Towards a Group Selection Design Pattern*

David Hales[1], Stefano Arteconi[1], Andrea Marcozzi[1], and Isaac Chao[2]

[1] Dept. of Computer Science, University of Bologna,
Mura Anteo Zamboni 7, 40127 Bologna, Italy
{hales, arteconi, marcozzi}@cs.unibo.it
[2] Arquitectura de Computadors, Universitat Politecnica de Catalunya,
Jordi Girona, 1-3, 08034 Barcelona, Spain
ichao@lsi.upc.es

Abstract. Recent work has shown how socially inspired mechanisms developed within social science simulations can be applied to a number of application domains in distributed systems engineering. Here we proposed a general mechanism based on an evolutionary "group selection" process in which individual entities (agents, nodes or components) dynamically copy the behavior of other entities if they achieve higher performance. We present this approach by beginning to sketch a generic "design pattern" and then show how it can be applied in different domains. Finally we discuss some open issues and future work.

1 Introduction

Recent models from computational social science and theoretical biology have demonstrated novel processes of group selection [23, 25, 28]. They demonstrate how selfish behavior at the individual level can, never-the-less, lead to highly co-operative and co-ordinated emergent behaviour at the collective level.

Here we discuss the formulation of a group selection "design pattern" which balances the often contradictory forces between individual goals and collective goals in distributed systems without the need for any centralised structures or control.

Previous work has shown how biologically inspired approaches can be presented in the form of design patterns [4] and, indeed, some have suggested that the "tag" approach (see later) could be formulated as a pattern [7].

The group selection approach is of great potential value for the engineering of massive, decentralized and open distributed computer systems because it embodies several attractive features: It is general, applicable to wide range of scenarios; it is scalable, allowing many millions of entities to coordinate efficiently; it is robust, meaning that entities may enter and leave the system without disrupting interaction drastically; it is self-organising requiring no central control or administration; it is resistant to certain kinds of free-riding, when entities try to exploit the system for their own benefit. Importantly, because entities only need to act on their own local performance criteria, there is no need for global information or the representation or enforcement of group or system level goals.

Group selection relies on the dynamic formation and dissolution of groups. Over time individual entities may change groups by moving to those that offer better individual performance. Interaction between entities that determine performance is mainly restricted to those sharing the same group. Essentially then, in a nutshell, groups that support high performance for the individuals that comprise them grow and prosper

* Partially supported by the EU within the 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS).

whereas exploitative or dysfunctional groups dissolve as individuals move away. Hence functional groups, in terms of satisfying individual goals, are selected over time.

Key aspects that define applications of group selection are: How group boundaries are formed; the nature of the interactions between entities within each group; the way that each entity calculates their individual performance (or utility) and how entities migrate between groups.

The success of any application of group selection is judged by how well the system self-organises towards achieving a collective goal - whatever that may be. Often this will be maximising the sum of individual performances but could involve other measures such as equality or fairness for example.

In the following sections we describe, in overview, several realisations of group selection applied to various application domains. For each realization we describe the domain or scenario, the implementation of each key aspects and give some results of simulations. More detail on each model can be found in the associated papers cited in each section.

Before we introduce our set of application models we overview and compare some recent social and biological models of group selection from which our approach is inspired. Then we present an abstract computational model which identifies the key assumptions, aspects and emergent process we associate with group selection. We then present each application model within this standard template indicating how each key aspect is implemented.

We believe the work presented in this paper contributes as a step towards a general "design pattern" for applying group selection in information systems. However we believe much more work needs to be done to achieve our goal. In the final section we discuss some open issues and indicate potential ways forward.

2 Recent social and biological models of group selection

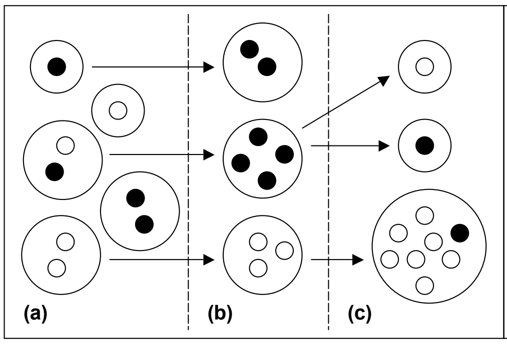
In almost all proposed social and biological models of group selection, in order to test if group selection is stronger than individual selection, populations are composed of individuals that can take one of two kinds of social behaviour (or strategy). They can either act pro-socially, for the good of their group, or they can act selfishly for their own individual benefit at the expense of the group. This captures a form of commons tragedy [16].

Often this is formalised as a Prisoners Dilemma (PD) or a donation game in which individuals receive fitness payoffs based on the composition of their group. In either case there is a fitness cost c that a pro-social individual incurs and an associated fitness benefit b that individuals within a group gain. A group containing only pro-social individuals will lead each to gain a fitness of $b-c$. However, a group containing only selfish individuals will lead each to obtain a fitness of zero. But a selfish individual within a group of pro-socials will gain highest fitness. In this case the selfish individual will gain b but the rest will gain less than $b-c$. Given that b and c are positive then it is always in an individuals interests (to maximise fitness) to behave selfishly. In an evolutionary scenario in which the entire population interacts within a single group then selfish behaviour will tend to be selected because this increases fitness. This ultimately leads to an entire population of selfish individuals and a suboptimal average population level fitness of zero. This is the Nash Equilibrium [20] and an Evolutionary Stable Strategy for such a system [26].

There have been various models of cooperation and pro-social behaviour based on reciprocity using iterated strategies within the PD [3]. However, we are interested in models which do not require reciprocity since these are more generally applicable. In many situations, such as large-scale human systems or distributed computer systems, repeated interactions may be rare or hard to implement due to large population sizes (of the order of millions) or cheating behaviour that allow individuals (or computer nodes) to fake new identities.

2.1 Tag model

In [15] a tag model of cooperation was proposed which selected for pro-social groups. It models populations of evolving agents that form groups with other agents who share an initially arbitrary tag or social marker. The tag approach was originally proposed by Holland [17] and developed by Riolo [22, 23]. The tag is often interpreted as an observable social label (e.g. style of dress, accent etc.) and can be seen as a group membership marker. It can take any mutable form in a model (e.g. integer or bitstring). The strategies of the agents evolve, as do the tags themselves. Interestingly this very simple scheme structures the population into a dynamic set of tag-groups and selects for pro-social behaviour over a wide range of conditions. Figure 1 shows a schematic diagram of tag-group evolution and an outline algorithm that generates it.



Outline algorithm for tag model:

```
for each generation loop
    interaction within groups (obtain fitness)
    reproduce individuals based on fitness
    with Prob(mt) individuals form new group
    with Prob(ms) individuals flip strategy
end generation loop
```

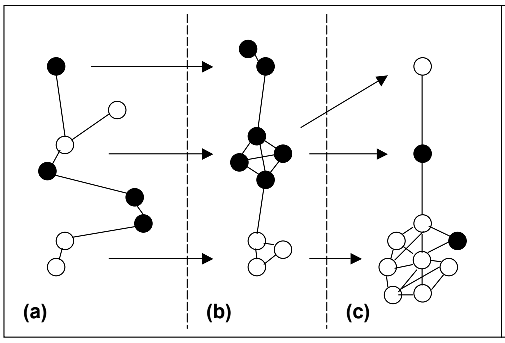
Group boundary: tag stored by each individual defines group membership
Group formation and migration: probabilistic mutation of tag

Fig. 1. Schematic of the evolution of groups in the tag model. Three generations (a-c) are shown. White individuals are pro-social, black are selfish. Individuals sharing the same tag are shown clustered and bounded by large circles. Arrows indicate group linage. Migration between groups is not shown.

