Complex Systems Informatics and Modeling Quarterly
CSIMQ, Article 72, Issue 12, September/October 2017, Pages 86–116
Published online by RTU Press, https://csimq-journals.rtu.lv
https://doi.org/10.7250/csimq.2017-12.05
ISSN: 2255-9922 online

Natural Model based Design in Context: an Effective Method for Environmental Problems

Eric D. Kameni*, Theo P. van der Weide and Wouter T. de Groot

Institute of Computing and Information Sciences, Radboud University,
Toernooiveld 212 6525 EC Nijmegen, the Netherlands

E.Kameni@science.ru.nl,tvdw@cs.ru.nl,W.deGroot@science.ru.nl

Abstract. Analyzing complex problem domains is not easy. Simulation tools support decision makers to find the best policies. Model-based system development is an approach where a model of the application domain is the central driving force when designing simulation tools. State-of-the-art techniques however still require both expert knowledge of the application domain and the implementation techniques as provided by ICT (such as multilevel agent technology).

Domain experts, however, usually do not master ICT sufficiently. Modeling is more insightful for the domain expert when its goal is to formalize the language being used in that domain as a semi-natural language. At the meta level, this language describes the main concepts of the type of application domain. The model then is a concretization of this meta model. The main focus of this article is (1) to propose a natural-language-based approach to modeling application domains, (2) to show how these models can be transformed systematically into computational models, and (3) to propose the tool TiC (Tool in Context) that supports the domain expert when developing a model and generating a simulation tool. Our research methodology is based on Design Science. We verify our approach by describing the various transformation steps in detail, and by demonstrating the way of working via a sample session applying a real problem of Laf Forest Reserve deforestation in North Cameroon.

Keywords: Actor model, Multi-level Agent-based model, Domain-specific languages, Model transformation, environmental problems.

1 Introduction

Complex systems comprise a large number of interacting elements whose overall characteristics cannot be deduced directly from their components. Examples are the cells of a living organism, ecosystems, economic systems, the Internet and

* Corresponding author

Reference: E.D. Kameni, Th.P. van der Weide, and W.T. de Groot, "Natural Model based Design in Context: an effective method for environmental problems," Complex Systems Informatics and Modeling Quarterly, CSIMQ, Issue no. 12, pp. 86–116, 2017. Available: https://doi.org/10.7250/csimq.2017-12.05

Additional information. Author's ORCID iD: E.D. Kameni – orcid.org/0000-0001-9821-9470 and Th.P. van der Weide – orcid.org/0000-0002-6517-2079. Article PII S225599221700072X. Article received: 2016 October 19. Last revision received: 2017 September 2. Accepted: 2017 October 12. Available online: 2017 October 31.

social networks. These systems have features such as the presence of several scales of space or time and reflexive interaction, which is one of the reasons for their non-linearity. The analysis of these systems often requires an interdisciplinary approach combining mathematical methods with those of social sciences, geography and computer science.

Modeling and simulation is increasingly used to provide some understanding, solution or decision support in such complex contexts. However, modeling and simulation often require a number of skills that are not readily mastered by domain experts such as social scientists, geographers, and biologists. This necessitates these researchers to enter into dialogue with ICT experts, often finding the gap between the domain's and the ICT languages and methods to bridge. This interdisciplinary approach is increasingly observed in Agent-Based Models (ABM), enabling explicit representation of interactions and thus providing a framework for taking into account the dynamics of systems and the individual or collective logic of Decision Making. For example, Becu et al. [1] provide a formal methodology for performing a transfer of reality observed in Northern Thailand watershed in a computer model to explain the management of a watershed by land users, based on representations of local actors integrated with a multi-agent approach. Other work using the ABM methodology in environmental problem can be found in [2] and [3].

These works show various approaches for data collection, system design and integration of the data in the system. A common feature is, however, that the actual work is usually done by a domain expert (e.g. social scientist, environmental scientist) which is then delivered to a computer modeler to integrate the data and ideas in the model that should be developed. This basically one-way communication may cause hard-to-trace inconsistencies between the various steps in the chain (from theory to meta-model, from meta-model to concrete model, from concrete model to data collection, from rough data to interpreted data, from data to model and so on). Moreover, a lack of two-way interaction may result in missed opportunities, e.g. more efficient data collection or more explicit theory. To avoid these inconsistencies, it would be ideal for the domain expert to be able to produce his own model from the analysis phase to the simulation without transferring data to an ICT expert.

Generic Agent-Based Modeling and Multi-Agent Modeling approaches are increasingly being introduced nowadays to allow domain experts to easily and rapidly produce their own models. Among these approaches are those specially designed for the development of agents solving specific practical or engineering problems ([4], [5]), typically in software systems [6]. These approaches are often limited and do not cover the entire development process [7], [8]. There is also a lack of conceptual richness on the social-scientific and socio-economic level. This makes them not suitable for the analysis of environmental problems. Then we also have approaches in which agents communicate and cooperate on a single scale level, [9], [10] and [11]. The latter weakness causes problems for land use studies because the keys to understand land use change and therewith to design efficient solutions often lie on abstraction levels that are higher than the land users themselves.

And finally, we have the approach that takes into account the multiplicity of levels called Multi-Level Agent-Based Modeling MLABM [12], [13], [14], [15]. The models proposed in this approach are very different to each other from the point of view of the study area, the modeled scales and the approaches implemented to manage the complexity involved by the management of the various organizational levels [16]. To use them requires advanced skills in mathematics and computer science. The metamodels proposed in these approaches are not enriched by the concepts of the domain and do not use a domain language. This makes them difficult to understand and be used by the domain expert. These limitations make the approaches not immediately suitable for modeling socio-economic and environmental problems.

Our intention in this article is to propose a model-based development approach for designing ICT-based solutions to environmental problems. We will call this approach Natural Model-based Design in Context, abbreviated as NMDC. This approach should allow a (trained) domain expert to design a conceptual model for a concrete environmental problem. This model describes the underlying application domain in terms of environmental concepts and neither requires specific mathematical and ICT skills nor involves implementation details.

In particular, our intention is to design the conceptual model as a linguistic framework for a controlled semi-natural language. On the one hand, this allows the domain expert to describe and reason on the application domain in a language that is close to that language normally used in that application domain while, on the other hand, this language has an underlying formal basis as required by the system analyst.

During the modeling process, the domain expert and system analyst develop the syntax of this controlled semi-natural language. Furthermore, the system analyst tries to capture the semantics of this language in interaction with the domain expert. A general procedure for the design of a semi-natural domain language, as an interplay between domain expert and system analyst, and the competencies required for the domain expert and system analyst, have been described in [17] (see also [18]), and also apply to our situation.

A major part of the design of the proposed model-based development approach is a successive systematic transformation of the domain model into implementation oriented models. This also involves the utilization of MLABM. Our intention is to make benefit from existing open source tools to materialize our approach. The result is what we will call *Tool in Context* (TiC). As a special feature, TiC also offers a syntax-directed editor to assist domain experts in designing a model for a concrete application by themselves.

Summarizing, our intentions in this article are as follows.

I1 To provide a framework in terms of a metamodel to describe a class of environmental problem.
I2 To describe how a concrete model can be compiled and transformed into a running application using multi-level agent technology.
I3 To describe a tool for environmental experts to help them in developing and entering concrete models.
I4 To demonstrate how this metamodel is used in a complex situation, leading to a concrete model.

The layout of this article is as follows. Section 2 summarizes our methodology. Section 3 gives the background for our research. In Section 4 we describe the conception of the actor model and shows how multiple levels are assigned to modeling the approach. Section 5 introduces the *Action in Context* (AiC) meta-model, followed by the proposed AiC agent architecture in Section 6. Then in Section 7 we transform the AiC meta-model into the AiC Domain Specific Language. Section 8 describes the design of TiC and the extension with its simulation tool, and shortly describe a sample session. In Section 9 we propose an empirical validation of our approach to demonstrate its usefulness. Finally in Section 10 we draw some conclusions.

2 The Methodology

In this article, we make a distinction between a general approach for the problem area and its application to a concrete problem (see Figure 1). The general approach covers a broad class of problems (environmental problems in our case). This is extended to a concrete model for a given application domain by describing the actors involved,

and the actions that they may perform. This general framework is the base for a tool that can be used to enter concrete problems and to generate applications (simulation tools in our case).

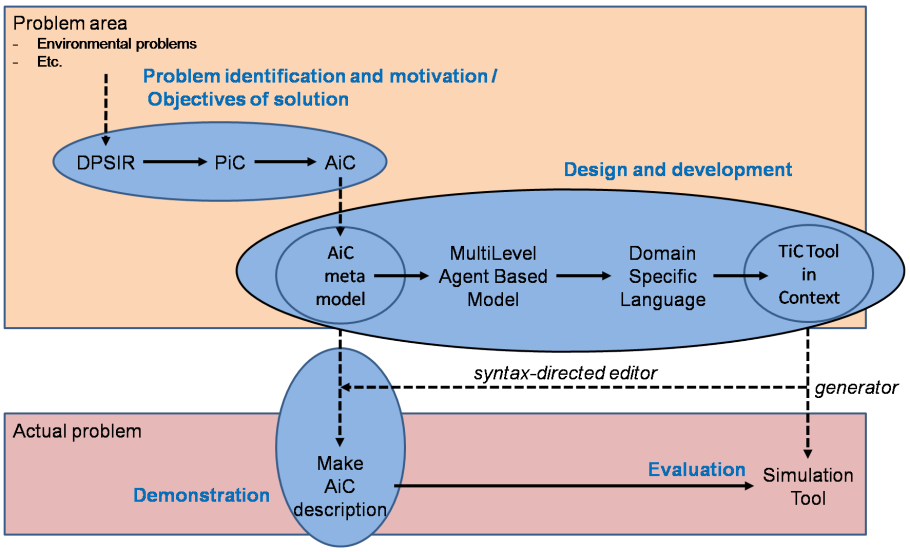

Figure 1. The various levels of abstraction and application (DPSIR - Driving forces-Pressures-State-Impacts-Response [52])

The methodology we use is based on the Design Science Research Method (DSRM) which is the standard research methodology used in the Information Systems discipline for designing new artifacts that solve unsolved problems or improve upon existing solutions (see [19], [20]). Based on a broad design science literature review, inspired by the seminal paper [19], [21] proposes a methodology for design science research consisting of the following six steps:

1. **Problem identification and motivation:** Define the specific research problem and justify the value of a solution.
2. **Definition of the objectives for a solution:** Infer the objectives of a solution from the problem definition and knowledge of what is possible and feasible.
3. **Design and development:** Create the artifact. Such artifacts are potentially constructs, models, methods, or instantiations.
4. **Demonstration:** Demonstrate the use of the artifact to solve one or more instances of the problem. This could involve its use in experimentation, simulation, case study, proof, or other appropriate activity.
5. **Evaluation:** Observe and measure how well the artifact supports a solution to the problem. Depending on the nature of the problem venue and the artifact, evaluation could take many forms.
6. **Communication:** Communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences, such as practicing professionals, when appropriate.

Our work is organized in accordance with these 6 steps. The problem identification, motivation, and the objectives for a solution are discussed in the introduction and are supported by Section 3 and Section 4. Design and development are addressed in Section 5, Section 6, Section 7 and Section 8. Then we demonstrate the working of the proposed model-based development approach in Section 9 with the Laf Forest Reserve deforestation case. A demonstration of the approach, described in this article, in a comparable, application area can be found in [22]. As a first evaluation, we give a sample session to show how well this approach will help to build a simulation tool for an environmental problem. Finally, the present document, as

well as the discussions in [23] are communication mediums that were used to assess the novelty and the rigor of our modeling framework.

3 Background

In this section, we present the difficulties encountered by domain experts in the modeling and simulation process and take a look at existing solutions that can overcome these difficulties, while bringing out the limiting factors that will be taken into account in our work.

