A GENOME BASED VISION OF MULTI-AGENT SYSTEMS

Monica Vitali
Università degli studi di Palermo, viale delle Scienze, Palermo, Italy
mon.vitali@gmail.com

Massimo Cossentino, Riccardo Rizzo
CNR-ICAR Palermo, viale delle Scienze, Palermo, Italy
cossentino@pa.icar.cnr.it, ricrizzo@pa.icar.cnr.it

Salvatore Gaglio
Università degli studi di Palermo and CNR-ICAR Palermo, viale delle Scienze, Palermo, Italy
gaglio@unipa.it

Keywords: Agent Models and Architectures, Multi-Agent Systems, Adaptation.

Abstract: A set of software agents can be programmed to provide a large but finite set of services, often defined during design phase. After an evolution of the external environment, the pre-defined services could be unable to satisfy the requested quality. In this work it is proposed an agent framework capable to adapt the agents in order to improve the quality of services provided by an agent society in correspondence with a modification of the external environment. These agents are based on a biologically inspired structure (genome), that defines all their behaviors and knowledges. The effectiveness of the approach is proved by a set of successful experimental results.

1 INTRODUCTION

The aim of the proposed system is to improve the quality of service through the definition of an adaptive multi-agent system where agents are specified by their genome.

We think that the definition of an adaptation mechanism that allows the satisfaction of requirements is an essential step towards a mechanism of agent adaptation focused on service improvement. Dissimilarly from the OO expected passivity of service providers (objects) that delegates to the system designer the responsibility to find a solution to each new issue, in the agent-oriented (AO) context we think the system should be able to autonomously adapt and solve new and unforeseeable problems thus developing an explicit and solid system evolution capability. When using the *system* term we are here referring to both a single agent that could solve alone the problem as well as to a group of agents that can collaborate in order to achieve a common goal.

In order to obtain this system capability, we decided to adopt a biological metaphor. Our agent is regarded as a living entity, whose structure is defined by its genome. Agent capabilities are described in its genome and their improvement is possible by means of a Darwinian evolution of the specie. When a solution to a problem is not achievable (the corresponding service is not available or it does not provide the required quality of service), several agents can reproduce themselves thus creating a new generation of agents that have new capabilities and can satisfy the requirements. The consequence of this behavior is the creation of an adaptive system. Adaptation is the ability of the system to learn from past experiences and to react to unexpected events (Gleizes et al., 1999). We realized such an approach by an extensive adoption of Genetic Programming techniques (Nilsson, 1998) (Koza, 1992) and the use of advanced methods for automatic code generation, compilation and execution from existing agent-oriented platforms (for our purposes we preferred the Java language and the JADE platform). These techniques are only a tool used to reach an adaptive behavior inside the system.

The aim of this paper is exploring a new approach towards the definition of adaptive multi-agent systems where agents are specified through their genome. Adaptation is exhibited by agents in improving their capacities to fulfill some requirements (more properly provide services) they are not initially able to cope with. This may occur at both the level of a single agent (who may adapt itself to provide a service with the required quality) or at the social level where several agents need to adapt themselves and their collab-

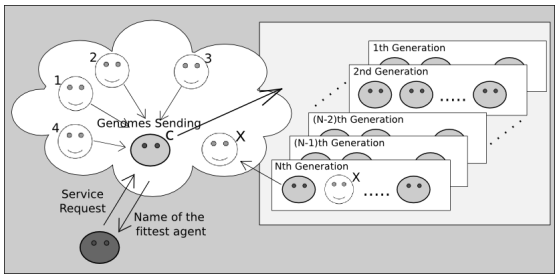

Figure 1: A representation of the evolution procedure. GenomeAgents are represented in white. The agent selected at the end of the procedure is marked with an "X", while the CrosserAgent is marked with a "C".

oration strategies in order to succeed.

