ON AGENT MODELLING AND SYSTEM SIMULATION OF SELF-ORGANISATION

M.C. Schut
Department of Computer Science
VU University, Amsterdam (NL)
schut@cs.vu.nl

N. Ferreira-Schut
Department of Computer Science
Radboud University, Nijmegen (NL)
nivea@cs.ru.nl

KEYWORDS

design, implementation and validation methodologies; multi-level simulation; self-organisation; multi-agent systems; system design; formal specification; empirical analysis.

ABSTRACT

Complex, natural, social, technological and economic systems have recently given rise to the need of a new paradigm for computational systems that are adaptive, can self-organise and exhibit emergent behaviour. The design of such systems concerns a homogeneous set of agents in which each agent receives an input and has to map it to a 'good' output, and where self-organisation emerges from the interaction between agents. Although general and simple, this concept is representative for a very wide spectrum of applications such as protocol design for large computer networks, design of collective robotics, and automative traffic engineering. Surprisingly, only a handful of recent research is aimed at a domain-independent (or: general) design of such systems. We propose as the solution for the design-problem a framework that tackles the local (agent) level *formally* and the global (system) level *empirically*. This allows us to do rigorous formal verification of the behaviour of the individual agents, as well as large-scale empirical validation of the system as a whole. Besides, it exploits the specific advantages of the approaches regarding the scale of the system: formalisation is good for small systems, while simulation works well for (very) large systems. The objective of this paper is to further develop and exploit the idea of combining a formal approach on the agent level and an empirical approach on the system level in self-organisation.

INTRODUCTION

The self-organising 'paradigm' is the response to the increasing need for systems that are adaptive, flexible and pliable towards uncertain operations of the units that build it up and the interactions between these units. These new computational systems (also known as pervasive technology or emergence engineering) consist of many (in order of thousands) units exhibiting complex behaviours on different aggregation levels. The aggregation levels vary from local (micro) behaviour on the unit level to global (macro) behaviour on the system level. Real-world examples of such systems are abundant: traffic networks, energy distribution networks, markets, computer networks, animal and human societies.

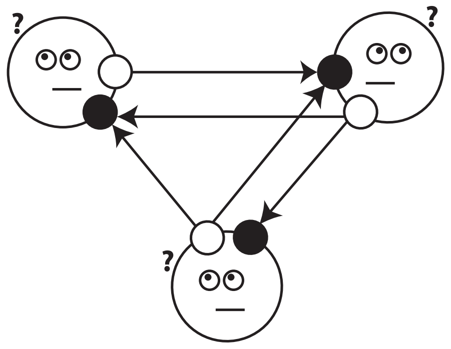

Figure 1: The design problem, with 3 agents who have to find a good mapping between their inputs (black circles) and outputs (white circles).

The problem addressed in this paper concerns the *design* of self-organising systems. Consider Figure 1 as an illustration of the self-organising system design problem: a homogeneous set of agents in which each agent receives an input (i.e., observes) and generates output (i.e., acts), and faces the problem of mapping its received input to a 'good' output. Although it is extremely general and simple, this illustration captures a very wide range of applications, including, for example, protocol design for very large computer networks, design of collective robotics and inter-vehicular communication for automative traffic engineering.

Throughout the paper, we define *self-organisation* in a strong way: a system is self-organising if, and only if, there are no organisational roles allocated to the system's parts (i.e., agents, individuals, elements). This definition excludes systems in which (designed or emerged) organisational roles exist, for example, managers, brokers, superiors or subordinates (which would be weakly self-organising). Different individuals can still be *specialised* in a functional way (related to the system's task), but not in an organisational way. The main argument for this approach is that we want to know the true power of self-organisation by leaving out organisational structures and to keep the agents as simple as possible – making it possible for the intelligence to emerge at the system level instead of at the agent level.