3.1 Difficulties Related to Modeling and Simulation

During the last decades, the contributions of computer science and models have played a very important role in the representation and understanding of complex phenomena. However, the modeling and simulation process presents itself as a delicate and rather difficult exercise. It requires the researcher to know perfectly the concepts of the domain, the phenomena to be modeled and the technical tools to represent these phenomena and concepts in order to be able to simulate them.

A number of obstacles and difficulties must be overcome if modeling is to be made useful to life scientists more broadly than is the case today. The development of a sophisticated computational model requires both a conceptual foundation and implementation. Challenges related to conceptual foundations can be regarded as mathematical and analytical; challenges related to implementation can be regarded as computational or, more precisely, as related to computer science [24].

Several works in literature have identified difficulties related to modeling and simulation, [25], [24]. These difficulties usually involve strict compliance with the modeling process (analysis, design, implementation, deployment), data collection, the time required to construct the model and the calibration of the simulation parameters. On the technical side, there is a longer time devoted to computer questions than domain questions, the representation of domain concepts, and finally a difficulty to transfer and integrate the data. Nowadays this work is carried out by a multidisciplinary team.

Solutions have been already proposed to many of these difficulties (see [26], [27]), but some of them still persist, notably the transfer and integration of data and the representation of the domain concepts. Because the design and implementation phase still use technical tools that are not necessarily mastered by domain experts, which implies a transfer of data and problems of data inconsistency that may have occurred. To these are added the difficulty of taking into account the multiplicity of the levels of scale in the models. We present in the next section some solutions in terms of MLABM that have already been proposed to these difficulties.

3.2 Multi-level Agent-based Modeling Approaches

Agent-based modeling (ABM) has been recognized as an effective means to *model social life as interactions among adaptive agents who influence one another in response to the influence they receive.* [28], [29]. [30] also discusses the application of ABM to real-world business problems. In [31] the following aspects are distinguished when studying such applications: (1) theoretical, (2) experimental, and (3) computational.

Applications may show a hierarchical structure in agent cooperation, see [32]. [33] discusses the reasons for the inclusion of multiple organizational levels in models by grouping them into: (1) the representation of interactions between a large number of system components and the justification of learning and (2) adaption of agents in a group by observing the behavior of other agents at different levels. There is a need for

entity descriptions evolving at different spatial and temporal scales, corresponding to different organizational levels of the system [34], [16]. New approaches to modeling a complex system design therefore are interested in extending the agent characteristics with a multi-level agent approach, see [35], [36], [37].The aim is to integrate into a single model the knowledge and/or operating rules from different levels of analysis considered by the system designer. These systems are at the center of various fields such as biomedical research [38], flow modeling [39], social simulation [40], for explanatory insight into the collective behavior of agents obeying simple rules. This makes them particularly difficult to build and verify due to the quantity of data flows to consider and the number of interactions to take into account. The current development and programming solutions offer very little abstractions and highly technical methodological support is often limited to treat these aspects.

According to [34] the studies on MLABM engineering can be grouped into two main categories: meta-models and platforms. Among the works on meta-models we can mention the GEneric Architecture for MultiAgent Simulation (GEAMAS) [12], the pioneering of MLABM, which is based on the description of a three-level MLABM framework that integrates agent representation at the micro level, the system point of view at the macro level and the representation of agent aggregation in a specific context at the meso level. The inter-level communication is asynchronous. Although GEAMAS was proved to be very helpful in understanding and analyzing the behavior of agents and to master the complexity of natural phenomena, it requires careful designing and long computing time. These issues can be explained by the fact that during simulations, agents usually face the same kinds of conditions and the repetition of the same action leads the system to become less powerful. Moreover, it is an architecture oriented methodology but the agent meta-model is not explicit and very abstract especially for domain experts. The design phase is still less detailed compared to various current challenges of scalability. [13] has proposed the Influence Reaction Model for Multi-level Simulation (IRM4MLS) meta-model. It is a generic meta-model that aims to specify and execute MLABM. This meta-model specifies the relations of perception and influence between levels with digraphs. IRM4MLS provides additional concepts on existing methodologies in the literature, including the representation of the hierarchical multi-level system, with different spatial and temporal dynamics, multi-level agents or environments and agents that are dynamically introduced in levels. However, its implementation remains very difficult due to various mathematical concepts used to represent its rules. Moreover, the specification of the reaction function is very difficult since it uses a discrete time model. Thus it will not be easy for other domain experts than mathematicians and ICT experts to use IRM4MLS model without assistance. Another generic meta-model PADAWAN is presented in [14]. It is a multi-scale ABM meta-model based on a compact matrical representation of interactions, leading to a simple and elegant simulation framework. And finally, the AA4MM meta-model, described in [41], provides a multi-modeling or model coupling meta-model applied to ML-ABM.

Concerning studies based on platforms, we can mention GAMA [15] which is a platform development of MLABM to offer the developer an easy and appropriate language allowing to build and reuse complex models combining various agent architectures, environment representations, and levels of abstraction. [42] has proposed SPARK, a Simple Platform for Agent-based Representation of Knowledge which is a framework for multi-scale ABM, dedicated to biomedical research. Some complementary work on the platforms are found in [39].

The challenges of multi-level systems are discussed by Blair and Grace [43]. The authors state that the challenge of multiple levels as such is not the most difficult to overcome. Rather, the challenge lies in the heterogeneity of the processes

involved. Indeed, a multi-level system is not necessarily strongly divided in terms of geographical distance or number of parties involved; the relevant characteristic is its' vertical spread over different levels of scale. Thus [44] defines a multi-level system as a system distributed vertically on several levels of different sizes in one or more dimensions (geographical, network, etc.).

This complies with what we have seen about AiC and its actor's field, albeit that the actor's field concept, being actors-based rather than system-oriented, allows to use the same core components (action, actor, options, motivations) on all levels. Even though actors obviously take their options and motivations from different system levels (farmers from the local level, merchants from the regional market level, World Bank from the global level), there is no need to fully model these levels themselves.

Although these works allow developing MLABM effectively, the models proposed are very different to each other from the point of view of the study area, the modeled scales and the approaches implemented to manage the complexity involved in the management of the various organizational levels [16]. To use them requires advanced skills in mathematics and computer science. Furthermore, many of them are based on ad-hoc meta-models. This makes the transferability of ideas from one to another not obvious and usually hard. Very few of them offer the design of a central model.

In our work, we design a generic methodology of modeling environmental problems using multilevel agent-based technology through a Specific Language, which can be reused, step by step, by domain expert to design a new model for a specific domain.

3.3 Conceptual Modeling

Object-Role Modeling (ORM) is a method for conceptualizing and explicitly representing data and processes in an application domain. The philosophy behind ORM is that it tries to describe this domain by reflecting the communication between its members as a semi-natural language. An ORM scheme basically is a grammar describing that communication. This grammar is also referred to as information grammar. An ORM diagram is a graphical representation of this grammar. The information grammar not only describes the structuring of the information but also describes constraints on how the structure may be populated. For an overview on expressivity in conceptual data modeling, based on [45], see Appendix A.

The information grammar describes the elementary sentences that are valid in the associated Universe of Discourse. From these sentences, other sentences may be formed. Object Role Calculus (ORC) ([46]) and ORM2 ([47]) are examples of such generic systems for constructing sentences. In formal terms, the information grammar may be seen as the signature of a first-order logic theory ([48]), together with a set of axioms, derived from the population rules. The associated logic theory is a basis for formal reasoning about the domain. Proof assistants (such as the Coq Proof Assistant ([49]) may help policymakers.

4 A Framework for Environmental Problem Analysis

We start this section with a presentation of the context in which the AiC framework is positioned. All starts with the study of environmental problems and the desire to design solutions to them. Because the design of a solution requires knowledge of causal chains (roughly, what causes the problem and why), causal frameworks hold a position of priority, rather than spatial [50] or mathematical methods [51].

4.1 Problem-in-Context (PiC)

Problem-in-Context (PiC) is an extension of the well-known Driving forces-Pressures-State-Impacts-Response (DPSIR) framework, proposed by De

Groot [52] and used inter alia by [53] to analyze, explain and solve environmental challenges. Compared to DPSIR, PiC makes the normative side of environmental problems explicit and adds much more details to their social causes. Figure 2 gives the essentials. In its problem-analytical part, PiC contains two parallel

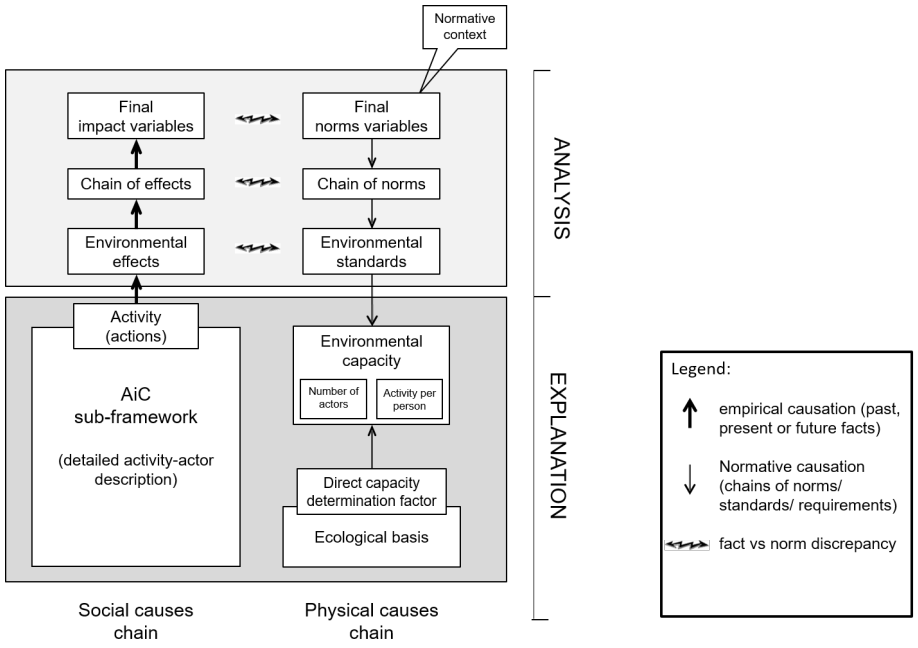

Figure 2. Problem-in-Context framework (PiC), adapted from [52]

causal chains. The left one is factual and indicates which human activities cause environmental effects and, onwards, impacts on society (called 'final variables'). The other chain is normative, running from desired societal goals (e.g. poverty alleviation or biodiversity conservation) to environmental standards and further downward to human activity prescriptions ('environmental capacity'). These two chains are connected in a comparative manner, e.g. factual poverty compared to poverty alleviation goals, environmental facts compared to environmental standards and factual versus prescribed activities, defining the environmental problem at these different levels in the double chain.

Focusing on the social explanation of problematic human activities, PiC further contains the Action-in-Context sub-framework that is used as the meta-model (see Section 4.2) in the present study. This allows the analysis of human actions to remain explicitly connected to their environmental relevance, i.e. their contributions to the environmental problem or its solution.

Thus in the Problem-in-Context framework, one entry point to analyze an environmental problem is the activity or action that sets it off. This action leads to a chain of effects on the environment, e.g. those predicted in environmental impact studies. The relevance (evaluation) of an impact depends on the normative context (e.g. policy principles on poverty and biodiversity) that induces the chain of environmental standards and provides a reference for the analysis. At the level of the activity (actions) variables, the chain of norms ends with the environmental capacity (carrying capacity), which may be split into the required number of actors and/or the required intensity of action per actor and may also be mapped, as in land evaluation.

4.2 Action-in-Context (AiC)

The sub-framework Action-in-Context (AiC) is depicted in some more detail in Figure 3. It focuses on the empirical explanation of the activity (actions) element

in PiC. Going upwards in the figure, the arrows indicate the causal direction. The direction of the explanation, amounting as it does to repeatedly asking the why-question, therefore runs downward in Figure 3.

