Multiagent Simulation Model Design Strategies

Franziska Klügl
School of Science and Technology
Örebro University, Örebro, Sweden
Email: franziska.klugl@oru.se

***Abstract*—Model design is particularly challenging for multi-agent simulation models as the simulation paradigm does hardly impose constraints on it. This contribution systematically analyzes procedures for developing a multi-agent simulation model: iterative methods derived from principles, such as KISS or KIDS and methods focussing on the different design elements (agents, interaction, environment).**

I. Introduction

Methodological questions are more and more in the focus of research on agent-based simulation. This contribution deals with the phase in modeling that requires most experience and creativity from the modeler: model concept and design.

The development of a multi-agent simulation model seems to be an inherently intuitive way of modeling: Entities observable as active in the real system are captured as active entities – as agents – in the model. There seems to be no need for complex aggregation or abstraction necessary for formulating the model, nor a languages that requires deep mathematical skills must be used. Features of the model such as variable structure, heterogeneities, mixed local and global effects, ..., can just be formulated or generated from more or less simple agent behavior and interaction – just as observable in the original system or stated in the original ideas. Due to this intuitiveness and its general potential, it is often forgotten, that the development of a multi-agent simulation model leads to particular problems beyond general simulation engineering:

- Determining the appropriate level of detail is everything else but trivial. Actually, it is the basic design problem to determine what behavior shall be included or which factors ignored.
- Thinking in terms of agents can also be a problem when the modeler is used to other paradigms, such as process-oriented or macro modeling. Then an interaction-oriented model for generating the aggregate behavior instead of describing it, may be difficult to design.
- The general intuitiveness of the modeling leads to a tendency of ad-hoc development. This is supported by visual programming tools such as SeSAm [1]. Modelers immediately start implementing instead of thinking about an appropriate design beforehand.
- Necessary effort for understanding and analyzing the model and its overall behavior is underestimated. As in a multi-agent simulation the overall behavior is generated from the interactions and local behavior of agents, special effort has to be invested for excluding artifacts originating from minor bugs at the micro-level – in the agents design as well as implementation.
- A last issue is the difficult control of the included assumptions – due to the size of the model and the effort of developing, a systematic control of the assumptions taken while modeling has to be done. After modeling all assumptions can hardly be recapitulated and tested when they are not explicitly collected while modeling.

These are issues mostly in design of a multi-agent model. Clearly there are others, when considering all phases of the simulation study. Examples for additional problems are the need for validation when relevant data is missing, the difficulties of implementing concurrent processes or technical problems due to the size of the model and simulation. Nevertheless the design phase of a modeling and simulation study is the one the requires the most experience. This is the phase that is often coined as "art" [2].

In this contribution, we are focussing on the issues in model design and give a systematic view on different procedures. In the following we will first set the context of our endeavor by tackling process models for (multi-agent) simulation, as well as input from agent-oriented software design. Then we will give procedures for designing a model – one-shot in section III and iterative in section IV. The contribution ends with a discussion of the procedures and a short conclusion.

II. Methodologies and Processes for Developing Multi-Agent Simulation Models

A. *Simulation Study Life Cycle*

Phases of a systematic development of agent-based simulation models have been suggested by several researchers [3], [4], [5]. While focussing on integration of data or different roles of participants in the study, their suggestions for procedures are naturally quite similar to guidelines for general simulation study development developed by [6], [7] or [2]. The basic phases are initial works such as fixing the objective, making a profound analysis of the system or gathering necessary data and information. This is followed by making a model concept and elaborating it - this is basically model design. Model implementation and calibration are additional phases that are accompanied by analysis and validation phases. When the model is ready, the simulation study is completed by deployment runs and documentation and interpretation of the results. Clearly, bugs and deficiencies discovered in one phase may require re-working in previous phases. Figure 1 depicts

