Weak Interaction and Strong Interaction in Agent Based Simulations

Fabien Michel, Abdelkader Gouaich, Jacques Ferber

Weak Interaction and Strong Interaction in Agent Based Simulations

Fabien Michel, Abdelkader Gouaïch, and Jacques Ferber

LIRMM Laboratoire d'Informatique, Robotique et Micro-électronique de Montpellier.
C.N.R.S. – Université Montpellier II, 161 rue Ada 34392 Montpellier Cedex 5 - France
{fmichel, gouaich, ferber}@lirmm.fr
http://www.lirmm.fr/~{fmichel, gouaich, ferber}

Abstract. This paper addresses the problem of the *engineering divergence phenomenon* in ABS. This problem is related to the fact that a particular conceptual model may give different outputs according to its implementation. Through two experiments, the paper shows that the implementation of the agents' interaction is one of the factors that are involved in this phenomenon. The underlying idea of this paper is that this problem can be greatly diminished if the analysis of the conceptual model incorporates some key concepts which are crucial for the implementation. To this end, this work proposes to identify two different classes of interaction: *weak interactions* and *strong interactions*.

1 Introduction

Agent Based Simulations (ABS) constitute an experimental tool of choice. Agent Based Modelling allows to directly represent the individuals, their behaviours and their interactions [1]. Each individual is named an *agent* and is supposed to represent an autonomous, proactive and social entity [2]. The autonomy relies on the fact that agents have full control of their behaviours. Unlike passive objects, agents proactively perform actions in their environment. In the scope of this paper, the social feature is defined as the ability of an agent to interact with others. Thus ABS are widely used to explore and design complex decentralised systems such as ant colonies, autonomous robots, social systems and so on.

As for any computer simulation [3], the ABS engineering schema can be described as follows:

1. Model design: during this phase, the simulation is expressed in a conceptual model (CM for short) that specifies the characteristics of the simulated system.
2. Model execution: during this phase, the CM specifications are implemented in concrete computational structures and programs that constitute the *simulator* of the experiment.
3. Execution analysis: during this phase, the outputs of the simulation are checked according to some validation rules and then interpreted.

Regarding this engineering process, a fundamental issue is raised using ABS: there is no consensus about the specifications that must be given to a CM (e.g. [4]). Thus, starting from a single CM and following this engineering process, several *computational models* (implementations) can be elaborated. Consequently, very different outputs may be obtained and the question of ABS experiments' reliability must be raised. Recent works such as [5,6] clearly address this matter. In this paper, this problem is identified as the *engineering divergence phenomenon*. It is important to distinguish this phenomenon, observed at the engineering phases, from the divergence of a particular CM due to its inherent properties. For instance, a chaotic system diverges, for each execution, using the same programming environment. This paper addresses the problem of the divergence of the outputs when different expertises and technologies are involved.

This paper focuses on the management of the agents' interactions and proposes to classify them along two different classes: *weak interactions* and *strong interactions*. The paper argues that simulations do not require the same programming technology according to the nature of the interactions present in the model. Thus, this distinction enables to refine the CM and diminish the engineering divergence phenomenon by reducing the possible implementations of the CM. This paper is structured as follows: Sect. 2 introduces the *engineering divergence problem* and describes the aims of the paper. The next section details the analysis of two experiments which are based on a minimalist CM. Section 4 discusses the results and proposes key concepts for refining a CM. The conclusion of this paper summarises the hypothesis and proposals of the paper.

2 The Engineering Divergence Phenomenon

2.1 Principle of ABS

Let us assume that Σ defines the whole possible states of the studied system, every ABS is based on the assumption that the environment evolution from one moment t to the next $t+dt$ results from the composition of the actions, $A_1(t), A_2(t)...A_n(t)$, produced by the agents and of the environment's action produced by its natural evolution, $E_n(t)$, at t. In a simplified way, the problem is to build a time function, *Dynamic* $D : \Sigma \mapsto \Sigma$, such as:

$$\sigma(t + dt) = D(\uplus(A_n(t), E_n(t)), \sigma(t)) \; . \tag{1}$$

The symbol $\uplus$ is used here to denote the action composition operator. It defines how the actions produced at the instant t must be composed in order to calculate their consequences on the previous world state $\sigma(t)$. Without detailing this calculus, it is easy to measure the difficulty of conceptualising such an operation knowing the diversity and the nature of the concepts hidden behind the word action: movement, decision-making, environment modification, and so on.

