Design Patterns for Decentralised Coordination in Self-organising Emergent Systems

Tom De Wolf and Tom Holvoet

AgentWise@DistriNet Research Group
Department of Computer Science, KULeuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
{Tom.DeWolf,Tom.Holvoet}@cs.kuleuven.be
http://www.cs.kuleuven.be/~tomdw

Abstract. There is little or no guidance to systematically design a self-organising emergent solution that achieves the desired macroscopic behaviour. This paper describes decentralised coordination mechanisms such as gradient fields as design patterns, similar to patterns used in mainstream software engineering. As a consequence, a structured consolidation of best practice in using each coordination mechanism becomes available to guide engineers in applying them, and to directly decide which mechanisms are promising to solve a certain problem. As such, self-organising emergent solutions can be engineered more systematically, which is illustrated in a packet delivery service application.

1 Introduction

Modern distributed systems exhibit an increasingly interwoven and completely decentralised structure [13] (e.g. ad-hoc networks, transportation systems, etc.). Different subsystems interact with each other in many, often very complex, dynamic, and unpredictable ways. More systems need to achieve their requirements autonomously [15]. A promising approach is to use a group of agents that cooperate to autonomously achieve the required system-wide or macroscopic behaviour using only local interactions, local activities of the agents, and locally obtained information. Such decentralised multi-agent systems (MASs) exhibit *self-organising emergent behaviour* [5].

When engineering a self-organising emergent solution, the problem-solving power mainly resides in the interactions and coordination between the agents instead of in the intelligent reasoning of single agents. Therefore, a major architectural design decision is the choice of suitable decentralised coordination mechanisms such as digital pheromones [3], gradient fields [22], market-based control [26], tags [12], or tokens [31]. Many of such mechanisms are already applied to a number of case studies in literature [3,21,22,18,20,19,25,12,31,26,32,4,10,11,17,24]. However, a fundamental problem is the lack of guidance on how to systematically choose and use the most suitable coordination mechanism. The main reason is that, currently, all existing knowledge and best practice on coordination mechanisms is spread over hundreds of papers without a clearly structured and directly usable description of the mechanisms.

S. Brueckner et al. (Eds.): ESOA 2006, LNAI 4335, pp. 28–49, 2007.

The main contribution of this paper, and the extended report in [7], is twofold. First, the paper shows how decentralised coordination mechanisms can be structurally described as design patterns. Secondly, the paper illustrates how an engineer can use these patterns to systematically choose how to coordinate agents and achieve the desired global behaviour. Section 2 motivates design patterns as a description to support engineers in their choice of decentralised coordination mechanisms. The section also outlines in detail the structure of the pattern description. After that, sections 3, 4 and 5 give an usable pattern summary of a number of coordination mechanisms and apply the design pattern description in detail to two example mechanisms, i.e. gradient fields and market-based control. In section 6, a case study on a packet delivery service illustrates how design patterns allow to engineer more systematically a self-organising emergent solution. Finally, section 7 concludes and discusses future work.

2 Decentralised Coordination Mechanisms as Design Patterns

Typically, engineering MASs means having 99% of the effort go to conventional computer science and only 1% involves the actual agent paradigm [8]. Therefore, to engineer self-organising emergent MASs, developers should exploit conventional software technologies and techniques wherever possible [29]. Such exploitation speeds up development, avoids reinventing the wheel, and enables sufficient time to be devoted to the value added by the multi-agent paradigm [30]. From this point of view, [6] proposes a step plan based on an existing industry-ready software engineering process, i.e. the Unified Process [14]. The UP process is customised to explicitly focus on how to address the desired macroscopic behaviour of self-organising emergent MASs. Figure 1 shows that almost every discipline in the UP process is customised.

This paper focusses on the Design which emphasises a solution that fulfills the requirements, rather than its implementation or the requirements themselves. More specifically the focus is on the coarse-grained architectural design. The author of [16] states that architectural design is partially a science and partially an art. The *science* of architecture is the collection and organisation of information about architectural significant requirements (e.g. non-functional and macroscopic functionality). The *art* or architecture is making skillful choices and constructing a solution that meets the requirements, taking into account trade-offs, interdependencies, and priorities. The 'art' of architectural design is the creative step where designers exploit knowledge and best practice from a number of areas (e.g. architectural styles and patterns, technologies, pitfalls, and trends) to reach a suitable solution. For self-organising emergent systems the main source of knowledge to exploit in this creative step are the different mechanisms to coordinate the desired macroscopic functionality such as digital pheromones [3], gradient fields [22], and market-based coordination [26]. This paper captures this knowledge on decentralised coordination mechanisms as architectural design patterns.

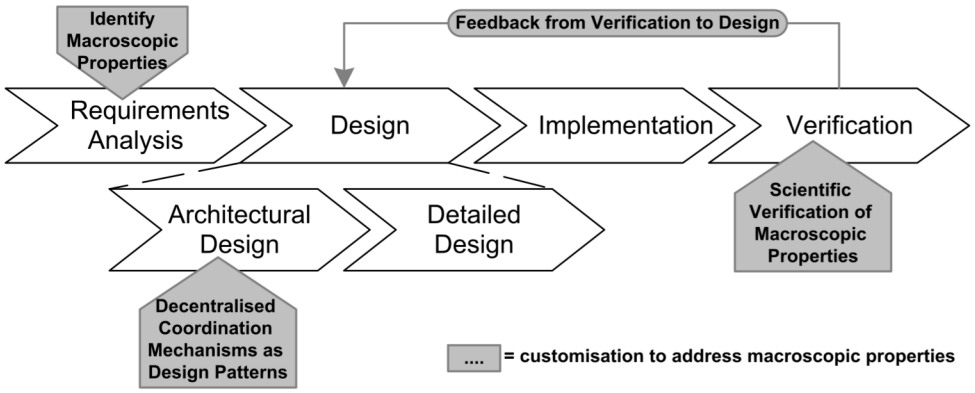

Fig. 1. A Unified Process engineering iteration annotated with customisations for issues specific for self-organising emergent MASs

2.1 Motivation for Design Patterns

As stated earlier, the main problem-solving power resides in the coordination between agents and a major architectural design decision concerns the choice of one or more decentralised coordination mechanisms to achieve the desired macroscopic behaviour. The mechanism used to coordinate has a strong impact on what is and can be communicated [21]. A lot of such mechanisms are used in literature but experience and knowledge about how to use them is currently spread out over hundreds of articles. A more structured and useful description is needed to consolidate this best practice.

