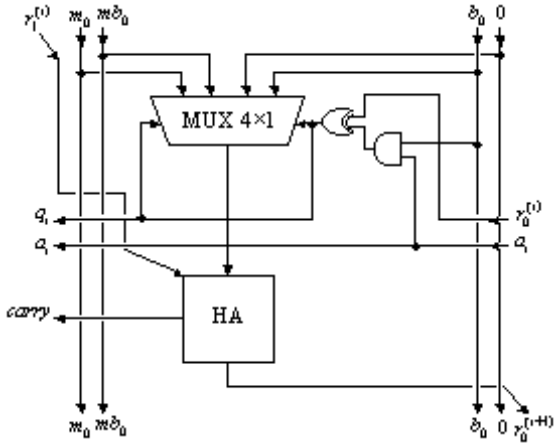


Figure 41: Right-border PEs – $cell_{i,0}$

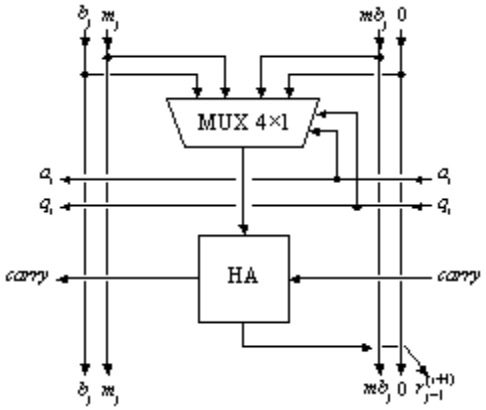


Figure 42: Left-border PEs – $cell_{0,j}$

The sum $M+B$ is computed only once at the beginning of the multiplication process. This is done by a row of full adder.

4.7 Time and Area Requirements

Consider the architecture of the systolic modular Montgomery multiplier of Figure 38. The output bit $r_j^{(n+1)}$ of the modular multiplication is yield after $2n + 2 + j$ after bits b_j, m_j and mb_j are fed into the systolic array plus an extra clock cycle, which is needed to obtain the bit mb_j. So the first output bit appears after $2n + 3$ clock cycles.

Table 4 shows the performance figures obtained by the Xilinx project synthesizer for the iterative multiplier the systolic modular multiplier, wherein *IM* and *SM* stand for *i*terative *m*ultiplier and *s*ystolic *m*ultiplier respectively. The synthesis was done for VIRTEX-E [40] family.

In Table 4, we present the clock cycle time required, the area, i.e. the number of CLBs necessary as well as the *time×area* product delivered by the synthesis and the verification tools of the Xilinx project manager [40] for

the iterative and systolic version of Montgomery multiplier.

operand size	***Area*** (CLBs)		***clock cycle time*** (ns)		***area×time***	
	IM	*SM*	*IM*	*SM*	*IM*	*SM*
128	89	259	46	23	4094	5957
256	124	304	102	42	12648	12767
512	209	492	199	76	41591	37392
768	335	578	207	82	69345	47396
1024	441	639	324	134	142884	85626

Table 4: Performance figures: iterative vs. systolic

The chart of Figure 43 compares the area×time product of iterative multiplier implementation vs. the systolic implementation. It shows that the latter improves the product as well as time requirement while the former improves area at the expense of both time requirement and the product.

The results show clearly that despite of requiring much more hardware area, our implementation improves substantially the time requirement and the performance factor when the operand size is bigger than 256 bits. This is almost always the case in RSA encryption/decryption systems. Nowadays, the hardware area has a very reasonable price so can be bought. However, the encryption/decryption throughput of cryptographic systems is the most fundamental characteristic and so cannot be sacrificed.

5 Further Improvements

The modular multiplication algorithm and respective hardware can be further improved if the representation of the operands is considered. The bits of the binary representation can be grouped to increase the representation base. For instance, if the bits are grouped into pairs or triples, the base will be 4 or 8 respectively. Although other bases are possible, usually a power of 2 is preferred to make conversion to and from binary easy. Increasing the base reduces the number of digits in the operand and so reduces the number of clock cycles required to complete a modular multiplication. Another improvement consists of using the so-called redundant representation of the operand together with the Montgomery algorithm. This avoids the unbounded propagation of carries.

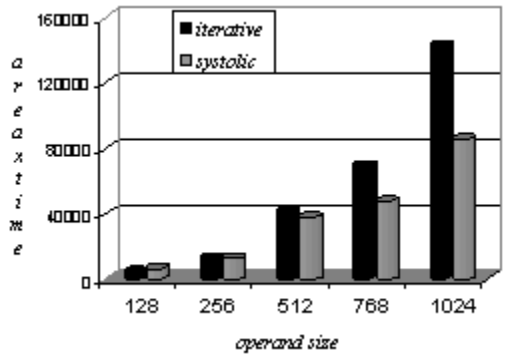


Figure 43: The *area×time* factor for iterative vs. systolic

6 Discussion

As stated in the introduction, the methods used to compute modular products fall in two categories: *(i)* those that first compute the product then reduced product, and *(ii)* those that compute the modular product directly.

The advantage of the first category method is that one can use any on-the-shelf method for multiplication and reduction. However, the only such methods that are efficient consist of those presented here, i.e. Karatsuba-Ofman's and Booth's methods for multiplication and Barrett's method for reduction. As far as the authors are concerned, these methods are the only ones appropriate for hardware implementation. Another disadvantage of using the multiply-then-reduce method is that the product is generally large and thus requires a great deal of space to store it for further use by the reduction step.

In contrast, the methods that interleave multiplication and reduction steps to produce the modular product do not have to store the product. However, also as far as the authors are concerned, only Montgomery's method that yields the modular product in such a way. Hardware implementation of Montgomery's algorithm always require very much less area than the implementations of the first category methods. Furthermore, these implementations are always very much slower than the implementations of Montgomery method.

7 Conclusions

In this paper we surveyed most known and recent methods for efficient modular multiplication. For each method presented, we provide an adequate hardware implementation.