2.2 Technical Structure of ABS Platforms

Since the ABS area of appliance is not restricted to a particular research field, ABS software applications do not follow particular development rules. Thus they are very heterogeneous regarding the way of using them, the nature of the models they consider and their internal computational structures and components. However, from a technical point of view they always incorporate, explicitly or not, at least three core components that we identify as follows (Fig. 1):

- The *behaviour module*: this module defines behaviours of the simulated entities in concrete computational structures.
- The *scheduler module*: it defines the manner in which the agents are executed during the simulation, namely the activation structure.
- The *interaction management module*: this module defines how the interactions among the entities are handled.

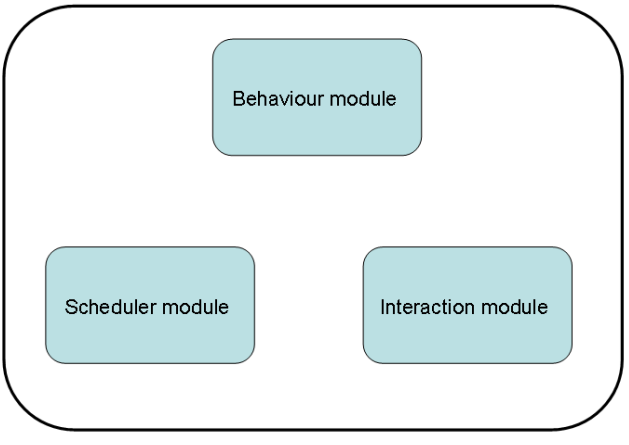

Fig. 1. The three core components of ABS platforms

2.3 Divergence of Simulation Outputs

As said in Sect. 1, if the specifications of the simulation CM are incomplete, the CM can be implemented in several ways and thus can yield different outputs. For instance, if the CM does not define clearly the model's time evolution, it is possible to implement it using a synchronous or an asynchronous simulation technique. Problems related to this particular point have been clearly shown and studied in works such as [7,8,9,10]. These works mainly focus on how the implementation of the scheduler module influences the outputs. For instance, [8] shows that *The Sugarscape Model*[1] (an adaptation of the model used in [11]) does

[1] *The Sugarscape Model* consists in a spatial distribution of generalised resource that agents need where agents have the ability to reproduce themselves.

not produce the same outputs for two different implementations. The authors explain that this divergence is due to the techniques used to schedule the execution of the agents' actions, namely the *activation structure* i.e. whether actions are performed synchronously or asynchronously[2]. Such a correlation between activation structures and outputs is also made is the previous cited works. Specifying the characteristic of the *scheduler module* in the CM, according to the nature of the studied system, is an important step that has improved the practical realisation of ABS by reducing the engineering divergence phenomenon related to this module.

As a matter of facts, the two other modules can also contribute to engineering divergence phenomena. Indeed, as shown in the next section, the implementation of the *interaction module* may also deeply influence the simulation outputs, even if the activation structure remains unchanged. Thus the work presented in this paper aims to:

- show that the interaction module is a key feature in the ABS framework.
- show that ABS involve different kinds of interaction that do not require the same programming attention.
- introduce some key concepts which can be used to refine the CM by specifying the nature of the interactions and thus the way they have to be implemented.

