Description and Composition of Bio-Inspired Design Patterns: The Gradient Case

Jose Luis Fernandez-Marquez, Josep Lluis Arcos
IIIA - CSIC, Campus UAB
Catalonia, Spain
fernandez@iiia.csic.es, arcos@iiia.csic.es

Giovanna Di Marzo Serugendo
CUI, University of Geneva
7, Rte de Drize
CH-1227 Carouge
giovanna.dimarzo@unige.ch

Mirko Viroli, Sara Montagna
Alma Mater Studiorum – Università di Bologna
via Venezia 52
47521 Cesena, Italy
sara.montagna@unibo.it, mirko.viroli@unibo.it

ABSTRACT

Bio-inspired mechanisms have been extensively used in the last decade for solving optimisation problems and for decentralised control of sensors, robots or nodes in P2P systems. Different attempts at describing some of these mechanisms have been proposed, some of them under the form of design patterns. However, there is not so far a clear catalogue of these mechanisms, described as patterns, showing the relations between the different patterns and identifying the precise boundaries of each mechanism. To ease engineering of artificial bio-inspired systems, this paper describes a group of bio-inspired mechanisms in terms of design patterns organised into different layers and exemplified through the description of 7 mechanisms: 3 basic ones (Spreading, Aggregation, and Evaporation), a mid-level one (Gradient) composed from basic ones, and 3 top-level ones (Chemotaxis, Morphogenesis, and Quorum sensing) exploiting Gradient.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence:[Coherence and coordination][Multi-Agent Systems]

General Terms

Algorithms,Theory

Keywords

Self-Organising Systems, Bio-Inspired Mechanisms

1. INTRODUCTION

During the last decade, the engineering of self-organising systems has attracted many researchers. Bio-inspired algorithms have been transposed in ad hoc ways into specific engineering systems. In an attempt to classify and describe the corresponding self-organising or bio-inspired mechanisms, different researchers have proposed descriptions such as the description of bio-inspired primitives [19], taxonomies for classifying self-organising mechanisms [15], or described some mechanisms under the form of design patterns [24, 7, 1].

However, these efforts are still fragmented: no clear catalogue of these patterns is provided, interpretations vary among authors, the relations among patterns and their precise boundaries are not described.

This paper is part of a larger work aiming at providing a complete catalogue of bio-inspired patterns organised into different layers. At the bottom layer, we propose basic design patterns that define fundamental biological behaviours, i.e. basic patterns that describe mechanisms at the level of the environment's locality. At the middle layer we propose patterns describing more complex behaviours and interactions obtained by composing or extending bottom level patterns, describing the establishment of global structures in the environment. Finally, top layer patterns describe processes exploiting middle layer patterns and how agents reach global coordination. To describe the behaviour of each pattern and their interactions between the entities, we propose a computational model inspired by the biological model that covers a wide set of applications found in the literature.

This way of cataloguing the mechanisms supports the creation of new mechanisms and the adaptation of existing ones to solve new problems. Moreover, this structure dissociates and identifies clearly the different biological behaviours underlying each pattern. Each pattern describes how and when they should be applied. Thus, depending on the existing problems the mechanisms can be easily chosen and combined to engineering self-organising systems.

Recently, in [13] we presented the Gossip Pattern as a composition of the Aggregation and Spreading Patterns. As a complement to this work, this paper presents the patterns related to the Gradient mechanism. We present first the computational model on which the patterns are based. We then express the three basic patterns (Spreading, Aggregation, and Evaporation), followed by the mid-level pattern resulting from the composition of the basic patterns (Gradient), and the top-level patterns exploiting Gradient (Chemotaxis, Morphogenesis, and Quorum Sensing). The paper ends with some concluding remarks. Figure 1 shows the different relations among the patterns linked to the Gradient Pattern.

BADS'11 , June 14, 2011, Karlsruhe, Germany

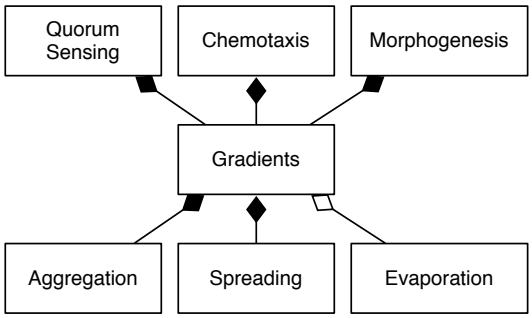
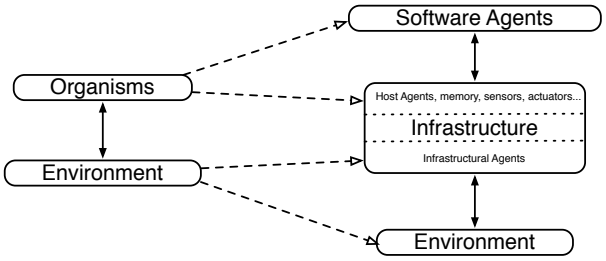


Figure 1: Patterns and their Relationships



(a) Biological Model (b) Computational Model

Figure 2: Relevant entities of the biological and computational models.