Proceedings 23rd European Conference on Modelling and Simulation ©ECMS Javier Otamendi, Andrzej Bargiela, José Luis Montes, Luis Miguel Doncel Pedrera (Editors)
ISBN: 978-0-9553018-8-9 / ISBN: 978-0-9553018-9-6 (CD)

The current state-of-the-art regarding the design problem is that there are in general 3 widely-used approaches: 1) simulation – construction of a simulation model followed by statistical analysis, 2) formal – building a formal model of the problem/solution and checking whether desired properties do hold or not; and 3) nature-inspired – taking observed methods and techniques from nature as the starting point (e.g., ant foraging, honeybee dancing, see (Camazine et al. 2001)) and 'finding' problems that can be solved with these methods and techniques.

We consider none of the above described approaches absolutely ideal to be used as a 'generic' design framework. For simulation, the major drawback is that the modeller has no means to verify the *local* workings of the system – the observed and analysed results are on the system level and not on the agent (individual) level. With a formal approach, it is only possible to look at 'small' systems because of the computational complexity of these approaches. A weakness of the nature-inspired approach is that it is not a 'problem-centered' one (instead, it is a reverse engineering approach, see (Zambonelli et al. 2005)). Also, all approaches are developed ad-hoc within the context of a given problem or domain. To date, there has been no successful attempt to develop a generic design framework with the explicit aim to be useful for a wide range of application problems.

The research objective of this paper is to further develop and exploit the idea of combining a formal approach on a local level and an empirical approach on a global level in self-organisation.

Although this paper is (only) a very first step towards the long-term aim of delivering a generic framework, we can clearly identify the originality of the proposed research: 1) it deals with design limitations of nature-inspired (reverse-engineering) approaches (e.g., swarm intelligence, see (Bonabeau et al. 1999)), 2) it does not exclusively focus on either formal or simulation approaches, and 3) it introduces self-organising systems as an interesting new system class for existing modelling techniques (e.g., Markov decision processes, as described below).

RELATED WORK

Design of Self-Organisation Although there is a general trend and steep increase of interest in the design of self-organising systems, there is only a handful of related research specifically aimed at the *domain-independent* (or: general) design of self-organisation. We briefly enumerate this work here in order to sketch a context for our paper. Note that we did not include here: work that looks at the "emergence problem" (the relation between the local and global levels in a self-organising complex system) (de Wolf and Holvoet 2005), because we consider it best to investigate this after successful application of the proposed framework, enabling us to systematically examine this problem; work where individual agents in a self-organising system have or develop organisational roles, because these systems do not fit the strong notion of self-organisation used in this paper; and work that is strongly theoretical, e.g., (Yamins 2007).

We briefly sketch the history of design approaches in the last decade. In 2003, (Serugendo 2003) sketches the idea of engineering emergent behaviour and in subsequent work identifies comparison criteria for self-organising systems, categorises mechanisms leading to self-organisation, and proposes an approach for selecting self-organising mechanisms. In 2004, (Edmonds and Bryson 2004) use the experimental method as the starting point of an adaptive strategy for producing self-organised software. In 2005, (Bar-Yam 2005) proposes multiscale analysis as a way to engineer complex systems; (Wolf and Holvoet 2007) categorise design patterns for decentralised coordination in self-organising systems. (Gershenson 2007) proposes a design methodology based on the idea that *reducing the 'friction' of interactions between elements of a system will result in a higher 'satisfaction' of the system.* In 2007, (Sudeikat and Renz 2007) suggest to use the approach of requirements engineering to develop self-organising multi-agent systems. And last year, (Martinoli 2008) proposes the use of multi-level modelling to design swarm (robotic) systems; also, (Gardelli et al. 2008) propose a simulation-driven approach to design self-organising environments.

The most important difference between the research in this literature overview and the work suggested in this paper is exactly the novelty of our work. Although, as shown above, some references also make the formal/empirical distinction, nowhere is it combined with the individual/system level. For instance, a state of the designed system is represented on the global level (all individual nodes are within one state), while we propose to do this only on the local level (i.e., individual-centered) and clone the agent to lift it to the global level (and then investigate the system empirically).

