BIO-CORE: Bio-inspired Self-organising Mechanisms Core

Jose Luis Fernandez-Marquez[1], Giovanna Di Marzo Serugendo[1], and Sara Montagna[2]

[1] University of Geneva, Switzerland
joseluis.fernandez@unige.ch, giovanna.dimarzo@unige.ch
[2] Università di Bologna
sara.montagna@unibo.it

Abstract. This paper discusses the notion of "core bio-inspired services" - low-level services providing basic bio-inspired mechanisms, such as evaporation, aggregation or spreading - shared by higher-level services or applications. Design patterns descriptions of self-organising mechanisms, such as gossip, morphogenesis, or foraging, show that these higher-level mechanisms are composed of basic bio-inspired mechanisms (e.g. digital pheromone is composed of spreading, aggregation and evaporation). In order to ease design and implementation of self-organising applications (or high-level services), by supporting reuse of code and algorithms, this paper proposes BIO-CORE, an execution model that provides these low-level services at the heart of any middleware or infrastructure supporting such applications, and provides them as "core" built-in services around which all other services are built.

Keywords: Bio-inspired design patterns, self-organising systems' engineering.

1 Introduction

The current situation in design and development of bio-inspired or decentralised systems can be compared to the situation some 40 years ago when programs were written in the assembly language. To compute an addition, using the assembly language, we need to move both operands into appropriate registers and then apply the addition operator. Today, to implement a system exploiting ant foraging using pheromones, in addition to programming the foraging behaviour, it is also necessary to implement the behaviour of the pheromone itself. An additional inconvenience today with bio-inspired systems resides in the fact that if two applications use pheromones, they both need to implement their own version of the pheromone even though the two applications run in the same node. What we need is the possibility to program bio-inspired systems using high-level operators manipulating core mechanisms as "first-class entities" in a way similar to how high-level programming languages helped abstracting away the implementation of the addition for the programmer and thus favoured the re-use

E. Hart et al. (Eds.): BIONETICS, LNICST 103, pp. 59–72, 2012.

of code. Additionally, as the same addition operator can be used for many different applications, the same pheromone mechanism can be shared by different applications.

Bio-inpired design patterns provide solutions for existing recurrent problems. A design pattern clearly identifies the problem that a mechanism solves, where it has been applied and what are the consequences or emergent behaviour we can observe after applying the pattern. Identifying the bio-inspired mechanisms and their boundaries is a first step towards systematic design and development of self-organising systems. However, we are still far from high-level programming, where designers and programmers concentrate on which mechanisms they want to use, how they want them to be combined, relying on a middleware for the actual distributed implementation of these mechanisms.

In our previous work [Fernandez-Marquez et al., 2011b, Fernandez-Marquez et al., 2011a], we focused on relations between mechanisms and presented several self-organising mechanisms under the form of design patterns, clearly identifying the boundaries of each mechanism, showing how the different patterns relate with each other and how some patterns are composed by others. We classified a set of bio-inspired mechanisms into three different layers: (1) Basic Mechanisms are those mechanisms that cannot be further decomposed and are used to compose other mechanisms or used alone; (2) Composed mechanisms are those mechanisms composed by basic mechanisms; and (3) Top-level mechanisms are those mechanisms exploiting basic and composed ones. From this classification we concluded that most of the bio-inspired patterns are actually using basic bio-inspired mechanisms. Additionally, these basic mechanisms are usually executed by the environment when we consider biological processes (e.g. spreading, aggregation and evaporation of pheromones).

This paper presents BIO-CORE, an execution model for "core bio-inspired services", providing basic bio-inspired mechanisms as built-in services. Such a core set of services, typically running in a middleware, allows the system to execute several composed or top-level bio-inspired mechanism at the same time, all sharing the basic mechanisms implemented inside the core. Executing more than one bio-inspired mechanism at the same time is also a step forward the self-composition and self-adaptation of mechanisms, thus allowing the system to dynamically build new mechanisms as a composition of existing ones or to adapt the existing mechanisms to solve new problems.

Our long-term goal is to provide a programming framework for bio-inspired applications that abstracts away the underlying bio-inspired mechanisms driving the behaviour of the many entities composing the application. To this aim, this paper presents an execution model for such a set of core services, called BIO-CORE, which includes a core set of bio-inspired services, a core's shared data space and core interfaces, as well as a model of interaction between the applications and the core services.