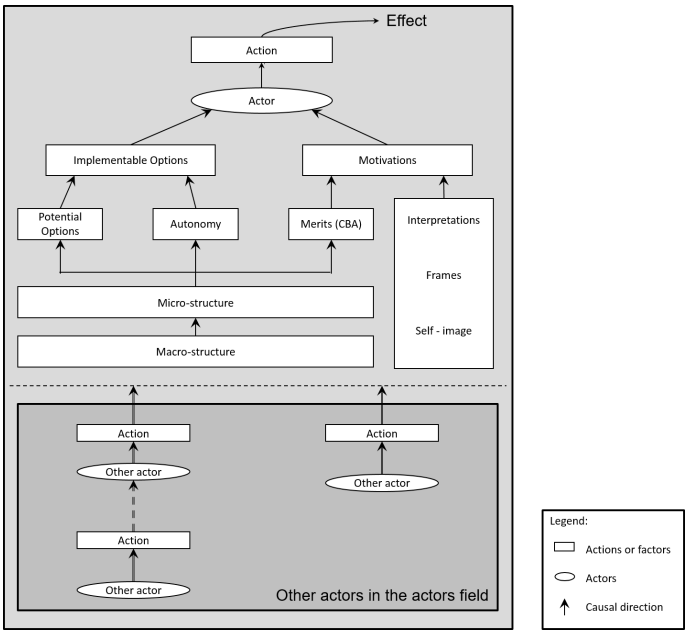

Figure 3. The AiC framework, adapted from [52]

For any action, AiC shows the actors involved and their options and motivations related to this action. Options are defined as what the actor can do (alternative courses of action), and motivations are the actor's reasons that come into play for choosing one or another option. This allows AiC to explicitly represent the process of actors' decision-making that integrates economic factors, political context and sociocultural perceptions (the merits of the action and the subjective interpretation of these merits).

Actor decisions can be modeled using any theory that contains the options and motivations concepts one way or another, such as a rational choice theory or a theory of reasoned action (see next section). In AiC, this is supported by four concepts that underlie the options/motivations triangle. They are (C1) *potential options*, defined as everything the actor could consider if he would have enough resources (C2) *autonomy*, defined as all resources ('capitals') of the actor, such as financial, social and cognitive, minus possible restrictions such as prohibitions or taboos, (C3) *merits*, defined as more or less objective reasons for the actor to act, such as financial gain, hours of work, calories of food, and (C4) *interpretations*, defined as the subjective (personal or cultural) beliefs about the good or bad of the merits, e.g. the degree they express loyalty, reputation or honesty. These four elements are then grounded in micro and macro structures that might also be called systems at the meso and macro levels.

From all this, we could say that it is a balanced summary of the regular social sciences. Uniquely for AiC however, the actor's options and motivations (hence decision) may also be seen as influenced by other actors that find their own options and motivations defined at system levels higher than those of the original actor. We will refer to this as AiC concept C5. One example may be a government issuing a prohibition on a certain type of land use, based on a nationwide land evaluation or food security study. Through the prohibition, the government actor reduces the options of farmers based on its assessment on the national level. In other words, if we include the government as an actor in a model additional to land users and connect them through the prohibition action, the model becomes multi-level through

this specific, one could say vertical or causal, type of being multi-agent [54]. Note that in its turn, the government actor may be influenced by global actors such as the International Monetary Fund (IMF) setting conditions on government discretion (options) through, for instance, a Structural Adjustment Program. In AiC this causal arrangement between actors is called the actor's field (see Figure 3), defined as the composition of all significant causal linkages from the action-to-be-explained (e.g. land use by farmers) to other actors through causal actions originating from these other actors.

4.3 The Need for an AiC Meta-model

Figure 3 does not provide a precise definition of how to network the possible linkages between actors and actions. For example, causal actions may influence elements that influence the actor's decisions, e.g. adding or subtracting potential options (e.g. through agricultural extension), elements of autonomy (e.g. through micro-credit or prohibitions), merits of options (e.g. by taxing some or subsidizing others) or interpretations (e.g. by propaganda or cultural debate) which are actions affecting options and/or motivations of other actors. For such a precise definition modeling languages such as Unified Modeling Language (UML) and ORM are used in practice.

Lacking such a precise definition makes it difficult (1) for a domain expert to precisely describe the application at hand, and (2) for the ICT expert to transform that description into a computational model that is easily comprehensible and thus can be used to develop a simulation or support tool.

Since we assume that the domain expert not necessarily (nor likely) will be an ICT expert, we need a precise description of the AiC framework that is comprehensible on both sides. For the domain expert, the technique to model an application domain should be close to the language used by domain experts. That way, the domain expert will not be hindered by technology issues when describing the application at hand as an AiC framework.

The precise description of the AiC framework is referred to as the AiC meta-model. The term meta-model is used since this model actually describes concrete AiC frameworks. In the next section, using the design approach proposed by ORM, we show how to move from the AiC framework into the AiC meta-model.

5 The AiC Meta-model

In this section, we begin the design activity of our approach. According to March and Smith [55], Hevner et al. [19] point to four types of artifacts that can be produced at the end of the design process:

- *Constructs* that provide the vocabulary and symbols used to define problems and solutions of the domain.
- *Models* that consist of a set of propositions or statements relating constructs.
- *Methods* that are defined as the set of steps that allow you to perform a task.
- *Instantiation* which are none other than the actual realization of an artifact in its environment.

In our work, we develop our artifacts inspired by the cycle view of research activities in design science proposed by Hevner et al. [56].

Thus, in this section, we defined the vocabulary of the domain ('Constructs') through the AiC framework (Capital, Resource, Action, Actor,...) using ORM technique. Also in this section, we focus on the actors in the actor field and define a set of propositions or statements ('Models') which overview the essential concepts and their relations as used by the actors to make their decisions as described in the previous subsection. It may also be used as a basis for inter-agent communication, but that is outside the scope of this paper.

The AiC meta-model is displayed in Figure 4 in the style of the ORM modeling technique (see Section 3.3). Each AiC application then can be seen as an instantiation (population) of this meta-model.

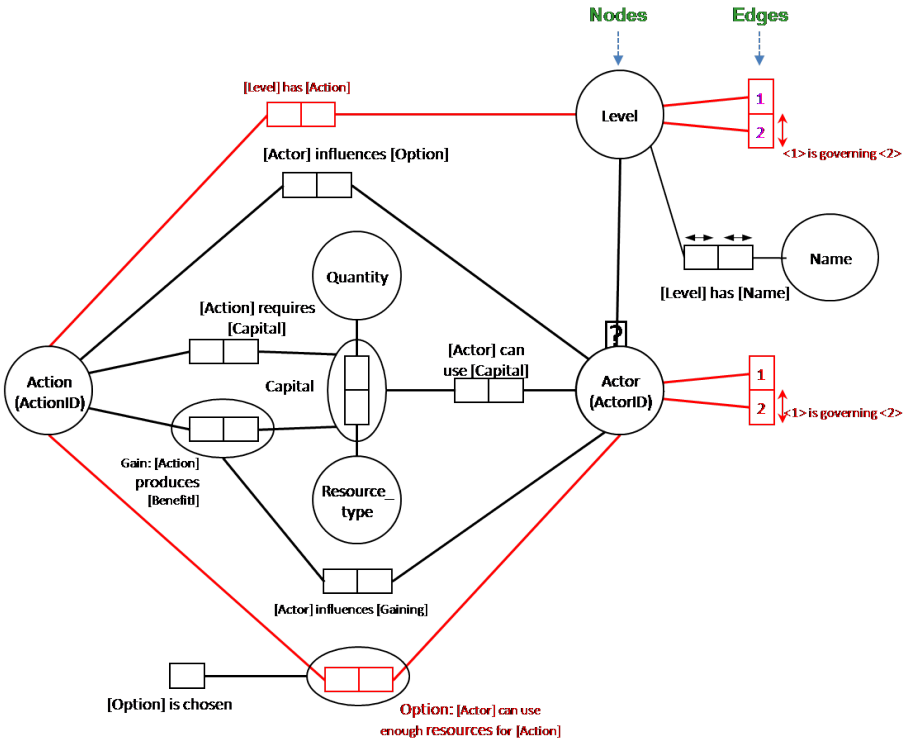

Figure 4. The AiC Conceptual Model

The essentials of Section 4.2 are summarized by the following elementary sentences. Actions are identified by their action ID; they require capitals as inputs and produce benefits as outputs. A capital is some quantity of some resource type. Actors, identified by the actor ID, dispose of capitals. This is formulated by the following elementary sentences:

AiC 1. Capital: Quantity of Resource_type.
AiC 2. Action requires Capital.
AiC 3. Actor can use Capital.

The sentence AiC 3 expresses the autonomy of the actor (AiC concept C1). The merits (AiC concept C3) are represented in the following elementary sentence:

AiC 4. Gain: Expected net benefit of Action

So an actor has as options all action for which the actor disposes of all required capitals. The following rule defines options (AiC concept C1) in terms of other elementary sentences. So an option does not correspond to an elementary sentence since it is a derivable sentence (derivable sentences are marked red in Figure 4).

AiC 5. LET Option BE Action requiring less Capitals than Capitals that can be used by Actor

The option that is chosen by the actor is registered in the following elementary sentence:

AiC 6. ChosenOption IS Option with highest Gain

According to the previous Section (See Section 4), actors also have motivations for options (representing AiC conceptC4). The reason is that an actor chooses the option with the highest expected net benefit. We imply by net benefit, the Gross benefit minus the expenses used to perform the action. Finally, actors may influence each other (AiC concept C5) by influencing the gain of actions. In other side, actors also influence options, e.g. by offering new knowledge or prohibiting certain options. A typical example is when the government wants to stimulate specific actions by increasing some of the resource types it produces, in the first case, and, in the second case, we can have a law prohibiting action. This is formulated as:

AiC 7. Actor influences Gain.
AiC 8. Actor influences Option.

This elementary sentence will be the basis for defining hierarchical levels for actors.

5.1 Managerial Actor Levels

Action-in-Context offers an alternative way of creating a multi-level model that is much less concerned with perceptions and more with direct causality. Let us assume that farmers are influenced by the fertilizer price, where it functions as one decision factor (motivation) on the farmers level. Let us assume further that the fertilizer price is co-determined by a subsidy issued by the Ministry of Agriculture. The level of subsidy is determined by the Ministry's own motivations at its own level, connected, for instance, with the need for national food security, and its subsidy level options, connected, for instance, with the other ministries. Thus, the fertilizer subsidy connects the farmer causally with the national food system, without any perception of this system by the farmer being needed. On the next level, similarly, for instance, the IMF may have put an abolishment of fertilizer subsidies as a condition to a loan to the government. The IMF "lives" on a level of its own, e.g. composed of national governments as actors and neo-classic economics as the dominant value system. Thus, in spelling out the micro and macro structural and cultural elements of the decision contexts of actors in the actor's field, the AiC multi-agent model becomes multi-level and multi-scale automatically, exactly to the extent causally relevant to the problem at hand [54], without cross-level considerations of perceptions needed.

Our intention is to implement AiC models by multi-level agent-based models. In AiC the basis for actor hierarchy is the following derivable sentence, that defines how actors govern other actors by their ability to influence the merits of actions:

AiC 9. LET Actor is governing Actor BE Actor influences Existence or Gain of Options or Actor.

As a next step, we define a partition scheme for actors based on this sentence.

This is done as follows. As a short-hand notation, we will write $g(a,b)$ as an abbreviation for the sentence Actor a is governing Actor b. There are several ways to add an partial order relation:

1. (governor-based) $a \equiv_g b \triangleq \forall_c [g(c,a) \Leftrightarrow g(c,b)]$.
2. (subordinate based) $a \equiv_s b \triangleq \forall_c [g(a,c) \Leftrightarrow g(b,c)]$.
3. (combined) $a \equiv_c b \triangleq a \equiv_g b \wedge a \equiv_s b$.

It is easily verified that these three relations all are equivalence relations. In this paper, we will assume that the governor-based approach $\equiv_g$ is being used. So each equivalence class consists of all agents that are governed by exactly the same agents. An equivalence class thus also may be seen as a managerial level. The levels that we introduce are the equivalence classes (managerial levels) of this relation.