In general it was found that pro-social behaviour was selected when $b > c$ and $mt >> ms$, where *mt* is the mutation rate applied to the tag and *ms* is the mutation rate applied to the strategy. In this model groups emerge from the evolution of the tags. Group splitting is a side effect of mutation applied to a tag during reproduction. A subsequent tag model [23] produced similar results although it cannot be applied to pro-sociality in general because it does not allow for fully selfish behaviour of identically tagged individuals [24, 8].

2.2 Network-rewiring model

Network rewiring models for group selection have been proposed with direct application to peer-to-peer (P2P) protocol design [13, 14]. In these models, which were adapted from the tag model described above, individuals are represented as nodes on a graph. Group membership is defined by the topology of the graph. Nodes directly connected are considered to be within the same group. Each node stores the links that define its neighbours. Nodes evolve by copying both the strategies and links (with probability t) of other nodes in the population with higher utility than themselves. Using this simple learning rule the topology and strategies evolve promoting pro-social behaviour and structuring the population into dynamic arrangements of disconnected clusters (where $t = 1$) or small-world topologies (where $0.5 < t < 1$). Group splitting involves nodes disconnecting from all their current neighbours and reconnecting to a single randomly chosen neighbour with low probability mt. As with the tag model pro-social behaviour is selected when $b > c$ and $mt >> ms$, where ms is the probability of nodes spontaneously changing strategies. Figure 2 shows a schematic of network evolution (groups emerge as cliques within the network) and an outline algorithm that implements it.



Outline algorithm for network model:

```
for each generation loop
    interaction within groups (obtain fitness)
    reproduce individuals based on fitness
    with Prob(t) copy new links
    with Prob(ms) individuals flip strategy
end generation loop
```

Group boundary: individuals directly linked in the network
Group formation and migration:copying of links probabilistically

Fig. 2. Schematic of the evolution of groups (cliques) in the network-rewiring model. Three generations (a-c) are shown. White individuals are pro-social, black are selfish. Arrows indicate group linage. Notice the similarity to the tag model in figure 1.

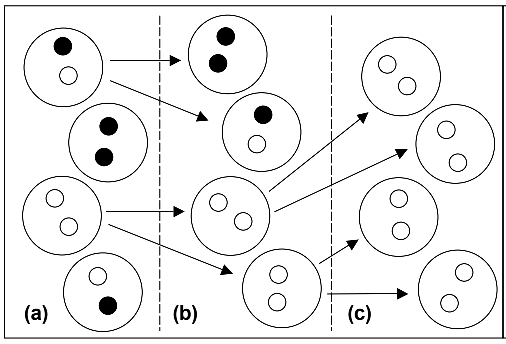
In this model we have translated dynamics and properties similar to the tag model into a graph. This is important because P2P networks can be viewed as graphs. Hence with relatively modest translation effort we were able to formulate a general P2P protocol that would promote pro-social behaviour over a wide rage of potential application domains. In [14] we applied the protocol to a simulated file-sharing scenario. In [13] we proposed a protocol that could be applied to collective spam filtering. In [12] the same fundamental rewiring protocol was applied to a scenario requiring nodes to adopt specialised roles or skills within their groups, not just pro-social behaviour alone, to maximise social benefit.

Interestingly it has also been shown recently [21] in a similar graph model tested over fixed topologies (e.g. small-world, random, lattice, scale-free) that under a simple evolutionary learning rule pro-social behaviour can be sustained in some limited situations if $b/c > k$, where k is the average number neighbours over all nodes (the average degree of the graph). This implies that if certain topologies can be imposed then pro-social behaviour can be sustained without rewiring of the topology dynamically. Although analysis of this model is at an early stage it would appear that groups form via clusters of pro-social strategies forming and migrating over the graph via nodes learning from neighbours.

Also a recent network model shows similar properties [25].

2.3 Group-splitting model

In [28] a group selection model is given that sustains pro-social behaviour if the population is partitioned into m groups of maximum size n so long as $b/c > 1 + n/m$. In this model group structure, splitting and extinction is assumed *a priori* and mediated by exogenous parameters. Splitting is accomplished by explicitly limiting group size to n, when a group grows through reproduction beyond n it is split with (high) probability q into two groups by probabilistically reallocating each individual to one of the new groups. By endogenously controlling n and m a detailed analysis of the model was derived such that the cost / benefit condition is shown to be *necessary* rather than just sufficient. The model also allows for some migration of individuals between groups outside of the splitting process. Significantly, the group splitting model can potentially be applied recursively to give multilevel selection – groups of groups etc. However, this requires explicit splitting and reallocation mechanisms at each higher level. Figure 3 shows a schematic of group-splitting evolution and an outline algorithm that implements it.



Outline algorithm for split model:

```
for each generation loop
    interaction within groups (obtain fitness)
    reproduce individuals based on fitness
    with Prob(q) split any group > m in size
                eliminate random group
end generation loop
```

Group boundary: individuals exogenously given group membership
Group formation and migration: splitting of group when size > m.

Fig. 3. Schematic of the evolution of groups in the group-splitting model. Three generations (a-c) are shown. White individuals are pro-social, black are selfish. Individuals sharing the same group are shown clustered and bounded by large circles. Arrows indicate group linage. Migration between groups is not shown.

3 Abstract Computational Model

In order to describe different realizations of the group selection mechanism in information systems applications we abstract assumptions and key aspects in the context of computational entities (e.g. agents, nodes in a network, individual processes etc.). Each instantiation of the mechanism specifies a way of implementing each aspect such that the emergent process occurs.

3.1 Assumptions

The group selection approach can be applied to systems where the following general assumptions hold:

- A system is composed of individual entities that can benefit from interaction with other entities
- The population of entities is partitioned into groups such that interaction is mainly limited to entities within the same group
- Entities measure their own performance periodically producing a utility value
- Entities may spontaneously change their behavior and group membership
- Entities may view and copy some state of other entities
- Entities desire to increase their performance (utility)

3.2 Key aspects

We have found, as will be seen in the later application examples, that any instantiation of the group selection process involves identifying the following aspects of a system - we use these key aspects as a template by which we related each application of group selection:

- Collective Goal - A desirable goal that the population of entities should attain.
- Group Boundary Mechanism (Group) - How an entity can locate and communicate with in-group members.
- Intra-Group Interaction (Interaction) - What kinds of utility effecting interactions an entity participates in with other in-group members.
- Utility Calculation Metric (Utility) - How an entity calculates a utility value based on its individual goal and in-group interactions.
- Group Migration Mechanism (Migration) - How migration between groups is performed.