We explained that the modular multiplication $A \times B \bmod M$ can be performed in two different ways: obtaining the product then reducing it; or obtaining the reduced product directly. There are various algorithms that implement modular multiplication. The most prominent algorithms are Karatsuba-Ofman's [27], [28] and Booth's [21], [22] methods for multiplying, Barrett's [21] method for reducing, and Montgomery's algorithms [5], [23], [24], [25], [26], [38], [39] for interleaving multiplication and reduction.

Throughout this paper, we considered each one of the methods cited previously. The review was organized as follows: First we described the Karatsuba-Ofman's and Booth's methods for multiplying. Subsequently, we presented Barrett's method for reducing an operand *modulo* a given *modulus*. For each method, we detailed the hardware architecture and then compared the respective hardware with respect to area and response time requirements. The implementation of the modular multiplication using Karatsuba-Ofman's method for multiplying and Barrett's method for reducing the obtained result presents a shorter signal propagation delay than using Booth's method together with Barrett's method, without much increase in hardware area requirements.

After that, we detailed the Montgomery's algorithm for interleaving multiplication and reduction. For the method, we presented two hardware implementations: one iterative and the other systolic. The systolic implementation is much better that the sequential one but requires more hardware area.

Subsequently, we reviewed some techniques that should allow further improvement to the implementation of the modular multiplication with long operand.

8 Acknowledgments

We are grateful to the reviewers and the editor that contributed to the great improvement of the original version of this paper with their valuable comments and suggestions. We also are thankful to FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro, http://www.faperj.br) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, http://www.cnpq.br) for their continuous financial support.

9 References

[1] Ashenden, P.J., *Recursive and Repetitive Hardware Models in VHDL*, Joint Technical Report, TR 160/12/93/ECE, University of Cincinnati, and TR 93-19, University of Adelaide, 1993.

[2] Barrett, P., *Implementating the Rivest, Shamir and Aldham public-key encryption algorithm on standard digital signal processor*, Proceedings of CRYPTO'86, Lecture Notes in Computer Science **263**:311-323, Springer-Verlag, 1986.

[3] Booth, A., *A signed binary multiplication technique*, Quarterly Journal of Mechanics and Applied Mathematics, pp. 236-240, 1951.

[4] Brickell, E. F., *A survey of hardware implementation of RSA*, In G. Brassard, ed., Advances in Cryptology, Proceedings of CRYPTO'98, Lecture Notes in Computer Science **435**:368-370, Springer-Verlag, 1989.

[5] S. E. Eldridge and C. D. Walter, *Hardware implementation of Montgomery's modular multiplication algorithm*, IEEE Transactions on Computers, **42**(6):619-624, 1993.

[6] Bewick, G.W., *Fast multiplication algorithms and implementation*, Ph. D. Thesis, Department of Electrical Engineering, Stanford University, United States of America, 1994.

[7] Dhem, J.F., *Design of an efficient public-key cryptographic library for RISC-based smart cards*, Ph.D. Thesis, Faculty of Applied Science, Catholic University of Louvain, May 1998.

[8] W. Diffie and M.E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, vol. **22**, pp. 644-654, 1976.

[9] ElGamal, T., *A public-key cryptosystems and signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, **31**(4):469-472, 1985.

[10] Gutmann P., Cryptographic Security Architecture: Design and Verifcation, Springer-Verlag, 2004.

[11] Kim, J.H., Ryu, J. H., *A high speed and low power VLSI multiplier using a redundant binary Booth encoding*, Proc. of 6th Korean Semiconductor Conference, PA-30, 1999.

[12] Knuth, D.E., *The art of computer* programming: seminumerical algorithms, vol **2**, 2nd Edition, Addison-Wesley, Reading, Mass., 1981.

[13] Jung, M., Madlener, F., Ernst, M. and Huss, S.A., *A reconfigurable coprocessor for finite field multiplication in GF(2^n)*, Proc. of IEEE Workshop on Heterogeneous Reconfigurable systems on Chip, Hamburg, Germany, 2002.

[14] Koç, Ç.K., *High speed RSA implementation*, Technical report, RSA Laboratories, RSA Data Security Inc. CA, version 2, 1994.

[15] Lim, C.H., Hwang, H.S. and Lee, P.J., *Fast modular reduction with precomputation*, In Proceedings of Korea-Japan Joint Workshop on Information Security and Cryptology, Lecture Notes in Computer Science, **488**:323-334, 1991.

[16] MacSorley, O., *High-speed arithmetic in binary computers*, Proceedings of the IRE, pp. 67-91, 1961.

[17] Menezes, A. van Oorschot, P. and Vanstone, S., *Handbook of Applied Cryptography*, CRC Press, 1996.

[18] Montgomery, P.L., *Modular Multiplication without trial division*, Mathematics of Computation, **44**: 519-521, 1985.

[19] Mourelle, L.M. and Nedjah, N., *Compact iterative hardware simulation model for Montgomery's algorithm of modular multiplication*, Proceedings of ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Tunisia, July 2003.

[20] Navabi, Z., *VHDL - Analysis and modeling of digital systems*, McGraw Hill, Second Edition, 1998.

[21] Nedjah, N. and Mourelle, L.M., *Yet another implementation of modular multiplication*, Proceedings of 13th. Symposium of Computer Architecture and High Performance Computing, Brasilia, Brazil, IFIP, pp. 70-75, 2001.

[22] Nedjah, N. and Mourelle, L.M., *Simulation model for hardware implementation of modular multiplication*, **In:** Mathematics nad Simulation with Biological, Economical and Musicoacoustical Applications, C.E. D'Attellis, V.V. Kluev, N.E. Mastorakis Eds. WSEAS Press, 2001, pp. 113-118.

[23] Nedjah, N. and Mourelle, L.M., *Reduced hardware architecture for the Montgomery modular multiplication*, WSEAS Transactions on Systems, **1**(1):63-67.