Multi-Agent Systems We understand that self-organising systems demonstrate complex behaviour even if their individuals are decribed in terms of simple strategies and knowledge – though some aspects of more common 'intelligent' multi-agent systems, see (Wooldridge 2002), such as autonomy, decentralisation and local view are still present. In terms of multi-agent systems, our research restricts its interests by considering a self-organising system as a collection of interactive individuals, which are uncomplicated, basic, light-weight in their specification. Opposite to this, being *intelligent* means that the agent is capable to reason about the world given the information it obtains and following its behaviour description. Our interest lies in stochastic systems in which the options of actions that such a *reasoner* can take follows a probabilistic distribution. Additionally, reward functions are used to allow the specification of strategic mechanisms, i.e., based on these functions, an agent will endeavour on establishing the best set of actions that leads to its goal(s).

Markov Decision Processes Our first attempt at a formalisation of the local level will be by means of a Markov decision process, hence a brief introduction. In general, a Markov chain, according to (Baier and Katoen 2008), is a directed graph where nodes represent (system) states and relations among those states are captured by probabilistic transitions. Markov decision processes (MDPs), see (Bellman 1957), extend Markov chain by the explicit represention of actions which generate the transitions among states. One can use an MDP in order to model problems in which the decision making process in uncertain environments is automated. By uncertain environments we mean that the outcomes of actions are not entirely predictable. By automation we mean that we can use algorithms in order to 'solve' an MDP (i.e., derive a policy which maps states to best actions).

Well-known exensions of the 'basic' MDP model that are relevant to this paper are partially observable MPDs (or: POMDPs, see (Kaelbling et al. 1998)) and multi-agent MDPs (or: MMDPs). Regarding the latter, (Jamroga 2008) presents a formal language, MTL, which allows the representation of general scenarios in which a system is formed by multiple agents (or: processes) which interact to one another. In this case, sets of agents can establish joint strategies in order to achieve (common) objective(s). A model for the multi-agent MTL is a refinement of MMDPs, see (Boutilier 1999). The main difference with the work by Jamroga is that Boutilier focus on fully cooperative multi-agent systems, while the former allows the specification of sets of agents that might not share the same objectives – accounting, therefore, for more general scenarios. Still, in both, agents are assumed to know the global state of the system at all times. Another approach also related to Boutiliers and to multi-agent MDPs is the framework proposed by (Xuan et al. 2000). This approach also concentrates on the description of fully cooperative systems. But here the decentralised aspect of multi-agents systems is emphasized. Instead of assuming the omniscience of agents, decentralised control of finite state Markov processes is used.

For us it is clear that formalisations of MDPs such as presented by (Jamroga 2008) is highly relevant. First, we can represent agents using a MDP model and specify properties regarding the behaviour of those agents. Then, we are able to verify whether those desired, or required, properties in fact hold.

In the next section we present our framework, establishing in more detail the role of specification and verification in our self-organising design.

SOLUTION FRAMEWORK

The solution that we propose is a design framework (shown in Figure 2) based on the before-mentioned simple idea to approach the agent level formally and the system level by means of simulation. This synthetic approach allows us to do rigorous formal verification of the behaviour of the individual agents as well as large-scale empirical validation of the system as a whole. It also exploits the specific advantages of the approaches regarding the scale of the system: formalisation is good for small systems while simulation works well for (very) large systems. Thus, differently from other approaches, we do not attempt to either fully formally *specify* the system behaviour, nor do we blindly rely on observed system behaviour while not knowing what happens at the local level.

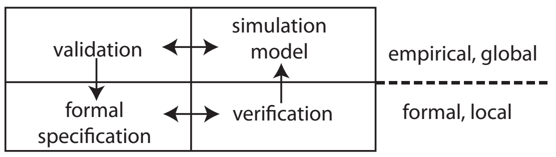

Figure 2: The proposed solution framework.

The reason that this approach potentially works well for self-organising system design (but possibly not for system design in general) has to do with two important aspects of a self-organisation itself. Firstly, although such systems are massive (i.e., involve many agents), they are formed by *homogeneous* set of agents (e.g., bird flocks and fish schools). This property makes the modelling and designing of these systems somewhat easier than 'complicated' systems – especially concerning the representation of the agents and, subsequently, scaling up the system (almost by simple 'cloning'). Secondly, the *interaction* among the agents occurs 'only' with their direct neighbours. Thus there is no need (not even theoretically) to consider the possibility that two distant agents interact with each other. Also, the nature of the interaction differs from cooperation in the traditional sense – agents rather *react* to each other (like birds in flock on the basis of separation, alignment and cohesion).