This paper is structured as follows: Section 2 presents previous works existing in the literature. We then propose a Computational Model specifying the

interactions between the agents participating in self-organising applications and the core itself. Section 4 presents the design of BIO-CORE. Section 5 discusses simulation results of two bio-inspired applications using BIO-CORE. Finally, we identify future work.

2 Related Works

Besides development methods focusing on iterative design and extensive simulations [Puviani et al., 2009], three main approaches for engineering self-organising applications are emphasised so far: (1) Self-organising design patterns presenting bio-inspired mechanisms according to software design patterns schemes, identifying how, when and where the patterns can be used, and providing a reusable solution for common recurrent problems; (2) Middleware that provide a support for storing, propagating and maintaining distributed tuple-based structures, facilitating design and development of bio-inspired self-organising systems by providing specific built-in features, and (3) Bio-inspired execution models for communications protocols, which provide new paradigms where self-organisation is reached by reactions among pieces of information occurring in a spontaneous way.

2.1 Self-organising Design Patterns

Self-organising mechanisms expressed as design patterns help identifying the *problems* that each mechanism can solve, the specific *solution* that it brings, the *dynamics* among the entities involved in the pattern and details of their *implementation*.

Several authors have proposed self-organising mechanisms following the software design pattern scheme [Babaoglu et al., 2006, Gardelli et al., 2007, De Wolf & Holvoet, 2007]. However, relations among the patterns are not identified, i.e. the authors do not describe how patterns can be combined to create new patterns or adapted to tackle different problems.

Focusing on the relation between the mechanisms, [Fernandez-Marquez et al., 2011b, Fernandez-Marquez et al., 2011a] show how the different patterns relate with each other and how some patterns are composed by others. [Fernandez-Marquez et al., 2011a] first proposed a decomposition of the Gossip mechanism into two basic mechanisms, Aggregation and Spreading, and then [Fernandez-Marquez et al., 2011b] proposed a decomposition of the Gradient Case into Spreading, Aggregation and Evaporation mechanisms, and showing how at the same time the Gradient mechanism itself is exploited by higher-level mechanisms as Morphogenesis, Quorum Sensing and Chemotaxis. The complex catalogue of self-organising design patterns is presented in [Fernandez-Marquez et al., 2012].

2.2 Middleware for Self-organising Systems

In order to facilitate the design and development of bio-inspired self-organising systems, a series of middleware proposals have been presented recently in

the literature. TOTA (tuple On The Air) [Mamei & Zambonelli, 2005] provides a support for storage, propagation and maintenance of distributed tuple-based data structures. Similar approaches include MeshMDL [Herrmann, 2003] and Lime [Dept & Murphy, 2001]. Those proposals are all based on tuple space technology (i.e. shared spaces where agents indirectly exchange information), thus providing a way to implement indirect communication between agents.

While these middleware ease the task of the programmer of a self-organising system by taking care of the execution of some low-level mechanism, they still require that the programmer carefully describes and programs the mechanisms' behaviour (e.g. propagation of a gradient in a distributed system), preventing the reuse of code.

2.3 Bio-inspired Execution Models for Communications Protocols

Recent bio-inspired execution models for communications protocols provide a new paradigm, inspired by molecular processes, where self-organisation is reached by spontaneous reactions among pieces of information. The chemical metaphor was originally proposed by Gamma in 1986 as a formalism for the definition of programs without artificial sequentiality [Banâtre et al., 2001]. The basic idea underlying the formalism was to describe computation as a form of chemical reactions on a collections of individual pieces of data. Based on this chemical metaphore, we highlight two different execution models for communications protocols: (1) Fraglets [Tschudin, 2003] unify code and data into a single unit, the so called "fraglet". This single unit is a string composed of symbols values. The first symbol value is a tag that represents the instruction that is going to be executed over the fraglet. Fraglets are stored in a fraglet store in the same way as tuples are stored in a shared tuple-space; (2) Rule-based Systems [Dressler et al., 2009], where self-organisation is reached by a given set of rules acting on passive data packets. Rule-based systems follow an approach similar to the fraglets, however, a piece of data does not contain code, and a rule can operate over several pieces of data in the same execution.