2. COMPUTATIONAL MODEL

The biological model is composed of *Organisms* and *Environment*. The computational model mirrors the biological one and its entities are as follows. The *Agents* are autonomous and pro-active software entities running in a Host. The *Infrastructure* is composed by 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. Finally, the *Environment* is the space where the infrastructure is located.

A system is then composed of Agents, Infrastructure, Infrastructural Agents, Hosts, and Environment. Behaviour of Agents and Infrastructural Agents is defined by a set of rules (hereafter referred to as transition rules), while Hosts are defined by the interface they provide.

3. BOTTOM LAYER PATTERNS

Patterns are described following Table 1. Since Spreading and Aggregation patterns have been fully defined in [13], we shorten their description here and keep only the description fields relevant to this paper.

3.1 Spreading Pattern

The Spreading Pattern is a bottom level 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.

Name	The pattern's name.
Aliases	Alternative names used for the same pattern.
Problem	Which problem is solved by this pattern and situations where the pattern may be applied.
Solution	The way the pattern can solve the problems.
Inspiration	Biological process inspiring the pattern.
Forces	Prerequisites for using the pattern and aspects of the problem that lead the implementation, including parameters (trade-offs).
Entities	Entities that participate in the pattern and their responsibilities. Entities are agents, infrastructural agents, and hosts.
Dynamics	How the entities of the pattern collaborate to achieve the goal. Typical scenario describing the run-time behaviour of the pattern.
Environment	Infrastructural requirements of the pattern.
Implem./ Simulation	Hints of how the pattern could be implemented, including parameters to be tuned.
Known Uses	Examples of applications where the pattern has been applied successfully.
Consequences	Effect on the overall system design.
Related Patterns	Reference to other patterns that solve similar problems, can be beneficially combined or present conflicts with this pattern.

Table 1: Description fields.

Aliases: also known as diffusion, dissemination, flooding, broadcast, epidemic spreading, or propagation.

Problem: Agents' reasoning suffers from the lack of knowledge about the global system.

Solution: 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.

Forces: If spreading occurs with high frequency, the information spreads over the network quickly but the number of messages increases.

Entities-Dynamics-Environment: Entities involved in the spreading process are Hosts, Agents, and Infrastructural Agents. The spreading process is initiated by an agent that first spreads the information. When information arrives to the neighbouring hosts, agents or infrastructural agents in those hosts re-send the information. The process continues even when all the hosts in the system have the information.

The behaviour of each pattern is informally defined by a set of abstract *transition rules* of the kind "$name :: data \xrightarrow{rate} action\ if\ condition$", which trigger if data occurs locally (and the optional condition is satisfied), and accordingly execute action. When the reaction could be continuously executed, the (optional) rate is used to express the frequency at which it fires in each location.

The transition rule for the Spreading Pattern (1) specifies that information in input *inf* is sent to a set of neighbours.

$$spreading :: inf \rightarrow send(inf, neighbours) \qquad (1)$$

Related Patterns: It is used in the Morphogenesis Pattern (Section 5.2), the Quorun Sensing Pattern (Section 5.3),

the Chemotaxis Pattern (Section 5.1), the Gossip Pattern ([13]), and the Gradient Pattern (Section 4.1).

3.2 Aggregation Pattern

The Aggregation Pattern [14], is a bottom level 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 synthesizing the information. The Aggregation Pattern reduces the amount of information in the system and assesses meaningful information.

Alias: Information fusion.

Problem: In large systems, excess of information produced by the agents may produce network and memory overloads. Information must be distributively processed to reduce information and to assess meaningful information.

Solution: Aggregation consists in locally applying an aggregation operator to process the information and to synthesize macro information. This operator can take many forms, such as filtering, merging, aggregating, or transforming.

Forces: Aggregation applies on all the information available locally or only on part of that information. The parameter involved is the amount of information that is aggregated; it relates to the memory usage in the system.

Entities-Dynamics-Environment: Aggregation is executed either by Agents or by Infrastructural Agents. In both cases, agents aggregate the information they access locally, comming from the environment or from other agents. Information that comes from the environment is typically read by sensors (e.g. temperature, humidity, etc.). The aggregation process terminates when aggregation leads (in one or more application of the law) to an atomic information.

The transition rule for aggregation (2) is as follows: information in input (possibly a set) I is transformed into a new set of information through an aggregation operator $op()$—I' stands for what I becomes after application of the rule.

$$aggregation :: I \rightarrow I' = op(I) \tag{2}$$

Implementation: Available information takes the form of a stream of events. Aggregation operators are classified into four different groups [6]: *Filter*, selects a subset of the received events; *Transformer*, changes the type of the information received in input; *Merger*, unifies all information received and outputs the information received as a single piece of information; *Aggregator*, applies a specific operation (e.g. max, min or avg) to one or more incoming inputs.

Related Patterns: The Aggregation Pattern can be implemented together with Evaporation and Gradient Patterns to form digital pheromones [20]. Evaporation can be used with aggregation to aggregate information recently collected from the environment. The Gossip Pattern [13] is composed by the Aggregation Pattern (Section 3.1).