The focus of our attention is not on the agent itself, but on its genetic makeup. This genetic makeup can be decomposed in different layers. In the first layer of the genome there are chromosomes. The genome of the agent is composed by two kinds of chromosomes: a Knowledge Chromosome which describes knowledge about the environment and a set of Ability Chromosomes which describe agent's abilities to interact with the world. At a deeper layer we have genes. Each chromosome is made of genes. In the Knowledge Chromosome, each gene describes an element of the knowledge (predicates, concepts and actions), while, genes within Ability Chromosomes describe plans components.

In order to care only about the genome, three hypothesis are made:

- the genome defines agent's capabilities;
- all the interesting agent's features are included in the genome;
- the agent's implementation is unique given an agent's genome.

During the designing process it is essential to define and manipulate the genome. In the definition of the genome an initial set of genes is given. This set does not have to contain the solution but only the elements which allow the creation of the first generation. From this initial set, through manipulation, genes are combined and activated originating new genomes. These genomes have to be evaluated through an objective function for measuring their level of adaptation to the required skill.

The process for manipulating the agent's genome is called "reproduction" and it is described in the next section.

The evolution procedure is represented in Fig.1.

When a service is requested by an agent inside or outside the society, and that is not available, the requesting agent can ask the CrosserAgent to activate an evolution process.

The CrosserAgent starts the procedure by creating a first fake generation composed by all the agents which provide that service with an insufficient quality (or a similar service according to the problem ontology). This generation may include some duplicated individuals if they are not enough (the number of individuals per generation is a parameter of the evolution process). The next generations will contain some agents inherited from the previous generation and some new agents obtained crossing the agents' genomes. During the evolution process two agents merge their chromosomes according to the rules of genetic programming that will be discussed later. The parents transmit their genes to the child which evolves and gains the capability of reaching different results. The crossing procedure operates both at the knowledge level and at the ability level, in which the subject of crossing is the agent behavior.

The evolution process ends when an agent in the current generation provides a satisfying service or if an a priori defined number of generations has been reached. The fittest agent is selected and becomes a member of the society. The CrosserAgent notifies its name to the requesting agent in order to fulfill the service request. This process will be discussed more in details in Sect.3.

In the proposed approach it is possible to see adaptation on both agent and system level because agents adopt their behavior independently from other agents but it is also possible to obtain system adaptation because a whole society of cooperative agents can evolve in order to reach a specific goal (this will be further discussed with regards to the last of the proposed case studies).

In both cases adaptation is strong, because there is no external control and no embedded rules are present in the system to lead adaptation.

After this summary of the proposed architecture the next section focuses on the description of the genome structure.

2 THE GENOME STRUCTURE

Fig.2 shows the genome structure in form of an UML class diagram. It highlights the two main parts in which the genome can be decomposed at a logical level: knowledge and abilities. The division is pointed out through the inclusion of chromosomes in two different packages.

Starting from the higher level, the genome (at the top of Fig.2) contains all the information needed to