[24] Nedjah, N. and Mourelle, L.M., *Two Hardware implementations for the Montgomery modular multiplication: sequential versus parallel*, Proceedings of the 15th. Symposium Integrated Circuits and Systems Design, Porto Alegre, Brazil, IEEE Computer Society Press, pp. 3-8, 2002

[25] Nedjah, N. and Mourelle, L.M., *Reconfigurable hardware implementation of Montgomery modular multiplication and parallel binary exponentiation*, Proceedings of the EuroMicro Symposium on Digital System Design – Architectures, Methods and Tools, Dortmund, Germany, IEEE Computer Society Press, pp. 226-235, 2002

[26] Nedjah, N. and Mourelle, L.M., *Efficient hardware implementation of modular multiplication and exponentiation for public-key cryptography*, Proceedings of the 5th. International Conference on High Performance Computing for Computational Science, Porto, Portugal, Lecture Notes in Computer Science, **2565**:451-463, Springer-Verlag, 2002

[27] Nedjah, N. and Mourelle, L.M., *Hardware simulation model suitable for recursive computations: Karatsuba-Ofman's multiplication algorithm*, Proceedings of ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Tunisia, July 2003.

[28] Nedjah, N. and Mourelle, L.M., *A Reconfigurable recursive and efficient hardware for Karatsuba-Ofman's multiplication algorithm*, Proceedings of IEEE International Conference on Control and Applications, Istambul, Turkey, June 2003, IEEE System Control Society Press.

[29] Nedjah, N. and Mourelle, L.M. (Eds.), Embedded Cryptographic Hardware: Methodologies and Applications, Nova Science Publishers, Hauppauge, NY, USA, 2004.

[30] Nedjah, N. and Mourelle, L.M. (Eds.), Embedded Cryptographic Hardware: Design and Security, Nova Science Publishers, Hauppauge, NY, USA, 2005.

[31] Nedjah, N. and Mourelle, L.M. (Eds.), New Trends on Embedded Cryptographic Hardware, Nova Science Publishers, Hauppauge, NY, USA (to appear).

[32] Paar, C., *A new architecture for a parallel finite field multiplier with low complexity based on composite fields*, IEEE Transactions on Computers, **45**(7):856-861, 1996.

[33] Rabaey, J., *Digital integrated circuits: A design perspective*, Prentice-Hall, 1995.

[34] Rivest, R., Shamir, A. and Adleman, L., *A method for obtaining digital signature and public-key cryptosystems*, Communications of the ACM, **21**:120-126, 1978.

[35] Shindler, V., *High-speed RSA hardware based on low-power piplined logic*, Ph. D. Thesis, Institut für Angewandte Informations-verarbeitung und Kommunikationstechnologie, Technishe Universität Graz, January 1997.

[36] Zuras, D., *On squaring and multiplying large integers*, In Proceedings of International Symposium on Computer Arithmetic, IEEE Computer Society Press, pp. 260-271, 1993.

[37] Walter, C.D., *A verification of Brickell's fast modular multiplication algorithm*, International Journal of Computer Mathematics, **33**:153:169, 1990.

[38] Walter, C.D., *Systolic modular multiplication*, IEEE Transactions on Computers, **42**(3):376-378, 1993.

[39] Walter, C. D., *Systolic modular multiplication*, IEEE Transactions on Computers, **42**(3):376-378, 1993.

[40] Xilinx, Inc. *Foundation Series Software*, http://www.xilinx.com.

Researchers and Development – Young Researches

Metod Černetič
Univerza v Mariboru, Fakulteta za organizacijske vede,
Kidričeva 55a, 4000 Kranj
E-mail: metod.cernetic@fov.uni-mb.si

Brina Černetič
Univerza v Ljubljani, Fakulteta za družbene vede
E-mail: brina.cernetic@yahoo.com

Technical paper

Keywords: Lisbon strategy, research and development, share of GDP, young researchers

Received: February 3, 2005

Although Slovenian science is permanently subject of analysis and reorganization, there are no evident improvements of its organizational structure and share of researchers in so called governmental and economic sector. There are opinions that the key problems of Slovenian science are in the field of technical sciences; that the cause of industrial non-competitiveness are engineers. The search of data of expenses (share of GDP for R&D) respectively investments in science and based on the data of number of young researchers in Slovenia show: (1) Lisbon strategy incorporate right goals, clear mechanisms – the critical points are instruments and coordination for achieving these goals; (2) As s model Slovenia has to take new members, which already introduce their own, with budget supported goals of Lisbon strategy; (3) Ever since the establishment of new country Slovenia we are not able to reach consensus among all the pillars of political power and decide what should the role of science (primarily of R & D) in what should be relation between universities and economy.
Lisbon strategy gives us the answers to the question how EU can be competitive in long term and at the same time preserve European model of life; that means a balance between economical, social and environmental goals. The first condition for maintenance of social sustainability and kindness towards an environment is economical growth.

Povzetek: Ob stalnih analizah in reorganizacijah slovenske znanosti se ne izboljšuje njena organizacijska struktura, način organiziranosti in deleži raziskovalcev.

1 Introduction

The summary of tasks from the report by the PHARE project (A Science and Technology…), which can be understood as instructions for changes in measures in current science and technology policies (ZTP), must be divided up into measures taken by the government and measures taken by the public ministries (MZT).
As a result of measures taken in the form of laws, decrees etc. the government should have created a suitable environment that (Černetič 1999, 274-275):

- Is stimulating for the development of proprietorship and innovation,
- Is stimulating for industrial development, especially in sectors that show competitive advantage,
- Is stimulating for technological innovation and the transfer of technologies, especially from abroad.
- Is stimulating for scientific – research work, whereby the needs of the state in different areas (economic development, education, natural and cultural heritage, national identity etc.) should be taken into consideration.

The MTZ should also probably change its evaluation regarding the management of current medium-term science and technology policies, which have also been identified by the PHARE report (not only PHARE but also many domestic experts) as generic problems of the ZT sector:

- The connectedness between the ZT sectors (and also inside this sector) and the end-user is weak,
- The absence of a system of priorities,
- Mobility is poor for researchers,
- An insufficient amount of R&D activities in industries,
- Deficiencies in the technological transfer system,
- Researchers lack certain experience, which is important for successful R&D,

- The lack of stimulation for the flow of (young) researchers in industry and other sectors,
- Low levels of motivation in researchers for useful research.

The above-mentioned discoveries about the weaknesses of science and technology policies in Slovenia are also current issues, even though in the meantime the new state of Slovenia had been established, the ministries had been restructured, social and state responsibility had taken on new political option (government), etc. Slovenia is incapable of achieving consensus within society about the vision and aims of development and furthermore making them operational (Sočan, Bučar 2003, 118-120). Today, a Europe with "many gears" is already a reality. The following paragraph contains some data and evaluations, regarding questions relating to the title of this paper.