In mainstream software engineering, current best practice and knowledge about known solutions is captured in what is called 'design patterns'. The most famous reference is the Gang of Four design patterns book [9]. This paper supports applying decentralised coordination mechanisms by describing them as design patterns, similar to patterns used in mainstream software engineering. An important issue in engineering self-organising emergent solutions is to understand for which kind of macroscopic behaviour each coordination mechanisms is useful, which is more appropriate in which situation, etc. Design patterns describe each decentralised coordination mechanism in a structured way, inherently including this kind information. Using such a set of structured patterns, self-organising emergent system can be designed more systematically.

2.2 The Pattern Description Format

Patterns can be described at several levels of abstraction. The patterns in the Gang of Four book [9] are described at the class or implementation level. Another level of abstraction to describe design patterns is the architectural or even conceptual level in which the focus is more on the coarse-grained and conceptual structure. The decentralised coordination mechanisms in this paper are described at the architectural and conceptual level. The class level is not described because such mechanisms can be implemented in multiple ways and little is known about the best way to do this.

This paper uses the guidelines and patterns for describing patterns from [23]. As such, the decentralised coordination mechanisms are described in a format known in mainstream software engineering which promotes their usage. For example, also the Gang of Four patterns book [9] uses a similar format. The general structure is extended in subsections to better describe issues specific for decentralised coordination in self-organising emergent solutions. The format used involves the following sections:

- **Pattern Name/Also Known As:** A clear name and aliases referring to the solution used or a useful metaphor. The name is given in the title of the pattern's section.
- **Context/Applicability:** The circumstances in which the problem being solves requires a solution. Often described via a 'situation'. In this paper this context typically indicates when a self-organising emergent solution is promising.
- **Problem/Intent:** What is solved by this pattern? Engineers compare this section with their problem in order to select coordination mechanisms.
- **Forces:** Often contradictory considerations that must be taken into account when choosing a solution to a problem.
- **Solution:** A description of how the problem is solved and described in close relation to the forces it resolves. This section has a more detailed structure:
  - *Inspiration:* Most coordination mechanism are inspired by some natural, biological, physical, or social phenomena.
  - *Conceptual Description:* A conceptual description of how the inspirational mechanism works and is used in computer systems. This section allows to map the concepts used in the coordination mechanism to domain-specific entities in the system under construction, i.e. the concepts and their relationships indicate what is needed in order to use this coordination mechanism.
  - *Parameter Tuning:* Typically, such coordination mechanisms have a lot of parameters that need to be tuned. This section enumerates them and gives some guidelines to tune them.
  - *Infrastructure:* Some mechanisms need a supporting infrastructure. More specifically, what is needed to support the design at the class level.
  - *Characteristics:* Using the mechanism imposes some characteristics on the solution including advantages, disadvantages, and other useful properties.
- **Related Mechanisms/Patterns:** An enumeration of patterns that are related in which the differences and similarities are emphasised.
  - *Variations:* Variations of the same pattern which can be more general or (domain) specific variations.
  - *Other Coordination Mechanisms:* Alternative coordination mechanisms that solve the same (or related) problem.
- **Examples/Known Uses:** Examples of known uses of the pattern in case studies.

In addition to a separate description for each pattern, a 'Problem/Solution Summary' is provided to help the reader find the pattern(s) which solve(s) their specific problems. Such a summary is typically a table with a brief description of the problem each pattern solves and how. In what follows, section 3 gives such a summary for a limited set of widely used coordination mechanisms and sections 4 and 5 describe two patterns in more detail following the structure given in this section.

3 Problem/Solution Summary

This section summarises the patterns by giving a so called 'Problem/Solution Summary'. An engineer uses this to find the pattern(s) or coordination mechanism(s) which solve(s) their specific problem(s). The Problem/Solution Summary can be found in Table 1 where a brief description of the problem and the corresponding solution is linked to the pattern to use. Due to space limitation only two pattern descriptions are given in (limited) detail in the following sections. However, a full detailed pattern catalogue can be found in [7].

4 Pattern 1: Gradient Fields

Also Known As: Computational Fields, Co-Fields, Morphogen gradients [18], Field-based coordination, Force Fields [21].

4.1 Context/Applicability

A solution is needed to coordinate multiple autonomous entities, situated in an environment, in a decentralised way to achieve a globally coherent spatial movement of the agents. The coordination mechanism has to be robust and flexible in the face of frequent changes. Also, local estimates of global information are the only possible way to coordinate. As such, decentralised coordination is the only possible alternative.

4.2 Problem/Intent

- *Spatial Movement:* How to adaptively orchestrate in a decentralised way the spatial movement of a large set of agents in large-scale distributed systems [21,20,18]? As such global *Pattern Formation* can be achieved.
- *Structure Formation:* How to adaptively self-configure a modular structure achieving the desired shape/structure (e.g. modular robots) [18]?
- *Routing:* How to achieve routing for messages, agents, etc. [18]?
- *Integration of Contextual Information:* How to provide agents with abstract, simple yet effective contextual information from various sources supporting and facilitating the required motion coordination activities [21,20,18] (i.e. spatial information such as distance/direction to source)?

Table 1. Problem/Solution Summary

Problem(s)	Solution	Pattern
Spatial Source to Destination Routing, Task Recruitment, Relation Identification, Integration of various information sources	*Agents explicitly search* for goals, tasks, or related items and drop pheromones to gradually form historical paths for other agents to follow. Reinforcement of an existing path by other agents can be seen as a reinforcement of the relation between source and destination. Evaporation, Aggregation, and Propagation keep the pheromones up-to-date and support integration of information.	Digital Pheromone Paths [3]
Spatial Movement, Pattern Formation, Structure Formation, Routing, Integration of Contextual Information	Spatial, contextual, and coordination information is automatically and instantaneously propagated by the environment as computational fields. *Agents simply follow the "waveform"* of these fields to achieve the coordination task, i.e. *no explicit exploration.*	Gradient Fields [22]
Resource Allocation in general (resource=task, power, bandwidth, space, time, etc.) , Integration of resource Usage/Need Information	A virtual market where resource users sell and buy resource usage with virtual currency. The price evolves according to the market dynamics and indicates a high (high price) or low (low price) demand. This information is used by agents to decide on using the resource or not. Economic market theory states that the prices converge to a stable equilibrium.	Market-based Coordination [26]
Trust and reputation, Team-formation, Discourage selfish behaviour in Teams, Specialisation of skills within Teams	Agents put and modify tags on other agents and a team is formed by only collaborating with agents with the same tag or some other condition. If tags indicate how well agents behaved in collaborations with others then trust and reputation information can be available.	Tags [12]
Resource Access Control/Allocation, Role Allocation, Enforce Organisation Structure, Information Sharing	Information, resources, or roles are represented by a token. Only the holder has exclusive access to the information and resource. Holding a role token commits to executing that role. Tokens are passed among agents to get adaptive coordination.	Tokens [31]
etc.	etc.	etc.