It will be convenient to represent the managerial levels as a directed graph that indicates the managerial relations between the various managerial levels. The equivalence classes will be the nodes of the graph. The governing relation defines the edges between the nodes. There will be an edge from node a to node b if for some (and thus for each) agent in the equivalence class a there exists an agent b such that $g(b,a)$.

We will write $a \to b$ when there is a path from node a to node b. Depending on the properties of the path relation, we can have different managerial styles. For example, in a strict hierarchical management structure the relation $\to$ is a strict (irreflexive) partial order, which is characterized by:

1. no agent is in control of its peers: $\neg a \to a$
2. transitivity: $a \to b \wedge b \to c \Rightarrow a \to c$
3. strictly hierarchical: $a \to b \Rightarrow \neg b \to a$.

Another managerial style would replace the latter condition by: option could be that no managerial class is in control of all the other, or:

3. no dictator: $\forall_a \exists_b [a \to b]$. So each managerial class is governed by at least one other class. The consequence is that we will have managerial cycles in this case.

With each managerial class we can consider the options and motivations of that class. We will use the conceptual language to define these sets. For this purpose, we extend the conceptual model of Figure 4 to cover levels and the edges.

AiC 10. Level POWERSET OF Actor.
AiC 11. LET Level is governing Level BE Level containing Actor is governing EACH Actor in Level.
AiC 12. LET Level has Action BE Level containing ACTOR having enough resources for Action.

Note that in a similar way we can extend actor properties to level properties.

6 The AiC Architecture

6.1 Agent Characteristics

In the ABM context, an agent is defined as an entity, situated in an environment, acting autonomously and flexibly to achieve its objectives (adapted from [57]) located in a level with other actors or objects with whom it interacts. It is characterized by various methods supplemented by three other additional properties such as identity, state, and behavior.

Usually, the goal of ABM is to search for explanatory insight into the collective behavior of agents obeying simple rules.

An agent can receive sensory input from its physical and social environment and can perform actions that are likely to change this environment in achieving its goals.

It can act independently without the intervention with other agents. This is well observed in the AiC metamodel because the actors are in different scales that can be located at the macro or micro levels and therefore each of their action may change the state of that level. Several architectures of agents exist. What differentiates the architectures in particular is the way in which perceptions are related to the actions. Thus there are three main architectures of agent:

- Reactive agents: who react only to the changes which occur in their environment.
- Deliberative agents: who make a decision based on their purpose to choose their actions.
- Hybrid agents: integrating the two previous features.

In our work, we focus on deliberative architecture, more precisely the architecture of decision-making based on utility.

In many situations, the goals are not sufficient to generate a high quality of decision. For instance, if there are several choices to achieve an objective, some will be more efficient than others. In this situation, the agent reasoning only with goals has no means to make the best choice, if goals only provide a simple distinction between states being satisfactory or not. Instead, the agent must be able to recognize degrees of satisfaction. In such cases, the agent will prefer one state above the other based when that state has a higher utility than the other.

6.2 AiC Agent Architecture

For an operationalization of the AiC metamodel (see Figure 3) we will first determine the associated programming model.

From the agent architecture based on utility [58], (see Figure 5), we are bringing out the similarity between AiC and a utility-based architecture. The resulting actor architecture defines the AiC programming model (see Figure 6) close to the agent model according to the description of AiC conceptual model.

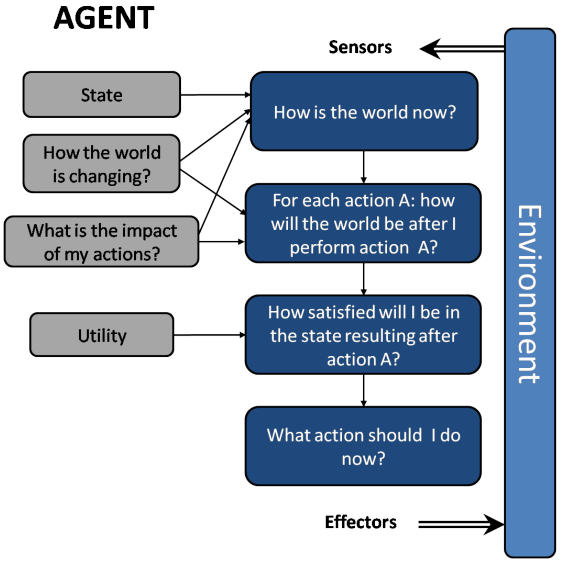

Figure 5. Agent utility-based architecture proposed by Russel [58]

Utility is a functionality that enables intelligent agents to make the best choice to achieve their goal. Utility strongly depends on the merits of actions and the resources (including capital) available to the agent. Utility and motivations determine and justify the behavior of the agents. Options and motivations of AiC framework can meet respectively with the agent questions concerning the possible future and future

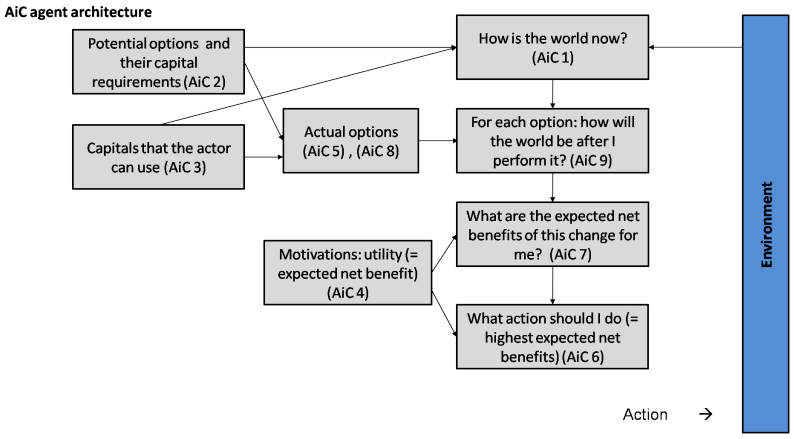

Figure 6. Adaptation of AiC framework to an agent architecture based on the utility

desired and thus contribute to the conceptual specification of the utility function that determines the actions to be performed by the agent.

The AiC actor architecture differs somewhat from wider-known ones such as Belief Desire Intension (BDI) [59] or the Theory of Planned Behaviour (TPB) [60]. Basically, it puts more emphasis on specifying the options the actor actually has. This makes the model suitable to also simulate actors in situations of poverty, where people may have strong intentions but very few options to act. Furthermore, AiC skips the intermediary 'attitude' and 'intention' concepts that often lead to a strong focus on attitudes and then later run into the dreaded 'attitude-behavior gap'. Instead, AiC focuses on motivations, that is, the criteria that actors use to arrive at their actions in a given action context. These criteria can be extremely simple (e.g. pure utility as in rational choice theory) or enumerative-additional (e.g. a set of criteria that add on to each other as in broad rational choice or multi-criteria models), or highly qualitative, e.g. the fit of an action into the overall life plan and identity. And finally, the actual 'decision calculus' is left open. It can be the maximization of something (e.g. welfare as in rational choice) or 'satisfying', for instance ('work until I have enough'). Thus, AiC may provide much modeling flexibility without conceptual burdens, much like the suggestion of Elster [61] whose primary actor model is to distinguish between opportunities (option in AiC) and desires (motivations in AiC). A full discussion of this issue is not our intention here, however, because our focus lies on the multi-level issue rather than modeling separate actors.

7 The AiC Domain Specific Language

Combined with the previous sections, this section presents the set of steps for transforming a type of object ('Methods').

In literature several definitions of Domain Specific Language (DSL) have been proposed, see for instance [62] and [63]. In each definition, the DSL depends on the domain application and the targeted user community. The definition from [62] seems to be the most accepted: *a DSL is a programming language whose specification is dedicated to a specific application area, offering good abstractions of this area to the user (SQL, regular expressions, Datalog, etc.).* By focusing on a particular domain, domain-specific languages offer custom programming solutions in order to effectively meet the specific needs of the associated domain. Consequently, domain-specific languages offer many advantages compared to general programming languages in terms of productivity, verification and programming ease. Today these languages are an interesting and serious alternative to traditional approaches of software

development [64], because they offer a high level of abstraction for the domain in question, facilitating the construction and verification of software systems.

The goal of domain-specific languages is to narrow the gap between the application domain and its implementation. Therefore such languages must satisfy (see [65]):

1. conceptual proximity: the domain concepts must be proximal to their corresponding language concepts
2. representational proximity: the representation of concepts in the application domain is proximal to the representation in the domain specific language.

Domain-specific languages are commonly represented by a meta-model that describes the relevant concepts in the application domain and their relations. In this paper, we use ORM to define the conceptual scheme. In ORM representational proximity is guaranteed by the modeling approach that requires for concept type a standard way of representing its associated concepts in a way that is derived from the description given by the domain expert. Conceptual proximity is also guaranteed from the specific modeling approach advocated by ORM.

In this section we describe how the AiC meta-model can be systematically converted into an Xtext format description. The Xtext description is the basis for the automatic construction in the Eclipse environment of a corresponding parser, linker, type checker, compiler as well as providing editing support for Eclipse, IntelliJ IDEA and a variety of web browsers (see the Xtext website [66]).

7.1 Transforming the Abstract Syntax into Concrete Syntax

In the ORM approach, the abstract syntax is the syntax that is described by the sentence types from the conceptual scheme, the AiC meta-model in our case. In this section we will give a representation of abstract syntax in terms of Xtext (see [67] and [68] and the website [66]). This description is referred to as the concrete syntax.

Xtext is a framework for the development of programming languages and domain-specific languages. Xtext requires a description of the underlying conceptual scheme in the specific Xtext format. In this section, we transform the AiC meta-model from Figure 4 into Xtext format. We follow the approach described in [69] to transform ORM into UML.

7.2 The Data Transformation Algorithm

The transformation of an ORM scheme into a relational scheme (also referred to as the *data transformation algorithm*) consists of the following main steps (see [70]):

1. Transform the object types to their identification. This step will lead to the primary keys of the relational tables.
2. Process the fact types. Functional fact types lead to properties of the associated independent object types. Non-functional fact types describe relations between the associated (independent) object types and therefore lead to separate tables that describe relations between their dominant object types. In terms of the relational model, the independent object types in such a relation are foreign keys to the associated object defining tables.
3. Process constraints. All constraints are formulated in terms of the relational scheme formed in the first steps.

The main step of this data transformation algorithm is the grouping of all fact types to associate them with their dominant object type. In the case of a transformation into a multivalued relational scheme, the representational model allows repeated values. The consequence is that step 2 in the procedure above may be relaxed. A non-functional fact type may also be represented by a multivalued relation. We assume that a multivalued attribute can also be a key for a relation.