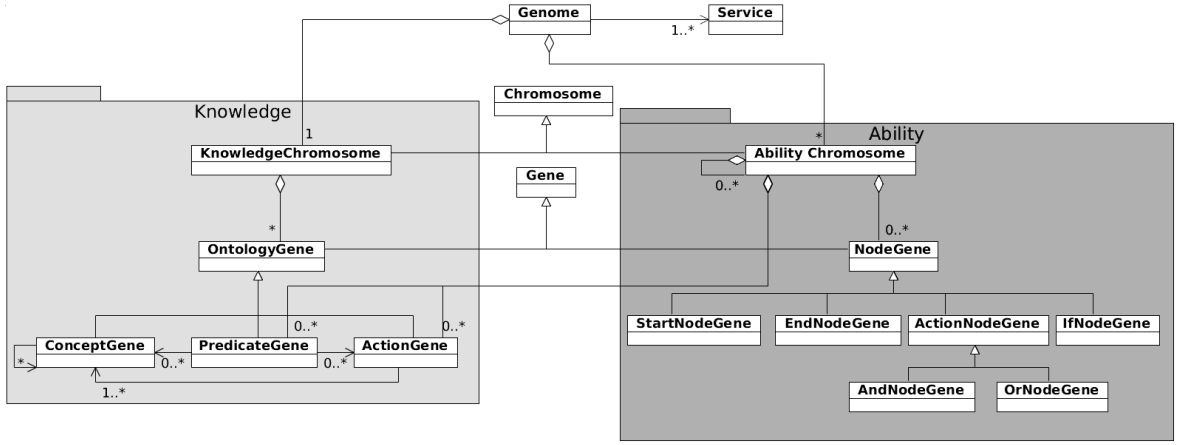

Figure 2: The genome structure. Genome is composed of two kinds of chromosome: Knowledge Chromosome and Ability Chromosome; both are composed by genes.

describe the agent; from this information a new agent can be created.

It contains a set of *Service*s which makes explicit the functions offered by the agent to the external environment.

The genome is composed of chromosomes. It contains a *KnowledgeChromosome* and a set of *AbilityChromosome*s.

The *KnowledgeChromosome* aggregates genes which refer to ontological concepts (*OntologyGene*) and that are specialized in three categories:

- *ConceptGene*: describes an instance of a concept of the ontology, it can refer to other genes of the same kind;
- *ActionGene*: describes an instance of an action of the ontology, it refers to concept genes on which the action operates;
- *PredicateGene*: describes an instance of a predicate of the ontology, it refers to concepts or actions genes which are needed to compute the predicate value.

The *AbilityChromosome* is composed of a set of genes which describe the plan structure (*NodeGene*) and by the contents of these nodes which describe the action associated to them.

The elements can belong to three different kinds: predicate or action genes (indicated as *PredicateGene* and *ActionGene* in Fig.2) or other Ability Chromosomes. Plugging in an Ability Chromosome with a node allows us to associate a behavior, described by another plan, to a node, thus creating a sort of recursive structure.

Nodes in the plan are specialized on the basis of their activation policy. So there are four kinds of nodes:

- *StartNodeGene*: this gene describes the first node in the plan. It can have successors but not ancestors. Neither an action nor a predicate can be associated to this kind of node;
- *EndNodeGene*: this gene describes the last node in the plan. It cannot have successors but only ancestors. As the previous one neither an action nor a predicate can be associated to this kind of node;
- *ActionNodeGene*: this gene describes an action node in the plan. An action is associated to this node and can be described by an Ability Chromosome or an ontological action. It can be specialized in two categories:
  - *AndNodeGene*: action node with a conjunctive activation policy;
  - *OrNodeGene*: action node with a disjunctive activation policy;
- *IfNodeGene*: this gene describes a choice in the plan. It is associated with a predicate gene.

The node classification reported here is inspired by (van Der Aalst et al., 2003).

3 THE AGENT ADAPTATION PROCESS

This section deals with the description of the reproduction process that has been adopted. This process starts when a service is required to the society but no agent (nor collaboration of agents) can provide that or can ensure the fulfillment according to a required level of quality. The final result is one (or sometimes

more than one) agent (or agents) that are able to fulfill the required service. The service evolution process is composed of the following steps:

- definition of the parents' sub-society. This sub-society includes all the agents that will be used in the reproduction process and that contribute with their ability genes and knowledge genes to the definition of the resulting agent;
- creation of the new generation of agents. This generation is composed in three ways by crossing, mutation and elitism;
- evaluation of the results provided by the new agents;
- if one (or more) agent(s) successfully provide the required service the process stops, otherwise another generation is created by using the last generation of agents as a parents' sub-society.

The agent adaptation process is lead by a particular agent in the platform: the CrosserAgent(Fig.1).

The CrosserAgent is responsible for three main functions:

- creation of agent generations;
- execution of new agents;
- evaluation of the results achieved by each agent.