3.3 Emergent Process

By correctly defining the key aspects above an emergent group selection occurs, achieving the collective goal, following the below general form:

- Entities are grouped in some initially arbitrary way
- Interactions between entities within groups determine entity utilities
- Based on utility comparisons between entities, and possibly randomized change, group memberships and interaction behavior (strategy) change over time
- Groups which produce high utility for their members tend to grow and persist as entities join
- Groups which produce low utility for their members tend to disperse as entities leave
- Hence group beneficial behavior tends to be selected

4 Tag Cooperation

TagWorld is a simple simulation model demonstrating group selection by using a “tag” method to maintain group boundaries and migration. Tag models have been developed within computational social science [17, 23]. He we outline a model based on [15]. Although not an application we provide this model as a baseline case illustrating the key aspects of group selection.

Figure 4 summarizes the key aspects of the model. Note that groups are formed using ”tags”. These are values associated with agents and identify them as members of a particular group. They can be copied and observed by other agents. This is a simple (possibly minimal) way to construct dynamic group boundaries.

Collective Goal	Maximize collective utility while there are incentives and opportunities for selfish egoistical behavior.
Entity	Agent - a process or thread running on a processor with inter-agent communications infrastructure. Each agent stores a Tag and a Strategy (see below).
Group	Tag - Agents store a single group ID value called a Tag. All agents storing the same value are considered to be within the same group.
Interaction	PD game - Agents store a pure PD strategy (either C or D) and perodically play a one-shot game of PD with a randomly chosen in-group member.
Utility	Average PD Payoff - Agents periodically calculate a utility by calculating the average payoff obtained from playing PD games.
Migration	Periodically (after each agent has played a game of PD) agents reproduce, to form a new agent population, probabilistically in proportion to the average payoff they received (using a roulette wheel selection algorithm). The population is kept constant in size over generations.

Fig. 4. Key aspects for the TagWorld model

4.1 Interaction

Agents have the task of playing the single round two player Prisoner’s Dilemma (PD) game with their in-group members. Table 5 shows the so-called payoff matrix for this simple game. The game is a valuable test because maximum collective utility (payoff) requires agents to select a C move but maximum individual utility is obtained by selecting a D move. Hence if group selection is operating sufficiently well we would expect to find high levels of C (cooperation) even though agents are trying only to optimize their own utility.

	Cooperate	Defect
Cooperate	R, R	S, T
Defect	T, S	P, P

Fig. 5. A payoff matrix for the two-player single round Prisoner’s Dilemma (PD) game. Given $T > R > P > S \wedge 2R > T + S$ the Nash equilibrium is for both players to select Defect but both selecting Cooperate would produce higher social and individual returns. However, if either player selects Cooperate they are exposed to Defection by their opponent — hence the dilemma

4.2 Migration

In the TagWorld model migration is a result of reproduction. After each agent in the population has played a game of PD a new population is reproduced using a standard "roulette wheel" selection method. If an agent is reproduced, it's Tag and Strategy are reproduced and a mutation operation is applied with low probability (randomly changing the Tag and / or Strategy). This differential reproduction based on utility has the effect of migrating agents between different tags (groups) over generations and changing the distribution of strategies in the population.

4.3 Simulation Algorithm

The model was realized as a simulation. The pseudo-code for the simulation is shown in figure 6. The probability of mutation applied to the the strategy was m = 0.001. Mutation applied to the Tag was an order of magnitude higher.

```
LOOP some number of generations
    LOOP for each agent (a) in the population
        Select a game partner agent (b) with same tag (if possible)
        Agents a and b invoke their strategies and get payoffs
    END LOOP
    Reproduce agents in proportion to their average payoff
    Apply mutation to tag and strategy of each reproduced agent with low probability
END LOOP
```

Fig. 6. Pseudo-code for the TagWorld simulation model.

4.4 Results

It was found that from random (Tag and Strategy) initializations TagWorld rapidly converged to very high levels of cooperation. Due to mutation the system never converges to a stable 100% cooperative state but over simulation runs of 10^5 generations we found over 99% of all PD interactions were mutually cooperative. When all agents were initialized as defectors we observed that the population still eventually reached over 99% cooperation. We produced an analysis of this and found the time to reach high cooperation in generations was dependent on mutation rate *m* and population size *n*. Figure 7 shows a chart of results over various population sizes.

5 File sharing

FileWorld is a simulation model of file-sharing in P2P systems. The goal of the model is to apply a group selection approach to a file-sharing scenario in which a certain level of peer altruism is required to maximize the efficiency of the network as a whole. The task involves nodes sending and serving queries for files. A node measures its performance by counting the number of query hits it obtains over a time period. The simulated scenario is modelled after [27]. The group selection key aspects are shown in figure 8. The results presented in this section are an overview of those given in [14].

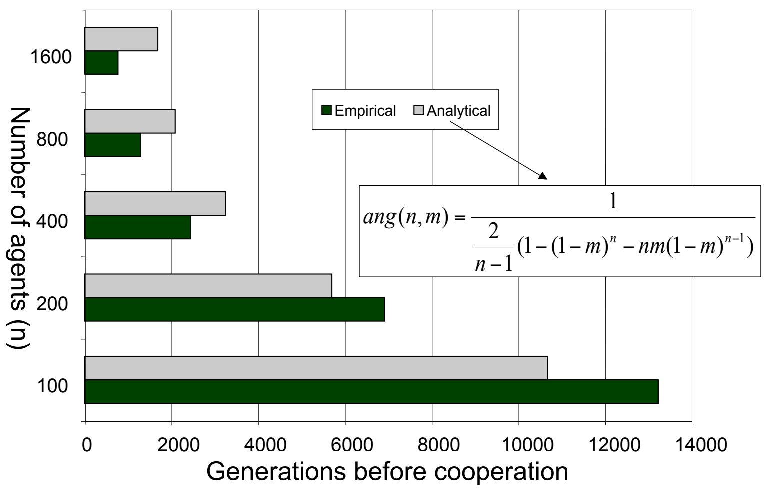


Fig. 7. Simulation and analysis of results for the TagWorld simulation model over different population sizes. Notice that as the population size increases the time to high cooperation reduces.

Collective Goal	Maximize the total number of query hits in the file-sharing network as-a-whole.
Entity	Peer Node - a node in a peer-to-peer overlay network with inter-agent communications infrastructure. Each node stores a neighbor list (or view) and an Altruism Level (see below). Periodically agents may randomly change their neighbor list and altruism level.
Group	View - nodes store a list of links to other peers called their view. All peers within the view are considered to be within the same group. Hence producing an overlapping network of groups.
Interaction	Sending and serving queries for files. Peers have a fixed capacity determining how many query messages they can handle in a given period. Nodes store an "altruism level" [0..1] which specifies the proportion of capacity devoted to serving requests from others as opposed to sending their own requests.
Utility	Query Hits - Peers generate their own queries with the capacity left over after processing others queries. Utility is the number of such answered queries (or hits) over a given period.
Migration	Copying peers with higher utility - Peers periodically select another peer randomly from the entire population (which may include peers outside the in-group). If the utility of this other node is higher then the peer copies its View and Altruism Level (overwriting previous values). By copying the View the agent migrates to the group of the copied node.