7.3 Transforming AiC Meta-model into AiC Relational Scheme

We follow the steps from the transformation scheme above to transform the conceptual AiC scheme into the relational AiC scheme. We make an straightforward association of fact types to dominant object types as displayed in Figure 7, where we try to keep the number of generated relations minimal. Note that we introduced a superfluous identification for levels (LevelID); this will simplify the handling of levels. This leads to the following relational AiC model:

R1. Actor (ActorID,
*(can use (Quantity, Resource_type))**,
(influences ((ActionId), (Quantity, Resource_type)))?,
(is governed by ActorID)?,
*(has option (ActionId))**,
(has chose (ActionId))?
)

R2. Action (ActionID,
(requires (Quantity, Resource_type))?,
(produces (Quantity, Resource_type))?
)

R3. Level (LevelID,
*(has member (ActorID))**,
(is governed by ((ActorID)))?*
)

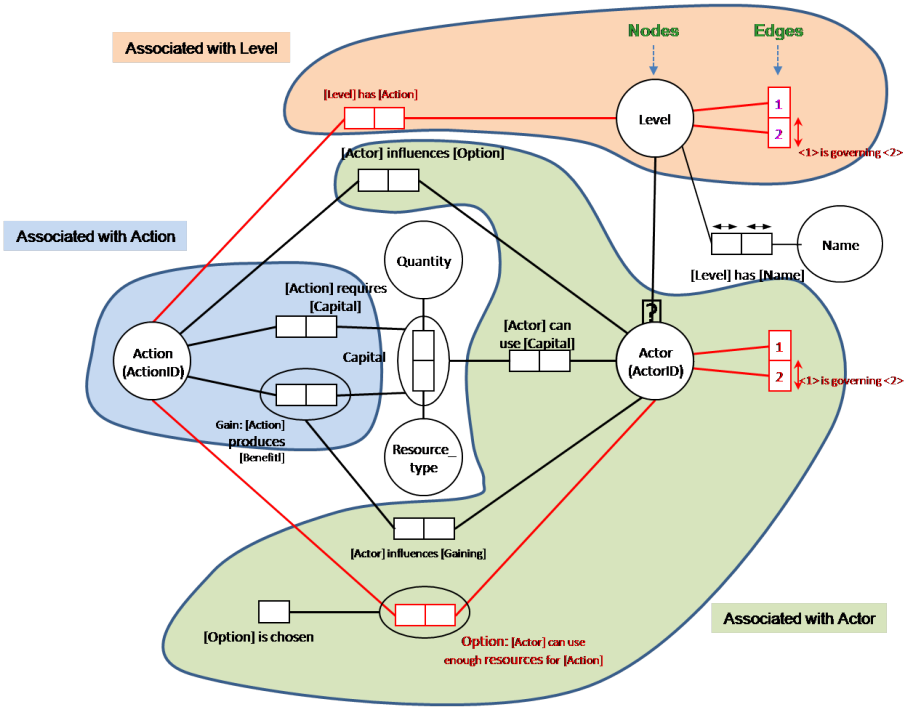

Figure 7. Grouping the fact types

Note that this transformation does not capture all constraints that can be derived from the AiC meta-model. We will come back to this in Section 7.5.

7.4 Describing the AiC Relational Scheme in Terms of XText

Next we will reformulate the relations from the AiC relational model in terms of XText. For each relation $R(I_1,\ldots,I_m,\ A_1,\ldots,A_n)$, where $I_1,\ldots,I_M$ are the (independent) identifying attributes for R-instances, and $A_1,\ldots,A_n$ the dependent attributes, we make an XText description by first describing the standard naming for R-instances, followed by (between curly brackets) a description for each of its properties $(A_1,\ldots,A_n)$:

```
R-rule returns R:
standard name for R involving independent attributes I1,...,Im
'{'
  for each dependent attribute Ai from A1,...,An
    a rule that describes how Ai is connected to this R-instance
'}'
```

Applying this general principle leads to the following AiC XText Meta-model:

1. According to relational **R1** scheme, the actor is identified by the ActorID, and may have as dependent attributes (1) a quantity of resource_type, (2) an influence a quantity of resource of action (3) a governing actor, (4) a list of option (5) a list of action preferences. These attributes are represented as Xtext rules.

```
ActorRule returns Actor:
  'Actor' (ActorID=EString) '(' iAttribute+=ID ')' '{'
     ('can-use''('quantity+=EInt','resource+=EString')')*
    ('influences''('action+=ActionRule','quantity+=EInt', 'resource_type+=EString'))?
    ('is governed by' ':' actor+=ActorRule)?
    'has-option' ':' (action+=ActionRule)*
    'has-chose' ':' (action+=ActionRule)?
  '}';
```

2. According to relational scheme **R2**, the action is identified by the ActionID, and may have as dependent attributes (1) requires a quantity of resource_type and (2)produces a quantity of resource_type. These two attributes are represented as Xtext rules.

```
ActionRule returns Action:
  'Action' (ActionID=EString) '(' (iAttribute+=ID)* ')' '{'
    ('requires''('quantity+=EInt','resource_type+=EString'))?
    ('produces''('quantity+=EInt','resource_type+=EString'))?
  '}';
```

3. According to relational scheme **R3** Level is identified by the LevelID and may have as dependent attributes (1) has a member actor and (2) is governed by actor. These two attributes are represented as Xtext rules.

```
LevelRule returns Level:
  'Level' (LevelID=EString)? '('(iAttribute+=ID)*')'
  '{'
     ( (actors+=ActorRule))*
    ('is-governed-by' ':' (actor+=ActorRule)*)?
  '}';
```

Thus the specification of our AiC language using grammar Xtext is as follows:

```
AIC_SPECIFICATION_RULE returns Model:
  'Model_AIC'
  '{'
    ('ModelVariable' ':' (Variable+=ID',')*)?
    (Level+=LevelRule)*
  '}';

EInt returns ecore::EInt:
    '-'? INT;

EString returns ecore::EString: STRING | ID;
```

Here the rule Variable is used for declaring global variables or parameters of the model and the rule EInt is used for generating the numerical type.

7.5 Adding Validation Rules

After the XText grammar construction constraints specified in the AiC meta-model are added to guarantee valid instantiations only. This can be done by the static validator AICValidator that is written in XTend. We verify (leaving out the AICValidator details):

- the name of an Actor (ActorID) is unique
- an actor or level cannot govern itself
- an actor cannot be governed by more than one other actor (strict hierarchy)
- the names of the options (ActionID) must be distinct.

8 The Design of TiC

Xtext not only provides a syntax-driven editor, it also generates a DSL editor. Our customized editor knows about the keywords of the corresponding language and where to place them, it knows about all the grammatical constructs, and it includes useful tools such as syntax coloring, code completion, and validation ((See Section 9.1, Figure 10)). It also allows to detect certain errors directly after the entry through the Quickfixes features (the possibility of renaming a variable to the duplicated name such as LevelID, ActorID and ActionID), thus it does not wait for the compilation like other conventional language editor to detect any errors. It provides an Outlines View to help visualize program structure.

8.1 Adding a Simulation Tool

In the previous sections, we have shown how a conceptual domain model (described in ORM) in general, and the AiC meta-model in particular, can be transformed via a relational model into an XText description, which is the basis of generating various tools such as a syntax driven editor and the extensions described in previous section. This forms the basis of the Tool-in-Context (TiC). In this section we go one step further, and show how TiC can develop a simulation tool for the application domain that was originally described by the conceptual model.

8.2 Transformating the AiC XText Model to Netlogo

In this section we describe how the XText grammar in general, and the AiC XText description in particular can be transformed to the NetLogo language and functions offered by NetLogo. NetLogo is a programming environment for modeling and simulating natural collective phenomena. It is well suited for modeling complex systems composed of hundreds or thousands of agents acting concurrently [71]. The set of transformation rules are defined in the TiC package that consists of two main classes:

1. the TranfoFunction.java class that contains the different functions of transformation and
2. the MIabmToNetlogo.java class that contains the main class that performs the transformation by calling the above functions.

The steps are:

1. Variable: The local and global variable of our language are transformed in NetLogo variable. Their transformation is performed by the two functions TransfoGlobalVar() and TransfoLocalVar().
2. Function: Concerning the functions, we define according to each language concepts functions for creating and transforming these concepts in NetLogo.

These functions are in the class TransfoFunction.java. Thus the concept Action and the dependant attributes of Actor and Action will be transformed as the functions performed by an agent.

3. Agent: The concepts of our language transformed into agent concerned Actor and Level. Furthermore, note that the agent link will use to create governance relationships between different levels and different classes of agents.
4. Transition and execution: This describes the transformation of our model as presented in Section 6. In the proposed model, the agent architecture is based on utility (Actor evaluates Profit with Score). Our agents are seeking to maximize their profit by evaluating gains. The observer will allow other agents to assess at every step the consequences of their actions at the time. The transformation is done according to the programming model presented in Section 6.

9 Validation: a Sample Session

As a first evaluation of Natural Model based Design in Context (NMDC) we apply it to the deforestation process, and show how the design helps to explain, simulate and analyze scenarios for more rational management of the Laf Forest Reserve located in the Northern Cameroon region [72].

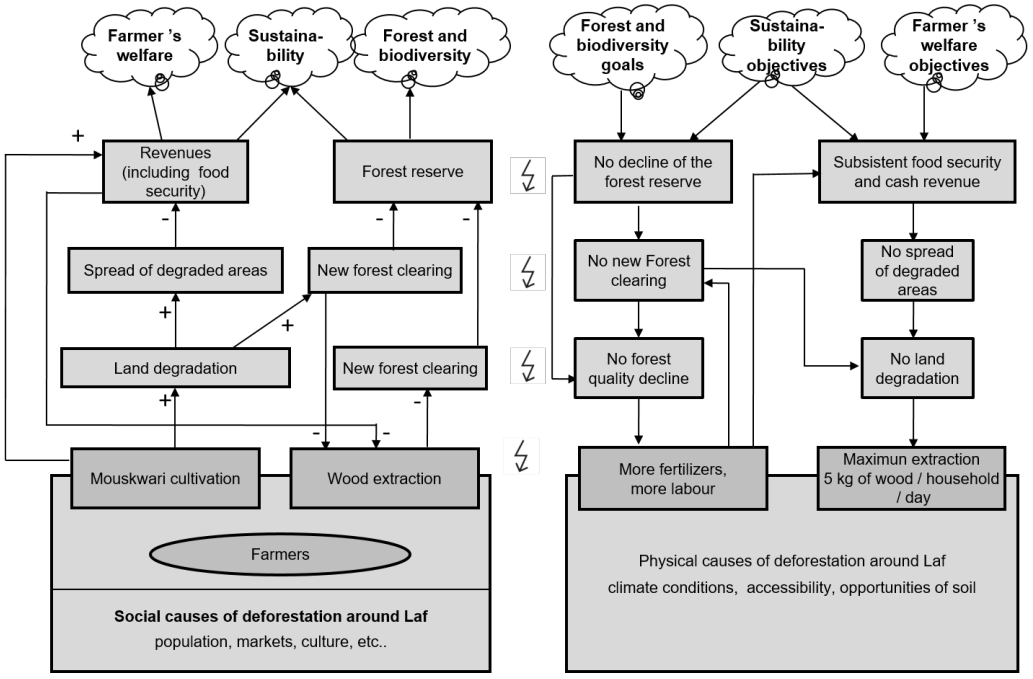

Figure 8. Analysis of the environmental problem in the Laf Forest Reserve [72]. *Left and right are facts and norms, respectively. Factual causal chains run primarily up towards the final variables (farmers' welfare etc.). In bold are the most important output variables of the model. The pluses indicate positive relations (more of A, then more of B); the minuses indicate the inverse*

Fotsing et al. [50] and Kameni et al. [23] have analyzed the deforestation process around the Laf Forest Reserve on the basis of remote data sensing and field interviews and proposed guidelines for planning and sustainable management. This is based on an analysis that takes into account the perception of the actors involved, their strategies and the impact of their decisions on the environment.

Figure 8 presents an analysis of the situation observed following the PiC framework (concretization of Figure 2 for the Laf Forest Reserve problem). This figure shows the effects of deforestation that result in the intrusion of populations in the protected area for agropastoral and timber harvesting activities. These practices lead to land saturation and reduced agricultural yields.