The execution model proposed in this paper follows both fraglets and rule-based systems. BIO-CORE provides basic bio-inspired mechanisms under the form of "core bio-inspired services" implemented by rules. This enables to process several tuples at the same time and, thus allows complex ways of aggregation besides other operations. Moreover, pieces of data upon which services apply contain properties, similar to tags, indicating to the engine how that piece of data must be processed.

These services are intended to equip the designers and programmers of higher-level self-organising services or applications with a set of ready-to-use low-level mechanisms, whose implementation and execution is taken in charge by a specific middleware.

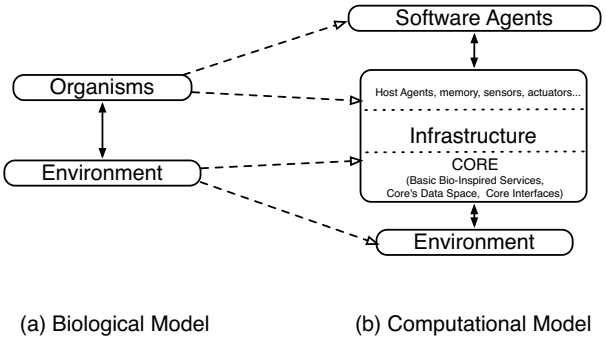


Fig. 1. Relevant entities of the biological and computational models

3 The Computational Model

A bio-inspired computational model, for describing the interactions between the entities participating in self-organising systems, was presented by [Fernandez-Marquez et al., 2011a]. The computational model is as follows: *Agents* are autonomous and pro-active software entities running in a Host. The *Infrastructure* is composed of a set of connected Hosts and Infrastructural Agents. A *Host* is an entity with computational power, communication capabilities, and may have sensors and actuators. Hosts provide services to the agents. An *Infrastructural Agent* is an autonomous and pro-active entity, acting over the system at the infrastructure level. Infrastructural agents may be in charge of implementing those environmental behaviours present in nature (e.g. spreading, aggregation and evaporation of pheromones). Finally, the *Environment* is the physical space where the infrastructure is located.

In this paper we extend the computational model presented in [Fernandez-Marquez et al., 2011a]. The new model, showed in Figure 1, adds the notion of BIO-CORE, which provides basic bio-inspired mechanisms, ready-to-use as "first-class" entities by higher-level services or applications (simply called CORE in Figure 1. BIO-CORE aims at decoupling the agents from the environment's behaviour by providing a *virtual environment* (as opposed to the actual real-world environment) where more than one bio-inspired algorithm can be executed at the same time, reusing implementation and enabling the creation of new bio-inspired mechanisms for solving new problems or dynamically adapting existing ones. BIO-CORE is composed of: 1. a Core's Data Space, where agents deposit and retrieve data; 2. a set of Basic Bio-Inspired Services implementing basic bio-inspired mechanisms through rules applying on data deposited in the data space; and 3. Core Interfaces providing primitives for the agents to interact with BIO-CORE, for accessing other cores in neighbouring nodes, and for accessing sensors and actuators of the local node. BIO-CORE is embedded into each device participating in the system.

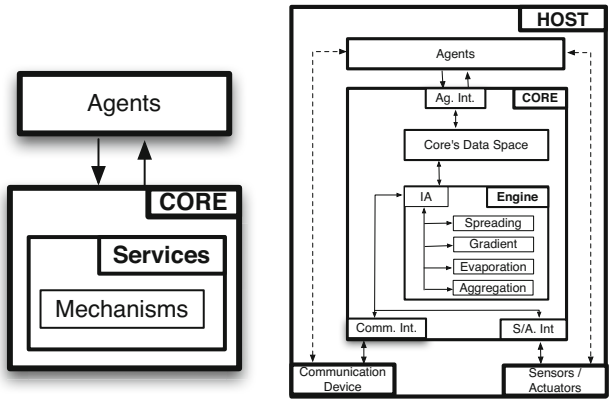
4 BIO-CORE Design

BIO-CORE encapsulates a set of low-level services providing bio-inspired mechanisms that applications or higher-level bio-inspired services can exploit and rely on. BIO-CORE provides a set of primitives for these applications and high-level services to interact with the low-level services, clearly separating the responsibilities of the agents from those of the environment. BIO-CORE also provides a shared data space for services and applications to exchange data and interact with each other.