Our framework consists of 4 subsequent development phases, i.e., formal specification, verification, simulation model and validation. The arrows in the Figure 2 indicate how the phases relate to one another. Note that a special case of such general description is having a system that goes successfully from specification to validation, never returning to a previous phase.

Formal specification As previously mentioned, we understand an agent to be an autonomous entity capable of observing (or *partially modelling*) the world (i.e., the domain in which it is situated) and acting in accordance with those observations. In order to obtain an accurate agent description, a diversity of languages adequate for an agent specification have been produced. In this context, logical languagues are characterised by their unambiguous semantics, and have been applied as useful formalisms for agents' description, allowing also agents' implementation and verification, see (Fisher et al. 2006; Barringer et al. 1996). In other words, logical formalisation of agent behaviour is desirable, not only in order to provide a clear semantics of agent-based systems, but also to provide the foundation for sophisticated reasoning techniques. For these reasons, we opt for applying a formal language for the representation of agents in this step of the design of self-organising systems.

Verification Well-defined syntax and unambiguous semantics of a formal language allow an accurate agent description to be produced, while also potentially allowing formal verification tools to be applied. Logical verification can (and should) be used in order to guarantee that malicious behaviour of individuals are avoided, something that is of crucial importance in complex and critical applications. In this phase of the system's design, we aim to verify that certain (desired) properties regarding individual behaviour are guaranteed, before proceeding to the system simulation where the self-organisation aspect (and, consequently, broader scope) comes into play. A method for formal verification of (agent-based) systems is *model-checking*, see (Baier and Katoen 2008). Essentially, this method performs an exhaustive search through a finite-state specification aiming to obtain a way in which the required property might be contradicted. In model-checking an agent is given in terms of local states in some structure.

Simulation model The simulation phase is the first one on the global level of the system design. It is through the simulation process that a system's emergent behaviour can be produced. That is, in our simulation phase a model is obtained in which a large number of individuals is considered, and as a result of their interaction the self-organising aspect of the system can be generated. For example, consider an artificial society in which the task of individual agents is solely to survive on the basis of collected resources that exist in the environment. The model that you use for this simulation can be a simple rule-set to be executed by an agent: each agent is a rule-based system. If the agents are able to learn, an adaptive algorithm is added to the simulation in order to make such learning possible. The simulation then adds the control loop to the whole system (model + algorithm) and allows you to switch on the model and observe its development over time. In short, it is through simulation that the real scale of the system can be achieved. It is only in such dimensionality that the emergent intelligent behaviour of the society can finally be accomplished and analysed.

Validation Validation of a self-organising system refers to the conclusions that can be drawn for the system behaviour, observed during the simulation phase. In this context, some statistical methods, such as two-sample t-test and analysis of variance, allow the interpretation and evaluation of *emergent* behaviour. In this sense, emergent means that "the whole is more than the sum of the parts" according to (Damper 2000). In other words, emergents – which are novel with respect to the individual parts of the system – arise dynamically from the interaction among those parts, following (Wolf and Holvoet 2007). Other criteria (or:'a properties) can be checked (besides general properties of self-organisation, see (Holland 1995)): *robustness* – capability of the system to remain operational in the face of (unpredicted) variations of the environment; *adaptivity* – which refers to the quality of fitting to given changes in the environment; in our context, an individual might change itself or the system as a whole might change; *redundancy* – when knowledge or information is kept in different locations of the system; and, *success* on the completion of required application task(s).