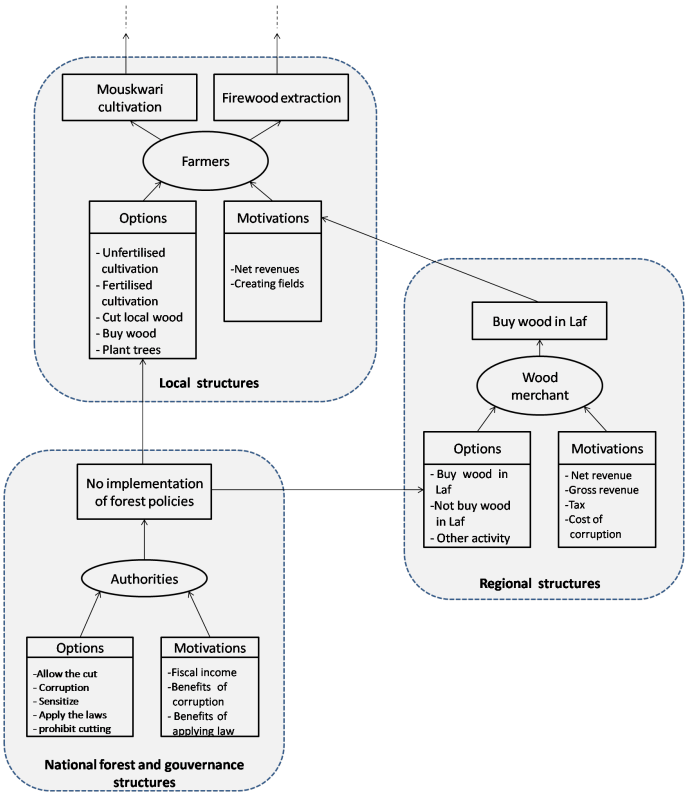

Figure 9. The social causation submodel of the environmental problem in Laf Forest Reserve [72]. *This figure shows modelled actors (farmers as primary actor, wood merchant as secondary actor and government as tertiary actor). Options and motivations are depicted as connected to structures (systems) on three scale levels: local, regional, national*

Figure 9 illustrates the social causes of the deforestation processes around the Laf Forest Reserve following the AiC sub-framework (concretization of Figure 3 for the Laf Forest Reserve problem). This figure also highlights the impact relations between the activities of (1) the wood merchant, (2) the authorities of the forest administration and (3) the options and motivations of farmers practicing deforestation. The main reasons for deforestation are (1) creating sorghum fields and the need to increase earning through (2) the sale of firewood. The choice of this latter option is favored by the high demand for wood and motivated by the cost of corruption (authorization uncontrolled cutting) and the importance of income derivatives from this activity. As recommended in [73], this figure highlights the scales of action between actors and the scales of effect as a crucial point for environmental problem design.

9.1 Populating the Laf Forest Reserve Process AiC Conceptual Scheme Using Natural Language

In this section we reformat the description of previous subsection into the structured semi-natural language format as prescribed by the AiC metamodel (see [72]). We illustrate here from some examples on the case of the Farmer how our natural language is used, the general and integral description of all the actors being in [72]. So, Farmers are involved in the following fact types:

- *fact type*: Actor can use Capital.
  Farmers start with some initial capital. This is described by:
  Farmer-1: Farmer can use $n1 - $n2 : '' of 'FCFA'.
  Farmer-2: Farmer can use $t1 - $t2 : '/week' of 'hours'.
  The actual values of $n1, $n2, $t1 and $t2 are simulation parameters that are set by the domain expert upon starting a simulation session. The actual

farmers then will get a start capital according to some normal distribution function as set by the domain expert.

- *derived fact type*: LET Option BE Action requiring less Capitals than Capitals that can be used by Actor.
  The population of this sentence type is derived from the actual action requirements and actor disposition. Assuming their initial capital suffices, the following would be derived:
  Farmer-3: Farmer can use enough capital for 'mouskwari cultivation'.
  Farmer-4: Farmer can use enough capital for 'firewood extraction'.
  Farmer-5: Farmer can use enough capital for 'unfertilized cultivation'.
  Farmer-6: Farmer can use enough capital for 'fertilized cultivation'.
  Farmer-7: Farmer can use enough capital for 'cut local wood'.
  Farmer-8: Farmer can use enough capital for 'buy wood'.
  Farmer-9: Farmer can use enough capital for 'plant trees'.
- *derived fact type*: ChosenOption IS Option with highest Gain. An option is proven beneficial, when (1) no other action can bring more profit and (2) no action with the same profit is higher appreciated by the actor. For the farmer, the following two options are chosen during the start of the simulation session. Both the government and the wood merchants will try to influence this choice.
  Farmer-10: Option ('firewood extraction', Farmer) is chosen.
  Farmer-11: Option ('mouskwari cultivation', Farmer) is chosen.
- *fact type*: Actor influences Option.
  From the interviews with the farmers we have to conclude:
  Farmer-12: Farmer influences 'firewood extraction'
  Farmer-13: Farmer influences 'mouskwari cultivation'
- *fact type*: Actor influences Gain.
  Farmer-14: Farmer influences Gain of 'firewood extraction'
  Farmer-15: Farmer influences Gain of 'mouskwari cultivation'

In practise this activity is supported by the syntax-directed editor that is provided by TiC (see Section 8 for more details). Once our editor is generated, we have developed a plugin. This plugin allows eclipse to recognize our language and thus facilitate the creation of a program in our language by a user (see Figure 10).

TiC can also check for inconsistencies, and offers other support facilities to help the domain expert to reformulate the natural sentences describing the application domain into the structured variants of the AiC metamodel. When the description has led to a consistent population of the AiC metamodel, the domain expert may let the simulation tool start a simulation session (see Section 8 for more details) and explore the effects of particular choices on the final norms, goals, and objectives.

Our approach is as follows. We group the actors according to their level. So we see 'farmer' as a group name for individuals (actors) who are acting as farmers. Basically we describe the properties of the actors as properties for the levels. The actual actors will act according to the rules described for their level. In the simulation (TiC), we will assume that the actual actor properties are statistically described. So for instance, we may indicate that the farmer is to be disposed of some money supply. Then the simulator will provide each actor of that level (each actual farmer) with an actual money supply according to a normal distribution with mean value the level money supply, and some variance to be specified.

Figure 11 and Figure 12 show the NetLogo code generated from our conceptual language concerning the Laf Forest Reserve deforestation process. This code consists of the functions and procedures such as the creation of agents, the initialization of the environment, the declaration of parameters and finally an implementation

```
ModelAIC
{
      ModelVariable: day_in_month, day-in-year, month-in-year, month-text, year,months,
      hour-in-day, day-in-week, weekdays-text, weekdays,  firewood_price, sorgho-price,
      fields-karal, C, sorgho-production_per_ha,  hour-begin1, hour-end1, hour-begin2,
       hour-end2, charge_from_one_ha, cost_daily_labor, number_of_hectar_work_per_day,
      flag_go_field, consum_kilo_per_day, coef price_kg_wood, test, Average_revenue,
      population_grow_rate, pop, pop1, karal_Area, wood_per_ha, limit_area, drap,

      Level Laf ()
      {
            Actor Farmer (number Amount Unit Resource_type)
                  { can-use(Amount,Resource_type)
                  has-option:unfertilized_cultivation fertilized_cultivation
                  Plant_trees Cut_local_wood buy_wood
                  has-chosen:
                                    Action Cutting_wood ()
                                    {requires(Amount,Unit,Resource_type)
                                     produces(Amount,Unit,Resource_type)
                                    }
                                    Action Mouskwari_cultivation ()
                                    {requires(Amount,Unit,Resource_type)
                                     produces(Amount,Unit,Resource_type)
                                    }
                    influences(cutting_Wood,Ipositive)
                    is-governed-by: Wood_Merchands Authotities
                  }

             Actor Wood_Merchands (number Amount Unit Resource_type)
                  {can-use(Amount,Resource_type)
                   has-option:no_buy_wood_in_Laf other_activity
                   has-chosen:
                                    Action Buy_wood ()
                                          {requires(Amount,Unit,Resource_type)
                                           produces(Amount,Unit,Resource_type)
                                          }
                   influences(buy_wood_in_Laf,Onegative)
                   is-governed-by:  Authotities
                  }
            Actor Authorities (number number Amount Unit Resource_type)
                  { can-use (Amount,Resource_type)
                    has-option:allow_the_cut corruption sensitize apply_the_laws
prohibit_cutting
                    no_implementation_of_forest_policyApply_law has-chosen :
                                     Action no_implementation_of_forest_policy()
                                          {requires(Amount,Unit,Resource_type)
                                           produces(Amount,Unit,Resource_type)
                                          }
                    influences ( no_implementation_of_forest_policy,Ipositive)
                    is-governed-by:

                  }
      }

}
```

Figure 10. Model of Laf Forest Reserve deforestation process obtained from the developed language domain

```
to can-use-of-Farmers[Amount Resource_type]
;to completed by user
end

to unfertilized_cultivation-Farmers[Amount  Unit  Resource_ty
let Amount Amount
let Unit Unit
let Resource_type Resource_type
;to completed by user
end

to fertilized_cultivation-Farmers[Amount  Unit  Resource_type
let Amount Amount
let Unit Unit
let Resource_type Resource_type
;to completed by user
end

to Plant_trees-Farmers[Amount  Unit  Resource_type]
let Amount Amount
let Unit Unit
let Resource_type Resource_type
;to completed by user
end

to Cut_local_wood-Farmers[Amount  Unit  Resource_type]
let Amount Amount
let Unit Unit
let Resource_type Resource_type
;to completed by user
end

to buy_wood-Farmers[Amount  Unit  Resource_type]
let Amount Amount
let Unit Unit
let Resource_type Resource_type
;to completed by user
end

to influences-Farmers[cutting_Wood Ipositive]
```

```
to create_Wood_Merchands[ number Amount Unit Resource_type number]
let number number
 let Amount Amount
 let Unit Unit
 let Resource_type Resource_type
 ceate-Wood_Merchandss number[    ]
;to completed by user
end

to create_Authorities[ number number Amount Unit Resource_type number]
let number number
 let number number
 let Amount Amount
 let Unit Unit
 let Resource_type Resource_type
 ceate-Authoritiess number[    ]
;to completed by user
end

to setup
create_Farmers  number Amount Unit Resource_type
create_Wood_Merchands  number Amount Unit Resource_type
create_Authorities  number number Amount Unit Resource_type
;to completed by user
end

to go
ask  Farmers[
;to completed by user
]
ask  Wood_Merchands[
;to completed by user
]
ask  Authorities[
;to completed by user
]
; Call the plot functions here if necessary

tick
end
```

Figure 11. Laf Forest Reserve generating NetLogo code for function declarations

scheme of agents decision. The function bodies are obtained by transforming the conceptual behavior descriptions (as described by the domain expert) into the concrete implementation language.

```
globals
[
 day-in-month day-in-year month-in-year month-text year
 months hour-in-day day-in-week weekdays-text weekdays
 firewood_price sorgho-price fields-karal C sorgho-production_per_ha
 hour-begin1 hour-end1 hour-begin2 hour-end2 charge_from_one_ha cost_daily_labor number_of_hectar_work_per_day
 flag_go_field consum_kilo_per_day coef price_kg_wood test Average_revenue population_grow_rate pop pop1 karal_Area wood_per_ha
 ]

breed[Levels Laf ]
Levels-own[ ]

breed[Farmerss Farmers]
Farmerss-own[number-Farmers Amount-Farmers Unit-Farmers Resource_type-Farmers ]
breed[Wood_Merchandss Wood_Merchands]
Wood_Merchandss-own[number-Wood_Merchands Amount-Wood_Merchands Unit-Wood_Merchands Resource_type-Wood_Merchands ]
breed[Authoritiess Authorities]
Authoritiess-own[number-Authorities number-Authorities Amount-Authorities Unit-Authorities Resource_type-Authorities ]
directed-link-breed[is-governed-by Govern]
```

Figure 12. Laf Forest Reserve Process generating NetLogo code for parameter declarations

As main functions, the TiC tool allows:

1. To modify sentence types: for instance to implement a new policy for the authorities.
2. To modify the distribution parameters for auto-generated object types (for example, the number of farmers, or the property that farmers dispose of).
3. To transform the domain concepts in the target language while keeping the logical construction of these concepts.
4. To generate a general structure of the model source code.

Moreover, we compared our approach with other existing metamodeling approaches according to the four criteria proposes by [74]: generating instances, editing metamodels (or models), user intervention, and error detection. Table 1 summarizes the result of this comparison.