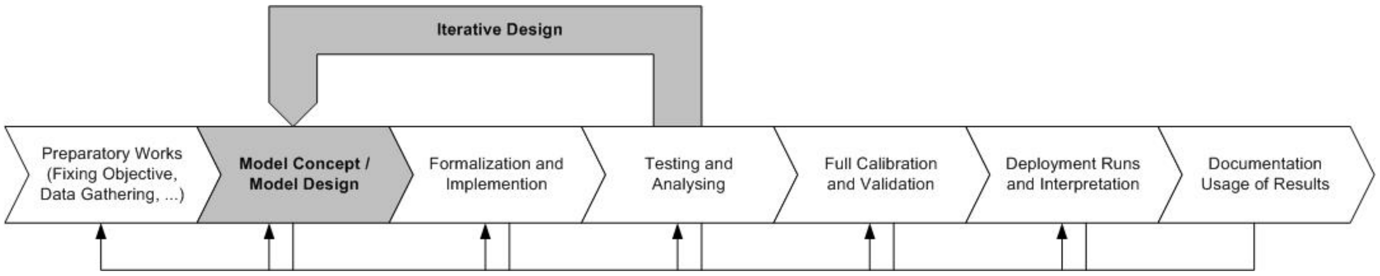

Iterative Design
Preparatory Works (Fixing Objective, Data Gathering, ...)
Model Concept / Model Design
Formalization and Implemention
Testing and Analysing
Full Calibration and Validation
Deployment Runs and Interpretation
Documentation Usage of Results

Fig. 1. Phases of an agent-based simulation study. Model design may also be iterative.

these basic phases for agent-based simulation development. The model design phase is highlighted.

It is quite unrealistic to assume that model design can be finished within one cycle. An iterative procedure for developing the model is appropriate not only for finding the right level of detail – starting with a initial model that is developed further until it is run-able and its dynamics analyzable. Based on the insight from this analysis, the model is further improved. Such an overall iterative proceeding is indicated in Figure 1.

This process is very abstract and does not use abstractions and techniques that are particular for agent-based simulation. This would be necessary for a more specific simulation life cycle. In contrast to agent-based software, there is no fully elaborated meta model describing necessary elements of an agent-based simulation in general (or even for a particular domain) in sufficiently concrete detail. Thus, a methodology comparable to the ones suggested for agent-based software engineering cannot be given yet. First attempts for a formal meta model for multi-agent simulation models have been made, such as in [8], but they are not on a level that makes them usable as a basis for the development of more specific life cycle models.

Before we will tackle model design and possible iterative improvement strategies, we will first have a short look on agent-oriented software engineering where the design of an agent system is central part of every methodology.

B. Agent-based Software Engineering

For the development of agent-based software, many methodologies have been suggested. Clearly, they are based on general software development phases such as requirements engineering/analysis, specification/design, etc. These methodologies rely on appropriate abstractions for representing and analyzing the agent-based software system. Development then happens by elaboration of these representations. Many methodologies have been proposed for guiding the development of agent-based software based on roles or organizational structures or focussing on specific agent concepts and architectures, such as BDI. Good collections can be found for example in [9]or [10]. Comparisons based on benchmarks [11] or features [12] aim at supporting the selection of the appropriate methodology for a particular problem. We would just like to mention [13] showing how modeling and simulation can be used for the design of self-organizing agent software. The suggested procedure has some similarities with the work described in this paper; yet here the focus lies in simulation applications with their specific characteristics. Our goal is to deal with a guideline how to modify the different elements of a model to finally build a valid simulation model.

Although there are profound differences between agent-based software and agent-based simulation – such as the correspondence to an original system ensured by validation or the inclusion of a simulated environment in addition to the simulation environment – the abstractions used for designing the software system may also be useful for designing a simulation model. This is especially the case when real-world organizational structures are used or folk psychology-based agent architectures are appropriate in the simulation model. In the following we want to tackle strategies for simulation model design from a simpler point of view:

III. Agent-based Simulation Model Design

In the following we will introduce and discuss three design strategies. They are derived systematically from an idea about elements of multi-agent simulation models: It consists of three basic components: agents model, environment model and model of the interactions. Each of them is used as a starting point or driving force for a design strategy; thus their usage can be seen as exclusive. However, it might be advisable to use different approaches one after the other in an iterative setting.