3.3 Evaporation Pattern

Evaporation is a pattern that helps to deal with dynamic environments where the 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 modify their behaviour taking into account information from the environment, information gathered recently must be more relevant than information gathered a long time ago. Evaporation is a mechanism that progressively reduces the relevance of information. Thus, recent information becomes more relevant than information processed some time ago. Evaporation was proposed as a design pattern for self-organising multi-agent systems in [14] and is usually related with Ant Colony Optimisation (ACO) [9].

Aliases: Penalisation, degradation, decay, depletion.

Problem: Outdated information can not be detected and it needs to be removed, or its detection involves a cost that needs to be avoided.

Solution: Evaporation is a mechanism that periodically reduces the relevance of information. Thus, recent information becomes more relevant than older information.

Inspiration: Evaporation is present in nature. For instance, in ant colonies [8], when ants deposit pheromones in the environment, these pheromones attract other ants and drive their movements from the nest to the food and vice-versa. Evaporation acts over the pheromones reducing their concentration along the time until they disappear. This mechanism allows the ants to find the shortest path to the food, even when environment changes occur (such as, new food locations or obstacles in the path). Ants are able to find the new shortest paths by forgetting the old paths.

Forces: Evaporation is controlled by the parameters evaporation factor (i.e. how much the information is evaporated) and the evaporation frequency (i.e. frequency of evaporation execution), used to decrement the relevance of the information. The evaporation factor and evaporation frequency must deal with the dynamics of the environment: if evaporation is too fast, we may lose information; if evaporation is too slow, the information may become outdated and misguide the agents' behaviour. A higher evaporation factor releases memory, but also reduces the information available in the system for the agents. When the evaporation is applied to collaborative search or optimisation algorithms, the evaporation factor controls the balance between exploration and exploitation: high evaporation rates reduce agents' knowledge about the environment, increasing the exploration, and producing fast adaptation to environment changes. However, a higher evaporation factor decreases the performance when no environment changes occur (due to an excess of exploration).

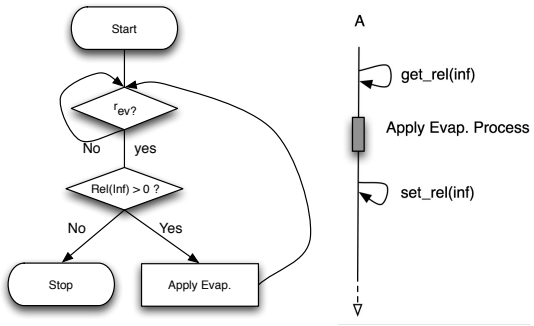
Entities-Dynamics-Environment: Evaporation can be applied to any information present in the system. Periodically, its relevance decays over time. Evaporation is performed by the agent or infrastructural agent periodically executing transition rule (3) or (4), with rate r_{ev}. Evaporation may be applied by subtraction or by multiplication [11].

$$evap :: \text{inf} \xrightarrow{r_{ev}} relevance(\text{inf})' = relevance(\text{inf}) * Ev_{\text{factor}} \tag{3}$$

$$evap :: \text{inf} \xrightarrow{r_{ev}} relevance(\text{inf})' = relevance(\text{inf}) - Ev_{term} \tag{4}$$

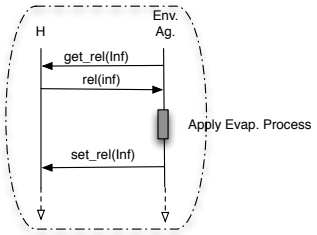
where *inf* is the information on which the evaporation is applied, *relevance*(inf) is the relevance of the information *inf* before the transition fires, Ev_{factor} is the evaporation factor [0..1], Ev_{term} is the evaporation term, and *relevance*(inf)$'$ is the relevance of *inf* once the transition has fired.

Implementation: The Evaporation Pattern is executed by an agent that needs to update the relevance of its internal information, or by infrastructural agents that change the relevance of the information deposited in an environment. We



(a) Agent flow

(b) Agent



(c) Agent - Host

Figure 3: Evaporation: agent behaviour (a), evaporation by the agent itself (b), evaporation by the host (c).

distinguish two cases. First, only one agent encapsulating the information and decaying its own relevance. In this case, the agent follows the flow chart 3 (a) and the corresponding interaction diagram 3 (b). Second, the information is deposited by one agent in a host and an infrastructural agent interacts with the host to decay the information's relevance. The host provides an interface for reading and changing the relevance value. In that case the interaction between the infrastructural agent and the host is shown in figure 3 (c).

Known uses: Evaporation has been used mainly in Dynamic Optimisation. Examples of algorithms using evaporation are ACO [10] and Quantum Swarm Optimisation Evaporation (QSOE) [11]. In [26] evaporation is performed using a parameter called freshness associated to the information.

Consequences: Evaporation enables adaptation to environmental changes. However, the use of evaporation in static scenarios may decrease performance, due to the loss of information associated to this mechanism. The Evaporation Pattern provides the ability to self-adapt to environmental changes increasing the tolerance to noise, as shown in [12]

Related Patterns: The Evaporation Pattern is used by higher level patterns, such as, Digital Pheromone Pattern (not described here) or Gradient Pattern (Section 4.1).