3 Experiments

3.1 Experimental Protocol

This section presents two experiments. The first deals with the modelling of the reproduction behaviour. The second consists in modelling resource consumption behaviours. The experiments are carried out using a testing platform defined by three modules as described in Sect. 2.2. As said in Sect. 2.3, the engineering divergence phenomenon may rely on the implementation of these three modules. Thus each module may potentially modify the outputs of the simulation. However, in this paper, the experimental protocol used for the two experiments presented here consists in building simulations where only the *interaction management module* is modified. Doing so will clearly identify the influence of this specific module on the obtained outputs.

3.2 Experiment 1: Reproduction Behaviour

In this section, a CM of reproduction behaviours is studied. The corresponding CM is defined as follows: let us consider two compatible (fertile and of the opposite sex) autonomous agents, A and B, with a simple behaviour which is only a choice between two actions: reproduce ($Agent_{repro}$) or do nothing ($Agent_{none}$) according to a defined probability $Pr(Agent_{behaviour})$.

[2] Specifically, the authors were interested in finding a suitable activation structure to simulate artificial societies.

Behaviour Module. For this CM, the behaviour module is defined as follows:

Table 1. The behaviour of agents as probabilities

$Pr(A_{repro}) = \alpha$ and $Pr(A_{none}) = 1 - \alpha$
$Pr(B_{repro}) = \beta$ and $Pr(B_{none}) = 1 - \beta$

Scheduler Module. The chosen activation structure consists in a discrete time simulation used with a classic synchronous time evolution. This method consists in activating all the agents in a sequential way and then incrementing the time of the simulation by one time unit. Moreover, we have randomized the activation list to avoid a possible advantage of one agent as proposed by [7]. As the agents are autonomous, their behaviours are independent, which means that $A_{behaviour}$ is not correlated to $B_{behaviour}$. Thus, for each step of the simulation, there are four possible interaction situations which have to be handled by the interaction module.

Table 2. Probabilities of interaction situations

$Pr(A_{repro}$ and $B_{repro}) = Pr(A_{repro}) \times Pr(B_{repro})$	$= \alpha\beta$
$Pr(A_{repro}$ and $B_{none}) = Pr(A_{repro}) \times Pr(B_{none})$	$= \alpha - \alpha\beta$
$Pr(A_{none}$ and $B_{repro}) = Pr(A_{none}) \times Pr(B_{repro})$	$= \beta - \alpha\beta$
$Pr(A_{none}$ and $B_{none}) = Pr(A_{none}) \times Pr(B_{none})$	$= 1 - \beta - \alpha + \alpha\beta$

Interaction Management Modules. These modules, the core part of the experiment, are defined according to three different ways for managing the interactions between agents. The first technique is inspired by the method used in [11]: in this case, the agents' actions are treated sequentially and each agent can reproduce, at his own turn, due to the proximity of a compatible partner. Table 3 describes the obtained results for each possible situation. The second approach for managing the agents' interactions (Table 4) corresponds to the implementation of [8]. In this case, when an agent successfully reproduce, the other agent cannot reproduce even if it has not acted yet[3]. The last interaction module is an application of the *influences / reaction* principle [12]. In this model the agents' action are considered simultaneously and we have assumed that it is necessary that the two agents want to reproduce to obtain an offspring (Table 5).

Results. Figure 2 shows the different outputs according to the three interaction management modules for hundreds of simulations: the A set of lines corresponds

[3] Lawson and Park have modified the agent reproduction rule to be more *realistic*: only one new entity can be produced by a pair of compatible agents.

Table 3. First interaction module

situations	birth(s)	probabilty
A_{repro}, B_{repro}	2	$\alpha\beta$
A_{repro}, B_{none}	1	$\alpha - \alpha\beta$
A_{none}, B_{repro}	1	$\beta - \alpha\beta$
A_{none}, B_{none}	0	$1 - \beta - \alpha + \alpha\beta$
Overall probabilities		
$Pr(births = 2) = \alpha\beta$		
$Pr(birth = 1) = \alpha + \beta - 2\alpha\beta$		
$Pr(birth = 0) = 1 - \beta - \alpha + \alpha\beta$		