A. Agent-driven Model Design

"Agent-driven model design" is the first strategy. It corresponds to bottom-up design. The focus lies completely on the agents, their decision making and their behavior. Environmental and interaction models are added when needed in the agent design.

1) Basic Strategy: The following procedure can be defined:

1) *Agent observation and coarse behavior description:* The modeler observes the real-world agents and derives a coarse behavior description from its observations. Observations may be replaced by literature work or operationalization of hypothesis about agent behavior/decision making.
2) *Categorize agents and determine the location of heterogeneity:* The coarse behavior descriptions are analyzed for determining how many classes or types of agents are necessary for the model and to what level the agents

should be different. The location of heterogeneity may be on the level of parameter settings, different activities or even completely different classes.

3) *Decide about agent architecture:* Based on the coarse behavior description that treated the agents superficially, the modeler has to decide about the architecture of the agent. He may select a behavior-describing approach, for example perception-action rules with or without internal state representation or an architecture that is explicitly grounded on goals and on the configuration and selection of plans or even using plan generation from first principles or an elaborated cognitive model. This selection is depending on the complexity and flexibility of necessary agent behavior.
4) *Formalize agent behavior/goals:* The next step consists of filling in the actual behavior into the agent architecture. This is done by analyzing, elaborating and refining the coarse behavior description developed in the first steps.
5) *Add interactions and environmental aspects when needed:* Particular elements of the environmental model or structures of interaction are added when the agent behavior needs to include particular perceptions, messages or contains manipulations of environmental entities. As these aspects are added in some ad hoc manner, some re-factoring may be necessary, such as merging different environmental entity classes. Also some considerations about necessary heterogeneity are essential.
6) *Test whether necessary macro-phenomena are sufficiently reproduced:* At any time, model validity has to be tested when it is testable. We assume that the focus on the agents will lead to agent behavior near validity. But, a major effort will be testing wether the interplay of agents and their environment results in a valid macro-level behavior.

This procedure is a pure agent-driven bottom-up method for developing a multi-agent simulation. Aspects that relate agents to others or the environment are only important in so far they are influencing the agent behavior. Before we continue to discuss this procedure, we give a short example how the application of this method may look like.

2) Example: We used an existing simulation model[14] of bee recruitment: The basic objective was to find support for the hypotheses that the environmental structure influences the success of a recruitment mechanism in social insects.

Using an agent-driven approach, this model is build from the point of view of a honey bee. Individual bees are to be observed and literature has to be scanned for description of behavior and parameter values. In the second step, different categories of bees are identified: scouts, foragers or bees waiting/working in the hive. In this particular case these categories correspond more to activities than to different agent types as agents may instantaneously switch between activities. Therefore one may decide for a homogenous agent population with differences in the currently executed activity.

The following tasks of an agent bee are to be considered:

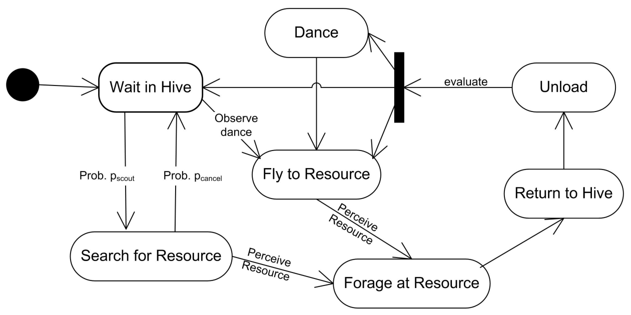


Fig. 2. Specification of example bee-agent behavior using activity graphs (adapted from [14]).

scouting, foraging, returning to the nest, unloading, wagggle dance (recruiting) and waiting. A simple behavior-descriptive architecture is appropriate. Due to the identified activities, an approach based on activity graphs representing the relations between activities is useful.