SPECIFICATION WITH MDP

We investigate whether the MDP model is an appropriate one for the formal specification in our proposed solution framework. Here, we define an MDP following (Jamroga 2008),

Definition A Markov decision process over domain $D = \langle U, \top, \bot, ^{-} \rangle$ and a set of utility symbols Π is a tuple $M = \langle S, Act, Pr, \pi \rangle$, where

- S is a finite and non-empty set of states;
- *Act* is a non-empty set of actions;
- $Pr : S \times Act \times S \to [0, 1]$ is a stochastic transition relation and $Pr(s_1, \alpha, s_2)$ defines a probability with which the system will go to state s_2, if it is in state s_1 and the agent performs action α. Being a probability distribution, for each state $s_1 \in S$, $\sum_{s_2 \in S} Pr(s_1, \alpha, s_2) = 1$, or $Pr(s_1, \alpha, s_2) = 0$ for all s_2, in case α is not enabled in s_1.
- $\pi : \Pi \times S \to \hat{U}$ is a valuation of a utility *fluent*.

The definition includes that a utility and/or propositional symbol (referred to as fluents) can be assigned to each state of the Markov decision process model. The domain consists of a set of utility values $U \in \mathbb{R}$, $\top$ and $\bot$ representing the logical truth and falsity, and $^{-}$ the complement function. Besides, $\hat{U} = U \cup \{\top, \bot\}$. In this context, a policy can then be defined as a sequence of steps which specify the set of (future) actions to be taken by the agent. In such context, policies can also be stochastic. In short, Markov decision processes define stochastic systems which involve strategic reasoning, decision making and reward/utility functions.

In the next section we present an example of self-organising system in which all phases of our framework is described, and MDP is the underlying model for an agent representation.

EXAMPLE

We consider here a short example to illustrate the ideas presented above. The example concerns a simple game. Assume that we have a $x \times y$ checkers board with in total i black and j white pucks. The pucks are scattered randomly over the board. On the board are also k robots situated that have to collect the pucks. Each agent also has a colour, black or white, and can only pick up pucks of the same colour as itself. An agent can observe pucks and other agents on the m surrounding cells; move horizontally (East, West) and vertically (North, South); it can pick up pucks, and also change colour. The aim of the game is for the group of agents as a whole to converge on a division of labour where the ratio for black/white robots is equal to the ratio of black/white pucks. Here, we are interested in the *with-replacement* version of the game: when a puck has been picked up by an agent, it is put back on the board at another random empty cell. (In the *without-replacement* version, an alternative aim of the game is to pick up all pucks as fast as possible.) We have not implemented the example in a computer model, but here we do give a preliminary outline of the different steps in the introduced framework.

Formal specification

Based on the general MDP definition given above, we formalise the agent states and actions as well as the reward function from our example. We let the state of an agent be a vector containing 1) its own colour, 2) the colour of the puck that the agent currently stands on (has value 0 is the agent does not stand on a puck), 3) the number of black agents, 4) the number of white agents, 5) the number of black pucks, and 6) the number of white pucks in the agent's neighbourhood. The agent receives a reward $\mathtt{r}$ when it picks up the puck that it currently stands on, it 'pays' a penalty $\mathtt{c}$ when it changes colour. Besides, the absolute value of $\mathtt{r}$ is much higher than the value of $\mathtt{c}$.

$Agents = \{1, \ldots, k\}, Colour = \{b, w\}$;
$St_i = \langle col_i, puck_i, \#ag_b, \#ag_w, \#pu_b, \#pu_w \rangle$,
where $i \in Agents$, $col_i \in Colour$,
$puck_i \in Colour \cup \{0\}$,
$ag_{colour} =$ [colour] agent,
$\#ag_{colour} =$ number of agents in neighbourhood,
$pu_{colour} =$ [colour] puck, and
$\#pu_{colour} =$ number of pucks in neighbourhood;
$Act = \{N, S, E, W, PickUp, ChangeColour\}$;
$Reward : St \times Act \rightarrow \mathbb{R}$,
$Reward(\langle b, \_, \_, \_, \_, b \rangle, PickUp) = \mathtt{r}$,
$Reward(\langle w, \_, \_, \_, \_, w \rangle, PickUp) = \mathtt{r}$,
$Reward(\langle \_, \_, \_, \_, \_, \_ \rangle, ChangeColour) = \mathtt{c}$,
where $\_$ is a wildcard, $\mathtt{r}$ and $\mathtt{c}$ are constants,
$\mathtt{r} > 0$, $\mathtt{c} < 0$, and $\mid r \mid \gg \mid c \mid$.