4. MIDDLE LAYER PATTERNS

4.1 Gradient Pattern

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: either a distance attribute is added to the information; or the value of the information is modified such that it reflects its concentration - higher values meaning the sender is closer, such as in ants' pheromones. 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. Different cases may apply: either only the information with the shortest distance to the sender is kept, or the concentration of the information increases.

Aliases: The Gradient Pattern is a particular kind of computational fields [2].

Problem: Large systems suffer from lack of global knowledge to estimate the consequences of the actions performed by other agents beyond their communication range.

Solution: Information spreads from the location it is initially deposited and aggregates when it meets other information. During spreading, additional information about the sender's distance and direction is provided: either through a distance value (incremented or decremented); or by modifying the information to represent its concentration (lower concentration when information is further away). When one agent receives the gradient information, it also knows the direction and the distance where the information comes from. During the aggregation process, a filter operator keeps only the information with the highest (or lowest) distance, or it modifies the concentration.

Inspiration: Gradients appear in many biological processes. The most known are ant foraging, quorum sensing, morphogenesis, and chemotaxis processes. In these processes, gradients support long-range communication among entities (cells, bacteries, etc..) through local interaction.

Forces: Adaptation to environmental changes is faster with high updating frequencies, increasing network overload. Lower updating frequencies reduce network overload, but can produce wrong values when environment changes occur. There is also a trade-off between the diffusion radius (number of hops) and the load in the network. Higher diffusion radii bring information further away from its source, providing a guidance also to distant agents. However, it incements the load and may overwhelm the network [2].

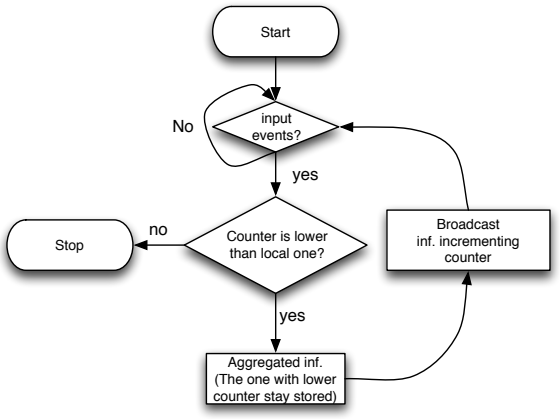
Entities-Dynamic-Environment: Entities acting in the Gradient Pattern are Agents, Hosts, and Infrastructural Agents. Analogously to the Spreading Pattern, when a gradient is created, it is spread to its neighbours. We distinguish two sets of transition rules. Rule (5) models the neighbours forwarding the received information "inf" modifying the distance "*d*" attribute by incrementing or decrementing its value. Rule (6) models the the aggregation when multiple gradients are locally present. This rule models the case of an aggregation where only the information with the shortest distance is kept.

$$gradient :: (\text{inf}, d) \xrightarrow{r_{gr}} send((\text{inf}, d+1), \text{neighbours}) \quad (5)$$

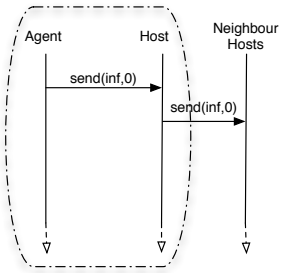
$$gradient :: \{(\text{inf}_1, d_1), ..., (\text{inf}_n, d_n)\} \xrightarrow{r_{gr}} send((\text{inf}_k, d_k + 1), \text{neighbours}), \text{if } k = min\{d_1, ..., d_n\} \quad (6)$$

Rule (7) models the case where the information spread to the neighbours changes during the spreading process to represent the concentration. Rule (8) models the aggregation when multiple gradients are locally present. A new information concentration is computed and provided to the neighbours.

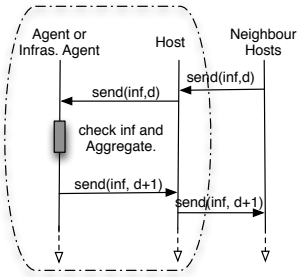
$$gradient :: \text{inf} \xrightarrow{r_{gr}} send(\text{inf} * conc_{factor}, \text{neighbours}) \quad (7)$$



(a) Agent behaviour



(b) Initialisation



(c) Agent - Host

Figure 4: Gradients: agent behaviour (a), initialisation (b), agent and infrastructural agent (c)

$$gradient :: \{\text{inf}_1, ..., \text{inf}_n\} \xrightarrow{r_{gr}} send(op(\text{inf}_1, ..., \text{inf}_n), \text{neighbours}) \quad (8)$$

Implementation: Agents start the process by sending information to all their neighbours, as shown in Figure 4 (b) for the case with distance value (Counter). When one agent receives the information it increments the distance attribute, or it reduces accordingly the concentration value of the information, and forwards the gradient again to all its neighbours (Spreading Pattern) as shown on diagram flow 4 (a) and sequence diagram 4 (b) for the case with distance value. When a host receives the gradient, infrastructural agents spread it further. Notice that this pattern can be applied just with agents. When an agent receives more than one gradient, it applies aggregation (Aggregation Pattern) as shown on sequence diagram 4 (c). For instance, it may filter only the gradient with the lowest distance attribute. To create self-healing gradients (i.e. gradients adaptable themselves to network changes) two interesting implementations are proposed in [25][3].