Fig. 8. Key aspects for the FileWorld model.

```
LOOP some number of cycles
    Initialise all node capacities and utilities
    LOOP some number of node firings (a time period)
        Select a random node (a) from the population
        IF node (a) has capacity to generate queries
            Decrease capacity by one query
            Generate query and pass to all neighbours (see below)
            Accumulate number of hits (utility)
        END IF
    END LOOP
    LOOP some number of times
        Select randomly a pair of nodes from the population
        Copy view and altruism value from higher to lower utility node
        Apply mutation with low probability to view and altruism value
    END LOOP
END LOOP

When node (b) receives a query:

IF node (b) has capacity to answer queries
    Decrease capacity by one query
    With ``Answering Power" probability produce a ``hit"
    IF no hit produced and query TTL > 0
        Reduce TTL
        Pass query to all neighbours
    END IF
ELSE
    Ignore query
END IF
```

Fig. 9. Pseudo-code for the FileWorld simulation model.

5.1 Interaction

Within a time period nodes are periodically "fired". When a node is fired, it will generate a query if it still has spare capacity left to do so. When a node generates a query it passes it to all its neighbour nodes. If a neighbour node has spare capacity left to answer another's queries then it will produce a "hit" with a given probability. Each node has an "answering power" value which is the probability it can match any query. If it does not produce a "hit" then it passes on the query to it's neighbours following a flood fill approach until some "time to live" (TTL) is reached. In the case of the FileWorld simulations TTL was set to 3. Conversely if a node has no spare capacity to answer others queries then it ignores the query without producing a "hit" or passing it on. Nodes store an "altruism level" which represents a behaviour strategy. Nodes with an altruism level of zero would always reject queries from others and use all its capacity to generate its own queries. A node with an altruism value of one would devote all its capacity to answering others queries and never generate any of its own. Hence for the system to achieve the collective goal, of maximising the number of hits over all nodes, a balance needs to be found between zero altruism and total altruism because if all nodes simply wait to serve others queries then no queries will be generated at all. Conversely if all nodes only generate their own queries and never serve others then no hits will be produced. For the results presented here the capacities and answering powers of all nodes were set to the same fixed rates after [27].

5.2 Migration

After a time period nodes refresh their capacities and randomly selected nodes pairs compare their utilities (total number of hits). Nodes with lower utility copy the altruism value and neighbour list of nodes with higher utility, also they make a link to the higher utility node. Old links are dropped. This re-wiring approach means that neighbours migrate to new neighbourhoods in the network. After a copy operation a “mutation” is applied with small probability to both links and altruism value. If links are mutated all links are replaced with a random link. If the altruism value is mutated it is replaced with uniform random value [0..1].

5.3 Simulation Algorithm

The model was realised as a simulation. The pseudo-code for the simulation is shown in figure 9. We used the same mutation rates as used in the TagWorld model (see previous section). A time period, or cycle, comprised *NC* node firings, where *N* was the number of nodes in the population and $C = 100$ was the capacity of each node. Both generating or serving a query was assumed to consume one unit of capacity.

5.4 Results

Figure 10 shows a typical run of the FileWorld model. Initial altruism values for each node were generated uniformly randomly in range [0..1]. Also the initial topology of the network was randomly generated. The results are for a population size of N^4 nodes. We found similar results for large networks up to N = 10^5. Indeed as in the TagWorld model we found that larger populations performed better in terms of time to high performance.

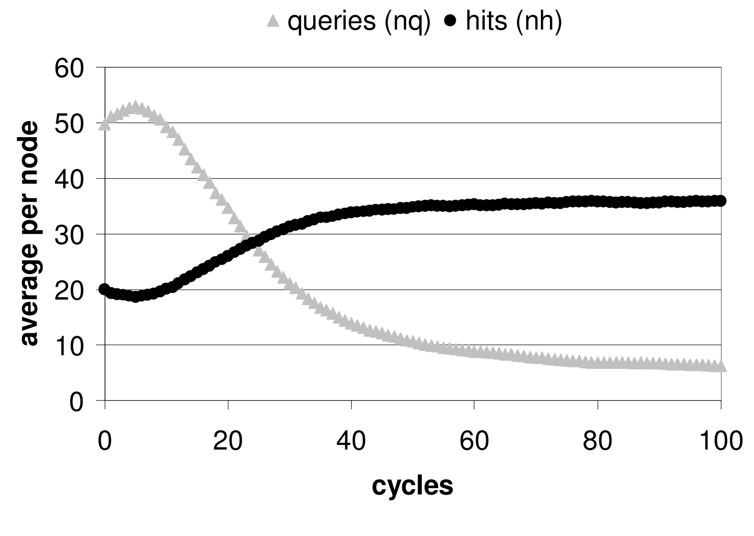


Fig. 10. Results for a typical run of the the FileWorld simulation model. The value (nq) shows average number of queries generated by nodes over time and (nh) shows the averagel number of hits obtained. Notice initially the population does worse than the purely random configuration but quickly adjusts increasing the number of hits but reducing flooding of the system with queries. This is due to an optimisation of the altruism value over time.

6 Grid policies

Grid computing originated a decade ago to aggregate distributed computational resources to facilitate scientific computational intensive tasks. The have also been applied in production systems (enterprise Grids). A Grid computing system is differentiated from other distributed computer systems because 1) Users request both software and hardware resources: databases, data files, applications; and processing capacity, storage space, network capacity, and expect those resources to provide quality of service, and 2) Grid users make concurrent use of several resources from different providers, requiring resource co-allocation and coordinated utilization. De facto standard Grid technologies have emerged such as the Globus toolkit [9]. This Grid middleware provides a set of low-level building blocks enabling the composition of interoperable services for distributed resource management and monitoring, as well as for remote task scheduling.

The Grid problem has been defined as coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [10]. Grid environments are organized in Virtual Organizations (VOs), associating heterogeneous users and resource providers with a common goal. Accomplishment of tasks implies coordination of VO participant policies for resource management. Such policy coordination gets complex for grids comprising heterogeneous administrative domains. Manual coordination becomes infeasible in practice, especially for dynamic Grid VO settings [29]. Addressing those issues will require a shift in engineering approach, from traditional policy management, to emergent, self-organizing policy management.

Here we apply the group selection pattern to the self-organized coordination of Grid policies in VOs. GridWorld is a simulation model of Grid policy interactions. The pattern is sumarized in figure 11. The results presented here give an overview. More extensive models and details can be found in [5, 6].

Collective Goal	Coordinate Grid nodes resource management policies to increase total collective utility.
Entity	Grid node - A host machine or device applying a local resource management policy, with a inter-node communications infrastructure. Each agent stores a (VO) identifier and a Policy.
Group	VO - Grid nodes store a single virtual organisation (VO) identifier. All nodes storing the same value are considered to be within the same VO or group.
Interaction	Policy Coordination game - Grid nodes mantain a resource management policy (from a pool of three) and perodically play a one-shot policy coordination game with a randomly chosen in-group (VO) member.
Utility	Average coordination game payoff - Grid nodes periodically calculate a utility by calculating the average payoff obtained from playing the policy coordination game with VO members.
Migration	Nodes join VOs where they find other nodes performing well and copy the policies of those nodes. Rrandom migration to a new VOs (mutation) is applied with a fixed probability *m*.