Table 1. Comparison between the main approaches of meta-modeling

Author	Tool	Criteria for comparison								
		Generating instances			Editing (meta)models		User intervention		Error detection	
		Automatic	Manual	No	Yes	No	Edited	Parameter	Automatic	Manual
[13]	IRM4MLS/ SIMILAR		•		•		•			•
[75]	GEAMAS			•	•		•		•	
[42]	SPARK			•		•		•		•
Our Approach	NMDC/TiC	•			•		•		•	

We see that in addition to presenting identical properties with the existing approaches, our approach allows to generate an instance of the model that we wish to develop.

The experimentation of our approach in a case study of Laf Reserve Deforestation [72], clearly reveals the process taking into account the diversity of situations related to the process of actors decision-making at various levels, potentially from local to regional systems. Comparing this approach with [23], we can see that our approach takes better account of the determining concepts such as Level, resource and profit to clearly represent the process. It also allows a simplified representation of the concepts of the domain through tools and 'technology' language, thus facilitating communication in the development process.

According to [76], our transformation generates about 30% of the NetLogo code see (Figure 12 and Figure 11), these results are very conclusive compared [23].

10 Conclusion and Future Work

In this article we have divided the description of a general class of problems into the description at a higher (meta) level and the description at a lower (concrete) level (see Figure 1). At the meta level we find general concepts that are relevant for the class of problems at hand, at the concrete level we find the adaption to the actual problem case. In our case, the meta level is designed as a grammar for a semi-natural (controlled) language that allows the domain expert to reason about the application domain in a language close to the actual domain language used.

By using accepted models for the class of environmental problems in Section 4 we derived the conceptual meta model (intention I1, see Introduction). We showed how concrete models can be stepwise systematically transformed, while preserving data consistency, into a running application, taking benefit of available open source software projects (intention I2), hiding the various technical issues and challenges involved from the domain expert.

The TiC tool not only supports the construction of a simulation tool for concrete application domains, it also can be used as a syntax-directed editor that supports the domain expert in developing a concrete model (intention I3). This allows the domain expert to develop a model-driven application without the need for advance ICT competencies, as is intended by Natural Model based Design in Context (NMDC).

With respect to validation, we put most effort in formally describing the NMDC approach and its transformation implementation oriented models. To demonstrate the usefulness of the NMDC approach, we shortly described a sample session as a proof of concept (intention I4).

With respect to further research, the transformation of NMDC to NetLogo code is still in its preliminary state. The transformation may be elaborated in more detail in order to capture all language aspects of NetLogo, including options for optimization of the resulting code. Furthermore, the functionality of TiC may be extended. Some suggestions are:

- Definition of a graphical language interface based on the semantics of the AiC language, allowing the non-computer scientist to easily edit their program.
- Expand the model to refine the empirical assumptions in order to improve both the multi-level character and the model realism.

References

[1] N. Becu, P. Perez, A. Walker, O. Barreteau, and C. L. Page, "Agent based simulation of a small catchment water management in northern Thailand," *Ecological Modelling*, 170(1-3), pp. 319–331, December 2003.
Available: https://doi.org/10.1016/S0304-3800(03)00236-9

[2] O. Barreteau, F. Bousquet, C. Millier, and J. Weber, "Suitability of multi-agent simulations to study irrigated system viability: application to case studies in the senegal river valley," *Agricultural Systems*, 80(3), pp. 255–275, June 2004. Available: https://doi.org/10.1016/j.agsy.2003.07.005

[3] F. Bousquet, C. Le Page, I. Bakam, and A. Takforyan, "Multiagent simulations of hunting wild meat in a village in eastern Cameroon," *Ecological Modelling*, 138(1-3), pp. 331–346, March 2001.
Available: https://doi.org/10.1016/S0304-3800(00)00412-9

[4] C. Bernon, M.P. Gleizes, G. Picard, and P. Glize, "The adelfe methodology for an intranet system design," *In Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)*, 57, Toronto, Canada, 2002.

[5] J. Gomez-Sanz and R. Fuentes, "Agent-oriented system engineering with ingenias," *Fourth Iberoamerican Workshop on Multi-Agent Systems (Iberagents '02)*, 2002.

[6] M. Niazi and A. Hussain, "Agent-based computing from multi-agent systems to agent-based models: a visual survey," *Scientometrics*, 1(21), pp. 479–499, 2011. Available: https://doi.org/10.1007/s11192-011-0468-9

[7] G. Picard, *Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente*. PhD thesis, Université Paul Sabatier de Toulouse III - France, 2004.

[8] S. Azaiez, *Approche dirigée par les modèles pour le développement de systèmes multi-agents*. PhD thesis, Université de Savoie, France, 2008.

[9] J. Madejski, "Survey of the agent-based approach to intelligent manufacturing," *Journal of Achievements in Materials and Manufacturing Engineering*, 21(1), pp. 67–70, 2007.

[10] S. Maalal, M. Addou, W. Dachry, and B. Aghezzaf, "A generic agent-oriented model used fordesigning a collaborative information system," In *2012 IEEE International Conference on Complex Systems (ICCS)*, pp. 1–8, Nov 2012. Available: https://doi.org/10.1109/ICoCS.2012.6458561

[11] Ö. Gürcan, O. Dikenelli, and C. Bernon, "A generic testing framework for agent-based simulation models," *Journal of Simulation*, 7(3), pp. 183–201, 2013. Available: https://doi.org/10.1057/jos.2012.26

[12] P. Marcenac and S. Giroux, "Geamas: A generic architecture for agent-oriented simulations of complex processes," *Applied Intelligence*, 8(3), pp. 247–267, 1998. Available: https://doi.org/10.1023/A:1008220501261

[13] G. Morvan, A. Veremme, and D. Dupont, "IRM4MLS: the influence reaction model for multi-level simulation," In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pp. 16–27. Springer, 2010.
Available: https://doi.org/10.1007/978-3-642-18345-4_2

[14] S. Picault and P. Mathieu, "An interaction-oriented model for multi-scale simulation," *In Walsh, T., editor, Twenty-Second International Joint Conference on Artificial Intelligence. AAAI Press*, 1, pp. 332–337, 2011.

[15] A. Drogoul, E. Amouroux, P. Caillou, B. Gaudou, A. Grignard, N. Marilleau, P. Taillandier, M. Vavasseur, D.-A. Vo, and J.-D. Zucker, "Gama: A spatially explicit, multi-level, agent-based modeling and simulation platform," In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pp. 271–274. Springer, 2013. Available: https://doi.org/10.1007/978-3-642-38073-0_25

[16] J. Gil-Quijano, G. Hutzler, and T. Louail, "De la cellule biologique á la cellule urbaine: retour sur trois expériences de modélisation multi-échelles á base d'agents," *In Actes des 17èmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA'09).*, 2009.

[17] P. J.M. Frederiks and Th. P. van der Weide, "Information modeling: The process and the required competencies of its participants," *Data & Knowledge Engineering*, 58(1), pp. 4–20, 2006. Available: https://doi.org/10.1016/j.datak.2005.05.007

[18] K. Maaß, "What are modelling competencies?" *Zentralblatt für Didaktik der Mathematik*, 38(2), pp. 113–142, 2006.
Available: https://doi.org/10.1007/BF02655885

[19] A. Hevner, S. March, S. Park, and S. Ram, "Design science research in information systems," *Management Information Systems Quarterly*, 28(1), pp.75–105, March 2004.

[20] T. M. Salvatore and F. S. Gerald, "Design and natural science research on information technology," *Decision Support Systems*, 15(4), pp. 251 – 266, 1995.
Available: https://doi.org/10.1016/0167-9236(94)00041-2

[21] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, 24(3), pp. 45–77, 2007.
Available: https://doi.org/10.2753/MIS0742-1222240302

[22] M. M. Khamis and Th.P. van der Weide, "A Linguistic-Based Systematic Approach to Complex System Dynamics and its Application to E-government Introduction in Zanzibar," *Complex Systems Informatics and Modeling Quarterly*, Issue no. 11, pp. 85–111, 2017. Available: https://doi.org/10.7250/csimq.2017-11.05

[23] E. D. Kameni, E. Fotsing, and W. T. de Groot, "Passage d'un modèle acteur à un modèle multi-agent pour la gestion des ressources naturelles : Utilisation du méta-modèle d'acteur action-in-context," *In URED Journal, Revue Pluridisciplinaire de l'Université Gaston Berger de Saint Louis, Série des sciences exactes, Presses Universitaires de Saint Louis*, pp. 89–96, 2013. ISSN 08502161.

[24] J. C. Wooley and H. S. Lin, *Catalyzing Inquiry at the Interface of Computing and Biology.*, chapter 5. Washington (DC): National Academies Press (US), computational modeling and simulation as enablers for biological discovery edition, 2005.

[25] François Bousquet, *Des milieux, des poissons, des hommes: etude par simulations multi-agents. Le cas de la pêche dans le delta central du Niger.* PhD thesis, Université de Claude Bernard-Lyon 1, Edition Orstrom, 1994.

[26] P. Diello, J. E. Paturel, G. Mahé, B. Barbier, H. Karambiri, and E. Servat, "Méthodologie et application d'une démarche de modélisation hydrologique prenant en compte l'évolution des états de surface en milieu sahélien d'afrique de l'ouest," *IAHS-AISH publication*, pp. 691–697, 2006.

[27] A. Guengant, J.-M. Josselin, and Y. Rocaboy, "Densités et finances locales. difficultés de la modélisation," *Les Annales de la Recherche Urbaine*, 67(1), pp. 65–71, June 1995.
Available: https://doi.org/10.3406/aru.1995.1878

[28] M. W. Macy and R. Willer, "From factors to actors: Computational sociology and agent-based modeling," *Annual review of sociology*, pp. 143–166, 2002.
Available: https://doi.org/10.1146/annurev.soc.28.110601.141117

[29] E.D. Kameni and Th.P. van der Weide, "Intrusion detection in a trust-based recommendation system," *Int. J. Trust Management in Computing and Communications*, 4(1), In Press 2017.

[30] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *Proceedings of the National Academy of Sciences*, 99(suppl 3), pp. 7280–7287, 2002. Available: https://doi.org/10.1073/pnas.082080899

[31] F. Castiglione, "Agent based modeling," *Scholarpedia*, 1(10), p. 1562, 2006.

[32] M. Salgado and E. Marchione, "Multilevel and agent-based modelling in the analysis of differential school effectiveness," In *Proceedings of the Social Network and Multiagent Systems Symposium at the Artificial Intelligence and Simulation of Behaviour Convention*, 2011.

[33] D. Servat, E. Perrier, J.-P. Treuil, and A. Drogoul, "When agents emerge from agents : Introducing multi-scale view-points in multi-agent simulations," *In J.S Sichman, R. Conte, and N. Gilbert, editors, MABS. (LNCS) Springer,*, 1534, pp. 183–198, 1998.
Available: https://doi.org/10.1007/10692956_13

[34] G. Morvan, "Multi-level agent-based modeling - a literature survey," *Cornell University Library: Multiagent Systems, arXiv:1205.0561 [cs.MA]*, pp. 1–27, November 2013.

[35] A. Grignard, P. Taillandier, B. Gaudou, D. A. Vo, N. Q. Huynh, and A. Drogoul, "Gama 1.6: Advancing the art of complex agent-based modeling and simulation," In *International Conference on Principles and Practice of Multi-Agent Systems*, pp. 117–131. Springer, 2013. Available: https://doi.org/10.1007/978-3-642-44927-7_9

[36] D. David and R. Courdier, "See emergence as a metaknowledge, a way to reify emegent phenomena in multiagent simulations?" In *International Conference on Agents and Artificial Intelligence (ICAART'09)*, pp. 564–569. INSTICC Press, 2009. Available: https://doi.org/10.5220/0001803305640569

[37] A. Der Kiureghian and J. Song, "Multi-scale reliability analysis and updating of complex systems by use of linear programming," *In Reliability Engineering and System Safety*, 93(2), pp. 288–297, Febuary 2008.
Available: https://doi.org/10.1016/j.ress.2006.10.022

[38] V. Andasari, R. T. Roper, M. H. Swat, and M. A. J. Chaplain, "Integrating intracellular dynamics using compucell3d and bionetsolver: applications to multiscale modelling of cancer cell growth and invasion," *PloS one*, 7(3):e33726, 2012. Available: https://doi.org/10.1371/journal.pone.0033726

[39] N. Gaud, S. Galland, F. Gechter, V. Hilaire, and A. Koukam, "Holonic multilevel simulation of complex systems : Application to real-time pedestrians simulation in virtual urban environment," *Simulation Modelling Practice and Theory*, 16(1), pp. 1659–1676, 2008. Available: https://doi.org/10.1016/j.simpat.2008.08.015

[40] R. Conte, G. Andrighetto, M. Campennì, and M. Paolucci, "Emergent and immergent effects in complex social systems," In *Proceedings of AAAI Symposium, Social and Organizational Aspects of Intelligence*, pp. 8–11, 2007.