Known uses: Gradient Pattern has been used in problems such as coordination of swarms of robots, coordination of agents in video games, or routing in sensor networks.

Consequences: The Gradient Pattern adds an extra information (distance). Distance can be used to limit the number of hops during the spreading process.

Related Patterns: The Gradient Pattern is a composition of the Spreading and Aggregation Patterns, extended with the distance value or concentration information. It is used by the Morphogenesis Pattern (Section 5.2), the Chemotaxis Pattern (Section 5.1), and the Quorum Sensing Pattern (Section 5.3). The Gradient Pattern may be combined with the Evaporation Pattern to create active gradients to support adaptation when agents change theirs positions or network topology changes.

5. TOP LAYER PATTERNS

This section describes the three high level patterns used in the literature that their contribution in different fields have been demonstrated. Many interesting applications using the Gradient exist in the literature. However, we only these three are presented because they have been accepted as mechanisms, in spite of other rich applications where their contributions are focused in only one field, and their generalisations have not been proposed.

5.1 Chemotaxis Pattern

The Chemotaxis Pattern provides a mechanism to perform motion coordination in large scale systems. Chemotaxis was proposed by [19]. The Chemotaxis Pattern extends the Gradient Pattern: agents identify the gradient direction to decide the direction of their next movements.

Problem: Decentralised motion coordination aiming at detecting sources or boundaries of events.

Solution: Agents locally sense gradient information and follow the gradient in a specified direction (either follow higher gradient values, lower gradient values, or equipotential lines of gradients).

Inspiration: In biology, chemotaxis is the phenomenon in which single or multi-cellular organisms direct their movements according to certain chemicals present in their environment. Examples in nature are: leukocyte cells moving towards a region of a bacterial inflammation or bacteria migrating towards higher concentrations of nutrients [27]. Notice that in biology, chemotaxis is also a basic mechanism of morphogenesis. It guides cells during development so that they will be placed in the final right position. In this paper, following [19], chemotaxis is used as motion coordination following gradients, and morphogenesis for triggering specific behaviours based on relative positions determined through a gradient.

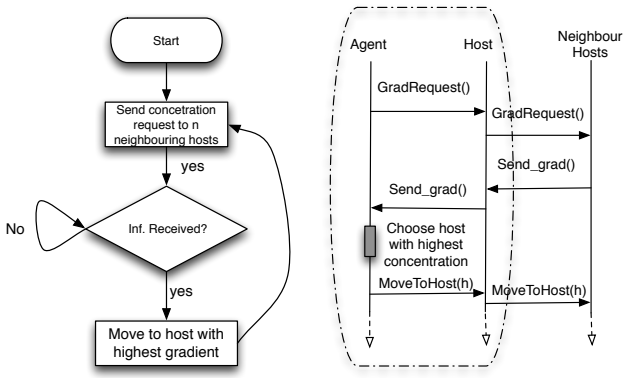
Forces: The Chemotaxis Pattern presents the same forces as the ones for the Gradient Pattern (Section 4.1).

Entities-Dynamic-Environment: The concentration of gradient guides the agents' movements in three different ways, as shown in Figure 6: (1) Attractive Movement, when agents change their positions following higher gradient values, (2) Repulsive movement, when agents follow lower gradient values, incrementing the distance between the agent and the gradient source, and (3) Equipotential movement, when agents follow gradients between thresholds.

Transition rule (9) models an agent that senses local gradient values from "n" neighbours. Then, it follows a specified direction towards a neighbouring:

$$chemotaxis :: \{(neigh_1 : grad_1), .., (neigh_n : grad_n)\} \rightarrow follow(neigh_i) \quad if \quad grad_i = op(grad_1, .., grad_n) \quad (9)$$

Implementation: Chemotaxis can be implemented in two different ways. Using gradients existing in the environ-



(a) Agent behaviour (b) Agent Interactions

Figure 5: Chemotaxis: agent behaviour (a), agent ir

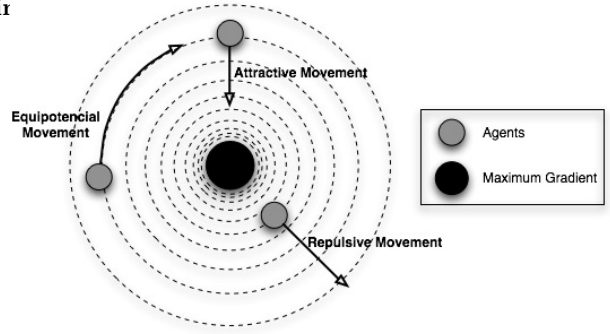


Figure 6: Chemotaxis Pattern - adapted from [7]

ment to coordinate the agent's positions or directions. For instance, [22] uses attractive and equipotential movements to detect the contour of diffuse events using a multi-agent approach over a sensor network infrastructure. On the other hand, gradient fields can be generated by agents. For instance, [17] uses a gradient-based approach to coordinate the position of bots in the Quake 3 Arena video game. Diagrams 5 (a) and (b) show a particular case of implementation where agents get information about neighbouring gradients before taking a decision about where to go next.