BIO-CORE advantages are: (1) Several applications or higher-level bio-inspired services can be running in the same virtual environment re-using the services provided by BIO-CORE (i.e. reusing code); (2) It makes it easier to model and implement bio-inspired applications, since agents' behaviour is decoupled from the environment, and the low-level services provided by BIO-CORE are still running in the middleware, ready to be executed on demand; (3) Since several bio-inspired mechanisms can be running at the same time, BIO-CORE is a first step towards self-composition of mechanisms.

The basic bio-inspired services provided by BIO-CORE are those identified during our work on defining design patterns for bio-inspired mechanisms, where we expressed high-level bio-inspired mechanisms as a composition of lower-level ones [Fernandez-Marquez et al., 2011b, Fernandez-Marquez et al., 2011a]. Namely, the mechanisms provided by BIO-CORE are: Spreading, Evaporation, Aggregation and Gradients. These mechanisms present common characteristics: (1) in biological processes they are mainly executed by the environment; (2) they occur both in macro- and micro-level systems (e.g. spreading mechanism can be found in ants colonies coordination or in signaling pathways between cells); and (3) they are at the basis of more complex self-organising mechanisms (e.g. gossip is a combination of aggregation and spreading).

Figure 2(a) shows the interactions between agents belonging to applications or to high-level bio-inspired services and BIO-CORE, which contains low-level services providing basic bio-inspired mechanisms. Figure 2(b) describes the relations between BIO-CORE and agents belonging to applications inside a given host. Agents have access to the *Communication Device*, *Sensors and Actuators* provided by the Host. Agents communicate with BIO-CORE using the *Agent Interface*. This interface is directly connected to the *Core's Data Space*, a shared data space allowing agents to deposit and retrieve data through specific primitives. The data deposited into the Core's Data Space is processed by the *Core Engine*. Namely, the Core Engine is composed of an Infrastructural Agent (IA), and a set of rules implementing the core services. The IA has access to the Communication Device and Sensors/Actuators provided by the Host through the *Communication Interface* and *Sensors/Actuators Interface* respectively. It allows data from the Core's Data Space to be sent to neighbouring Core's Data Spaces, to provide access to Sensor's data by inserting Sensor reads into the Core's Data Space, and to instruct specific Host's actuators to perform some action (e.g. move the wheels of a robot in a certain direction).



(a) (b)

Fig. 2. System's Architecture

4.1 BIO-CORE Engine

The BIO-CORE engine is composed of the Infrastructural Agent (IA) and a set of rules that implement the low-level services offered by BIO-CORE. Basically, the IA is responsible for applying the rules to the set of data stored in the Core's Data Space according to the data's properties. The IA is also responsible for managing the internal interfaces (e.g. sending or receiving data to other Core's Data Spaces or acting over the sensor or actuators).

The rules are transitions that provide BIO-CORE with chemical reactions similar to chemical machine models [Banâtre et al., 2001, Dressler et al., 2009] and make the IAs act over the Core's Data Space in a completely distributed and decentralised way. Rules provide a simple way to define the environment's behaviour, emulating the laws of nature, and providing the environment with an autonomous and proactive behaviour (i.e. Applications do not actually call those rules, rules are applied dynamically when necessary). Indeed, the behaviour of BIO-CORE over the data stored in the Core's Data Space depends on the data's properties, in the same way as in the real word, the environment acts over the entities depending on their properties and the nature's laws (e.g. gravity, diffusion, aggregation, etc.).

Figure 3 shows these basic mechanisms provided by BIO-CORE, and how high-level bio-inspired mechanisms are composed from these core mechanisms.

A full description, under the form of design patterns, can be found in [Fernandez-Marquez et al., 2011b, Fernandez-Marquez et al., 2011a].

Spreading Pattern. The Spreading Pattern [Fernandez-Marquez et al., 2011a] is a basic pattern for information diffusion/dissemination. The Spreading Pattern progressively sends information over the system using direct communication among agents, allowing the agents to increment the global knowledge of the system by using only local interactions.

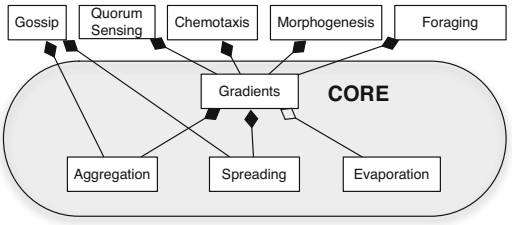


Fig. 3. Core's Patterns

Rule: A copy of the information, received or held by an agent, is sent to neighbours and propagated over the network. Information spreads progressively over the system and reduces the lack of knowledge of the agents while keeping the constraint of the local interaction.