Table 4. Second interaction module

situations	birth(s)	probability
A_{repro}, B_{repro}	1	$\alpha\beta$
A_{repro}, B_{none}	1	$\alpha - \alpha\beta$
A_{none}, B_{repro}	1	$\beta - \alpha\beta$
A_{none}, B_{none}	0	$1 - \beta - \alpha + \alpha\beta$
Overall probabilities		
$Pr(birth = 1) = \alpha + \beta - \alpha\beta$		
$Pr(birth = 0) = 1 - \beta - \alpha + \alpha\beta$		

Table 5. Third interaction module

situations	birth(s)	probabilty
A_{repro}, B_{repro}	1	$\alpha\beta$
A_{repro}, B_{none}	0	$\alpha - \alpha\beta$
A_{none}, B_{repro}	0	$\beta - \alpha\beta$
A_{none}, B_{none}	0	$1 - \beta - \alpha + \alpha\beta$
Overall probabilities		
$Pr(birth = 1) = \alpha\beta$		
$Pr(birth = 0) = 1 - \alpha\beta$		

to the application of the first method, the B set to the second and the C set to the third. This figure clearly shows the engineering divergence phenomenon related to the management of agents' interactions: even if the behaviour and scheduler modules remain unchanged, the outputs obtained by each method diverge. Thus, if the management of the interactions among the entities is not specified the implementation of this model can lead to an EDP.

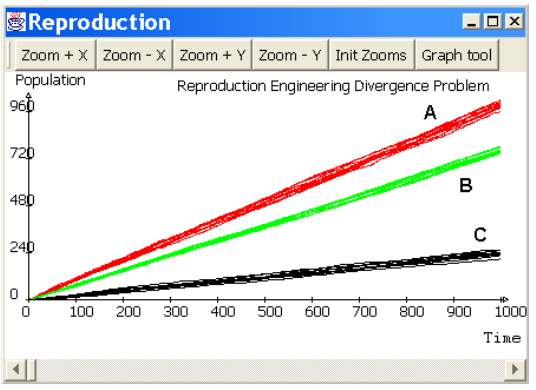

Fig. 2. Simulation outputs of the three interaction modules used in the first experiment with $\alpha = 0.4$ and $\beta = 0.6$

3.3 Experiment 2: Resource Consumption

The objective of this experiment is to study the consumption of a single resource by two agents A and B. Each agent has a life level L ($0 <= L <= max\_life\_level$) that he must maintain above a particular threshold $T (0 < T < 100)$ by consuming a resource up to the *max_life_level*. Thus the agents can exhibit two behaviours: consuming or doing nothing; $Agent_{consume}$ and $Agent_{none}$. When the life level of an agent decreases to zero the simulation is stopped. Besides the resource grows back at a rate of α units per time interval up to a defined capacity. The key statistical output of this model is a measure of the agents' average life level.

Behaviour Module. The behaviour module of the CM is defined as follows:

Algorithm 3.1: $Agent_{Behaviour}()$

if $life\_level < threshold$
  then *consume resource* : $Agent_{consume}$
  else *do nothing* : $Agent_{none}$

Scheduler Module. The scheduler module used here is exactly the same as in the previous experiment. Thus there are four situations that have to be handled by the interaction module:

Interaction Management Modules. Two different management modules have been used. The first method for managing the interactions is to immediately execute the agent's action. Hence the interactions between the resource

Table 6. Interaction situations

$A_{consume}$	and $B_{consume}$
$A_{consume}$	and B_{none}
A_{none}	and $B_{consume}$
A_{none}	and B_{none}

and one agent are reduced to two situations: the agent consumes the available resource to recover life points or the agent does not interact with the resource.

The second approach for managing the agents' interactions with the resource is to consider that actions can occur simultaneously. For instance both agents can access the resource at the same time. In this case the resource is equitably shared between the agents.