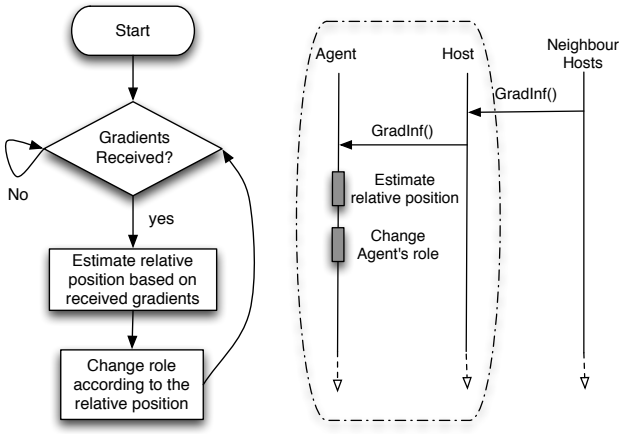
Known uses: Mamei et al. [16] use Chemotaxis to coordinate the positions in swarms of simple mobile robots. Chemotaxis is alsoused in [25], where chemotaxis is applied to route messages in pervasive computing scenarios.

Related Patterns: The Chemotaxis Pattern extends the Gradient Pattern (see section 4.1).

5.2 Morphogenesis Pattern

The goal of the Morphogenesis Pattern is to achieve different behaviours by the agents depending on their position in the system. The Morphogenesis Pattern exploits gradients: positional information is assessed through one or multiple gradient sources generated by other users. Morphogenesis was proposed as a self-organising mechanism in [15, 24]. The morphogenesis process in biology has been considered as one inspiration source for gradient fields.

Problem: In large-scale decentralised systems, agents decide on their roles or plan their activities based on their spatial position.



(a) Agent behaviour (b) Agent Interactions

Figure 7: Morphogenesis: agent behaviour (a), agent interaction (b)

Solution: Specific agents spread morphogenetic gradients. Agents then assess their positions in the system by computing their relative distance to the morphogenetic gradients sources.

Inspiration: In the biological morphogenetic process some cells create and modify molecules (through aggregation) which diffuse (through spreading), creating gradients of molecules. The spatial organisation of such gradients is the morphogenesis gradient, which is used by the cells to differentiate the role that they play inside of the body, e.g. in order to produce cell differentiations.

Forces: The forces presented in this pattern are the same as the ones of the Gradient Pattern (Section 4.1).

Entities-Dynamic-Environment: The entities involved in the morphogenesis process are Agents, Hosts, and Infrastructural Agents. At the beginning, some of the agents spread one or more morphogenesis gradients, implemented using the Gradient Pattern. Other agents sense the morphogenetic gradient in order to calculate their relative positions. Depending on their relative positions, the agents adopt different roles and coordinate their activities in order to achieve collaborative goals. Transition rule (10) models an agent sensing its local gradient values to adapt its behaviour depending on its relative position to the gradient sources. $modality'$ represents a modality state variable describing how the agent should behave in the following.

$$\begin{aligned} morphogenesis &:: \{grad_1, ..., grad_n\} \rightarrow \\ modality' &= compute(grad_1, .., grad_n) \end{aligned} \tag{10}$$

Implementation: An interesting implementation of the morphogenesis gradient to estimate the position is found in [2]. In that paper a self-healing gradient algorithm with a tunable trade-off between precision and communication cost is proposed. In [16] the motion coordination of a swarm of robots is implemented by using both Morphogenesis and Chemotaxis Patterns (Section 5.1). Diagrams 7 (a) and (b) show agents estimating their position in response to gradient information propagated by neighbouring hosts.

Known uses: The Morphogenesis Pattern was used in [4] to implement control techniques for modular self-reconfigurable robots (meta-morphic robots). In [21] morphogenesis is used to create a robust process for shape for-

mation on a sheet of identically programmed agents.

Consequences: The Morphogenesis Pattern provides to the agents a mechanism to coordinate their activities based on their relative positions. Like the other mechanisms previously presented, robustness and scalability are properties provided by this pattern.

Related Patterns: The Morphogenesis Pattern extends the Gradient Pattern (Section 4.1). The Morphogenesis Pattern can be combined with the Digital Pheromone Pattern where the role and behaviour of the agents depend on the distances to the pheromone sources.