These function are obtained using different behaviors. This agent selects the agents that provide a similar service, it collects their genomes and it uses them to start the agent adaptation process. As already said, this adaptation is achieved by means of a reproduction process where quality of service provides the fitness function.

The CrosserAgent waits for a request and selects the genomes of all the agents which provide the service (or a similar one). Of course, the CrosserAgent is asked to evolve the society because no existing agent is able to perform the service with the required fitness. The CrosserAgent creates each new generation by using three classical Genetic Programming techniques (Banzhaf, 1998)(Mitchell, 1998): elitism, reproduction and mutation.

3.1 Reproduction

The reproduction procedure allows to obtain a new agent from two individuals of the previous generation. Genes of the two original individuals are crossed obtaining a new genetic code.

The crossing procedure can be divided into two steps: knowledge crossing and ability crossing.

Knowledge Crossing Knowledge crossing allows to modify the set of knowledge about the environment modifying parameters that might improve the agent interaction with the world.

Knowledge Chromosomes crossing is performed over each agent's knowledge gene by using four techniques and the result is a new Knowledge Chromosome. The four techniques to generate a new knowledge from the two parents are the following ones:

- **fusion**: the two parents' knowledge genes are melted in a single gene that will contains a weight or algebraic average of the parents' knowledges. In this case, from two knowledges we obtain a single knowledge for the new individual;
- **selection**: one of the parents' knowledges is chosen and copied in the new individual while the other one is discarded;
- **union**: both of the parents' knowledges are copied in the new individual. This technique produce redundant agents if frequently used;
- **copy**: if a particular portion of knowledge is present only in one of the parent, it is copied to the generated agent.

So, for each knowledge gene of the parent agent there will be only one knowledge gene in its son. On the other side, each son's gene is related to at least one parents' knowledge gene. In case of the union operation both of the parents genes are chosen and so two relations are created.

The knowledge crossing procedure is inspired by (Noy and Musen, 1999). To be crossed, two knowledges have to be considered similar. We suppose that throughout all the system, the agents refer to the same ontology although sometimes to different (overlapping or not) parts of it. Two knowledge genes are *similar* if they are instances of the same ontological element.

Once the knowledge crossing is completed, the ability and tasks crossing can be executed.

Ability Crossing The abilities of an agent are represented through plans, composed by nodes, and are labeled with a goal, which indicates the ability purpose. Nodes belong to two different kinds: activity and control. Control nodes include start/end nodes as well as control flow structures used to define the flow of activities. Activity nodes refer to the actions defined in the ontology (an agent cannot perform an action it does not even know nor conceive).

Agents in the platform are provided with a higher-level plan which handles the agent's life-cycle and allows each agent to interact with the external environment. This plan is always crossed by a fusion op-

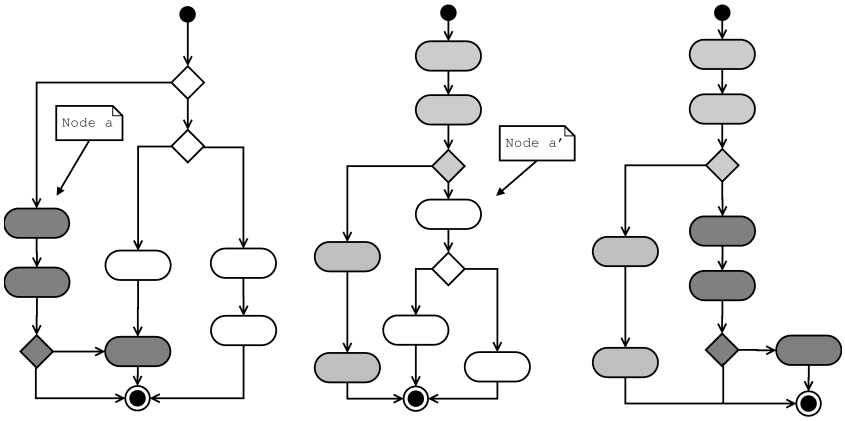

Figure 3: On the left the two plans of the parent agents and on the right the resulting plan. The parts of the plan selected for the crossover procedure are filled.

eration. All the other plans can be crossed also by using the selection, union or copy techniques already described in the previous paragraph.