Results. Figure 3 shows the average of the outputs obtained for thousands of simulations with different initial values (T and α). These results show the outputs are similar in average and do not depend on the initial parameters (notably the abundance or scarcity of the resource, according to the α parameter) nor on the interaction module type. These results show that whatever the interaction module used and initial parameters, the results are similar in average. So even if the second method seems more advanced than the first one (because it handles simultaneity), the engineering divergence phenomenon is not observed considering this simulation model. In fact, the interaction management module has not influenced the outputs of this simulation at all.

3.4 Discussion

The first experiment clearly shows the great influence of the interaction management on the simulation outputs. Thus it stresses that it is fundamental to include the specification of the *interaction management module* in a CM for avoiding engineering divergence phenomena. However, on the basis of the two experiments, a question must be raised: why does the interaction management module influence the outputs of the first experiment while having no consequences on the second experiment?

The underlying idea of this paper is that these two experiments involve interaction processes which are very different in nature. And the paper argues that they do not require the same programming solutions for being implemented correctly (see [13] for other recent works which are related to this idea). Here, the question is to make the interaction management consistent with the meaning of the model. It is not a question of checking if the simulation's results model the reality but to know if the simulation process truly represents the model's reality that we have in mind.

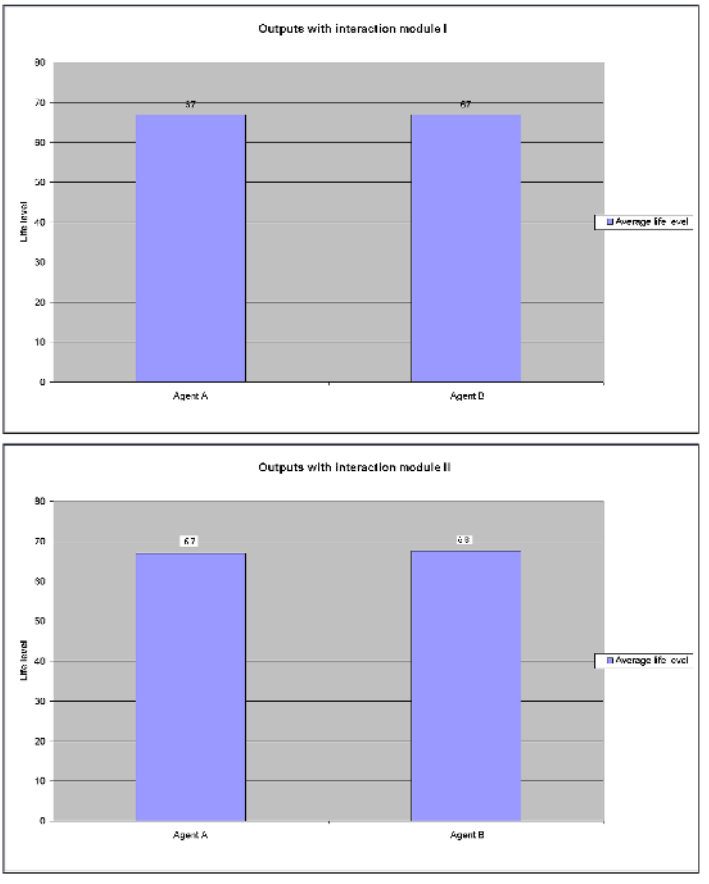

Fig. 3. Agents' average life level with first and second interaction modules

4 Weak and Strong Interactions

4.1 Strong Interaction

Let us analyse the three different implementations used in the first experiment (Sect. 3.2). With the first interaction module, where the agents' actions are treated sequentially, the results are quite surprising. Indeed, as Epstein and Axtell have noticed for their own experiments [11], one single agent can reproduce himself several times per turn. If three agents are present, the first agent produces a new entity, and then the second and the third may do the same thing[4]. So, it is obvious that, for our experiment, this first implementation of the interaction module does not agree with the meaning of the model.

[4] Epstein and Axtell say on this matter that this does happen *rarely* as an agent has a health parameter that decreases with a new birth.