5.3 Quorum Sensing Pattern

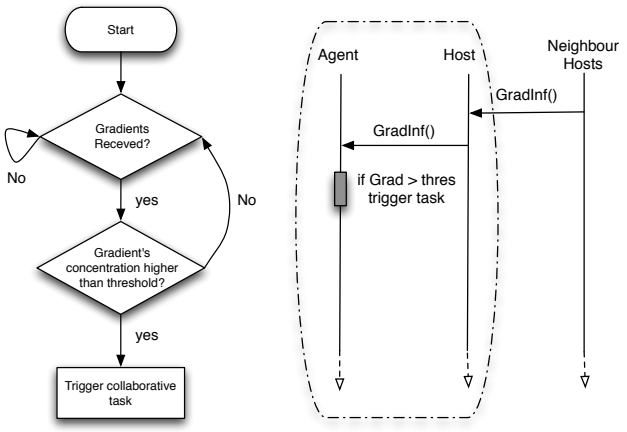
Quorum sensing is a decision-making process for coordinating behaviour and for taking collective decisions in a decentralised way. The goal of the Quorum Sensing Pattern is to provide an estimation of the number of agents (or of the density of the agents) in the system using only local interactions. The number of agents in the system is crucial in those applications, where a minimum number of agents are needed to collaborate on determined tasks.

Problem: Collective decisions in large-scale decentralised systems requiring a threshold number of agents or estimation of the density of agents in a system using only local interactions.

Solution: The Quorum Sensing Pattern allows to take collective decisions through an estimation by individual agents of the agents' density (assessing the number of other agents they interact with) and by determination of a threshold number of agents necessary to take the decision.

Inspiration: The Quorum Sensing Pattern is inspired by the Quorum Sensing process (QS), which is a type of intercellular signal used by bacteria to monitor cell density for a variety of purposes. An example is the bioluminescent bacteria (Vibrio Fischeri) found in some species of squids. These bacteria self-organise their behaviour to produce light only when the density of bacteria is sufficiently high [18]. The bacteria constantly produce and secrete certain signaling molecules called auto-inducers. In presence of a high number of bacteria, the level of auto-inducers increases exponentially (the higher the auto-inducer level a bacteria detects, the more auto-inducer it produces). Another interesting example is given by the colonies of ants (Leptothorax albipennis) [23], when the colony must find a new nest site. A small portion of the ants search for new potential nest sites and assess their quality. When they return to the old nest, they wait for a certain period of time before recruiting other ants (higher assessments produce lower waiting periods). Recruited ants visit the potential nest site and make their own assessment about its quality returning to the old nest and repeating the recruitment process. Because of the waiting periods, the number of ants present in the best nest will tend to increase. When the ants in this nest sense that the rate at which they encounter other ants exceed a particular threshold, the quorum number is reached.

Forces: The Quorum Sensing Pattern uses gradients presenting the same parameters as the Gradient Pattern (Section 4.1). The threshold, indicating that the quorum number has been reached, triggers the collaborative behaviour. Quorum Sensing provides an estimation of the density of agents in the system. However, this pattern does not provide a solution to calculate the number of agents necessary to carry out a collaborative task (i.e. to identify the threshold value).



(a) Agent behaviour (b) Agent Interactions

Figure 8: Quorum Sensing: agent behaviour (a), agent interaction (b)

Entities-Dynamic-Environment: The entities involved in the Quorum Sensing Pattern are the same as in the Gradient Pattern. That is, Agents, Hosts, and Infrastructural Agents. The concentration is estimated by the aggregation of the gradients. Transition rule (11) models that agents sense their local gradient, whose value indicates whether or not the threshold is reached.

$$quorum :: GradInf \rightarrow quorumReached, \quad \text{if } GradInf > threshold \tag{11}$$

Implementation: There is no specific implementation for the Quorum Sensing Pattern. However, biological systems presented above give us some ideas about how to implement the pattern. Here we propose two different approaches to implement the Quorum Sensing Pattern: (1) To use the Gradient Pattern to simulate the auto-inducers like in the bioluminescent bacteria. In this case the gradient concentration provides the agents with an estimation of the agents' density; (2) As in ants' systems, the agents' density can be estimated through the frequency to which agents are in communication range. The use of gradients provides better estimations than the use of frequencies. However, it is more expensive computationally and it requires more network communications. Diagrams 8 (a) and (b) show agents identifying whether the concentration gradient has reached the threshold, in response to gradient information propagated by neighbouring hosts.

Known uses: In [5] quorum sensing is used to increase the power saving in Wireless Sensor Networks. Quorum sensing permits to create clusters based on the structure of the observed parameters of interest, and then only one node for each cluster sends the information on behalf of the quorum. Another known example is the coordination of Autonomous Swarm Robots [23].

Consequences: Each agent can estimate the density of nodes or other agents in the system using only local information received from neighbours, even when the system is really large and there are agents are anonymous.

Related Patterns: The Quorum Sensing Pattern depending on its implementation uses the Gradient Pattern.

6. CONCLUSIONS

This paper is part of a larger work aiming at describing under the form of design patterns, and organised into different layers, a series of bio-inspired mechanisms, such as: digital pheromone, flocking, gossip, or gradient field. In a previous paper we described the three patterns involved in the Gossip case [13]. This paper reports on the six mechanisms related to the Gradient Pattern. Furthermore, as a part of the EU Project SAPERE, we investigate the role of these bio-inspired patterns to enact spatial and temporal behaviour in pervasive services.