2 New Ties, New Government – Old Problems

In the last few months there have been many conferences regarding the theme of new scientific and technological policies in Slovenia relating to the Lisbon strategy, which define that the EU is going to be the most competitive economy in 2010. These conferences were made by the European Commission in Slovenia, the newspaper Finance and the Slovenian Chamber of Commerce (GZS). Let us mention a few reasons why the Slovenian sciences have not been achieving the desired objectives and expected progress:

- On explanation states that Slovenia still has a great concentration of research and researches in big research organizations.
- The second explanation states that we lag behind in the share of investment for R&D
- The third explanation, which deals with the decrease in the competitiveness of the Slovenian economy, emphasizes that there is an insufficient amount of researchers and developers in the (FTE) economy.
- The fifth explanation states that there are not enough young researchers in the technical sciences.
- The fifth explanation talks about the failures in implementing R&D that are found in the deficiencies in managing the sciences at the national level (within individual national economies and the EU level, in which amongst all others "Lisbon" and cohesive funds etc. are also available.

2.1 Science at the State Level in the Market

Recently, the Minister for Higher Education and Science answered the question how to increase the participation between science and the economy in the following manner: companies should find out what kind of R&D projects they need, unite and then inform the ministry. Then a tender would be published, whereby scientific research organizations would apply who would also then carry out these projects. The minister's commentary shows that it would be necessary to notify him, for a while now we have not had a system of central planning and his appeal for a national program that would organize the disharmony in developmental policies does not mean anything but an appeal for the strengthening of an already strong sentiment of economic totalitarianism. Why? (Pezdir, Finance no. 43/05)

The conditions within the institutional framework of the Slovenian economy are therefore, according to the central plan to monitor the economy, crushing. What kinds of opportunities are given to companies that would like to become more innovated in the institutional framework of Slovenian science as suggested by the minister? If the minister's statement is a central moment for the future mechanism in increasing the innovation potential of the Slovenian economy, we can expect that the process of increasing collaboration between science and the economy will take place according to the scenario, which is quite unusual for a market economy. As we already know in the first place, there will be companies only because they exist, they will be forced to join the "fraternity", which is known to always confiscate a part of their revenues and assign them to the financing the process of finding ways for individual groups of rent seekers, which are lead through the Chamber of Commerce straight to the carries of economic policies. They will in turn proclaim them as national champions (as a rule, the strongest group of rent seekers), who will unite at individual projects and notify the authorized ministries about them. The notified ministry will then carry out a tender, in which the needs of the national champions will be serviced by pre-transitional research and development organizations financed by the national budget. These types of projects will only be able to be financed by the taxes paid by proprietors that did have any luck in becoming the national champions. In other words – unprivileged companies, who had already paid taxes into the Chamber of Commerce for the transformation of rent seekers into national champions, will pay again – whether it is for achieving the aims of rent seekers or for creating an illusion that the economic policies stimulate collaboration between science and the economy. What does it really stimulate? (Pezdir, Finance no. 43/05). Above all things, within the system of legalized rent seeking it stimulates a competition amongst rent seekers for the best starting point in getting a share of the national budget, the state financed sciences, which is a big slice and the competition for access to the largest possible number of state financed projects for rent seekers (Pezdir, Finance št. 43/05).

2.2 A Weak Point – Management and Funds for R&D

In realizing the Lisbon strategy, each country must also take into consideration specific national objectives and the main problem of the strategy is in its weak

management and co-ordination between the EU and member states. All foundations for the documents have been implemented and now it is important to better comprehend the meaning of partnership in the realization of the Lisbon strategy. One of the important instruments for its realization is also the EU's budget in which the "Lisbon" and "cohesive" funds are available. There is a catch that exits in that the first part of the funds are going to be distributed according to excellence, which for the most part means ending up in the developed members and the rest for lesser developed states. At this point, smaller EU member states emphasize the need for the restructuring of the European budget.

2.3 Germany and France Should Not Be Our Role Models

The main problem of the EU is in its heterogeneity and Slovenia should not model themselves after the traditional EU members (France and Germany) as much, which have been plagued with small growth in GDP and a high rate of unemployment. We should look at new member states (Slovakia, Latvia), which are more ambitious in development, more decisive and have already introduced their own budget supported goals of the Lisbon strategy. As a result, they have already organized three areas: a friendlier business environment (tax and other reforms), the liberalisation of the labour market, intend more funds fir subsidising technological development and finding synergy between universities, the area of research and the economy. In the opinions of some, this is the right path for realizing the Lisbon strategy.

2.4 Expenditures of EU Members for Research and Development

In 2002, 25 of the EU members assigned an average of 1.93% of GDP for R&D in comparison to 1.82 % of GDP in 1998 according to Eurostat (Kenda, Finance, no. 40/05). They have also published data for EFTA members, candidates China, Japan and the USA so that international comparisons could easily be made. Once again, Sweden assigned the most followed by Holland and The UK. On the other hand, Luxembourg assigned the greatest expenditure in the private sector for total R&D expenditure. As a result, the % in Luxembourg was 90% followed by Sweden, Finland, Ireland and Germany. According to this information Slovenia assigned 1.53 % of GDP in 2003 and 67% of this in the private sector of total expenditure for R&D in 2002.

3 Intolerable Effortless Planning

3.1 Bureaucracy Cannot Direct Development

We are moving into an "economy driven by intelligence" (Kos, Finance, no. 33/05). However, who will be the one to surpass the incapability of the economic elite and the government? Writers in this area also do not have the knowledge about innovative mechanisms and creative people. We cannot expect technological progress to come from universities and institutes because someone must direct their research into a team that works in this area.

Jobs are disappearing in our economy. However, the state bureaucracy will not solve anything with its ideas but will only intensify the crisis. An individual with their ideas is needed, who also takes responsibility for their actions. However, within a thousand ideas only one can succeed and therefore it is necessary to divide up the state funds accordingly and weed out the bad ideas before time and money is wasted on them. This is how the founders of innovative companies abroad operate and it is hard to find them at home.