It is undoubtedly the reason why [8] added a gestation period so that an agent cannot reproduce himself twice. However, this programming technique is still criticisable. The selected partner does not have any choice: the initiator has decided for both! The partner's individual behaviour, **its own goals** are not taken into account. What would happen if this agent was moving for a critical reason and found itself involved in a reproduction process that freezes its movement? This would totally contradict basic foundations of multiagent systems where agents are autonomous and not under the control of another one [2]. Indeed, when an agent modifies, directly by changing the internal state of another agent (the boolean value *pregnant* is now true), or indirectly by ordering the other agent to do it ("you are now pregnant"), the autonomy of the agent is lost. Indeed, the multi-agent approach relies on taking into account each individual. Without this assumption, the system's dynamic is not representing individual entities interacting together.

That is why, to be correctly implemented (with respect to the autonomy property), the reproduction interaction **requires** to take into account each behaviour before computing the result. The third implementation of the interaction module follows this guideline and produces a birth iff both agents want to reproduce. Indeed, if we consider **autonomous** agents, each one must independently generate a reproduction behaviour to finally interact and produce a new entity:

$$Pr(birth = 1) = Pr(A_{repro} \textbf{and} B_{repro}) = Pr(A_{repro}) \times Pr(B_{repro}) = \alpha\beta \ . \quad (2)$$

In the first experiment, a reproduction attempt has only a meaning when another agent is close. The result of such an interaction, a birth, requires at least that two agents are present in the system. An agent will never reproduce itself nor try to do it if it is alone in its environment. More precisely, reproduction is a behaviour that produces a result on the environment which can be realized when two agents are mating.

Definition 1 (Strong Interaction). *Actions of agents define a strong interaction when the feasibility of each action's goal depends on the action of another agent.*

This kind of interaction process requires considering all the involved agents before computing its result. It is important to notice that it is not a problem of coordination (as defined by [14]) between agents like in [15] for instance. The question is not to know how or why the agents internally produce or coordinate their actions in order to achieve their goals, but to specify how autonomous actions (decisions) are handled by the simulator.

4.2 Weak Interaction

The second experiment involves an interaction process that is less complex than the first one. In fact, even if the agents interact through the environment by consuming the resource, they carry out actions which are not directly correlated.

Moreover, the presence of other agents in the system is not necessary for consuming the resource: the feasibility of the goal of this action is not conditioned by the action of another agent.

This does not mean that complex situations cannot arise when dealing with *weak interactions*. For instance, the consumption of the resource can happen simultaneously. However, explicitly handling this situation does not modify the true meaning of the model because of this weak relation between the agents. Indeed, the autonomy property is respected since the two autonomous behaviours are taken into account. For instance, a collision between two robots is a *weak interaction* since space can be considered as a resource that agents consume. In this case the problem is not about managing sequentially or simultaneously the movement of the agents and the question of *realism/validity* relies on the granularity of the considered actions[5] and not on the management of simultaneity.

Definition 2 (Weak Interaction). *Agent actions define a weak interaction when the feasibility of each action's goal does not depend on the action of another agent.*

A good example of this kind of interaction type can be found in the classical "termites model" [16]. In this model the world is made of a set of termites and of a set of wood chips initially equally distributed around a toroidal environment. The termites follow a set of simple rules to gather wood chips into piles. This behaviour can be summarized with the two rules that follow:

- *Rule 1*: " *If I do not have a wood chip, I look for one randomly.* "
- *Rule 2*: " *If I have got a wood chip, I look randomly for another to put it down aside*"

Figure 4 shows four successive stages of a simulation where one can see that the wood chips end up in a single pile. In this example, the set of possible actions carried out by the agents is reduced to moving, taking and deposing an object.

It can be made an interesting remark on the behaviour of these electronic termites. A termite's behaviour does not suppose or integrate the existence of the others. Indeed a termite has no representation of its kind. So, even if the termites have the same goal, they interact in a *weak interaction* mode and, whatever the interaction management used, a single pile of woodchips is always the final result of this simulation[6]. The most interesting thing is undoubtedly that the same result is obtained when only one termite is working, but this takes only a greater amount of time.