4.3 Forces

– *Explore vs. Exploit*: In order to be adaptive the solution has to explore sufficiently compared to only exploiting already known information. Otherwise the approach can get trapped in local optima or never find new targets at

all. However, to much exploration may result in an approach that is very inefficient.
- *Centralised vs. Decentralised*: A decentralised solution often requires a huge amount of communication and coordination overhead which a centralised solution has not. A centralised solution often optimally controls the system. However, a centralised solution is often a bottleneck and single point of failure in a very dynamic context.
- *Optimality vs. Robustness/Flexibility*: An adaptive approach that has no central means to optimise its efficiency may result in suboptimal solutions. However, an optimal solution only exists with respect to a static situation, which is never reached in the face of frequent changes. As such, a robust and flexible approach may be preferred to an approach that is optimal but inflexible.
- *Responsibility of Environment vs. Agents*: Coordination needs complex processing and communication. There is a trade-off to make the agents themselves or the environment responsible. Making the agents responsible allows agents to explicitly reason about and control how information is distributed but sometimes requires complex algorithms. On the other hand, making the environment responsible allows agents to simply be guided by the results in the environment, i.e. a "red carpet' that avoids complex processing within agents, but the agents are no longer in control of the information distribution. Such coordination can be more dynamic and adaptive because the source of changes, i.e. environment, also supplies the coordination information.
- *Greedy vs. Focussed*: A "greedy" approach to coordination disregards that a small sacrifice now, i.e. not exploiting a piece of coordination information, could possibly lead to greater advantages in the future. However, it is a general drawback of distributed solutions, where the possibility of globally informed decisions by distributed agents is often ruled out due to the need for efficient and adaptive coordination [19].

4.4 Solution

Inspiration. The Gradient Field coordination mechanism takes its *inspiration from physics and biology.* In physics [20,19,21], the same mechanism can be found in the way masses and particles in our universe adaptively move and globally self-organise their movements accordingly to the locally perceived magnitude of gravitational/electro-magnetic/potential fields. The particles follows the "wave-form" of the fields (see figure 2). In biological organisms, a coherent, reliable and complex behaviour is achieved from the local cooperation of large numbers of identically "programmed" cells [18]. In particular, chemicals are diffused among cells and cells are driven in their behaviour by the locally sensed gradients of diffused proteins ("morphogen gradients"). Morphogen gradients is a mechanism used to determine positional information and polarity. For example, cells use the concentration of different morphogens to determine whether they lie in the head, thorax, or abdominal regions to achieve wing and limb development.

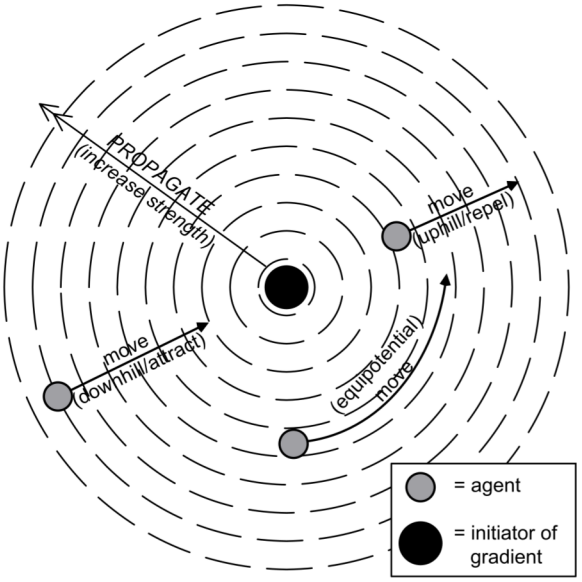

Fig. 2. A gradient field with propagation direction and agent movements shown

Conceptual Description. To use this as a decentralised coordination mechanism in software systems such a gravitational/electro-magnetic/chemical field has to be translated into an artificial data structure representing the `Gradient Field`, i.e. a computational field or Co-Field [22,21,19]. Figure 3 illustrates the conceptual structure of such a solution in UML class diagram notation. A `Gradient Field` is data structure which is spatially distributed, as `Gradient Parts`, over `Locations` in the `Environment`. Each field is characterised by a unique identifier, the necessary contextual information such as a location-dependent numeric value (representing the field strength in that location), and a `Propagation Rule` determining how the numeric value should change in space.

A gradient field is started, initiated, or injected into the environment from a certain "source" location by a `Gradient Initiator` (i.e. a `Location` itself, an `Agent`, or some other entity in the system) conveying some application-specific information about the local environment and/or about the initiator itself [21]. The `Environment` makes sure that the `Gradient Field` is propagated, according to its `Propagation Rule`, from the starting location to the neighbours of that location (typically increasing the strength of the gradient, initially set to zero; decreasing gradients are also possible). In turn, the neighbouring locations modify the strength and re-broadcast the gradient to their neighbours which is repeated until the gradient has propagated far enough. Each intermediate location stores and forwards only the gradient part with the minimum strength value it has received for that particular gradient field. As such a "waveform" gradient map is formed in the environment which conveys useful context information for the coordination task.

`Agents` observe multiple `Gradient Parts` on the neighbouring locations of the location on which the agent is situated. Then agents follow the waveform (deterministically or with some probability) by moving to a neighbouring location. This allows agents to coordinate their movement with respect the the

gradient initiator. For example, in figure 2 agents move downhill (attracted by the initiator), uphill (repelled by the initiator), or on an equipotential line (equal strength around initiator).

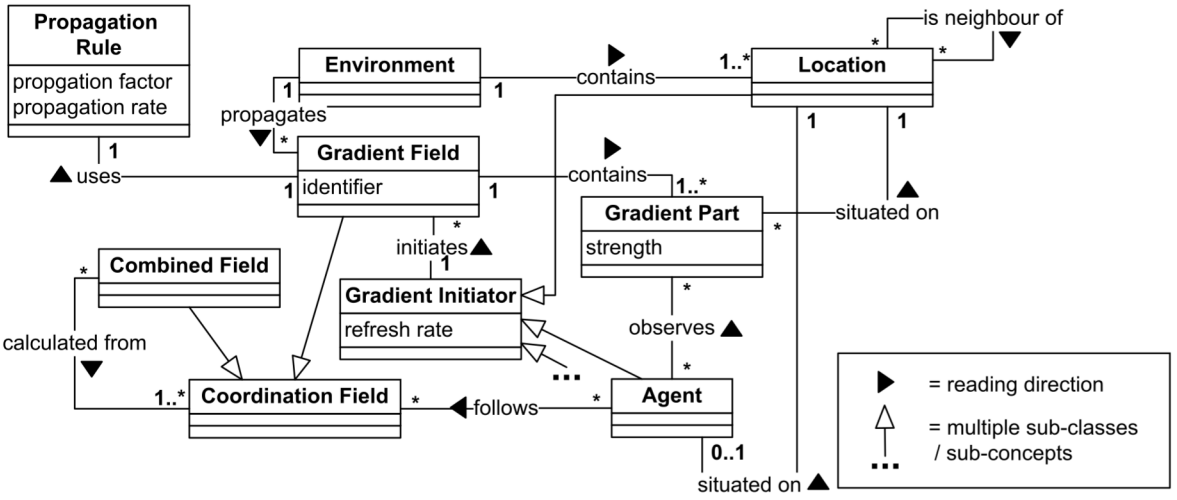

Fig. 3. A conceptual model of the gradient coordination mechanism

The gradient field mechanism can be schematised as follows [22,21,20,19]:

- The *environment is represented and abstracted by "computational fields"* which provide agents with a location-dependent and local perspective on the global situation of the system to facilitate the required motion coordination.
- The coordination policy is realised by letting the agents move locally *following the "waveform" of these fields.* Agents can autonomously decide whether to follow the gradient fields or not.
- Environmental *dynamics* and movement of entities *induce changes in the fields' surface.* For example, when the initiator of a gradient field moves, the field -with some propagation delay- has to be updated in the environment to reflect that move. As such, a feedback cycle is formed that consequently influences agent movement.
- This *feedback cycle* lets the system *self-organise.* A globally coordinated and self-organised behaviour in the agent's movements emerges in a fully decentralised way due to the interrelated effects of agents following the fields' shape and of dynamic field re-shaping [21,20].

However, the achievement of an application-specific coordination task is rarely relying on the evaluation, as it is, of an existing computational field. Rather, in most cases, an application-specific task relies on the evaluation of an application-specific `Coordination Field` [21,20,19]. This coordination field can be an existing gradient field but typically it is a `Combined Field`, calculated as a combination (e.g. linear) of some of the locally perceived fields or other coordination fields. The coordination field is a new field in itself, and it is built with the goal of encoding in its shape the agent's coordination task. Once a proper coordination

field is computed, agents can achieve their coordination task by simply following the shape of their coordination field.

Parameter Tuning. Every gradient field has a number or parameters, or "settings": a *Propagation Factor* determining how much is added to or removed from the gradient strength each propagation step and a *Propagation Rate* indicating how fast propagation occurs; a dynamic gradient has to be updated regularly which is determined by a certain *Refresh Rate*; the *Initial Strength of a Gradient* is the strength at the source location; each propagation rule can have its own *rule-specific parameters*, e.g. using another propagation factor at a distance of X from the initiator.

Infrastructure. A proper infrastructure or middleware is required (e.g. TOTA [20]) that supports data storage (to store field values), communication (propagate field values), event notification and subscription mechanisms (notify interested agents about field changes, as well as update fields to support changes), mobile-code services (dynamically configure field-propagation algorithms and coordination fields composition rules), and localisation mechanisms (to discover where agents are).

Characteristics

- Typically, following the gradient field downhill is the *shortest path* towards the initiator of the field [18].
- The structure of the environment in which the agents are working should reflect the current "problem" the agents are working on and the gradient structure and distribution should guide the agents to the current "solution" to that problem.
- *Spatial Context Information is distributed:* Gradients support *information spreading/distribution.* Gradient-Fields mainly deliver spatial information such as the direction or distance to a gradient initiator [20]. However, gradients can also embed any other necessary information.
- *Feedback Cycle [22]:* Feedback is given by the fact that gradients can change when changes occur in the environment or when the agent that emits the gradient decides to move or change the gradient. Other agents or gradient-emitting entities can then take that change in the gradient into account and react on it by for example changing its own gradient info. As such a feedback cycle is established to self-organise.
- *Simple Agents - Complex Environment:* Field-based approaches delegate to a middleware infrastructure in the environment the task of constructing and automatically updating the gradient field [20]. As such, the environment makes sure that not too much computational and communication burden is imposed on the agents themselves [19]. The context is represented as gradient fields, i.e. a kind of "red carpet", which represents how to achieve a coordination task by simply following the field.

- *Greedy Approach [20,19]:* A weakness of field-based approaches is that they are "greedy" because of the strictly local perspective in which agents act. Agents disregard whether as small sacrifice now, climbing a field hill instead of descending it - could possibly lead to greater advantages in the future.
- etc. (see [7] for more characteristics)

4.5 Related Mechanisms/Patterns

Variations

- *Propagation Inhibition or Selective Propagation [20,18]:* The propagation of a field can be bounded to a portion of space by having the propagation procedure to check conditions such as the local spatial coordinates, the gradient type or its strength to decide on further propagation or not.
- *Multiple Types of Fields [20,21,19]:* As mentioned earlier, multiple gradient instances can be combined in a coordination field to follow as a whole. All used gradients are typically of the same type. A logical extension is using different types of fields depending on the specific motion pattern to enforce. As such, they can be propagated and combined in coordination fields according to field-specific rules.
- etc. (see [7] for more variations)

Other Coordination Mechanisms. *Digital Pheromone Paths* is a specific instance of *Gradient Fields* where a number of small fields or pheromones aggregate into a gradient path and evaporate over time [20]. Also pheromone paths are constructed explicitly and, as such, agents also explicitly have to discover targets. Gradients make targets immediately and automatically (by the environment) visible through the presence of a gradient field for that target. Agents do not have to explore pro-actively. In addition, pheromones constitute a memory of the recent past, while gradient fields are instantaneous.

4.6 Examples/Known Uses

Some known uses or possible applications are: spatial shape formation [18], urban traffic management [21], reconfiguring modular robots' shape [21,25], control of autonomous characters in video games [21,19], tourist movement in museum [19,20], forklifts activity in a warehouse [19], software agents exploring the web [19], etc.

5 Pattern 2: Market-Based Control

Also Known As: Market Control, Market-Oriented Programming [32].

5.1 Context/Applicability

You need to coordinate multiple autonomous entities in a decentralised way to achieve a common and globally coherent goal while sharing a set of scarce

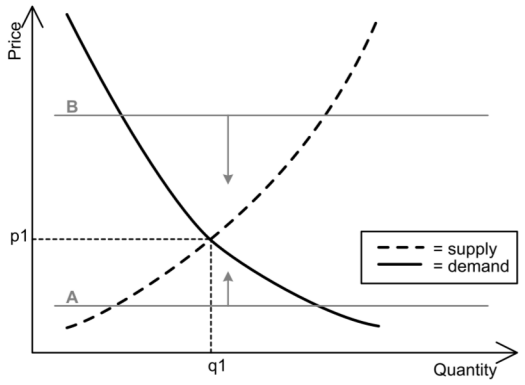

Fig. 4. The theory of Supply and Demand

resources. The coordination mechanism has to be robust and flexible in the face of frequent changes. Also, local estimates of global information are the only possible way to coordinate. As such, decentralised coordination is the only possible alternative. Some locally available information is needed indicating the global usage of resources.