The next step is the formulation, i.e. specification and implementation of agent behavior. In figure 2 we depict the behavior of an agent.

Figure 2 does not indicate, when an interaction with the environment or other agents has to take place. When elaborating this graph, the modeler concurrently determines that there is a 2dimensional map for scouting and discovering resources. There must be resources that provide a nectar of certain quality. When returning to the hive, there must be some place to unload and some place to dance for recruitment of others. The storage area and the dance ground are only dealt with on a very abstract level. They may only be attributes of an object of the type "hive". For switching activities between waiting and foraging, the actual recruitment has to be modeled. Information is displayed by the dancer and perceived by conceptive other ants that based on the received information decide to scout or not.

This specification has to be implemented and tested as described above. As aggregate measures, the nectar input may be available for macro-level validation as well as counts how many bees are dancing, how many fly out, etc.

3) Discussion: The result of this process is a model that is fairly process-oriented on the agent level. For the example, the agent-driven design seems to work quite well. This might be due to the fact that interaction happens only indirectly via the environment or by displaying or broadcasting information. In the example, there is no direct message-based interaction. In principle the procedure should also work with direct peer-to-peer interactions. However, the strategy does not contain any perspective on the system-level containing for example protocol specifications. Such a bottom-up technique where the modeler takes over the perspective of an agent can be appropriately tested using participatory simulations where one agent is controlled by a human, the others are simulated.

A critical issue occurs when the overall validity is not reached. Then, this pure bottom-up technique will lead to trial and error procedures as during the development of the model no connection between macro- and micro level is established. Additionally this procedure does not help in finding the appropriate level of detail. Therefore it needs to be combined with an iterative procedure.

B. Interaction-driven Model Design

There are simulation domains where a focus on interactions is more appropriate than a purely agent-driven design. Examples may be models that focus on the performance of organizational design. In the following we want to introduce interaction-driven design.

1) Basic Process: One can formulate the following basic process for interaction-driven model design:

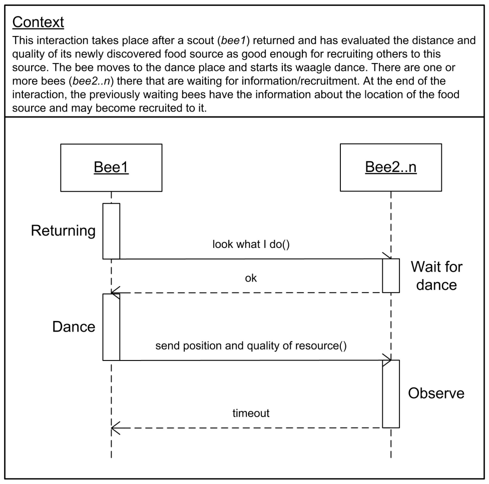
1) *Identify actors/entities and interactions among them:* Instead of observing individual real-world agents, the modeler is taking the birds' perspective and analyzes who is interacting and how.
2) *Coarse description of protocols* and their conditions, constraints, etc. The identified actors and interactions are refined to protocols going from general notions of interaction to atomic exchanges of information and manipulations of the environment.
3) *Derive agent behavior* for producing the interaction elements (messages, signals, actions...) and add environmental entities, such as shelter objects, to the model when needed for interaction. In this step something like a finite state machine based language can be used to specify agent interactions as state transitions.
4) *Implement agent behavior and test* whether the intended interactions and their outcome on the macro level are actually produced by the overall system. It must be also tested whether the agent level behavior is plausible or valid – depending on the available data.

In the interaction-driven approach, agents are basically seen as black boxes for producing the appropriate messages, information, etc. The general procedure may be further developed into some form organization-driven model design inspired by similar methods and methodologies from agent-oriented software engineering. Then, analysis of organizational structures forms the starting point for all system analysis as it forms the structural backbone of interactions.