In the example, the transition function represents moving from one neighbourhood state to another. In the formalisation above, we opted for a very concise description of the state, making the role of the transition function somewhat more important. More detailed state descriptions can also be considered that could include information about the distance to the other robots/pucks or even the exact location relative to the agent's own location.

With respect to a suitable action policy, i.e., a sequence of actions to the taken at each state in order to maximise an agent's reward, either an automated or hadmade policy can be established. This means that either MDP solution algorithms (e.g., value/policy iteration) can be applied in order to obtain a system's solution, or a 'hard-coded' solution can be described.

Verification

Given the nature of self-organising systems, we do not have a notion of global states (i.e., representing the state of all the agents). Therefore, the described properties do not refer to group objectives or team performance which should be optimised. Instead, we define and verify properties that only refer to an agent's individual behaviour or aim. For instance, we may be interested in verifying whether it holds that "an agent *eventually* collects a puck". Moreover, and perhaps more interesting, is a property which establishes that "an agent picks up as many pucks as possible with the least possible colour changes".

Although we are aware that properties regarding the behaviour of small groups of agents can also be verified, due to the scale of self-organising systems (as mentioned above), only through simulation the full behaviour of the system can be observed and further analysed. An example of such a (small group) property is whether it is guaranteed that "the ratio of the number of black/white pucks is the same as the ratio of the black/white agents".

In order to specify and verify such system properties, a suitable logical language is needed. The language proposed by (Jamroga 2008) could be considered as a potential formalism for this purpose. However, it still remains to be evaluated in detail how its description fits to the our concept of multi-agency in self-organising systems. For instance, we might require a more generic description of a system - more closely related to the concepts of POMDP. This should allow us to avoid the usual requirement of knowing the global state of the system.

Simulation model

The purpose of the simulation model is to obtain information about the global level of the system. In other words, for our example, we are interested to see if all agents together can achieve convergence towards equal black/white robot and puck ratios. As mentioned above, the generic approach to constructing the simulation model is to take the individual agent model as specified formally and 'clone' it to scale it up to reach a typical self organisational system size. We expect the formal specifications to be as such that the step towards implementation (and produce software) is as small as possible. Besides this 'cloning' process, we need to take some design decisions on the *environment* in which the system is situated. In the example, we need to define the size ($x \times y$) of the board. With the simulation model being complete, we need to *design* and *set up* the experimental parts of the simulation. Thus, we have to define *independent* and *dependent* variables as well as *contingency* (exogenous) variables. Typically, in these kinds of simulation experiments, there are many of such parameters and we must carefully craft the experimental design to restrict ourselves in terms of what we want to investigate. The objective(s) of the study should work as restrictors here. In this puck-collection study, we can vary, for example, the number of robots, number of pucks, or size of the board. Then we can measure the effect of these variables on, for example, the black/white robot and puck ratios, or the total number of collected pucks. Also, if we have specified a number of different control policies for the agents, we can test the performance of those policies in comparison to each other.

Validation

The properties to be *validated* on the global level have the form of statistical hypotheses: statements on correlations between independent and dependent variables in the experimental design/setup. In our example, the most important hypothesis concerns two dependent variables: the ratio of the number of black/white pucks, and the ratio of the black/white agents. It depends on our particular interest how we are going to investigate this (regarding a suitable hypothesis), e.g., do we want to show this in general, only for particular board sizes, only for particular numbers of robots or pucks?

In general, whereas the logical verification of the individual level concerns checking *correctness*, i.e., “does my agent do what I expect/want it to do?”, the properties on the system level generally concern *performance* criteria, i.e., “is the collection of agents effective with respect to some shared goal or objective?”. Besides such (domain-dependent) performance criteria, we already mentioned the apparent useful (generic) properties of self-organisation: robustness, adaptivity and redundancy. Testing these in terms of statistical hypotheses means that these properties must, first of all, be *quantified* and then included in the (in)dependent variables. Robustness and redundancy are related to size of the system (number of agents) and the flow of information between agents; adaptivity can be 1) hardwired into the agents themselves (and is thus not testable), 2) measured on the local level (an individual agent adapts to changing circumstances), or 3) measured on the global level (individual behaviour is not adaptive, but the system as a whole is, i.e., like a flock of birds.