Fig. 11. Key aspects for the GridWorld model

6.1 Interaction

Interactions are formulated as a Grid policy coordination game. In the case of just two different policies in the system, the interaction reduces to a pure coordination game. The extension to model richer scenarios involves using a more elaborated set of policies, resulting in a bigger payoff matrix. Consider for example the payoff matrix from figure 12, with 3 different policies to be coordinated. The payoffs represent the expected performance derived from interactions between nodes combining each pair of policies. These three policies (and the payoffs themselves) can be mapped to three real Grid VO resource sharing policies: you-give-what-you-can (YGWYC) policy, proportional sharing (1/N) policy and you-get-what-you-give (YGWYG) policy [29].

	P1	P2	P3
P1	1, 1	0, 0	-1, -1
P2	0, 0	1, 1	0, 0
P3	-1, -1	0, 0	1, 1

Fig. 12. A payoff matrix for the 3-polices single round pure coordination game. The Nash equilibria for both players is to select matching policies. There is no dilemma in this case. Individual and colective welfare match. However misscoordination leads to very suboptimal results.

6.2 Migration

Periodically (after each node has played a coordination game) they evolve their resource management policies and VO memberships. Nodes join VOs where they find other nodes performing well and copy the policies of those Nodes. Mutation (random migration to a new VO) is applied with a fixed probability m. We do not apply policy mutation because we assume this to be under the control of an administrative authority and is not necessary for the emergence of coordination assuming a population comprising matching policies exists.

6.3 Simulation Algorithm

The model was realized in a simulator, implementing the algorithm in figure 13. First each Grid node is placed randomly into an initial VO. This initial bootstrapping is motivated to better model real scenarios where VOs are already in some specific configuration. Node interaction is constrained within each VO. In the case that a VO only contains a single node then interaction is skipped and the node goes directly to evolution phase. Evolution is realized as explained in the migration section.

6.4 Results

We performed experiments for 100 Grid nodes evolving over 500 rounds. We measured the performance as the average utility of the nodes in the system after round 500. We considered three different mutation rates. Figure 14 shows the results obtained. Notice how even under very high mutation rates (m=0.1), high levels of coordination are still achieved. This indicates robustness of the mechanism in highly dynamic environments. Figure 15 shows two typical individual runs for two different mutation rates.

```
Bootstrap Nodes in VOs (groups identified by a Tag)
LOOP a number of rounds
    LOOP for each VO
        LOOP for each node in the VO (operation phase)
            Interact with another Node from the VO (i.e. has same Tag)
            Collect Payoff
        ENDLOOP
    ENDLOOP
    LOOP each Node in the Grid (evolution phase)
        Select random partner node in the Grid
        IF partner outperforms Node
            Copy partner Tag (migrate to its VO) and policy
            Mutate: node applies probabilistic Tag mutation
        END IF
    ENDLOOP
ENDLOOP
```

Fig. 13. Pseudo-code for the GridWorld simulation model.

	m= 0.001	m= 0.01	m= 0.1
Average Utility	0.97 (0.01)	0.97 (0.00)	0.85 (0.01)

Fig. 14. Performance results for a VO policy coordination game. Mean (and standard deviation) of average utility after 500 rounds over 10 experiment runs for different mutation rates.

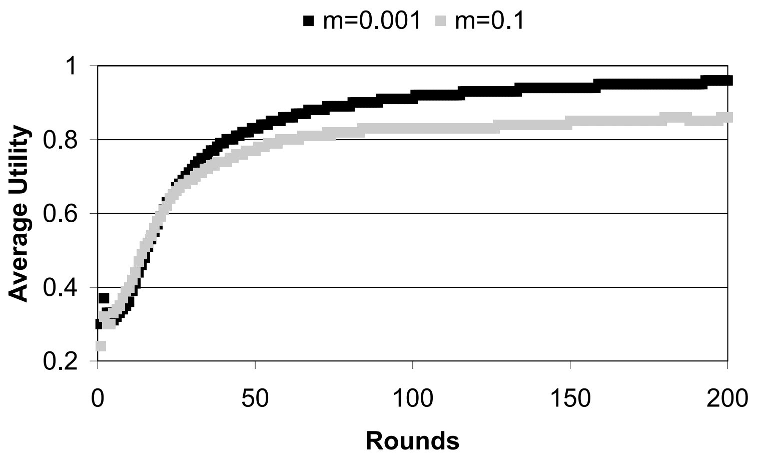


Fig. 15. Results from two runs of the GridWorld model comparing two different mutation rates (m). Notice that over a number of rounds the average grid node utility increases.

7 Selfish broadcasting

A broadcast operation involves one node in a network sending a message to the entire network population. However it is not practical in many situations for every single node to connect directly to all other nodes. A simple way of implementing a broadcast function is for each node to pass all new broadcast messages it receives to all of the other nodes it connects to (its neighbors). This way a message initiated from any node will eventually reach all other connected nodes in the network. This method is called the flood fill (FF) approach and although simple and robust it can be highly costly in terms of the total number of messages required to be passed: FF will send L messages (where L is the total number of links in the network) assuming the network is connected. This can be highly sub-optimal.

Collective Goal	Maximise the number of nodes receiving broadcast messages initiated from randomly selected nodes while minimising the number of messages sent.
Entity	Peer node - a node in a peer-to-peer overlay network. Each node stores a neighbor list (or view) and a forwarding strategy (either PASS or DROP). Periodically nodes may randomly change their strategy or copy it from other nodes, but they do not change their neighbor list.
Group	All nodes reachable via a broadcast from a given node represents the the group membership of that node.
Interaction	Interaction involves initiating broadcast messages. Receiving messages from neighbors. Passing (or not) received messages to neighbours based on the strategy (either PASS or DROP) held by the node.
Utility	Benefit (B) received upon message reception minus cost (C) due to possible message forwarding. $B > C > 0$. Hence nodes have an incentive to receive messages but no incentive to pass them on.
Migration	No direct migration (no rewiring in the network). Groups are defined by nodes receiving or not receiving messages and depends on strategy distribution among nodes. By changing a strategies nodes migrate between broadcast groups.

Fig. 16. Key aspects for the Broadcast model.

The optimal broadcasting possibility in terms of messages sent, is for the message to follow a spanning tree in which each node only passes the message to those neighbors that have not already received the message. This approach requires only N messages (where N is the total number of nodes in the network).

We consider a different approach from the two limit cases above, in which nodes dynamically decide if to pass messages to their neighbors based on a simple adaptive evolutionary protocol in which there are incentives for nodes to not pass on messages.

Using simulations we found that this approach, though motivated by self-interest in the nodes, produced reasonable broadcasting performance with a total message cost significantly less than that required for the FF approach. Interestingly, we found that our simple protocol appears to evolve the network towards a critical threshold of PASS nodes and this *could* be related to site percolation theory.

The results presented here are an overview of work given in a previous paper from which more details can be obtained [1].