2) Example: We use the same example as above for illustrating the approach and its differences to the agent-driven procedure.

As given above, we start by identifying the actors and their interactions. Again we have to tackle the problem of the location of necessary heterogeneity. Are the actors to be found on the level of scouts or foragers or as bees on a more homogeneous level? In our particular case we took bees as agents. Table I shows the basic interaction between the different types of entities.

The next step is the elaboration of the protocols. We suggest to use UML-based interaction diagrams for the initial description. In figure 3 we show the description of the recruitment protocol together with an (still) informal text about its context of appearance.


Context

This interaction takes place after a scout (*bee1*) returned and has evaluated the distance and quality of its newly discovered food source as good enough for recruiting others to this source. The bee moves to the dance place and starts its waagle dance. There are one or more bees (*bee2..n*) there that are waiting for information/recruitment. At the end of the interaction, the previously waiting bees have the information about the location of the food source and may become recruited to it.

Fig. 3. First specification of recruitment protocol together with context description. .

Formulating this interaction is not trivial as it is more like a broadcast (or stigmergic interaction). The message is send – respectively the information is displayed in the dance – to all agents that want to listen or observe it.

The next step would be to derive agent-behavior from the interaction diagrams. We suggested to transfer the interaction diagram into some finite state machine like representation. If we do that for every interaction the agents are participating in, a collection of finite state machines is generated. These single simple graphs have to be combined into a complete behavior description. This is done by first identifying similar states as interfaces which's unification merges the single state machines into some larger one. This larger one is probably too large as every interaction aspect is modeled explicitly. It might be possible to simplify some parts. Therefore, a refactoring might be necessary for bringing the overall state graph into some well-structured and minimal form.

Figure 4 shows the straight forward combination of a number of single-interaction state charts. For connecting we identified two times similar nodes – end nodes of one interactions, starting nodes of the other. In the graph such nodes are depicted as black nodes with second ring.

During the development of this behavior graph, we had to add state machines for interaction partners that are not given in figure 4. These interaction partners are the bee hive sending information that is used for the evaluation of the value of the load when the bee-agent is returning to the hive. Additionally, we omitted the interaction with the overall environment where the agent is requesting and receiving perception information and with resources with that it interacts while harvesting the nectar of this resource.

As a next step, it has to be determined what happens within the different states. This is straight forward, often consisting of waiting for incoming messages.

3) Discussion: Despite of the birds' eye perspective, this procedure did not result in an abstract and minimal behavior description. Nevertheless, the interaction and dependency of

Interactions	Bees	Resources	Nectar Storage
Bees	Recruitment	Harvest	Unloading
Resources	Localization	-	-
Nectar Storage	Status Information	-	-

TABLE I
INTERACTION TABLE FOR BEES AS AGENTS.

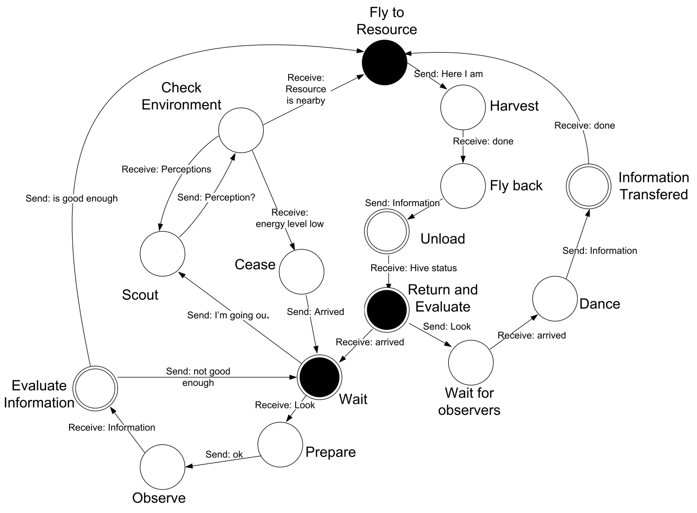


Fig. 4. Putting together single interaction state machines into a complete agent-level graph that describes all interactions of an agent. The black nodes were the initial start and end nodes of the two graphs. They are basically the interfaces between the two state machines .

behavior on information and material provided by others is treated explicitly, much more than in the agent-driven approach. This is actually an advantage of this procedure.