Finally, note that we do not aim to ‘solve’ or explicitly acquire further knowledge on the before-mentioned “emergence problem”. In our experience (and also observed by others), attempts to investigate this problem either exclusively formal or empirical are prone to failure because of the system size (formal approaches do not scale) or individual behaviour/interaction (simulation loses touch with the local level). As mentioned, we are mainly interested in the resultant outcome of the system, without necessarily finding out more about ‘emergence’ as a phenomenon.

CONCLUSIONS

We have seen that nature can bring us useful methods and techniques, but it seems that the range of applicability is limited: we know of a small number of problems that can be solved – travelling salesman problem, dynamic routing, collective robotics – but we may have hit the boundaries. It is time to change our frame of mind: instead of the *solution-centered* way of approaching self-organising system design (i.e., we have a solution provided by nature and find a problem that we can solve with it), we need to move to a *problem-centered* approach in which there is a problem to be solved and we can use a (generic) design framework to pour the problem in and systematically investigate potential solutions.

In terms of developing such a framework, we need to 1) *conceptualise* the problem class (to know what we are talking about), and to 2) *formalise* the problem/solution space (to see if solutions can be found) – eventually leading up to an *operational* framework containing design principles and/or solution algorithms. However ambitious the ultimate goal is to create such a framework, it would meet the challenge set to the community: *to develop applications that “work for themselves” – define a global goal, design components and local functionality and check that the desired result will emerge during execution*, as coined by (Serugendo 2003).

This paper is a small step in between steps 1 and 2 in the above described timeline. We have started with the conceptualisation in earlier work (outlined in (Schut 2007)) and are now exploring ways to put the conceptualisation on a more formal footing. As such, we do not aim to deliver a framework that will generate a solution – i.e., solve the mapping problem for the individual agents in the above-mentioned design problem (Figure 1). Instead, what we aim to achieve is a framework in which we can 1) represent a design problem, and 2) systematically examine potential solutions by effectively integrating formalisation and simulation. A possible next step after successfully examining potential solutions systematically, further automation applied to the framework could potentially involve generating solutions.

REFERENCES

Baier, C. and Katoen, J. P. (2008). *Principles of Model Checking*. MIT Press.

Bar-Yam, Y. (2005). About engineering complex systems: Multiscale analysis and evolutionary engineering. In *Engineering Self-Organising Systems*, volume 3464 of *Lecture Notes in Computer Science*, pages 16–31. Springer.

Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M., editors (1996). *The Imperative Future: Principles of Executable Temporal Logic*. Research Studies Press. ISBN: 0-86380-190-0.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press. Dover paperback edition (2003).

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence : From Natural to Artificial Systems*. Oxford University Press.

Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. In Dean, T., editor, *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 478–485. Morgan Kaufmann.

Camazine, S., Deneubourg, J., Franks, N., Sneyd, J., Bonabeau, E., and Theraulaz, G. (2001). *Self-Organization in Biological Systems*. Princeton University Press.

Damper, R. (2000). Emergence and levels of abstraction. *International Journal of systems Science*, 31(7):811–818.

de Wolf, T. and Holvoet, T. (2005). Emergence versus self-organisation: Different concepts but promising when combined. In Brueckner, S., Serugendo, G. D. M., Karageorgos, A., and Nagpal, R., editors, *Proceedings of the workshop on Engineerings Self Organising Applications*, volume 3464 of *Lecture Notes in Computer Science*, pages 1–15. Springer.

Edmonds, B. and Bryson, J. (2004). The insufficiency of formal design methods - the necessity of an experimental approach for the understanding and control of complex mas. In Jennings, N., Sierra, C., Sonenberg, L., and Tambe, M., editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2004)*, pages 938–945. ACM Press.