7.1 Interaction

We formulated the model in the form of a *broadcast game* in which nodes were awarded a payoff (or utility) based on the outcome of each broadcast event. A broadcast event involves randomly choosing a node from which a broadcast message is initiated. The node passes a message to each of its neighbours. In the next time-step each neighbour then decides if to pass the message to its own neighbors based on its current strategy (either PASS or DROP). This process continues until no more messages can be sent. We assume that nodes wish to receive broadcast messages (they contain useful information) but that nodes have some incentive not to pass messages on (due to a bandwidth or power cost). Hence we assume there is a benefit for receiving a message (B) and some cost for sending a message (C). We assume that $B > C > 0$ meaning a node receiving and passing a message receives positive benefit. After a broadcast event each node in the network will either receive a utility award of B, if it received but did not pass the message or $B - C$ if it received and passed the message or zero if it did not receive the message at all.

7.2 Migration

Migration between groups is not explicitly defined (i.e. there is no network rewiring) since only strategies are copied. The network structure itself does not change through evolution. According to the strategy distribution the network is composed of groups in the sense of connected clusters of nodes holding the PASS strategy. That is, every node i is a member of a group defined by the set of all reachable nodes S from node i via a broadcast. A group boundary is thus formed by a region of the network being bounded by nodes holding the DROP strategy - stopping any message originating inside the group from passing to other groups. Through utility comparison and strategy copying a group selection process is achieved at the level of quantity of PASS nodes in a group. Groups of nodes not receiving messages have zero utility. Hence it will copy any other nodes strategy if it received a message. This means the proportion of PASS nodes in groups receiving messages will tend to be reproduced by strategy copying. Such proportions of PASS nodes will tend toward the critical value at which groups form because by definition a group exists if it is above this critical threshold. On the other hand a node within a group receiving a message has an incentive to switch from a PASS to a DROP strategy - lowering the proportion PASS nodes in such groups. This ultimately will cause the creation of new groups not receiving messages when the PASS value falls below the critical value. Hence we see cyclical oscillations around the critical value.

7.3 Simulation algorithm

We implemented the broadcast model in the PeerSim platform [30]. Figure 17 shows the pseudo-code for the one active and two passive threads that form the protocol. For the simulations discussed here we selected one node randomly to initiate a broadcast message every 20 cycles. Messages were given a TTL value of 5. A cycle was equivalent to the time it took to send a message one hop in the network. Nodes maintained input buffers to store incoming messages. The results given here are for static random networks but similar results were obtained when high levels of node churn (nodes entering and leaving the network) were incorporated. For full details see [1].

```
*Active “replication” thread for node (i):*

Periodically do
    j ← selectRandomNode()
    IF Utility$_j$ > Utility$_i$ THEN
        Strategy ← Strategy
        With probability m change Strategy
        Utility$_i$ ← 0
    END IF

*Passive “initiate broadcast” thread for node (i):*

When requested to broadcast message (msg) do
    LOOP for each neighbour node (j) in view
        Send(j, msg)
    END LOOP

*Passive “receive message” thread for node (i):*

When receive message (msg) from neighbour (k) do
    IF new unseen message THEN
        Increment Utility$_i$ by B (benefit)
        IF (Strategyi == PASS) and (TTL of msg > 0) THEN
            Decrement TTL of msg
            Decrement Utility$_i$ by C (cost)
            LOOP for each neighbour node (j) in view except (k)
                Send(j, msg)
            END LOOP
        END IF
    END IF
```

Fig. 17. Pseudo-code for the Broadcast protocol comprising one active and two passive threads. The topology was initialised to a random network with degree 20. All links between nodes were symmetric.

7.4 Results

We tested the broadcast game on a random network topology of degree 20 where all links were symmetric. The system shows an interesting behaviour. A typical run of the algorithm is shown in Figure 18. In this experiment a single message is generated every 20 simulation cycles and spread starting from a randomly chosen node, while in each cycle each node compares its utility with a random node with 20% probability.

We initialised all nodes with the PASS strategy (NP=1). Hence at the start of the run flood fill is performed and there are a lot of redundant messages. If some node stops passing messages it gets a higher utility and gets copied, so initially a dramatic drop in NP occurs as nodes discover how to get higher utility by not passing the message. Eventually too many nodes will adopt the DROP strategy and the message will fail to be spread throughout the network. At this stage the network is divided into a groups receiving messages and groups not receiving anything. Nodes in non receiving groups will tend to reproduce the NP value of the receiving cluster and will eventually be able

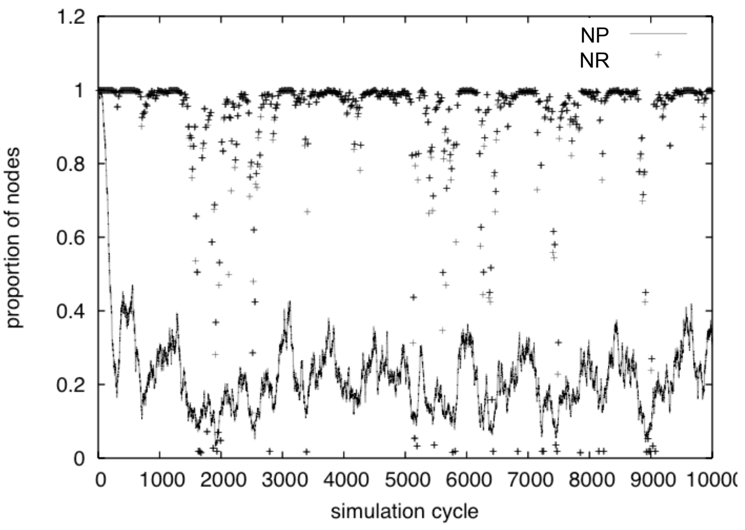


Fig. 18. A single run of broadcast game. The solid line (NP) represents the ratio of nodes with PASS strategy, while each dot (NR) represents a message and the quantity of nodes that received it. Initially NP drops dramatically but it meta-stabilises at a critical threshold below which messages are not spread and above which messages become redundant.

to receive messages again. Hence a group-like selection is created between clusters receiving messages and clusters not receiving anything, and the strategies are distributed oscillating around a critical value above which there are redundant messages and below which messages can not be spread throughout the network.

These dynamics are very complex and subtle, we only hinted at them here. For a fuller discussion see [1].

8 Content replication

Nodes in a network often store and serve content to other nodes. However, each has a finite capacity and if requests for content exceed the capacity then queries fail. It is generally not possible for *a priori* predictions of load demand because at given times some content may suddenly become popular and at others hardly requested at all. Hence over a given time period a population of nodes has a certain total capacity to serve requests (the sum of all individual node capacities) and some demand load (queries going to the nodes). Assuming nodes can replicate content and redirect queries we present a simple node level protocol that self-organises nodes into cooperative clusters leading to efficient outcomes in some simple scenarios. CacheWorld is a simulation model that applies to group selection pattern to this replication sceanrio. The key aspects are shown in figure 19. This section is an overview previous work where further details can be found [11].