Since selection, union and copy simply transfers a plan from a parent to its child, the sole operation worth of a discussion is plan fusion and therefore it will be discussed in what follows.

Plan crossing is less complex than knowledge crossing because all the referenced knowledges have already been crossed.

As already reported for knowledge crossing, two plans can be crossed only if they are similar (if they have the same goal). This means that they have the same purpose but it might be provide in different ways and unnecessarily in a satisfactory way. The new plan is obtained by combining parents' nodes. Plan crossing is shown in Fig.3.

The two parents' plans play a different role in this part of the reproduction process. The receiver's agent plan is used as the basis for implanting the contribution from the donor agent. More specifically, the fragment of plan extracted by the receiver agent will be composed with the portion of plan extracted from the donor agent by removing a randomly selected node (node a' in Fig.3) together with all the descending nodes and replacing them with a portion of plan extracted from the donor agent. The plan stub extracted from the receiver agent will always contain the start node while the fragment of plan extracted by the donor agent will be composed of a randomly selected node (node a in Fig.3) and all of its successor nodes (nodes that can be reached by the selected node). The donor's plan will always contain the end node but not the start node. An example of donor and receiver plan fragments are shown respectively in the right and the middle part of Fig.3. The two fragments are linked by replacing the node selected from the receiver's plan with the one selected from the donor's one (and its successors), as shown in the right part of Fig.3.

3.2 Elitism and Mutation

In the presented approach we perform an off the book use of elitism and mutation.

Elitism consists in the election of a group of individuals of the previous generation. After the evaluation of the old generation, a fixed number of agents is elected to become part of the new generation using the tournament selection technique.

Mutation allows to add total unexpected features to the new-born agent. It can occur over a knowledge or a plan gene. If it acts over a knowledge gene, a value can be obtained in a random way. For instance, if the knowledge to be mutated contains a string field, this string can be mutated adding a random sequence of chars inside it. If it acts over plans, a node can be replaced with another similar node or a link between nodes can be added or removed.

4 EXPERIMENTAL RESULTS

In this section some applications of the presented framework are briefly reported: the first example shows a system where an agent-level adaptation occurs when an agent is involved in providing a service. In the last case study several agents cooperate in reaching a goal, thus realizing a system-level adaptation.

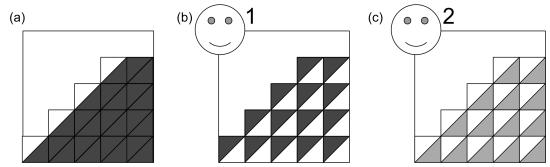

Figure 4: In (a) the target picture. In (b) and (c) the results achieved by two agents of the initial society.

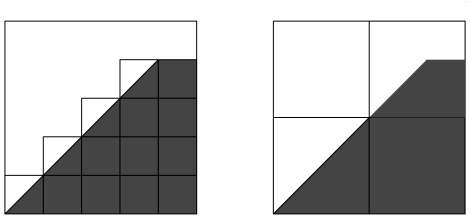

Figure 5: Two agents which provide the required service in different ways.

4.1 Agent-Level Adaptation Example: the Trapezium

In this first case study, the society is composed by agents drawing simple geometrical shapes. The aim of each agent is to reproduce a given picture, in the case study a trapezium (Fig.4a).

The agent divides the picture in small chunks by using a grid and tries to fill each position of the grid according to the guidance provided by the target picture. The work is carried on through several iterations and each iteration is populated by a different generation of agents. The initial generation is composed by two simple agents. Each of them is able to fill in a grid cell with the shape of a triangle: the first agent draws a dark gray triangle oriented towards the up-left corner of the cell, the second agent draws a light gray triangle oriented towards the bottom-right corner of the cell. The figures that are used to fill a sector of the grid are called drawing primitives.