Fisher, M., van der Hoek, W., B.Konev, and Lisitsa, A., editors (2006). *Logics in Artificial Intelligence*, volume 4160 of *Lecture Notes in Computer Science*. Springer-Verlag. ISBN 0302-9743.

Gardelli, L., Viroli, M., Casadei, M., and Omicini, A. (2008). Designing self-organising environments with agents and artefacts: A simulation-driven approach. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 2(2):171–195. Special Issue on Multi-Agent Systems and Simulation.

Gershenson, C. (2007). *Design and Control of Self-organizing Systems*. PhD thesis, Vrije Universiteit Brussel.

Holland, J. (1995). *Hidden Order: how adaptation builds complexity*. Perseus Books.

Jamroga, W. (2008). A temporal logic for multi-agent MDP's. In *AAMAS 2008 Workshop on Formal Models and Methods for Multi-Robot Systems*, pages 29–34.

Kaelbling, L., Littman, M., and Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence Journal*, 101:99–134.

Martinoli, A. (2008). A multi-level modeling methodology for swarm robotic systems. In *AAMAS 2008 Worshop on Formal Models and Methods for Multi-Robot Systems*. Invited talk.

Schut, M. (2007). Science of simulation of collective intelligence. Available at http://sci.collectivae.net/.

Serugendo, G. D. M. (2003). Engineering emergent behaviour: A vision. In Hales, D., Edmonds, B., Norling, E., and Rouchier, J., editors, *Proceedings of the 4th International Workshop Multi-Agent-Based Simulation*, number 2927 in Lecture Notes in Artificial Intelligence. Springer.

Sudeikat, J. and Renz, W. (2007). Toward requirements engineering for self-organizing multi-agent systems. In Serugendo, G., Martin-Flatin, J., Jelasity, M., and Zambonelli, F., editors, *Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 299–302, Washington, DC, USA. IEEE Computer Society.

Wolf, T. D. and Holvoet, T. (2007). Design patterns for decentralised coordination in self-organising emergent systems. In Brueckner, S., Hassas, S., Jelasity, M., and Yamins, D., editors, *Engineering Self-Organising Systems*, volume 4335 of *Lecture Notes in Computer Science*, pages 28–49. Springer.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley & Sons.

Xuan, P., Lesser, V., and Zilberstein, S. (2000). Communication in multi-agent Markov decision processes. In *Proceedings of ICMAS Workshop on Game Theoretic and Decision Theoretic Agents*.

Yamins, D. (2007). *A Theory of Local-to-Global Algorithms for One-Dimensional Spatial Multi-Agent Systems*. PhD thesis, Harvard School of Engineering and Applied Sciences.

Zambonelli, F., Gleizes, M.-P., Mamei, M., and Tolksdorf, R. (2005). Spray computers: Explorations in self-organization. *Pervasive and Mobile Computing*, 1(1):1–20.

AUTHOR BIOGRAPHIES

MARTIJN C. SCHUT has over 10 years of experience in research on multi-agent systems, organisational modelling, computational intelligence and self-organisation. He has been involved in the following research projects that are related to the work described in this paper: NEWTIES (EU-FP6-003752, 2004-08; where an *emergence engine* was developed combining individual, social and evolutionary learning in a population-based adaptive system); and SYMBRION (EU-FP7-216342, 2008-13; development of symbiotic multi-robot organisms based on bio-inspired approaches and modern computing paradigms). His email is `schut@cs.vu.nl` and his personal webpage at `http://www.cs.vu.nl/∼schut`.

NIVEA FERREIRA-SCHUT works at the Institute for Computing and Information Sciences at the Radboud Universiteit (Nijmegen, NL) on the B-Screen project (NWO BRICKS/FOCUS grant number 642.066.605) whose work include the proposal of probabilistic (graphical) models for the breast cancer domain, based on screening mammography interpretation. She received in 2006 her PhD degree from the University of Liverpool, and her thesis regards the development and application of a logic-based programming language, which is a powerful – although simple – logical language obtained by the combination of a linear temporal logic-based framework with a probabilistic logic of belief. Her email is `nivea@cs.ru.nl`.