The Spreading Pattern is one of the most used in the literature, and it appears in important higher-level bio-inspired patterns, such as, Morphogenesis Pattern, Quorum Sensing Pattern, Chemotaxis Pattern, Gossip Pattern, and Gradient Pattern [Fernandez-Marquez et al., 2012].

Aggregation Pattern. The Aggregation Pattern [Gardelli et al., 2007], is a basic pattern for information fusion. The dissemination of information in large-scale systems deposited by the agents or taken from the environment may produce network and memory overload, thus, the necessity of synthesising the information. The Aggregation Pattern reduces the amount of information in the system and assesses meaningful information.

Rule: Aggregation consists in locally applying a fusion operator to process the information and to synthesise macro information. This operator can take many forms, such as filtering, merging, aggregating, or transforming. In BIO-CORE, the aggregation service fuses the information present in the Core's Data Space. Information comes from the real-world environment (through sensors reads like temperature, humidity, etc.), from other agents (i.e. through communication with other core's space) or from agents running in the Host interacting directly with the Core's Data Space. The aggregation process terminates when aggregation leads (through one or more applications of the aggregation law) to an atomic information.

The Aggregation Pattern used in conjunction with the Evaporation and Spreading Patterns is at the basis of the digital pheromone and thus the Foraging Pattern.

In BIO-CORE enabling simultaneously Aggregation, Evaporation and Spreading services on a piece of data allows: to create digital pheromones; to perform Gossip; and to run applications that exploit gradients.

Evaporation Pattern. Evaporation is a pattern that helps to deal with dynamic environments where information used by agents can become outdated. In real world scenarios, the information changes with time and its detection,

prediction, or removal is usually costly or even impossible. Thus, when agents have to adapt their behaviour according to information from the environment, information gathered recently must be more relevant than information gathered a long time ago.

Rule: Evaporation is a mechanism that progressively reduces the relevance of information.

In BIO-CORE, enabling the Evaporation and Gradient services allows to build dynamic gradients, making them adaptable to topology changes. The Evaporation service used in conjunction with the Spreading and Aggregation services allows to create digital pheromones. Used on its own, the Evaporation service allows the data deposited in the Core's Data Space, and to which it applies, to become outdated and to be removed by the IA.

Gradient Pattern. The Gradient Pattern focuses on large scale system, where agents suffer from lack of global knowledge to estimate the consequences of their actions or the actions performed by other agents beyond their communication range. Using the Gradient Pattern, information is spread from the location where it was initially deposited and it is aggregated when it meets other information. Thus, agents that receive gradients have information that come from beyond their communication range, increasing the knowledge of the global system not only with gradient's information but also with the direction and distance of the information source.

Rule: The Gradient Pattern is an extension of the Spreading Pattern where the information is propagated in such a way that it provides an additional information about the sender's distance and direction. Additionally, the Gradient Pattern uses the Aggregation Pattern to merge different gradients created by different agents or to merge gradients coming from the same agent but through different paths.

4.2 BIO-CORE Data

We could envisage different ways of dynamically applying services on data. Here, we consider the use of *properties* attached to data. Indeed, data are passive entities that, once deposited into the Core's Data Space, are subject to the actions of BIO-CORE services (e.g. modified, cloned or removed) depending on their properties. Services provided by BIO-CORE are then activated on demand by the applications or higher-level services by modifying the data's properties. The actual activation occurs through the Infrastructural Agent that takes care of identifying the appropriate service. The interactions between the Infrastructural Agent and the data are defined by the set of low level services performed in the BIO-CORE Engine.

Data properties are defined in table 1. Basically the idea is that if a piece of data has the Evaporate property set to true, then the Evaporation service will be executed over the data. A piece of data can have several properties equals to true at the same time, thus enabling multiple services to act over it (e.g. data that is spread, aggregated, evaporated and subject to gradient could be a

Table 1. BIO-CORE Data's Properties

Property	**Description**
ID	Unique identifier
Evaporate	Activate the Evaporation service
Aggregate	Activate the Aggregation service
Spread	Activate the Spreading service
Gradient	Activate the Gradient service. Automatically this property also enables the spread and aggregation properties.
Information	Actual information stored in the data

digital pheromone; data that is spread and aggregated may be the subject of gossip mechanism; data that is subject to gradient and evaporated can be used to create dynamic gradients, etc...). Moreover, data contains also parameters for defining the probability of evaporation, the kind of evaporation, the kind and frequency of aggregation, etc.