[5] It is not relevant to raise the problem of the simultaneity of a collision when the selected space scale is coarse.

[6] This is not true in the particular case where there are more termites than wood chips.

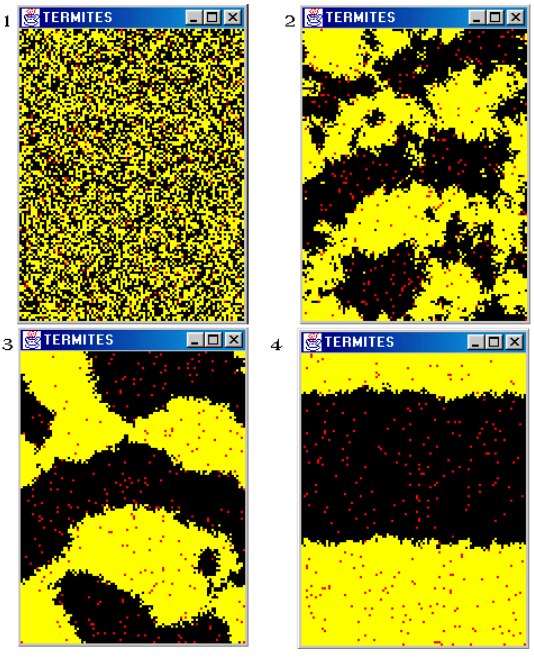

Fig. 4. The termites follow simple rules to gather wood chips into piles

4.3 Refining Conceptual Models

By classifying interactions along weak and strong classes, this paper made an explicit link between the CM and its implementation. In models that contain *strong interactions* the implementation techniques used to program them deeply influence the results of the simulation. Furthermore they may contradict some foundational agent concepts such as autonomy, resulting in skews in the simulation process. So, refining a CM according to such concepts offers a better understanding of how to implement it. It is necessary to include the analysis of the involved interactions into the conceptual model, if one wants to reduce engineering divergence phenomena.

Therefore, a related methodology would have to check every event that can occur in the environment (moves, agent birth, etc.) to decide which interaction type generated it, and how to implement it appropriately: *strong interactions* requires a specific management of the interaction itself, while *weak interactions* do not.

Another interesting consequence is that both kinds of interaction dynamic may coexist in one single simulation. So it does make sense to implement several interaction dynamics to simulate a particular model. For instance, in [17] a simulation of autonomous robots is proposed. In this model, robots are in charge of recovering objects where some of need to be pushed by several agents. The movements of the agents (*weak interactions*) are managed sequentially while the

displacement of a too heavy object requiring that two agents push at the same time (*strong interaction*) is implemented using an *influences/reaction* model: first, forces resulting from the actions of the agents are summed up, and then, the environment "decides" if the objects will finally move.

4.4 Future Works

By using the concepts presented in this paper, our future works suggest a systematic and formal way of implementing both kinds of interaction. To this end, an algebraic model has been proposed [18]. In this model, agents are autonomous entities that act only through explicit *interaction objects*. Interaction objects are structured algebraically as a commutative group. Hence, they are combinable by a + law to represent an aggregation of actions. Negative interaction objects are defined abstractly without any concrete intuitive interpretation, but for the internal definition and computation of the model. So, *strong interactions* are modelled as a sum of elementary interaction objects that define a third interaction object that will be treated correctly by the simulation environment. For instance, the sum of two attempts of reproduction are treated as a single interaction object by the simulator that will actually produce a birth if both of emitting agents want to reproduce themselves. In contrast, *weak interactions* are treated linearly by the simulator and their composition does not produce a new interaction object.

5 Conclusion

This paper has addressed the problem of the *engineering divergence phenomenon* in ABS. This problem is related to the fact that a particular conceptual model may give different outputs according to its implementation. Through two experiments, this paper has shown that the implementation of the agents' interaction is one of the factors which are involved in this phenomenon. The underlying idea of this paper is that this problem can be greatly diminished if the analysis of the conceptual model incorporates some of the key concepts of the implementation. To this end, this work has proposed to identify two different classes of interaction: *weak interactions* and *strong interactions*. The paper claims that this distinction helps to refine the conceptual model by increasing its specification.