The desired result and the pictures produced by these agents are respectively shown in Fig.4a,b,c. It is clear that none of two agents supply a fulfilling result. Besides, it is evident that neither the simple cooperation of the two agents could solve the problem.

In order to obtain the desired result, a reproduction procedure is needed. During that a lot of agents with different behaviors are created; there can be agents having different sizes for the grid cell, different colors, different shapes and different shape orientation.

There are two factors considered during the evaluation of the service provided by each agent: the degree of similarity in both color and shape with the target picture;

In our experiment, after nine generations, several individuals which perfectly reproduce the desired picture have been created (obviously because of the random characteristic of the new generations production different runs of the experiment may produce different results). Fig.5 reports two examples of such individuals; as it is possible to see, the two agents use a different grid to decompose the target picture.

The test case has been evaluated with different pictures and colors. It has been observed, as it was

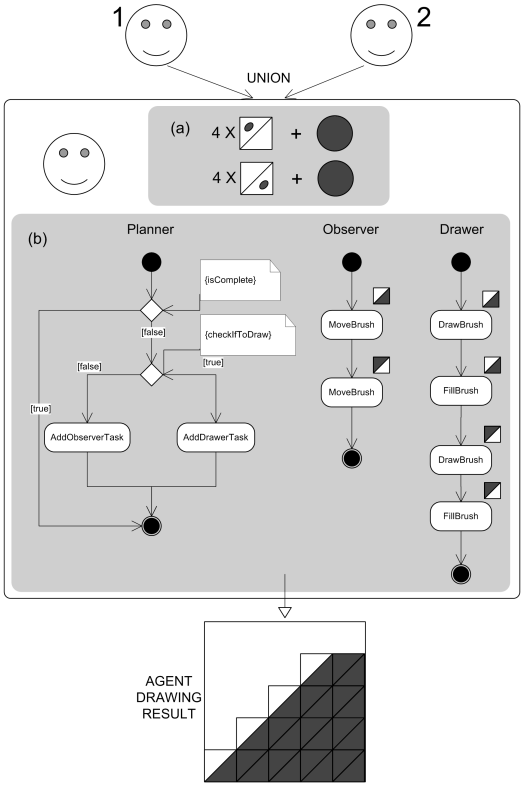

Figure 6: A successful agent. In (a) its knowledge and in (b) its plan and its other tasks obtained through the crossing procedure. On the bottom the resulting picture. The knowledge about the brush shape is the result of an union crossing procedure while color is obtained by a selection crossing procedure.

expected, that the number of generations needed to reach a perfectly fitting outcome grows up with the complexity of the target picture.

Further details about the implementation of this case study will be provided in the following paragraph.

4.1.1 Experiment Discussion

These examples explain how, from two initial agents, a lot of different individuals can be generated. The example is simple so that it was possible to analyze it in details.

We regard this case study as an example of single agent adaptation since even if the whole society is involved in the evolution process, giving the birth to a new agent, only this agent provides the service, without any kind of direct cooperation with the other elements of the society. It could be argued that an indirect cooperation descends from the agreement of each agent involved in the reproduction process in sharing its own genome but at the present we consider that as an ethical rule of the society that enforces all agents to accept to be part of the reproduction (as in the cooperative agents described in (Bernon et al., 2002)). This process can also be regarded as a strong adaptation because it occurs without any external control. The positive results of applying the proposed framework to this case study prove that the approach succeeds in successfully evolve an agent society, that initially is not able to provide a given service, by generating new agents that, inheriting portions of their parents genome, better approximate (or perfectly achieve) the selected goal.

Summarizing, the adopted adaptation process proved to be successful but it is to be noted that the development framework is undoubtedly complex in use and the setup of a new experiment requires a lot of programming. We are at present not really concerned about that. For sure different techniques may be explored (and they will be in the future) but the goal of the current study is evaluating the adoption of the proposed genome-based description of agent capabilities and knowledge.