5.2 Problem/Intent

- *Resource Allocation:* How to do efficient resource allocation [26,4,10,32] in a distributed and decentralised manner? Resources can be interpreted as tasks, bandwidth, manufacturing devices, etc.
- *Integration of Resource Usage/Need Information:* How to have locally available information about the global usage of and need for resources?

5.3 Forces

- *Centralised vs. Decentralised* (see section 4.3 in Gradient Field pattern).
- *Optimality vs. Robustness/Flexibility* (see section 4.3).
- *Responsibility of Environment vs. Agents* (see section 4.3).

5.4 Solution

Inspiration. The inspiration and metaphor came from human economies and market mechanisms, i.e. the free-market economies. In such an economy, goods or resources are allocated to the participants in a decentralised, robust, and self-organising manner. Participants act as buyers and/or sellers of goods by offering to buy or sell a good to other participants for a certain price. As long as the participants act completely self-interested then the market achieves a globally optimal allocation of goods. Transaction prices converge to a global equilibrium price, i.e. the market price. Note, that the degree of convergence depends on the market characteristics (e.g. elasticity, shape of demand and supply curves, etc.).

This price mechanism depends on the evolution of the demand and available supply of goods. Figure 4 illustrates this by plotting the evolution of supply

and demand. The demand curve shows the quantity of good that consumers are willing and have the capacity to buy at the given price. The supply curve shows the quantity that suppliers are willing to sell at a given price. If the quantity of a good demanded by consumers in a market is greater than the quantity supplied, competition between consumers causes the price of the good to rise. As such, according to situation A in figure 4 this can both reduce the quantity demanded (because some consumers can no longer afford it) and increase the quantity supplied (because some suppliers may be more interested in selling at higher prices). Similar effects occur when the quantity supplied is greater than the quantity demanded (situation B in figure 4). According to classical economic theories the market equilibrates (see arrows in figure 4): transaction prices approach an equilibrium value $(p1, q1)$ where the quantity demanded matches the quantity supplied. As such, an efficient means of societal resource/goods allocation exists.

Conceptual Description. The aim of market-based coordination in computer science is fundamentally different from the aim of economic theory [32]. In market-based coordination, microeconomic theory is taken as given. Whether or not the microeconomic theory actually reflects human behaviour is not the critical issue. The important question is instead how microeconomic theory can be utilised for the implementation of successful resource allocation mechanisms in computer systems. Although decision-making is only local, economic theory, which has an immense body of formal study [28], provides means to generate and predict macroscopic properties such as the equilibrium price and others that can be deduced from that price information. The aim is that computer systems exhibit the same decentralisation, robustness, and capacity for self-organisation as do real economies [4]. In designing a market of computational agents, a key issue is to identify the consumers and producers [11]. Various preferences and constraints are introduced through the definition of the agents' trading behaviour. This ability to explicitly program the trading behaviour is an important difference from human markets. Finally, the mechanism for matching buyers and sellers must be specified.

Figure 5 illustrates market-based coordination conceptually in UML class diagram notation. A group of `agents` negotiate and trade with each other on an virtual market of scarce resources. The agents `communicate` with messages encapsulating `offers`, `bids`, `commitments`, and `payments` for resources [10]. The agents' goal is to acquire the `resources` of a certain `Resource Type` that they need. All agents start with an initial amount of `Currency` [11]. `Resources` are at each moment in time owned/used by one `agent`. Some agents act as 'consumers' or 'buyers' and have a `Demand` for a certain *quantity* of a resource and are willing to pay a certain *price* for it, based on the `Currency` they have and possibly other conditions. Other agents act as 'producers' or 'suppliers' and have a `Supply` of a certain *quantity* of a resource and are willing to sell at a certain *price*. The goal is to maximise their profit. Each `Agent` is self-interested and decides to participate as a *buyer/consumer* in an `Transaction` with a *supplier/producer* to transfer a certain *quantity* of resources. The behaviour of each agent depends on its local

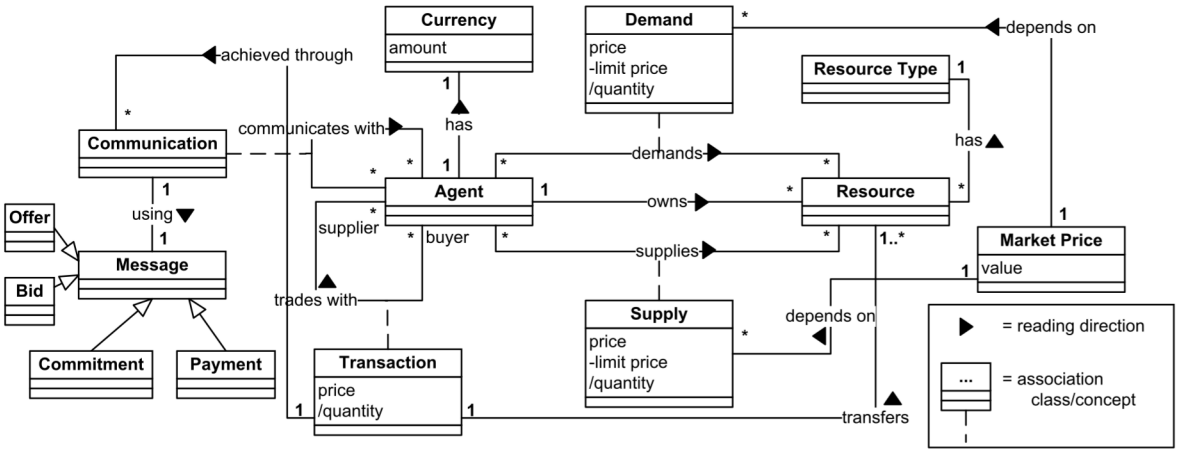

Fig. 5. Conceptual model of Market-based Coordination

information as well as market prices that communicate global information. For each unit to trade, each agent has a `limit price` that is unknown to others [4]. A buyer couldn't pay more than its limit price, and a seller couldn't sell for less than its limit price. The distribution of limit prices determines the evolution of the supply and demand curves.

Similar to a real free-market economy the process of `demand` and `supply` determines the `Market Price` to which the transaction prices evolve. When supply is greater than demand (resources are plentiful), the price of the resource will fall; and when demand exceeds supply (resources are scarce), the price rises. The aim then is that prices rise and fall, dynamically matching the quantity demanded to the quantity supplied, while these quantities also vary dynamically [4]. Agents set their prices solely on the basis of their implicit perception of supply and demand [10] through their success of bidding at particular prices. A special case occurs when an agent is at the same time a seller and a buyer [10] which implies a strategy to maintain an inventory level (i.e. number of owned resource units) that maximises its profit, i.e. that suits the market demand. The amount of available resources can be limited [11]. This is taken into account in the demand and supply. For instance, if the amount of available resources increases suddenly then the supply curve on figure 4 shifts to the right because suppliers are willing to supply more for each price. This results in an equilibrium increase in the supply and decrease in price (similarly demand curve shifts when agents suddenly need more resources). A globally limited amount of resources makes the market mechanism a resource allocation mechanism that generates an equilibrium distribution of resources that maximises global welfare [10].