Most of engineering divergence phenomena are due to incomplete CM specifications. Hence, conceptual model designers should emphasis the key properties of their system and specify if the interactions among entities are in a strong or a weak mode.

References

1. Parunak, H.V.D., Savit, R., Riolo, R.L.: Agent-based modeling vs. equation-based modeling: A case study and users' guide. In: Proceedings of the 1^{st} Workshop on Modelling Agent Based Systems, MABS'98, Paris, France (1998)

2. Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. The Knowledge Engineering Review **10** (1995) 115–152
3. Fishwick, P.: Computer simulation: Growth through extension. In: Proceedings of the European Simulation Multiconference ESM'94, Modelling and Simulation, Barcelona, Spain, A. Guasch and R. Huber (1994)
4. David, N., Sichman, J.S., Coelho, H.: Towards an emergence-driven software process for agent-based simulation. In Sichman, J.S., Bousquet, F., Davidsson, P., eds.: Multi-Agent-Based Simulation II, Proceedings of MABS 2002, Third International Worshop. Volume 2581., Springer-Verlag (2003) 89–104
5. Edmonds, B., Hales, D.: Replication, replication and replication some hard lessons from model alignment. In: Model to Model Workshop M2M, Marseille, France (2003)
6. Rouchier, J.: Re-implementing john duffy's model of speculative learning agents in a small scale society: Problems, interest and issues. In: Model to Model Workshop M2M, Marseille, France (2003)
7. Huberman, B.A., Glance, N.S.: Evolutionary games and computer simulations. In: Proceedings of the National Academy of Science USA 90. (1993) 7716–18
8. Lawson, B.G., Park, S.: Asynchronous time evolution in an artificial society mode. Journal of Artificial Societies and Social Simulation, JASSS **3** (2000)
9. Axtell, R.L.: Effects of interaction topology and activation regime in several multi-agent systems. In: Proceedings of the 2^{nd} Workshop on Modelling Agent Based Systems, MABS'00, LNAI 1979 (2000)
10. Michel, F., Ferber, J., Gutknecht, O.: Generic simulation tools based on mas organization. In: Proceedings of the 10^{th} European Workshop on Modelling Autonomous Agents in a Multi Agent World MAMAAW'2001, Annecy, France (2001)
11. Epstein, J.M., Axtell, R.L.: Growing Artificial Societies. Brookings Institution Press, Washington D.C. (1996)
12. Ferber, J., Müller, J.P.: Influences and reactions: A model of situated multiagent systems. In: Proceedings of the 2^{nd} Internationnal Conference on Multi-Agent Systems, ICMAS' 96. (1996) 72–79
13. Weyns, D., Holvoet, T.: Model for simultaneous actions in situated multi-agent systems. In: First German Conference on Multi-Agent System Technologies, MATES 03, Erfurt, Germany, to appear in LNAI volume, Springer-Verlag, Berlin (2003)
14. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Computing Surveys (CSUR) **26** (1994) 87–119
15. Vincent, R., Horling, B., Lesser, V.: An agent infrastructure to build and evaluate multi-agent systems: The java agent framework and multi-agent system simulator. In: Lecture Notes in Artificial Intelligence: Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems. Volume 1887., Wagner & Rana (eds.), Springer (2001)
16. Resnick, M.: Learning about life. Artificial Life Journal **1** (1994) 229–241
17. Chapelle, J., Simonin, O., Ferber, J.: How situated agents can learn to cooperate by monitoring their neighbors' satisfaction. In: Proceedings of the 15^{th} European Conference on Artificial Intelligence ECAI'2002, Lyon France (2002)
18. Gouaïch, A., Guiraud, Y., Michel, F.: Mic*: An agent formal environment. In: To appear in the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003), session on Agent Based Computing ABC'03. (2003) Orlando, USA.