However, one can foresee problems when actual pro-active behavior has to be formulated. That means behavioral dynamics that are not triggered by external messages. Another drawback is, that also resources and other entities that are basically are no agents, have to be treated as agents as every interaction is formulated based on protocols. Although these are then just "passive" agents, they have to be practically tackled as agents for specifying the interactions.

Due to its focus on direct interaction, other forms of interaction may be hard to model, such as stigmergic interaction in form of broadcasted messages that are persistent in the environment decoupling sender and receiver. However, for simulated multi-agent systems with interaction that relies on message-based communication this design strategy seems to be appropriate.

C. Environment-driven Model Design

The third strategy we want to analyze is driven by a focus on the environment.

1) Basic Process: In analogy to the previously discussed design strategies, the starting point of the environment-driven design is an analysis of the environmental structure. Based on this, the agent interface and its behavior definition are determined. The steps are in particular:

1) *Identify relevant aspects* (global status, global dynamics/ local entities) of the part of the model that represents the environment.
2) *Determine the primitive actions of the agent and the reaction of the environmental entities to these.*
3) *Determine what information from the environment must be given to an agent* so that it can appropriately select and perform its actions.
4) *Decide on an agent architecture* that is apt to connect perceptions and actions of the agent appropriately for actually producing the agents behavior. Concurrently, the elements of the internal agent status are settled.
5) Use a *learning mechanism for determining the actual agent behavior*. A reward function for providing feedback to the agents actions has to be found. The reward schema also tackles questions such as when and how often to provide feedback to the agents, whether all agents learn based on a shared reward or individual reward is given to the agents.
6) *Implement the environmental model* including reward function if needed.
7) *Specify and implement the agents* behavior program or agent interfaces in combination with the chosen learning mechanism.
8) *Test and analyze the overall simulation* results and individual trajectories carefully for preventing artifacts that come from an improper environmental model or weak interfaces (perceivable situations and effects of primitive actions).

2) Example: For an illustration of this model design strategy we again use the recruitment scenario although the environmental complexity is not high enough to actually require such a procedure.

The start is made by formulating the environmental model. In this simple case the environment consists of a global world entity managing a 2-dimensional map, a central hive entity that is basically a container for the storage and a number of resource entities randomly distributed over the map, each with an individual nectar supply.

The initial environmental configuration in this case is the following: the hive is positioned in the center of the map. Each resource is located at a random position. Every resource object has an attribute called "nectar supply" that is initially set according to a normal distribution.

The next step is to design the perception capabilities and possible primitive actions – the interfaces of the bee agents. Without particular regard on what is actually needed in the behavior definitions, we can list the following perceptions:

- perceive nearby resource, its position respectively (if nearby)
- perceive existence of resource (from a certain distance)
- perceive capacity of resource (if nearby)
- perceive hive storage (if nearby)
- perceive position display (if at hive)
- perceive current nectar load

... and actions:

- Perform random search
- Fly towards perceived resource
- Fly towards hive
- Load nectar
- Unload nectar
- Display resource information
- Memorize perceived position

One can notice that with defining this interface, the modeler also determines the abstraction level of the behavior definition – the environmental model alone did not fully determine the level of abstraction.

The next step is to connect perceptions and action to produce actual agent behavior. In our case this could be done using a rule-based approach with rules defined by the modeler. The simple interface that already indicates that a rule-based approach – including some stochasticity in agents decisions for some very abstract treatment of internal motivation – might be sufficient.