We have tenders, whereby regional developmental agencies participate and there is no individual who would take responsibility for their ideas. These are bureaucratic efforts in obtaining funds for some common goals, which are organized in a general way and are in no way productive for creating new jobs. They are just new public administration sectors that are going to use up the state budget.

Some even support the idea for a central distribution of funds. This is nothing other than just transferring the competencies to the state level, where the funds will be distributed even more inefficiently.

The foundation of every strategy for creating new jobs must be proprietor-individual, who has a developed idea about the product. This is an alternative to unemployment regarding sectors that have been written off.

3.2 The Minister of Technology on Inheritance Mines

Slovenian science and its influence on the competitiveness of companies, has decreased by four or more places on international lists after the departure of the former government, which is a somewhat poor inheritance for the new Minister of Science and Technology. The government only invested in well-established research spheres and not in the economy. Investments into the economy had decreased from 9% to 5%, which is in contrast to the operations of EU governments that have been incorporating innovations into the economy more and more, which is the foundation of the Lisbon strategy.

This is why reforms on developmental policies are necessary. Measures must be taken in two areas (Kos, Finance no. 33/05):

First of all, financing must originate from research and development projects. Program financing is abolished. State intuitions have battled for "program financing", which does not demand the selection of project topics regarding innovation and that is why the orders in the economy have been cut. State institutions can do whatever they like, which is very irresponsible of the government and at the same time it does not bring any progress to the innovativeness of Slovenia.

Secondly, there is restructuring. The percentage of institutes, which are only mainly research-oriented, should fall under universities and the other part should be in the ownership of companies as industry institutes. This is how we shall develop strong universities, which would otherwise weaken and also a strong science economy with research capable of progress in innovation.

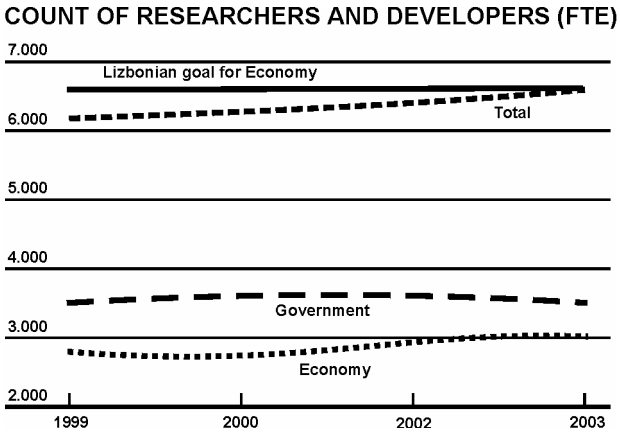
3.3 Decreased Competitiveness in the Slovenian Economy

Graduates of post-graduate and graduate studies will not achieve the Lisbon strategy, only the engineers of technical natural sciences will. Our planners (for achieving the Lisbon strategy) are still far from reality. Great numbers of sociologists, humanists and people in the legal profession cannot at the least influence product innovation of, exporting and GDP. Innovations are the fruit of talented engineers that work in companies, even though there are many who only have a BA or a secondary technical school diploma. Researchers in the government sector have not yet developed and manufactured no new product (Kos, Finance, no. 175/04).

Unfortunately, only 27% of all 23.691 engineering graduates work in industry, others are in the service sector (24%), public administration (10%) and in education, where innovation is lost.

As a result, we are a country lagging behind in innovation. As a result of no aid from the government and thousands of companies not investing a single tolar in innovation low-quality exports, no new jobs and a social crisis has resulted.

Even a quick glance of the graph (Graph 1) shows us that it is impossible to reach 2% of GDP for achieving the Lisbon strategy through a natural process, without radical measures taken by the state, which will at least triple the flow of engineers and kick-start innovation. However, we are not capable of doing this because there is a blockade of lobbies that have special interests.



Graph 1: Number of researchers and developers (Source: Kos, Finance, no. 175/04)

4 Where is the Knowledge – in Companies, Universities or Institutes?

4.1 Young Researchers

One of the instruments for the scientific policy of research agencies is financing post-graduate studies and research training for young researchers. The program has been successfully operating since 1985 and has additionally contributed to the increase in the amount of research and adding young minds to research groups. As a result of the program's success, a large part of the agency's budget funds are intended for human resource training. Until now, 5347 young researchers have been part of the program.

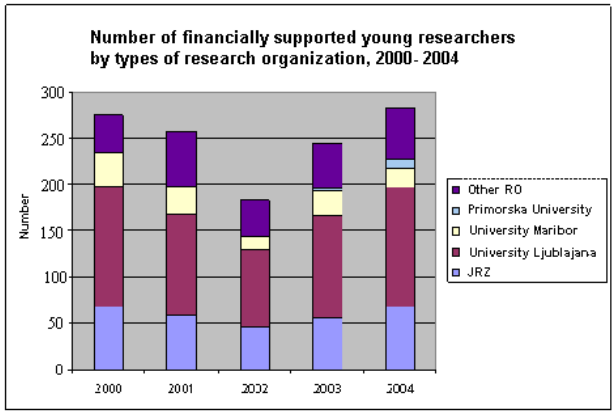
Characteristics of the young researchers program(http://www.arrs.gov.si):

- During their post-graduate studies young researchers do research on fundamental or research-development application projects,
- They have a fixed employment contract,
- The ministry ensures funds for their salaries, contributions, material and non-material expenses for their research work and post-graduate studies.

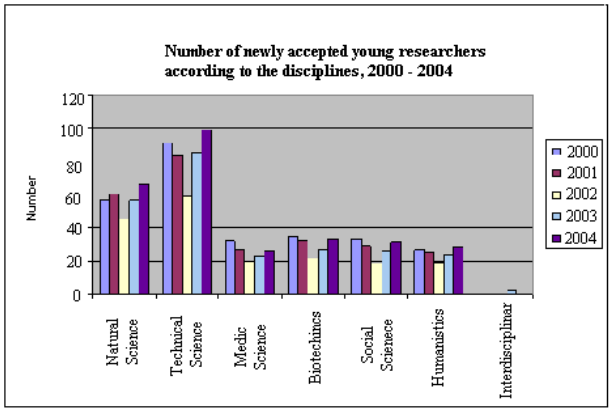
Scholarships for training young researchers are given for a certain period of time, which includes four years and six months at most for a PhD. On average, 6.6 million tolars is needed to finance a young researcher yearly.