There are two approaches to establish a market in a computer system [10,4,32,11,1]:

– *Auctioneer-mediated Markets (Centralised):* The market is mediated by an auctioneer agent. Agents send demand/supply functions or bids telling how much they like to consume/produce at different prices. The auctioneer then immediately establishes an equilibrium market price vector such that supply

meets demand for all resources. Then agents exchange the resources as stated by their bid and the calculated equilibrium price.
- *Direct Markets (Decentralised):* Real bids are used instead of calculating based on demand and supply functions. The market is executed for real and the agents bid and adapt their bidding to the outcomes of the auctions over time. A transaction occurs whenever one trader accepts an offer or a bid quoted by another trader. Transactions depend on pairwise encounters in which agents exchange their bid or price for a resource (e.g. traders distributed in space only transact with nearby traders). The agents bid and adapt their bidding to their success over time which iteratively balances supply and demand at the equilibrium [10].

Note, that the speed or stability of equilibration in a market can be affected by the type of market. An auction-mediated market typically equilibrates immediately in each auction cycle while a decentralised market needs multiple transactions cycles before the market equilibrates. However, the system should not rely on the operation of any single critical component, i.e. a centralised auctioneer [4]. The failure of any one trading agent in a market-based system should result in only a minor impairment to the overall behaviour of the system, rather than a total breakdown. A truly decentralised and robust system in which agents bargain directly with each other is preferred to a failure-prone and inflexible central one. However, firm guarantees that the (optimal) equilibrium price is reached are not always available because less theory is available for decentralised markets compared to centralised markets.

Parameter Tuning. Depending on the specific use of the market mechanism, a number of parameters have to be tuned: *limit prices* determine the demand and supply curve behaviour and the profit of agents, the *initial amount of currency* influences the limit prices, the *number of agents* participating influences the convergence to equilibrium, and the *price-adaptation behaviour* (speed and size of increase/decrease of prices) influences the demand/supply curves and thus the speed and efficiency of the market. For the decentralised market, the specification of who trades with who is also an important parameter (e.g. a distance-limit between agents in a 2D world can determine if they can trade or not).

Infrastructure. Because most of the coordination happens directly between agents no real infrastructure support is needed, except for communication infrastructure.

Characteristics

- *Information Distribution.* Markets deliver to agents information on the resource-usage through the price mechanism. A high price reflects a high demand and/or low remaining supply, i.e. high usage. A low price reflects a low demand and/or high remaining supply, i.e. low usage.
- *Feedback Cycle.* The price mechanism serves as a feedback cycle. A high price implies that agents buy less which diminishes the demand and as

such the price reflects this by decreasing. In turn, a lower price changes the behaviour of agents to start buying again which increases the demand. As such feedback about the global market is given through the price evolution.
- *Agents have all responsibility.* The responsibility of coordination is completely situated with the agents themselves, i.e. no environment-mediated coordination. However, for auction-based markets the auctioneer can be considered as the environment.
- *Decentralisation - Robustness - Self-Organisation [32,10,4]:* Market-based control without an auctioneer allows for truly distributed systems with the same decentralisation, robustness to participant failures, and self-organising properties as real free-market economies. A central auctioneer would be a single point of failure.
- *Price Information indicates Global and Local Performance:* For example for network routing in [10], the cost of each network call can be computed from the available price information which allows more efficient call charging. Trading resources for some sort of money enables evaluation of local performance and valuation of resources [32] based on the price, so that it becomes apparent which resources are the most valuable and which agents are using the most of these.
- *Stabilisation at Equilibrium - Pareto Optimality [4,17,32]:* For markets to be of genuine use in applications, they should exhibit smooth and fast convergence to the equilibrium. Transaction prices stabilise rapidly at an equilibrium that is predictable from economic theory and which is stable and robust with respect to sudden changes in the market. Actions of groups of individuals, engaging in simple trading interactions driven by self-interest can result in optimal resource allocation. Market-based systems are termed Pareto optimal. Pareto optimality means that no agent can do better without diminishing the performance of another. However, there are indications in theoretical economic studies that the dynamics of some decentralised markets may converge to stable but highly sub-optimal equilibria.
- etc. (see [7] for more characteristics)

5.5 Related Mechanisms/Patterns

Variations

- *Multiple Markets [10]:* Multiple related markets are used. An example on how such markets can be related is when resources on one market are a composite of resources on another market (e.g. 'network path' contains 'network links' [10]).
- etc. (see [7] for more variations)

Other Coordination Mechanisms. Another coordination mechanism that can solve resource allocation is the *Tokens* mechanism. A token then represents the capability and authority to use a certain resource exclusively and the number of tokens in the system is limited to the available amount of resources.

5.6 Examples/Known Uses

Some known uses and possible applications are: manufacturing control [24] (resource = machine), power distribution [32] (resource = power), routing in networks [10] (resource = bandwidth), stabilisation of unstable (civil) structures [11,17] (resource = power to push somewhere), climate control in buildings [32] (resource = cold air), and other cases on distributed resource allocation [32,4].

6 Case Study: A Packet Delivery Service

This section considers the design of a packet delivery service application to illustrate the use of decentralised coordination patterns. First the requirements are described and then a design is proposed using the patterns.

Problem Statement and Requirements. A packet delivery service [27] allows customers to submit an order to come and pick up a packet and transport it to a given destination or delivery location (see figure 6). At each moment in time, new pickup and delivery locations can be added to the system by customers. A fleet of trucks is available which has to self-organise efficiently to accommodate the current request for transport. As such, the orders of customers form a dynamic overlay network of pickup and delivery locations between which trucks have to move routing themselves through a street map. So basically there are two main requirements for this problem:

- Dispatching: every new order has to be assigned to a truck that will be responsible for handling the requested transport.
- Routing: trucks have to be adaptively routed through a street map in order to reach new pickup locations for orders to which they were assigned and to reach delivery locations for orders already inside the truck.

These requirements have to be achieved in the face of frequent changes: new orders arriving at any moment, changes to the delivery location of existing orders, congestion and obstacles on streets, trucks failing, etc. On the other hand there are a number of timing constraints that have to be reached: pickup time-window in which the pickup should occur, delivery time-window, regular breaks for drivers to rest, etc. This is a highly dynamic problem in which the information needed to decide how to route or dispatch is inherently decentralised over a number of trucks, customers, streets, etc. As such a self-organising emergent solution is promising.