8.1 Interaction

We assume a population of *N* server nodes which form a P2P overlay network. In addition to being part of the overlay, each node functions as a server responding to requests (queries) from clients outside of the overlay. An example could be that each

Collective Goal	Maximise the total number of queries served by harnessing unused capacity in underloaded nodes.
Entity	Peer node - a node in a peer-to-peer overlay network with the ability to receive and serve queries, for a content item, from clients external to the overlay network. Each node has a maximum capaciy limiting number of queries serviceable over a time period. Each node can be thought of as a web server, for example, and stores is own content item and a replicated copy of each of its neighbours content items.
Group	The neighbour list (or view) of a node defines its group
Interaction	Receiving redirected queries from overloaded nodes or conversely redirecting queries to a random neighbour when overloaded. When a node makes a connection to a new neighbour both nodes mutually replicated their contents.
Utility	A simple binary satisfaction function: if all queries received by a node are eventually served then the node is satisfied otherwise it is unsatisfied.
Migration	Periodically, unsatisfied nodes move randomly in the network. But a node will only accept an incoming connection from a moving node if it is in a receptive state. A node is only receptive if it has spare capacity or is itself unsatisfied.

Fig. 19. Key aspects for the CacheWorld model.

node is a web server with the overlay linking the servers and clients being web browsers on user machines. Servers store a copy of their own content item (e.g. a website) and have additional storage for k replicated items from other servers. The overlay links servers bidirectionally if they mutually replicate content. In our website example this would mean two linked servers hold a copy of the others' site. We also assume servers have access to three services: a replication service that copies items between servers; a peer sampling service that supplies a random server from the overlay; a content server that serves or redirects queries as required.

Over a given time period nodes receive queries (load) from clients to serve their content item. Each node has a fixed capacity, C, specifying the total queries it can serve in the given time period. If the load exceeds capacity then the nodes is said to be "overloaded". Overloaded nodes redirect queries to randomly selected neighbors. If a neighbor is not itself overloaded it will serve the query from its local content replica (or cache), otherwise it will ignore the query. The essential idea therefore is that overloaded nodes will have neighbors that are not overloaded and can serve queries.

8.2 Migration

Each node maintains an estimate of the proportion of queries for its own content that were actually served (ps). A node is said to be satisfied when $ps \geq t$, where t is some threshold value. Here we set $t = 1$ for all nodes, meaning nodes are only satisfied if 100% of their queries are served. Periodically nodes attempt to change their neighbors (move) in the overlay network if they are not satisfied. When a node i moves, it drops all its current links and selects a random node, j, from the population. i then attempts to link to j. The link is accepted if j is in a *receptive* state. A node is defined as receptive if it is currently not satisfied or if it has spare capacity which is unused. This captures the notion that a node only wants new connections if it is either not satisfied or if it has spare capacity to offer. If a receptive node accepts an incoming link but already has the maximum *k* links then it drops a randomly selected old link to make space for the new link.

8.3 Simulation algorithm

We implemented the CacheWorld model in the PeerSim platform [30]. Figure 20 shows the pseudo-code for the active and passive threads that form the protocol. The passive thread is activated when a node receives a query, the active thread is activated periodically. For the simulations discussed here we selected a simple loading scenario on the nodes in which half the nodes were overloaded and half were underloaded such that an optimal network structure exists in which all queries could be answered. Details of this and other load scenarios can be found in [11].

Active "node movement" thread for node:

```
Periodically do
    IF not satisfied
        drop all neighbour links
        j ← selectRandomNode()
        IF j is receptive then link to j
    END IF
```

Passive "receive query" thread for node:

```
When receiving a query (q) do
    IF not overloaded, service q directly
    ELSE IF neighbours > 0 and q is not already a redirected query
        j ← selectRandomNeighbour()
        redirect q to j
    END IF
```

Fig. 20. Pseudo-code for the CacheWorld protocol. It comprises one active and one passive thread. These were simulated within the PeerSim environment. Notice that only one hop of query redirection is allowed.

8.4 Results

Figure 21 shows results from three different experiments (isolated, random and dynamic) for the simple half overloaded / half underloaded scenario. Here we show the simplest case where number of replicas $k = 1$, meaning that nodes essentially form pairs rather than complex network structures. The *Isolated* experiments gives a performance baseline by running the simulation with all CacheWorld services turned-off: no links between nodes, no query redirection and no movement. This captures the situation where nodes simply answer their own queries and do not know about other nodes. The *Random* experiments give a secondary baseline. Here the overlay is initialized to a random topology (degree *k*) that is fixed - i.e. no node movement is possible. However, nodes my redirect their queries to neighbors if they become overloaded. In the *Dynamic* experiments the CacheWorld protocol is fully enabled allowing for dynamic node movement based on satisfaction, as previously described. By comparing *Random* and *Dynamic* experiments we can determine how much extra performance the dynamic movement produces over just a fixed random overlay. By comparing *Isolated* and *Dynamic* experiments we can determine the overall increase in performance obtained by using CacheWorld over letting servers deal with all their own queries individually.

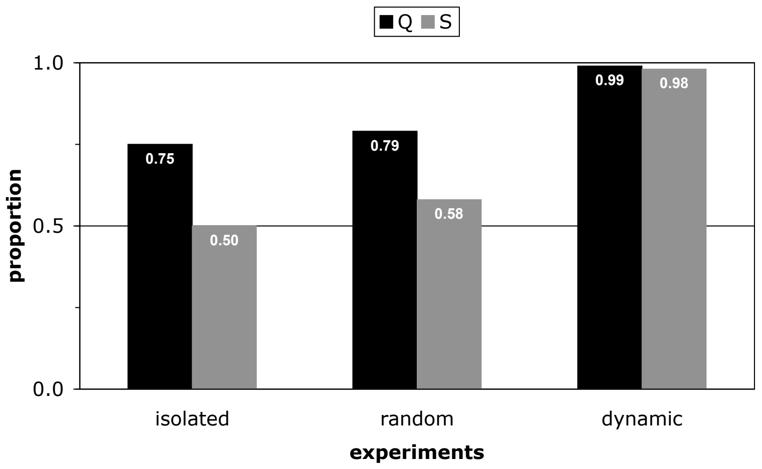


Fig. 21. Results from the Content Replication model. Isolated shows performance when the protocol is tuned off. Random shows the performance when there is no node movement. Dynamic shows results when the protocol is fully enabled. Q = proportion of queries answered, S = proportion of nodes satisfied.

9 Conclusions

We have given an overview of some existing application models that have been inspired by group selection approaches. We identified some key aspects that define such models and expressed them in a standard template. However, there are still open issues before we can present our approach as a design pattern in a principled way. Firstly, all the applications so far discussed have only been tested at the level of simulation in which many potentially significant effects are not modelled. For example, we have not tested the systems against several kinds of malicious attacks. Also there is an assumption in many of the models that nodes will, at some level, supply valid information (for example for utility comparison or link exchanges) and malicious nodes may not do this - we have addressed some of these issues in previous studies [2, 18] but there is more work to do here. We believe that to produce a credible design pattern we need to report on experience with deployed and tested implementations. We feel that the work presented here moves us closer to that possibility but, again, there is work to do.

The group selection approach may be just that: a design *approach* rather than a design *pattern*. Our experience has been that, given a co-ordination problem in a distributed system, the group selection approach is more like a general methodology than a solution. Rather like functional decomposition within structured programming, or inheritance hierarchies in object orientated design, the approach requires one to identify what might be the natural structures of the interactions within the system - i.e. you have to answer the questions: Where are the groups here? What are the strategies or relevant individual behaviour choices? How can migration occur? What is the collective goal?

What are the individual goals? Producing answers to those questions is not easy and often requires experimentation. Yet if crisp definitions can be found for each of these aspects then the group selection process is relatively straightforward to apply.