4.2 The Size and Structure for Financing

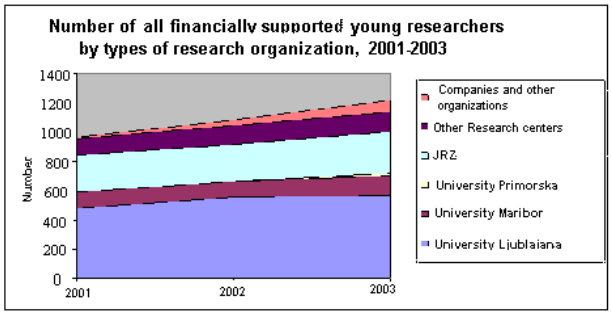
Every year, the agency finances about 1200 young researchers, which represents about the equivalent of 850 to 900 fully paid young researchers. Every year, 200 to 250 young researchers complete the training program, which is the same amount that is financed once again.



Graph 2: The number of newly accepted young researchers according to the type of research organization.



Graph 3: Number of newly accepted young researchers according to the disciplines

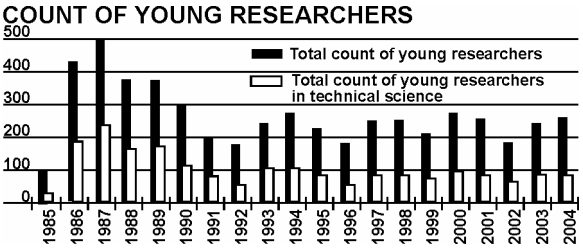


Graph 4: Total number of financially supported young researchers according to the type of research organization.

4.3 The Number of Young Researchers and ZTP Direction

After constant analyses of Slovenian science, its reflection in companies, development and influence of high technology sometimes bear negative thoughts and energy by researchers who participate with companies. As a result of these analyses a doubt arises in researchers in the area of technical sciences in that they are being constantly dealt with as the only ones, to whom the fault can be attributed to in Slovenia for the slow (too slow) development of the nation. That is to say that the problem in Slovenian science is exclusively in the area of technical science and that the problems in the lack of competitiveness industry are exclusively because of engineers. (Duhovnik, Finance, št. 226/04).

This had been established in 2004 at the business conference in Portorož, which is organized by the daily Finance. There were discussion on innovation processes with domestic and foreign consultants and some new ones, who had mad contributions about how to teach Slovenian development engineers so that they could understand the new technology. This transfer of knowledge should be made by those who manage business systems. Members of management boards for development in multinationals or of famous consulting companies also had lectures. There was also special emphasis put on investing in science. It is difficult to understand that within such an elite group of speakers, a domestic scientist in the technical field is not allowed to (or better yet it would not be suitable to) make a commentary. (Duhovnik, Finance, št. 226/04). Even more so, if they are employed at any institute or technical faculty and do not deal with "management" but does research and development, even having patents in certain areas.



Graph 5: Number of young researchers in technical science (Source: Duhovnik, Finance no. 226/04)

Therefore, if we take a look at the programs for young researchers in the area of science, we can establish that we have six areas: natural-mathematical, technical, medical, biotechnical, social sciences and humanities. There could only be four areas with a rough division: natural-mathematical, technical, medical/biotechnical and social sciences with humanities. Older researchers and responsible employees in companies are familiar with the infamous project 2000 MR, which had been introduced to the Slovenian public by Dr. Boris Frlec. It was accepted with enthusiasm and thus in was continued. One year it was a little better and in others it was not as successful. It is understandable that it cannot be totally successful. In comparison with the entire period it had a success rate of 75%. If we take a look at only the last few years or the years after 1990, we can see an exceptional success rate of 97%. First of all we can evaluate the success rate with a formally obtained academic title, with a doctorate in the last little while. An important fact is that in the last few years 420 to 500 candidates who have all the credentials have appeared and only 180 to 280 are rightfully selected. In some areas, the quality of the candidature of young researchers is so high that young researchers who only have an average mark of nine are cut. The prerequisite for each candidate is their average grade for the university program. This already shows the disproportion of grades in the programs that are only four years or those that are six. As a result an expert system has been implemented, however with a cut-off rate (50 %) of a limited number of accepted young researchers, which is almost a disgrace for evaluators who see a young researcher with exceptional potential in front of their eyes. (Duhovnik, Finance, št. 226/04).

4.4 The Project of Young Researchers and Domestic Consulters

Let us take a look at how they decided on making an investment in the project for young researchers from 1985 onwards. If we imagine that as a rule young researchers in companies are not going to introduce high

technology but are going to design a product, which is going to use the functions of high technology then we can probably expect them to be from the area of technical science. From the diagram, we can clearly see the relation between young researchers in the area of technical science and the total number of accepted young researchers. That is why it would be proper that the investment analysts for science understand investment in science at all level, which the state must harbour or it is already defined in their development strategy.

From 1997 onwards, a levelling of wages in the placing of funds has been used for the different sciences. This means that a development strategy for a certain science had not been even used. Even what is more alarming is that nothing has changed amongst young talented engineers in Slovenia. There has been no change in the growth of talents. Impossible! It can be achieved with the use of hard policies!

In the upcoming years we will have to increase GDP intensely. That is why above all other things it will be very important, how the knowledge of young researchers will be directly used. What will be the answer from the Slovenian business world? Will anyone conjure to say that our young researchers are incapable, in light of the data on average grades? The question arises: Have we opened the doors for employment for theses exclusive young researchers? Or will they rather return to projects using high technology for the manufacturing of products as R&D engineers in multinational corporations? I doubt that foreign lecturers are going to be able to respond in a strategic way to such a question at Slovenian business conferences. I shall be especially satisfied (Duhovnik, Finance, št. 226/04), if they could respond to one more important question: Can a modern state without its own products, which means industrial ownership collect enough money from taxes and contributions that it could cover all the costs pertaining to the public administration, a normal pension fund and finally a good healthcare system? For future business conferences it is important to invite domestic experts of acclamation, who also know how to take on the responsibility of technological development in Slovenian companies. Therefore, domestic lectures need to be invited to these conferences. Only these people would be able to explain what exactly could be done regarding the situation of industrial ownership in Slovenia.