Design with Decentralised Coordination Patterns. As mentioned earlier, the problem-solving power resides in the interaction between agents. Therefore for each of the requirements that have to be coordinated one has to discover which information is needed to take the appropriate decisions and actions. And especially, which decentralised coordination mechanism allows to exchange that kind of information and to coordinate the agents to achieve the requirements.

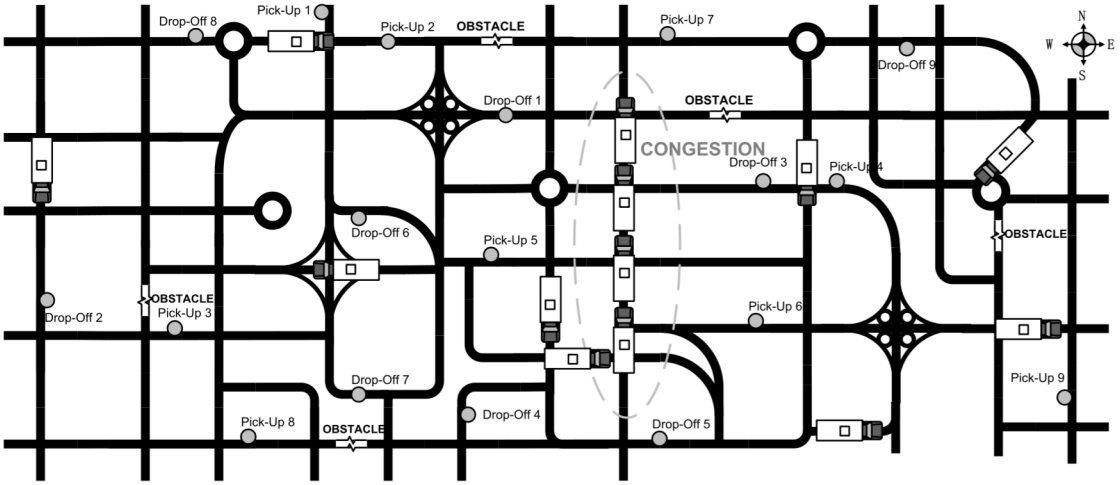

Fig. 6. The Packet-Delivery Problem

Consider the *dispatching requirement* for which new orders have to be assigned as a task to available trucks. Information is needed to decide which truck is chosen, such as the distance of the truck to the pickup location or its estimated arrival time, the trucks available with enough room to carry the packet, and the estimated delivery times of trucks. The best truck at that moment should be allocated. To solve this problem systematically, the 'Problem/Solution Summary' described in Table 1 is used. The engineer has to find a problem description matching the dispatching problem. Some kind of allocation of resources is required. The resource is the room available in trucks that has to be allocated to an order. Table 1 states that a Market-based Coordination mechanism may be suitable. Therefore, the consumers and suppliers of the resource market have to be determined. For example, consider the trucks as the suppliers of available transportation room, and order agents, representing orders, are the consumers of that room. Another important aspect of markets is the instantiation of the price mechanism. According to the pattern, a number of price values are involved such as limit prices under which the room is not bought or sold, prices offered in individual bids, and the market price. These prices should depend on information important for making the allocation decisions because in a market limit prices, offered prices, and market price determine what is allocated to whom. As such, for the order dispatching a truck could have a limit price that increases with the distance or estimated travel time to reach the pickup, increases with the estimated delivery time, decreases with the amount of available room in the truck, etc. The order's limit price can depend on the wanted delivery and pickup time constraints so that an order willing to pay a high price is willing to wait longer before pickup and/or doesn't require a short delivery time. A market is started when an order is submitted by a customer and stops only when it has been assigned to a truck. During the market execution trucks will offer to supply room at a certain price as far as possible above their limit price to maximise their profit (i.e. for example, the higher the price payed, the more time the truck has to deliver the order). Orders will bid to buy the room at certain prices that

are as far as possible below their limit price in order to get the best deal (i.e. for example, the lower the price, the quicker the delivery is done). The market mechanism of demand and supply then allocates the trucks to the orders in such a way that approximates a globally efficient optimum at that moment.

The second requirement was to actually *route the trucks through the street map*. The problem to be solved is a routing or spatial movement problem for which there are two possible coordination patterns according to table 1: digital pheromones or gradient fields. To decide which one is the most promising the patterns need to be studied in more detail, i.e. which characteristics match the needs for routing trucks, etc. Assume that the gradient field approach is the most suitable mainly because the digital pheromone approach requires that trucks actively search for delivery locations, orders, etc. On the other hand, the gradients are automatically propagated by the environment and trucks only have to follow their coordination gradient to be routed through the street map efficiently. As such new orders are immediately found by trucks which is an important requirement. Without giving a more detailed solution the main idea is using different types of gradients:

- *Delivery Location Gradients:* each delivery location emits a gradient as soon as the order is assigned to a truck. The truck then simply follows that gradient.
- *Pickup Location Gradients:* each pickup location emits a gradient to signal the presence of a new order. The truck assigned to the order simply follows the gradient to reach the pickup.
- *Market Communication Gradients:* to facilitate direct interaction and negotiation needed for markets between trucks and orders at pickup locations each can emit a market gradient. That gradient is used to actively send bids to buy room, offers to sell room, payments, etc. These messages can route themselves by simply following the right gradient. Each truck and order have their own gradient.

Note that a requirements was to cope with dynamics such as congestion and obstacles. The gradients used above all take into account these changes in their propagation rules. For example, the propagation occurs slower, not at all, or with a higher increase in strength through congested streets. Similar for other dynamics.

Once the coordination mechanisms are chosen, the pattern description offers a guide to actually apply them: different variants can be considered, the parameters to tune are known, guidelines are available, etc. In particular, a gradient solution requires an environment capable of storing and propagating gradients, i.e. an infrastructure. Such an environment is not available or too expensive on a real street network. Alternatively, a decentralised solution which is not distributed on the street network can be used. As such a server emulates the street network as a graph environment in which gradients propagate and on which truck agents move in sync with and actively coordinate the movement of the real trucks. Changes such as information on congestion and obstacles is updated in that emulated environment in a decentralised manner by directly linking the

real world trucks with the agents. Customers submit their orders to this server. The advantages of a self-organising solution are preserved because the solution is constantly adapting to changes without a central controlling entity and is still robust to truck failure. Of course, a failure of the server emulating the environment would break the system. However, making the server failure-safe remains cheaper than embedding a gradient infrastructure in a real street network.