4.3 BIO-CORE Interfaces

BIO-CORE defines three different interfaces: (1) an external interface for the Agents to communicate with the Core's Data Space. Agents adapt their behaviour according to the data retrieved from the Core's Data Space and conversely, by inserting appropriate data into the Core's Data Space delegate environmental responsibilities, such as, spreading, aggregation or evaporation to BIO-CORE; (2) an internal interface for exchanging data among Core's Data Space of different Hosts; and (3) an internal interface for the Core's Data Space to communicate with Sensors and Actuators of the Host.

Data is exchanged between the Core's Data Space and the agents, and vice-versa, through external interface primitives for creating, depositing and retrieving data.

5 Simulation Results

This section shows the design and development of two bio-inspired applications using BIO-CORE. The first application, "Reaching an agreement" is based on the Gossip mechanism. The real-world environment consists of a set of Hosts (or nodes) each equipped with an initial random colour. The goal is to reach an agreement on the colour of the nodes, where all nodes share the same colour.

The second application is "Regional Leaders Election", where Spreading and Evaporation services are used to assign the leader and member roles to the nodes participating in the systems. The goal of this section is to analyse the design and feasibility of these applications using BIO-CORE.

5.1 Reaching an Agreement

As it was presented in [Fernandez-Marquez et al., 2011a], Gossip is a composed mechanism, where the Spreading and the Aggregation mechanisms are used

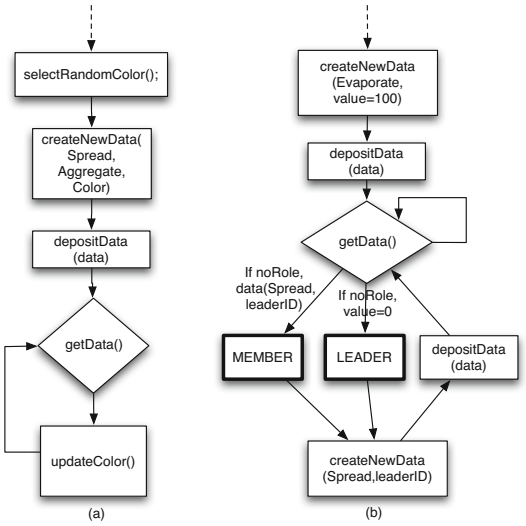
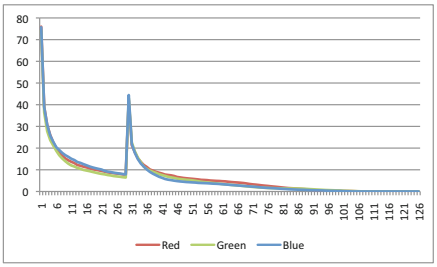


Fig. 4. (a) Reaching an agreement (b) Regional Leaders Election

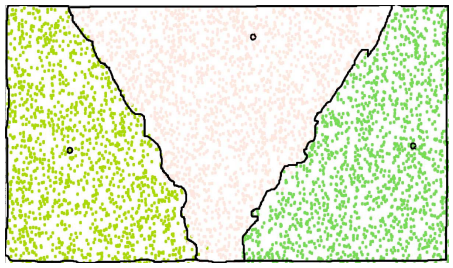
simultaneously. In the gossip process the information is sent over the network using the Spreading mechanism and it is aggregated at each node with the local information by using the Aggregation mechanism.

In this simulation the system has to reach an agreement on the nodes' colour. Initially, each node has a random colour and during the simulation the colour of the nodes must converge to the same colour. Moreover, if new nodes appear the system must be able to deal with it and reach the agreement again taking into account new colours: if the new nodes are blue, the whole system will change the colour towards a blue shade. The aggregation operator used in the simulation is the average between the three components of colour (i.e. red, green and blue, where each component is in [0-255]). This simulation executed with 500 nodes randomly placed in a 1200x700 meters bi-dimensional space, the mobility pattern used is random walk and the communication range for each node is 60 meters.