Another aspect that deserves attention is the scalability of the system. We are in an early stage of development of the system so we did not face this aspect but obviously if the problem to solve is in the same domain it is very easy to write down a fitness function to match the agent behavior to the desired goal, if the system should face a problem in a different space, a new fitness function has to be added to the system and probably a new initial knowledge should be added to the basic agent society.

Moreover the system can approximate the problem solution in a smooth way so that it is possible to stop the evolution of the system to a suboptimal solution. This smooth approximation is related to the chosen fitness function (for instance the fitness function uses a measure of the covered area and color matching) but this means that this system can be used to find an approximate solution to some problems.

5 CONCLUSIONS AND FUTURE WORKS

In this paper we proposed a service adaptation mechanism as an integral part of an agent-oriented adaptive and self-organizing society. As a first step towards our goal we tested an evolutionary system inspired by the Darwinian evolution theory. In the agents of the proposed society all the features are codified in a genome-like structure. In order to improve the quality of a given service, several agents can reproduce themselves creating individuals which better fit the target. These individuals are provided with new capabilities derived by their parents.

The approach has been tested through simple case studies. The application reported in this paper proves that it is possible to obtain a perfectly working agent from original agents which provides a service with a low quality. Using the proposed Genome Framework the problem moves from the implementation of a solution to the definition of the problem domain as a starting point from which is possible to reach the desired result.

The obtained results encourage the development of further release of the proposed framework. The use of a formalization language to describe the genome structure might be the following step in order to lay the groundwork for an agent-oriented language.

REFERENCES

Banzhaf, W. (1998). *Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications*. Morgan Kaufmann.

Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. Wiley.

Bernon, C., Camps, V., Gleizes, M., and Picard, G. (2005). Engineering adaptive multi-agent systems: The adelfe methodology. *Agent-oriented methodologies*, pages 172–202.

Bernon, C., Gleizes, M.-P., Peyruqueou, S., and Picard, G. (2002). ADELFE, A Methodology for Adaptive Multi-Agent Systems Engineering. In *Proc. 3rd International Workshop Engineering Societies in the Agents World (ESAW-02)*, pages 156–169. Springer-Verlag.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agent and Multi-Agent Systems (8)*, 3:203–236.

Cossentino, M., Gaud, N., Hilaire, V., Galland, S., and Koukam, A. (2010). ASPECS: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304.

Dastani, M., van Riemsdijk, B., Dignum, F., and Meyer, J. (2003). A programming language for cognitive agents: Goal directed 3apl. In *Proceedings of the First Workshop on Programming Multiagent Systems: Languages, frameworks, techniques, and tools (ProMAS03) to be held at AAMAS'03*.

Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press.

Gleizes, M.-P., Camps, V., and Glize, P. (1999). A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. In *In proc. of Fourth European Congress of Systems Science*.

Koza, R. (1992). *Genetic Programming: On the Programming of Computers by means of Natural Selection*. MIT Press.

Mitchell, M. (1998). *An introduction to genetic algorithms*. The MIT press.

Nilsson, N. J. (1998). *Artificial Intelligence, a New Synthesis*. Morgan Kaufmann Publishers, Inc.

Noy, N. and Musen, M. (1999). An algorithm for merging and aligning ontologies: Automation and tool support. In *Proceedings of the Workshop on Ontology Management at the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 1999–0799.

Rao, A. and Georgeff, M. (1991). Modeling rational agents within a bdi-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning.*

Robertson, P., Shrobe, H., and Laddaga, R. (2000). Introduction. In *in Proc. of the First International Workshop on Self-Adaptive Software*, pages 1–10. Springer-Verlag.

van Der Aalst, W., Ter Hofstede, A., Kiepuszewski, B., and Barros, A. (2003). Workflow patterns. *Distributed and parallel databases*, 14(1):5–51.