7 Conclusion and Future Work

To systematically design a self-organising emergent solution, guidance on which decentralised coordination mechanisms to use is essential because the problem-solving power resides in the coordination process. Decentralised coordination mechanisms can be described as design patterns, similar to patterns used in mainstream software engineering. Such a clear and structured description format helps in making the engineering process more systematic for two reasons:

- Firstly, pattern descriptions allow to *directly find a solution based on the problem.*
- Secondly, each *pattern is a consolidation of best practice* to use it, which *systematically guides engineers* in applying the coordination mechanisms.

However, the patterns in this paper and in [7] have a conceptual focus. More work is needed on patterns for designing coordination mechanisms at the class or implementation level. Even for the conceptual patterns given, more structure is possible. For example, identifying the participants and interactions and representing this structurally in UML interaction diagrams; or identifying new sections in the pattern format that address issues specific for self-organising emergent systems. Also, many other coordination mechanisms should be captured as design patterns to easily compare and choose between them when engineering a self-organising emergent solution. Exemplary work can be found in [2].

Acknowledgements. This work is supported by the K.U.Leuven research council as part of the concerted research action on Autonomic Computing for Decentralised Production Systems (project 3E040752).

References

1. R. Axtell and J. Epstein. *Distributed Computation of Economic Equilibria via Bilateral Exchange.* Brookings Institution, Washington DC, 1997.
2. O. Babaoglu, G. Canright, A. Deutsch, G.A. Di 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 Transactions on Autonomous and Adaptive Systems*, 1(1):26–66, September 2006.
3. S. Brueckner. *Return From The Ant - Synthetic Ecosystems For Manufacturing Control.* PhD thesis, Humboldt-Universitt, Berlin, 2000.

4. D. Cliff and J. Bruten. Simple bargaining agents for decentralized market-based control. Technical Report HPL-98-17, HP Labs, Bristol, UK, 1998.
5. T. De Wolf and T. Holvoet. Emergence and Self-Organisation: a statement of similarities and differences. In *Proc. of the 2nd Int. Workshop on Engineering Self-Organising App.*, 2004.
6. T. De Wolf and T. Holvoet. Towards a methodolgy for engineering self-organising emergent systems. In H. Czap, R. Unland, C. Branki, and H. Tianfield, editors, *Self-Organization and Autonomic Informatics (I)*, volume 135 of *Front. in Artif. Intell. and App.* IOS Press, 2005.
7. T. De Wolf and T. Holvoet. A catalogue of decentralised coordination mechanisms for designing self-organising emergent applications. CW 458, Department of Computer Science, K.U.Leuven, August 2006.
8. O. Etzioni. Moving Up the Information Food Chain: Deploying Softbots on the World Wide Web. In *Proc. of the 13th Int. Conf. on Artificial Intelligence*, 1996.
9. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Professional Computing Series. Addison-Wesley, 1995.
10. M. A. Gibney, Nicholas R. Jennings, N. J. Vriend, and José-Marie Griffiths. Market-based call routing in telecommunications networks using adaptive pricing and real bidding. In *IATA '99: Proceedings of the Third International Workshop on Intelligent Agents for Telecommunication Applications*, pages 46–61, London, UK, 1999. Springer-Verlag.
11. O. Guenther, T. Hogg, and B. Huberman. Power markets for controlling smart matter. Computing in Economics and Finance 1997 62, Society for Computational Economics, 1997.
12. D. Hales. Choose your tribe! - evolution at the next level in a peer-to-peer network. In *Proc. of the 3rd Workshop on Engineering Self-Organising Applications (EOSA 2005)*, 2005.
13. K. Herrmann, G. Mhl, and K. Geihs. Self-Management: The Solution to Complexity or Just Another Problem? *IEEE Distributed Systems Online*, 6(1), 2005.
14. I. Jacobson, G. Booch, and J. Rumbaugh. *The unified software development process.* Addison Wesley, 1999.
15. J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer Magazine*, 36(1):41–50, Jan 2003.
16. C. Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.* Prentice Hall, 3rd edition, 2005.
17. J.P. Lynch and K.H. Law. Decentralized control techniques for large-scale civil structural systems. In *Proc. of the 20th Int. Modal Analysis Conference (IMAC XX)*, 2002.
18. M. Mamei, M. Vasirani, and F. Zambonelli. Experiments of morphogenesis in swarms of simple mobile robots. *Applied Artificial Intelligence*, 18(9-10):903–919, 2004.
19. M. Mamei and F. Zambonelli. Motion coordination in the quake 3 area environment: A field-based approach. In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *Environments for Multi-agent Systems - First E4MAS workshop, New York, NY, July 19, 2004, Revised Selected Papers*, volume 3374 of *Lecture Notes in Computer Science*, page 264. Springer Verlag, 2005.
20. M. Mamei and F. Zambonelli. Theory and practice of field-based motion coordination in multiagent systems. *J. Appl. Artif. Intell.*, 19, 2005. to be published.

21. M. Mamei, F. Zambonelli, and L. Leonardi. Distributed motion coordination with co-fields: A case study in urban traffic management. In *Proc. of the The 6th Int. Symp. on Autonomous Decentralized Systems (ISADS'03)*, page page 63, Washington, DC, USA, 2003. IEEE CS.
22. M. Mamei, F. Zambonelli, and L. Leonardi. Co-fields: A physically inspired approach to motion coordination. *IEEE Pervasive Computing*, 3(2), 2004.
23. G. Meszaros and J. Doble. Metapatterns: A pattern language for pattern writing. In *The 3rd Pattern Languages of Programming conference*, Monticello, Illinois, USA, September 1996.
24. H. V. D. Parunak, A. D. Baker, and S. J. Clark. The aaria agent architecture: From manufacturing requirements to agent-based system design. *Integrated Computer-Aided Engineering*, 8(1):45–58, 2001.
25. E. Rimon and D. E. Kodischek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.
26. Editor S. H. Clearwater. *Market-Based Control: A Paradigm for Distributed Resource Allocation.* World Scientific, Signapore, 1996.
27. M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:1729, 1995.
28. H. Varian. *Intermediate Microeconomics.* W.W.Norton, New York, USA, 1999.
29. D. Weyns, A. Helleboogh, E. Steegmans, T. De Wolf, K. Mertens, N. Bouck, and T. Holvoet. Agents are not part of the problem, agents can solve the problem. In *Proc. of the OOPSLA Workshop on Agent-Oriented Methodologies*, 2004.
30. M. Wooldridge and N.R. Jennings. Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, 3(3):20–27, 1999.
31. Y. Xu, P. Scerri, B. Yu, S. Okamoto, M. Lewis, and K. Sycara. An integrated token-based algorithm for scalable coordination. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS)*, pages 407–414, Utrecht, NL, 2005.
32. F. Ygge and H. Akkermans. Decentralized markets versus central control: A comparative study. *Journal of Artificial Intelligence Research*, 11:301–333, 1999.