Figure 4(a) shows the flow diagram for each agent in the "Reaching an agreement" application. The different steps are presented as follows: (1) each agent in each node initially chooses a colour at random; (2) it creates a Core data, with properties Spread, Aggregate set to true and the colour as Information; (3) the data is deposited into the Core's Data Space; (4) Spreading and Aggregation services of BIO-CORE then act on these data: all nodes spread the data, on each node aggregation then acts on all data whose Aggregate property is set to true, averaging the colour Information (only one piece of data per node will remain, containing the average colour of the node and that of its neighbours); (5) periodically agents check their local Core's Data Space, and retrieve new aggregated core data, (6) each agent then updates its colour to the one contained in the Information field provided by the retrieved Core data and returns to step 5.



(a) Reaching an agreement: STD



(b) Regional Leaders Election

Fig. 5. Simulation results

Figure 5(a) shows the Standard Deviation (STD) of the three colour's components (i.e. red, green and blue) over the nodes. At the beginning the STD is maximum because of the set of colours is randomly chosen. The simulation shows how in the first steps the STD decreases. At step 30, new nodes with random colours are added into the system. These new nodes increment the STD but the system easily overcomes this environmental change and reaches the final agreement after few steps, reaching an STD equals to 0 (i.e. all the nodes share the same colour). It is out of the scope of this work to improve the performance of the gossip algorithm. We are interested here in showing how basic bio-inspired services apply on data provided by applications or higher-level services.

5.2 Regional Leaders Election

The regional leaders election was presented as an application example exploiting amorphous computing primitives in [Abelson et al., 2000]. The goal is to split the network into disjoint groups each led by one node, following a decentralised and distributed process. Initially nodes have no identified role. Once the system converges, the network is broken up into contiguous domains each composed of one leader and members.

Figure 4(b) shows the flow diagram of the agents behaviour in the Regional Leaders Election process using BIO-CORE. The steps are as follows: (1) each agent creates a Core data with properties Evaporate set to true and Information equals to 100 representing the relevance value; (2) each agent deposits this data into the Core's Data Space; (3) BIO-CORE periodically decreases the relevance value; (4) if an agent has no assigned role yet, it checks the local Core's Data Space periodically. If the Information (relevance value) of the Core data has reached 0 the agent decides to become a leader, it then creates and deposits a new Core data in the Core's Data Space with properties Spread set to true, including its agentId in the ID properties. BIO-CORE will then spread this new Core data. If the Information Core data is not yet 0, but another Core Data with information (Spread, agentId) is found in the local Core's space, the agent decides to become a member of the leader whose id is equal to agentId.

Both applications implemented using BIO-CORE have reached the desired emergent behaviour. The flow diagrams show that both design have been reduced from the original ones, in the sense that most of the responsibilities are performed by BIO-CORE. Moreover, the BIO-CORE services have been reused by two applications making easier not only the design phase, but also the implementation. It exists a wide number of applications based on the basic mechanisms implemented inside the BIO-CORE. Thus, BIO-CORE is a first step to create systems where bio-inspired applications can be executed, reusing code, sharing the same virtual environment as biological process share different mechanisms in a real environment.

6 Conclusions

This paper presents BIO-CORE, an execution model for a set of low-level services providing basic bio-inspired mechanisms that applications or high-level bio-inspired services can exploit and rely on. BIO-CORE provides primitives for those applications interacting with the low-level services, clearly separating the responsibilities of the agents from those of the environment. BIO-CORE design is presented based on shared data space technology and rules. This is one way of considering the implementation of BIO-CORE, other techniques or existing middleware could be used. This paper focuses on engineering bio-inspired self-organising systems, providing a core for designing and implementing bio-inspired applications. BIO-CORE feasibility is analysed using two different applications, "Reaching an Agreement" and "Regional Leaders Election", where both simulations have reached the desired emergent behaviour.

Future works will focus on three different directions: (1) extending the catalogue of mechanisms and the relations between them, in order to extend BIO-CORE giving support for the maximum number of bio-inspired applications; (2) implementing BIO-CORE for Android OS, analysing different implementations, and (3) even when the services' parameters can be set up by passing arguments down from application to the BIO-CORE, we plan to work on self-composition of services and self-adaptation of parameters in order to avoid a complex parameterisation of the services and to provide a better performance against environmental changes.

Acknowledgments. This work has been supported by the EU-FP7-FET Proactive project SAPERE Self-aware Pervasive Service Ecosystems, under contract no.256873.