7. ACKNOWLEDGEMENTS

Work supported by projects EU SAPERE (contract no. 256873) and EVE (TIN2009-14702- C02-01). The first author holds a FPI scholarship from the Spanish Government.

8. REFERENCES

[1] O. Babaoglu, G. Canright, A. Deutsch, G. A. D. Caro, F. Ducatelle, L. M. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, and T. Urnes. Design patterns from biology for distributed computing. *ACM TAAS*, 1:26–66, 2006.

[2] J. Beal. Flexible self-healing gradients. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1197–1201. ACM, 2009.

[3] J. Beal, J. Bachrach, D. Vickery, and M. Tobenkin. Fast self-healing gradients. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1969–1975, New York, NY, USA, 2008. ACM.

[4] H. Bojinov, A. Casal, and T. Hogg. Multiagent control of self-reconfigurable robots, 2001.

[5] M. Britton and L. Sack. The secoas project: development of a self-organising wireless sensor network for environmental monitoring. In *2nd International Workshop on Sensor and Actor Network Protocols and Applications*, Boston, 2004.

[6] G. Chen and D. Kotz. Solar: A pervasive-computing infrastructure for context-aware mobile applications. Technical report, Department of Computer Science, Dartmouth College Hanover, NH, USA 03755, 2002.

[7] T. De Wolf and T. Holvoet. Design patterns for decentralised coordination in self-organising emergent systems. *Engineering Self-Organising Systems*, 4335:28–49, Feb. 2007.

[8] J. Deneubourg, J. Pasteels, and J. Verhaeghe. Probabilistic behaviour in ants: A strategy of errors? *Journal of Theoretical Biology*, 105(2):259 – 271, 1983.

[9] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, 1992.

[10] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. *New Ideas in Optimization*, pages 11–32, 1999.

[11] J. L. Fernandez-Marquez and J. L. Arcos. An evaporation mechanism for dynamic and noisy multimodal optimization. In *Proc. of the Conference on Genetic and evolutionary Computation (GECCO'09)*, pages 17–24. ACM, 2009.

[12] J. L. Fernandez-Marquez and J. L. Arcos. Adapting particle swarm optimization in dynamic and noisy environments. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 765–772, 2010.

[13] J. L. Fernandez-Marquez, J. L. Arcos, G. Di Marzo Serugendo, and M. Casadei. Description and composition of bio-inspired design patterns: the gossip case. In *Proc. of EASE'2011*, pages 87–96. IEEE Computer Society, 2011.

[14] L. Gardelli, M. Viroli, and A. Omicini. Design patterns for self-organizing multiagent systems. In *Proceedings of EEDAS*, 2007.

[15] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli. Case studies for self-organization in computer science. *J. Syst. Archit*, 52:433–460, 2006.

[16] M. Mamei, M. Vasirani, and F. Zambonelli. Experiments of morphogenesis in swarms of simple mobile robots. *Journal of Applied Artificial Intelligence*, 18:903–919, 2004.

[17] M. Mamei and Zambonelli. Motion coordination in the quake 3 arena environment: a field-based approach. In *First International Workshop on Environments for Multiagent Systems*, 2003.

[18] M. B. Miller and B. L. Bassler. Quorum sensing in bacteria. *Annual Review of Microbiology*, 55(1):165–199, 2001.

[19] R. Nagpal. A catalog of biologically-inspired primitives for engineering self-organization. engineering self-organising systems: Nature-inspired approaches to software engineering (2003); 2977: 53-62. In *Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering. Volume 2977 of Lecture Notes in Computer Science*, pages 53–62. Springer, 2004.

[20] H. Parunak, M. Purcell, and R. Connell. Digital pheromones for auton. coordination of swarming uavs.

[21] M. C. Radhika and R. Nagpal. Programmable self-assembly using biologically-inspired. pages 418–425. Press, 2002.

[22] R. M. Ruairí and M. T. Keane. An energy-efficient, multi-agent sensor network for detecting diffuse events. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1390–1395, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[23] E. Sahin and N. R. Franks. Measurement of space: From ants to robots. *In Proceedings of WGW 2002: EPSRC/BBSRC International Workshop Biologically-Inspired Robotics*, 2002.

[24] J. Sudeikat and W. Renz. Toward systemic mas development: Enforcing decentralized self-organization by composition and refinement of archetype dynamics. In *Proceedings of Engineering Environment-Mediated Multiagent Systems*. Springer, 2007.

[25] M. Viroli, M. Casadei, S. Montagna, and F. Zambonelli. Spatial coordination of pervasive services through chemical-inspired tuple spaces. *ACM Transac. on Autonomous and Adapt. Systems*, 2011.

[26] D. Weyns, T. Holvoet, and A. Helleboogh. Anticipatory vehicle routing using delegate multi-agent systems. pages 87 –93, sep. 2007.

[27] L. Wolpert, T. Jessell, P. Lawrence, E. Meyerowitz, E. Robertson, and J. Smith. *Principles of Development*. Oxford Univer. Press, 3rd edition, 2007.