[41] B. Camus, J. Siebert, C. Bourjot, and V. Chevrier, "Modelisation multi-niveaux dans AA4MM," In Pierre Chevailler and Bruno Mermet, editors, *Journées Francophones sur les Systèmes Multi-Agents*, Système Multi-agents : Ouverture, autonomie et co-évolution, pp. 43–52, Honfleur, France, October 2012. Cépaduès.

[42] A. Solovyev, M. Mikheev, Z. L. Dutta-Moscato, C. Ziraldo, G. An, Y. Vodovotz, and Q. Mi, "Spark: A framework for multi-scale agent-based biomedical modeling," *International Journal of Agent Technologies and Systems*, 2(3), pp. 18–30, 2010. Available: https://doi.org/10.4018/jats.2010070102

[43] G. Blair and P. Grace, "Emergent middleware :tackling the interoperability problem," *Internet Computing, IEEE*, 16(1), pp. 78–82, 2012. Available:
https://doi.org/10.1109/MIC.2012.7

[44] S. Rottenberg, S. Leriche, C. Lecocq, and C. Taconet, "Vers une définition d'un systême réparti multi-échelle," *Ubimob '12 :8èmes journées francophones Mobilité et Ubiquité. Anglet, France. Cépaduès*, pp. 178–183.(hal–00724426), June 2012.

[45] A. H. M. ter Hofstede and Th.P. van der Weide, "Expressiveness in conceptual data modelling," *Data & Knowledge Engineering*, 10(1), pp. 65–100, February 1993. Available: https://doi.org/10.1016/0169-023X(93)90020-P

[46] A. H. M. ter Hofstede, H. A. Proper, and Th.P. van der Weide, "Formal definition of a conceptual language for the description and manipulation of information models," *Information Systems*, 18(7), pp. 489–523, October 1993. Available: https://doi.org/10.1016/0306-4379(93)90004-K

[47] T. A. Halpin and M. Curland, "Automated Verbalization for ORM2," In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, vol. 4278, pp. 1181–1190. Springer LNCS, Heidelberg, 2006. Available: https://doi.org/10.1007/11915072_21

[48] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press, 2004.

[49] G. Barthe, J. Forest, D. Pichardie, and V. Rusu, "Defining and reasoning about recursive functions: a practical tool for the coq proof assistant," In *International Symposium on Functional and Logic Programming*, pp. 114–129. Springer, 2006. Available: https://doi.org/10.1007/11737414_9

[50] E. Fotsing, M. Ntoupka, and A. Boubaoua, "Etat de la réserve forestière de laf et des zones riveraines: orientations d'aménagement et gestion de l'espace," In *Savanes africaines: des espaces en mutation, des acteurs face à de nouveaux défis. Actes du colloque, Garoua, Cameroun*, Cirad-Prasac, 2003.

[51] Jacques Ferber and Jean-François Perrot, *Les systèmes multi-agents: vers une intelligence collective*. InterEditions, 1995.

[52] W. T. de Groot, *Environmental Science Theory. Concepts and methods in a one-world, problem-oriented paradigm*. Elsevier Science Publishers, Amsterdam, 1992.

[53] M. Hobbes, *Figuring rural development. Concepts and cases of land use, sustainability and integrative indicators*. PhD thesis, CML Institute of Environmental Sciences, Leiden University, P.O. Box 9518, 2300 RA Leiden, the Netherlands, 2009.

[54] P. H. Verburg, W. T. de Groot, and A. J. Veldkamp, "Methodology for multi-scale land-use change modelling: Concepts and challenges," In *Global environmental change and land use*, pp. 17–51. Springer, 2003.
Available: https://doi.org/10.1007/978-94-017-0335-2_2

[55] S. March and G.F. Smith, "Design and natural science research on information technology," *Decis. Support Syst*, 15(4), pp. 251–266, 1995. Available:
https://doi.org/10.1016/0167-9236(94)00041-2

[56] A. R. Hevner, "A three cycle view of design science research," *Scandinavian journal of information systems*, 19(2), p. 4, 2007.

[57] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Autonomous agents and multi-agent systems*, 1(1), pp. 7–38, 1998. Available: https://doi.org/10.1023/A:1010090405266

[58] S. Russell and P. Norvig, "Artificial intelligence. A modern approach," *Artificial Intelligence. Prentice-Hall, Englewood Cliffs*, 25:27, 1995.

[59] A.S. Rao and M.P. Georgeff, "Modelling rational agents within a bdi-architecture," *In J. Allen,R. Fikes, and E. Sandewall, editors, Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann Publishers, San Mateo, CA*, 1991.

[60] I. Ajzen, "The theory of planned behavior," *Organizational behavior and human decision processes*, 50(2), pp. 179–211, 1991.
Available: https://doi.org/10.1016/0749-5978(91)90020-T

[61] J. Elster, *Nuts and bolts for the social sciences*. Cambridge Univ Press, 1989. Available: https://doi.org/10.1017/CBO9780511812255

[62] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.

[63] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM Computing Surveys*, 37(4), pp. 316–344, 2005. Available: https://doi.org/10.1145/1118890.1118892

[64] J. Mercadal, "Approche langage au développement logiciel : application au domaine des systèmes d'informatique ubiquitaire," *Langage de programmation [cs.PL]. Université des Sciences et Technologies - Bordeaux I*, p. 111, 2001.

[65] K. Ostermann, Domain specific languages. Retrieved Sept 2016.

[66] Xtext. Xtext website. https://eclipse.org/Xtext, Retrieved Sept 2016.

[67] M. Eysholdt and H. Behrens, "Xtext: implement your language faster than the quick and dirty way," In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pp. 307–309. ACM, 2010.
Available: https://doi.org/10.1145/1869542.1869625

[68] L. Bettini, *Implementing Domain-Specific Languages with Xtext and Xtend*. Packt Publishing Ltd, 2013.

[69] T. Halpin, "UML Data Models From An ORM Perspective: Part 1,". *Journal of Conceptual Modeling*, 1(1), 2003.

[70] T. A. Halpin, *Object-Role Modeling (ORM/NIAM), Handbook on Architectures of Information Systems*. Springer Berlin Heidelberg, 1998.
Available: https://doi.org/10.1007/3-540-26661-5_4

[71] U. Wilensky, Netlogo. http://ccl.northwestern.edu/netlogo/. Technical report, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.

[72] E. D. Kameni, T. P. van der Weide, and W. T. de Groot, "From conceptual model to implementation model Piloting a multi-level case study in Cameroon," working paper or preprint: https://hal.archives-ouvertes.fr/hal-01570397, June 2017.

[73] P. Dourish, "Hci and environmental sustainability: the politics of design and the design of politics," In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, pp. 1–10. ACM, 2010.
Available: https://doi.org/10.1145/1858171.1858173

[74] A. Ferdjoukh, *A Declarative Approach for Model Generation*. Theses, Université de Montpellier, October 2016.

[75] P. Marcenac, R. Courdier, S. Calderoni, and J. Soulie, "Towards an emergence machine for complex systems simulations," *In Tasks and Methods in Applied Artificial Intelligence, Springer*, 416, pp. 785–794, 1998.
Available: https://doi.org/10.1007/3-540-64574-8_465

[76] A. G. Kleppe, J. B. Warmer, and W. Bast, *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional, 2003.

A Appendix. Overview of ORM Concepts

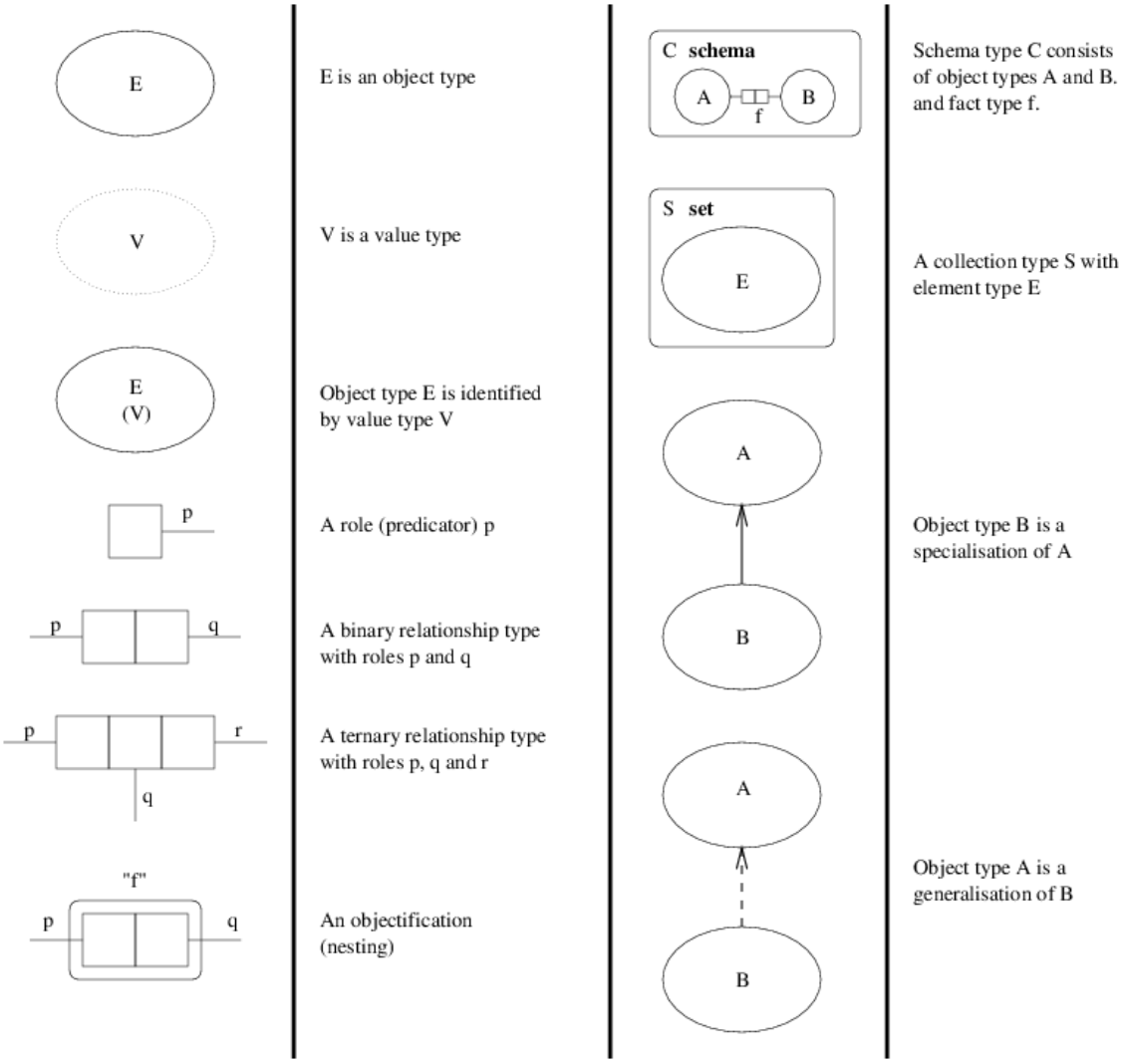

Figure 13.