5 Conclusion

The Lisbon strategy incorporates the right goals and clear mechanisms, although the critical points are instruments and coordination for achieving these goals. Representatives of the European Commission in Slovenia on the Lisbon strategy stated that the newcomers have been especially successful.

The Lisbon strategy gives us the answers to the question how the EU can be competitive in the long run and at the same time preserve the European model of life with a balance in economical, social and environmental goals. The first condition for the maintenance of social sustainability and kindness towards an environment is economical growth.

In realizing the Lisbon strategy, each country must also take into consideration specific national objectives and the main problem of the strategy is in its weak management and co-ordination between the EU and member states. All foundations for the documents have been implemented and now it is important to better comprehend the meaning of partnership in the realization of the Lisbon strategy.

One of the important instruments for its realization is also the EU's budget in which the "Lisbon" and "cohesive" funds are available. There is a catch that exits in that the first part of the funds are going to be distributed according to excellence, which for the most part means ending up in the developed members and the rest for lesser developed states. At this point, smaller EU member states emphasize the need for the restructuring of the European budget. Evidently, there are still three problems regarding expenditures for R&D (Černetič, 2003, 16):

- Lack of vision or consensus in development
- The creation of a list of priorities, which should be able to strengthen the gathering of funds fore R&D
- How to create an environment of innovation intended for most small and medium-sized companies with the measures and instruments of current economic policies.

References

[1] A Science and Technology Strategy for Slovenia, GOPA, MZT, Ljubljana 1994.

[2] Černetič M: *Ekonomika izobraževanja in raziskovanja*, Moderne organizacija, Kranj 1999.Černetič M: Sociological and economic point of wiev on globalization and development,

[3] *Adult Education* – A Key to the 21. Century, Proceedings of the International Coference, Hrvatsko andragoško društvo, Zagreb 2004.

[4] Černetič M: Knowledge and Education – Management of Development, Informatologija no.2, Zagreb 2003.

[5] Černetič M. Razvoj človeških virov –primer podiplomskega študija(PŠ) v Sloveniji, Organizacija št. 7, Kranj, 1998.

[6] Černetič, M. *Vlaganje v razvoj in modeli univerz*. Organizacija št. 10, Kranj, 2002.

[7] Černetič, M. *Znanje in informacijska družba*. Organizacija, št. 8, Kranj, 2002.

[8] Duhovnik J:*Znanost v podjetjih-ali kje sploh je?* Finance št. 226, Ljubljana, 18. 11. 2004.

[9] Kos M: *Birokracija ne more usmerjati razvoja. Finance št.33.Ljubljana, 16.02 2005.*

[10] *Kos M: Minister za znanost na minah dediščine*. Finance št. 256, Ljubljana, 1.12. 2004.

[11] Kos M: *Neznosna lahkotnost planiranja.* Finance št. 175, Ljubljana, 07. 09. 2004.

[12] Kos M: *O regionalizaciji: enakost, denar, ideje*. Finance št. 145, Ljubljana, 27.07. 2004.

[13] Kos M: *Izdelčno vodena konkurenčnost.* Finance št. 115, Ljubljana, 14. 06. 2004.
[14] Stanovnik, P: *Nujnost prehoda v družbo znanja s krepitvijo slovenskega raziskovalno-razvojnega sistema*. Posvet DOF v DS, 27.01.2004, Ljubljana, 2004.
[15] http://www.arrs.gov.si/sl/mr/razpisi/-20. 06. 2005

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in `.eps` format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than ten years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

http://www.informatica.si/

Referees:

Witold Abramowicz, David Abramson, Adel Adi, Kenneth Aizawa, Suad Alagić, Mohamad Alam, Dia Ali, Alan Aliu, Richard Amoroso, John Anderson, Hans-Jurgen Appelrath, Iván Araujo, Vladimir Bajič, Michel Barbeau, Grzegorz Bartoszewicz, Catriel Beeri, Daniel Beech, Fevzi Belli, Simon Beloglavec, Sondes Bennasri, Francesco Bergadano, Istvan Berkeley, Azer Bestavros, Andraž Bežek, Balaji Bharadwaj, Ralph Bisland, Jacek Blazewicz, Laszlo Boeszoermenyi, Damjan Bojadžijev, Jeff Bone, Ivan Bratko, Pavel Brazdil, Bostjan Brumen, Jerzy Brzezinski, Marian Bubak, Davide Bugali, Troy Bull, Sabin Corneliu Buraga, Leslie Burkholder, Frada Burstein, Wojciech Buszkowski, Rajkumar Bvyya, Giacomo Cabri, Netiva Caftori, Particia Carando, Robert Cattral, Jason Ceddia, Ryszard Choras, Wojciech Cellary, Wojciech Chybowski, Andrzej Ciepielewski, Vic Ciesielski, Mel Ó Cinnéide, David Cliff, Maria Cobb, Jean-Pierre Corriveau, Travis Craig, Noel Craske, Matthew Crocker, Tadeusz Czachorski, Milan Češka, Honghua Dai, Bart de Decker, Deborah Dent, Andrej Dobnikar, Sait Dogru, Peter Dolog, Georg Dorfner, Ludoslaw Drelichowski, Matija Drobnič, Maciej Drozdowski, Marek Druzdzel, Marjan Družovec, Jozo Dujmović, Pavol Ďuriš, Amnon Eden, Johann Eder, Hesham El-Rewini, Darrell Ferguson, Warren Fergusson, David Flater, Pierre Flener, Wojciech Fliegner, Vladimir A. Fomichov, Terrence Forgarty, Hans Fraaije, Stan Franklin, Violetta Galant, Hugo de Garis, Eugeniusz Gatnar, Grant Gayed, James Geller, Michael Georgiopolus, Michael Gertz, Jan Goliński, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Jozsef Gyorkos, Marten Haglind, Abdelwahab Hamou-Lhadj, Inman Harvey, Jaak Henno, Marjan Hericko, Henry Hexmoor, Elke Hochmueller, Jack Hodges, Doug Howe, Rod Howell, Tomáš Hruška, Don Huch, Simone Fischer-Huebner, Zbigniew Huzar, Alexey Ippa, Hannu Jaakkola, Sushil Jajodia, Ryszard Jakubowski, Piotr Jedrzejowicz, A. Milton Jenkins, Eric Johnson, Polina Jordanova, Djani Juričič, Marko Juvancic, Sabhash Kak, Li-Shan Kang, Ivan Kapustøk, Orlando Karam, Roland Kaschek, Jacek Kierzenka, Jan Kniat, Stavros Kokkotos, Fabio Kon, Kevin Korb, Gilad Koren, Andrej Krajnc, Henryk Krawczyk, Ben Kroese, Zbyszko Krolikowski, Benjamin Kuipers, Matjaž Kukar, Aarre Laakso, Sofiane Labidi, Les Labuschagne, Ivan Lah, Phil Laplante, Bud Lawson, Herbert Leitold, Ulrike Leopold-Wildburger, Timothy C. Lethbridge, Joseph Y-T. Leung, Barry Levine, Xuefeng Li, Alexander Linkevich, Raymond Lister, Doug Locke, Peter Lockeman, Vincenzo Loia, Matija Lokar, Jason Lowder, Kim Teng Lua, Ann Macintosh, Bernardo Magnini, Andrzej Małachowski, Peter Marcer, Andrzej Marciniak, Witold Marciszewski, Vladimir Marik, Jacek Martinek, Tomasz Maruszewski, Florian Matthes, Daniel Memmi, Timothy Menzies, Dieter Merkl, Zbigniew Michalewicz, Armin R. Mikler, Gautam Mitra, Roland Mittermeir, Madhav Moganti, Reinhard Moller, Tadeusz Morzy, Daniel Mossé, John Mueller, Jari Multisilta, Hari Narayanan, Jerzy Nawrocki, Rance Necaise, Elzbieta Niedzielska, Marian Niedq'zwiedziński, Jaroslav Nieplocha, Oscar Nierstrasz, Roumen Nikolov, Mark Nissen, Jerzy Nogieć, Stefano Nolfi, Franc Novak, Antoni Nowakowski, Adam Nowicki, Tadeusz Nowicki, Daniel Olejar, Hubert Österle, Wojciech Olejniczak, Jerzy Olszewski, Cherry Owen, Mieczyslaw Owoc, Tadeusz Pankowski, Jens Penberg, William C. Perkins, Warren Persons, Mitja Peruš, Fred Petry, Stephen Pike, Niki Pissinou, Aleksander Pivk, Ullin Place, Peter Planinšec, Gabika Polčicová, Gustav Pomberger, James Pomykalski, Tomas E. Potok, Dimithu Prasanna, Gary Preckshot, Dejan Rakovič, Cveta Razdevšek Pučko, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Vojislav D. Radonjic, Luc de Raedt, Ewaryst Rafajlowicz, Sita Ramakrishnan, Kai Rannenberg, Wolf Rauch, Peter Rechenberg, Felix Redmill, James Edward Ries, David Robertson, Marko Robnik, Colette Rolland, Wilhelm Rossak, Ingrid Russel, A.S.M. Sajeev, Kimmo Salmenjoki, Pierangela Samarati, Bo Sanden, P. G. Sarang, Vivek Sarin, Iztok Savnik, Ichiro Satoh, Walter Schempp, Wolfgang Schreiner, Guenter Schmidt, Heinz Schmidt, Dennis Sewer, Zhongzhi Shi, Mária Smolárová, Carine Souveyet, William Spears, Hartmut Stadtler, Stanislaw Stanek, Olivero Stock, Janusz Stokłosa, Przemysław Stpiczyński, Andrej Stritar, Maciej Stroinski, Leon Strous, Ron Sun, Tomasz Szmuc, Zdzislaw Szyjewski, Jure Šilc, Metod Škarja, Jiři Šlechta, Chew Lim Tan, Zahir Tari, Jurij Tasič, Gheorge Tecuci, Piotr Teczynski, Stephanie Teufel, Ken Tindell, A Min Tjoa, Drago Torkar, Vladimir Tosic, Wieslaw Traczyk, Denis Trček, Roman Trobec, Marek Tudruj, Andrej Ule, Amjad Umar, Andrzej Urbanski, Marko Uršič, Tadeusz Usowicz, Romana Vajde Horvat, Elisabeth Valentine, Kanonkluk Vanapipat, Alexander P. Vazhenin, Jan Verschuren, Zygmunt Vetulani, Olivier de Vel, Didier Vojtisek, Valentino Vranić, Jozef Vyskoc, Eugene Wallingford, Matthew Warren, John Weckert, Michael Weiss, Tatjana Welzer, Lee White, Gerhard Widmer, Stefan Wrobel, Stanislaw Wrycza, Tatyana Yakhno, Janusz Zalewski, Damir Zazula, Yanchun Zhang, Ales Zivkovic, Zonling Zhou, Robert Zorc, Anton P. Železnikar

Informatica

An International Journal of Computing and Informatics

Archive of abstracts may be accessed at America: http://ocean.ocean.cs.siu.edu/informatica/index.html, Europe: http://www.informatica.si/, Asia: http://www3.it.deakin.edu.au/ hdai/Informatica/.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2006 (Volume 30) is
– 60 EUR (80 USD) for institutions,
– 30 EUR (40 USD) for individuals, and
– 15 EUR (20 USD) for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printed by Dikplast Kregar Ivan s.p., Kotna ulica 5, 3000 Celje.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. Drago Torkar, Jožef Stefan Institute: Tel (+386) 1 4773 900, Fax (+386) 1 219 385, or send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova Ljubljanska Banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, ACM Digital Library, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Cybernetica Newsletter, DBLP Computer Science Bibliography, Engineering Index, INSPEC, Linguistics and Language Behaviour Abstracts, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik

The issuing of the Informatica journal is financially supported by the Ministry of Higher Education, Science and Technology, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 30 Number 1 January 2006 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **30** (2006) Number 1, pp. 1–139