References

[Abelson et al., 2000] Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight Jr., T.F., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R.: Amorphous computing. Commun. ACM 43(5), 74–82 (2000)

[Babaoglu et al., 2006] Babaoglu, O., Canright, G., Deutsch, A., Caro, G.A.D., Ducatelle, F., Gambardella, L.M., Ganguly, N., Jelasity, M., Montemanni, R., Montresor, A., Urnes, T.: Design patterns from biology for distributed computing. ACM Trans. on Autonomous and Adaptive Sys. 1, 26–66 (2006)

[Banâtre et al., 2001] Banâtre, J.-P., Fradet, P., Le Métayer, D.: Gamma and the Chemical Reaction Model: Fifteen Years After. In: Calude, C.S., Pun, G., Rozenberg, G., Salomaa, A. (eds.) Multiset Processing. LNCS, vol. 2235, pp. 17–44. Springer, Heidelberg (2001)

[De Wolf & Holvoet, 2007] De Wolf, T., Holvoet, T.: Design Patterns for Decentralised Coordination in Self-organising Emergent Systems. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS (LNAI), vol. 4335, pp. 28–49. Springer, Heidelberg (2007)

[Dept & Murphy, 2001] Dept, A.M., Murphy, A.L.: LIME: A Middleware for Physical and Logical Mobility. In: Proc. of the 21st Int. Conf. on Distributed Computing Systems, ICDCS 2001, pp. 524–533. IEEE Computer Society (2001)

[Dressler et al., 2009] Dressler, F., Dietrich, I., German, R., Krüger, B.: A rule-based system for programming self-organized sensor and actor networks. Comput. Netw. 53, 1737–1750 (2009)

[Fernandez-Marquez et al., 2011a] Fernandez-Marquez, J.L., Arcos, J.L., Di Marzo Serugendo, G., Casadei, M.: Description and Composition of Bio-Insp. Design Patterns: the Gossip Case. In: Int. Conf. on Engineering of Autonomic and Autonomous Syst. (EASE), pp. 87–96. IEEE Computer Society (2011a)

[Fernandez-Marquez et al., 2011b] Fernandez-Marquez, J.L., Arcos, J.L., Di Marzo Serugendo, G., Viroli, M., Montagna, S.: Description and Composition of Bio-Inspired Design Patterns: The Gradient Case. In: Workshop on Bio-Insp. and Self-*Algorithms for Distributed Systems (BADS), pp. 25–32. ACM (2011b)

[Fernandez-Marquez et al., 2012] Fernandez-Marquez, J.L., Di Marzo Serugendo, G., Montagna, S., Viroli, M., Arcos, J.L.: Description and Composition of Bio-Inspired Design Patterns: a complete overview. Natural Computing Journal (invited paper, submitted, 2012)

[Gardelli et al., 2007] Gardelli, L., Viroli, M., Omicini, A.: Design Patterns for Self-Organizing Multiagent Systems. In: De Wolf, T., Saffre, F., Anthony, R. (eds.) 2nd International Workshop on Engineering Emergence in Decentralised Autonomic System (EEDAS), pp. 62–71. CMS Press (2007)

[Herrmann, 2003] Herrmann, K.: MESH Mdl " A Middleware for Self-Organization in Ad Hoc Networks. In: Proc. of the 23rd Int. Conf. on Distributed Computing Systems, ICDCSW 2003. IEEE Computer Society (2003)

[Mamei & Zambonelli, 2005] Mamei, M., Zambonelli, F.: Programming stigmergic coordination with the TOTA middleware. In: Proc. of the 4th Int. Joint Conf. on Autonomous Agents and Multiagent Systems, AAMAS, pp. 415–422. ACM (2005)

[Puviani et al., 2009] Puviani, M., Di Marzo Serugendo, G., Frei, R., Cabri, G.: Methodologies for Self-Organising Systems: A SPEM Approach. In: Proc. of the 2009 IEEE/WIC/ACM Int. Joint Conf. on Web Intelligence and Intelligent Agent Technology, WI-IAT, pp. 66–69. IEEE Computer Society (2009)

[Tschudin, 2003] Tschudin, C.F.: Fraglets - a Metabolistic Execution Model for Communication Protocols. In: In Proceeding of 2nd Annual Symposium on Autonomous Intelligent Networks and Systems (AINS), Menlo Park (2003)