The key point is that by answering these questions one is forced to think about the runtime distributed *social dynamics* of a system at a level of abstraction much higher than specific implementation issues, static structures or purely individual properties. By doing this one is forced to *think socially* and is freed from the need to design centralised structures to impose the collective goal.

If group selection in distributed information systems can be developed as a general design methodology then perhaps this could have a wide general impact in the emerging are of socially intelligent systems design.

Acknowledgements

This work would not have been possible without many informative and passionate discussions with many people, particularly those from the Bologna group, including: Ozalp Babaoglu, Mark Jelasity, Fabio Picconi and Simon Patarin. We would also like to thank Steve Richardson from Manchester University for many discussions and pointers concerning the general design pattern concept - although he may look on with horror at our use of the term for the group selection approach we present here - hence our discussion about approaches rather than patterns in the conclusion.

References

1. Arteconi, S., Hales, D. (2006) Broadcasting at the Critcial Threshold. October 2006, Technical Report UBLCS-2006-22, University of Bologna, Dept. of Computer Science.
2. Arteconi, S.; Hales, D. (2007) Greedy Cheating Liars and the Fools Who Believe Them. In Engineering Self-Organising Systems, Proceedings of the 4th International Workshop, ESOA 2006, Hakodate, Japan, May 9, 2006, Revised and Invited Papers. Brueckner, S.; Hassas, S.; Jelasity, M.; Yamins, D. (Eds.), Lecture Notes in AI , Vol. 4335, Springer.
3. Axelrod, R. (1984). *The evolution of cooperation.* Basic Books, NY.
4. Babaoglu, O., Canright, G., Deutsch, A., Di Caro, G., Ducatelle, F., Gambardella, L., Ganguly, N., Jelasity, M., Montemanni, R., Montresor, A., and Urnes, T. (2006) Design Patterns from Biology for Distributed Computing. In *ACM Transactions on Autonomous and Adaptive Systems, vol. 1, no. 1, 26–66.*
5. Chao, I., Ardaiz, O., Sangesa, R. (2007) A Group Selection Pattern for Agent-Based Virtual Organizations Coordination in Grids. OTM Workshops (1) 2007: 136-148
6. Chao, I., Ardaiz, O., Sangesa, R. (2007) A Group Selection Pattern Optimizing Job Scheduling in Decentralized Grid Markets. OTM Workshops (1) 2007: 37-39
7. De Wolf, T. and Holvoet, T (2007) Design Patterns for Decentralised Coordination in Self-Organising Emergent Systems. In Engineering Self-Organising Systems, Proceedings of the 4th International Workshop, ESOA 2006, Hakodate, Japan, May 9, 2006, Revised and Invited Papers. Brueckner, S.; Hassas, S.; Jelasity, M.; Yamins, D. (Eds.), Lecture Notes in AI , Vol. 4335, Springer.
8. Edmonds, B. and Hales, D. (2003) Replication, Replication and Replication - Some Hard Lessons from Model Alignment. Special Issue on Model-2-Model Comparison, *Journal of Artificial Societies and Social Simulation* vol. 6, no. 4.
9. Foster, I. (2005) Globus Toolkit Version 4: Software for Service-Oriented Systems. In *IFIP International Conference on Network and Parallel Computing.* Springer Verlag LNCS 3779, pp 2-13.
10. Foster, I., Kesselman, C., Tuecke, S. (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In *International J. Supercomputer Applications, 15(3).* 2001
11. Hales, D., Marcozzi, A., Cortese, G. (2007) Towards Cooperative, Self-Organised Replica Management. Proceedings of the First International Conference on Self-Adaptive and Self-Organizining Systems (SASO2007), July 2007, Boston, MIT. IEEE Press.
12. Hales, D. (2006) Emergent Group-Level Selection in a Peer-to-Peer Network. *Complexus 2006;3.: 108-118.*

13. Hales, D. and Arteconi, S. (2006) SLACER: A Self-Organizing Protocol for Coordination in P2P Networks. *IEEE Intelligent Systems* 21(2):29-35.
14. Hales, D. and Edmonds, B. (2005) Applying a socially-inspired technique (tags) to improve cooperation in P2P Networks. *IEEE Transactions in Systems, Man and Cybernetics - Part A: Systems and Humans, 35(3):385-395.*
15. Hales, D. (2000) Cooperation without Space or Memory: Tags, Groups and the Prisoner's Dilemma. In *Moss and Davidsson (eds.) Multi-Agent-Based Simulation.* LNAI 1979:157-166. Springer. Berlin.
16. Hardin, G. (1968) The tragedy of the commons. *Science*, 162, 1243-1248.
17. Holland, J. The Effect of Lables (Tags) on Social Interactions. *Santa Fe Institute Working Paper 93-10-064*. Santa Fe, NM 1993
18. Jesi, G. P, Hales, D., van Steen, M. (2007) Identifying Malicious Peers Before Its Too Late: A Decentralized Secure Peer Sampling Service. Proceedings of the First International Conference on Self-Adaptive and Self-Organizining Systems (SASO2007), July 2007, Boston, MIT. IEEE Press.
19. McDonald, A. and Sen, S. (2005) The Success and Failure of Tag Mediated Evolution of Cooperation. In *Woking Notes of the AAMAS-05 Workshop on Learning and Adaptation in MAS.* LAMAS 155-1643779, pp 2-13
20. Nash, J. F. (1950) Equilibrium Points in N-Person Games, *Proc. Natl. Acad. Sci.* USA 36, 48-49.
21. Ohtsuki, H. et al. (2006). A simple rule for the evolution of cooperation on graphs and social networks. *Nature* 441(25):502-505.
22. Riolo, R. (1997) The Effects of Tag-Mediated Selection of Partners in Evolving Populations Playing the Iterated Prisoner's Dilemma. *Santa Fe Institute Working Paper 97-02-016. Santa Fe, NM*.
23. Riolo, R., Cohen, and M.D., Axelrod, R. Cooperation without Reciprocity. *Nature* 414, 441-443, (2001).
24. Roberts G, and Sherratt N. T. (2002) Similarity does not breed cooperation. *Nature* 418, 449-500.
25. Santos F. C., Pacheco J. M., Lenaerts T. (2006) Cooperation prevails when individuals adjust their social ties. *PLoS Comput Biol 2(10): e140. DOI: 10.1371/journal.pcbi.0020140*
26. Smith, J. M. (1982) Evolution and the Theory of Games.
27. Sun, Q. and Garcia-Molina, H. (2004) SLIC: A Selfish Link-based Incentive Mechanism for Unstructured Peer-to-Peer Networks. In Proc. of 24th IEEE Int. Conf. on Distributed Systems, IEEE Computer Society.
28. Traulsen, A.; Nowak, M. A. (2006). Evolution of cooperation by multilevel selection. *Proceedings of the National Academy of Sciences* 130(29):10952-10955.
29. Wasson, G., and Humphrey, M. (2003) Toward Explicit Policy Management for Virtual Organizations. In *4th International IEEE Workshop on Policies for Distributed Systems and Networks (POLICY 03), pp. 173-182.*
30. Peersim Peer-to-Peer Simulator, Available at: http://peersim.sf.net