We may suggest the following rules determining the agent behavior:

1) `if` hive-storage $<$ A `then` perform random search (with probability p_A)
2) `if not` at hive `and not` perception of resource `then` perform random search
3) `if` perception of resource `then` fly towards perceived resource
4) `if` at resource `then` memorize resource information
5) `if` at resource `then` load nectar with rate *load*
6) `if` nectar load $>$ B `then` fly towards hive
7) `if` at hive `and` nectar load $>$ B `then` unload nectar with rate *unload*
8) `if` at hive `and` resource information memorized `then` display resource information
9) `if not` at hive `and not` perception of resource `then` fly to hive (with probability p_{cancel}

This set of rules is quite small, but sufficient. There are some probabilities and thresholds to be set ideally based on available data.

While not necessary in the example, using a learning mechanism might be an appropriate solution for generating the behavior program based on the perceivable situations and primitive actions of the agents. A learning approach, e.g. based on classifier systems seems to be feasible at least in this application example.

The major question in this application example is when to give feedback as an information to the agent about its performance: Giving feedback after each step does not make sense: the random search without information is intentional. The agents shall not learn where the resources are positioned. Ideally, positive feedback shall be only given when the agent has accomplished to recruit other bees to a good resource or even more delayed, the reward to all agents could be proportional to the current influx to the overall hive storage.

3) Discussion: Again one can find potential problems considering the agents internal motivations for generating true pro-active behavior without external triggers. Such elements of complex agent-based simulation models are not well integrated into this design procedure.

Involving learning mechanisms forms a basis for research questions with evolutionary background. However, integrating agent learning into the model design makes the model susceptible for artifacts coming from incomplete, imprecise or not fully elaborated reward or fitness functions. Agents that adapt to an environmental model with flaws can never reliably reproduce an original system independent how good the learning mechanism or the rest of the model is. Nevertheless, it is not trivial to find appropriate feedback functions characterizing the goal of the agents' development.

Another risk of this environment-driven approach consists in the selection of the learning mechanism. It could easily happen that no appropriate learning mechanism exists that can be used without further research. It is not so difficult to reach the boundaries of what is possible using current state-of-art learning: involving co-adaptation, true multi-agent learning with more than a couple of agents, non-Markov settings, etc. Then the learning problem may be too hard for finding a mechanism that converges within a feasible time.

IV. Strategies for Iterative Model Development

The strategies introduced above are for designing an agent-based simulation model in one step. Especially for identifying the appropriate level of detail iterative procedures may be combined with these strategies.

A. KISS: "Keep It Simple, Stupid"

The well known KISS principle for simulation modeling means that the modeler should avoid unnecessary complex models, but keep the model as simple as possible for generating the appropriate behavior. Simple models contain less sophisticated assumptions, can be easier explained and understood. Simplicity also refers to the modeling and simulation paradigm used formulating and simulating the model.

The starting point of this procedure is the identification and description of phenomena of the original system that should be contained, respectively reproduced by the final model. Grimm and Railsback [15] call these basic modules of system data characteristics "pattern". Examples for such pattern are a certain statistical distribution of tree sizes in a forest, another pattern is then the spatial distribution of large trees.

```
1) Identify and describe the set of observable
   properties (statements)
   about the real system $S$.
2) Define a model $M_0$ that is apparently too
   simple for reproducing the system
   with all its properties
3) By calibration, determine the set $S_M$ of
   properties, that are reproduced
   by $M_0$.
4) M $\leftarrow M_0$
5) Repeat Until $S_M = S$
   a) M $\leftarrow$ modify model M for producing more
      elements in $S$ than in the last
      iteration.
   b) Calibrate M and determine $S_M$ as the set
      of properties reproduced by M.
```

When this algorithm stops, the result should be the simplest model that captures all phenomena identified at the beginning of the process.

However, it remains open how to modify and enlarge the model for producing the next model from the previous one. Sometimes also side-steps might be necessary removing one model component and adding an alternative one. Finally, it is not trivial to see how to modify a model best for producing any additional phenomena. In worst case, this might result in a try-and-error procedure. Nevertheless the documentation of every single model shall be elaborated that especially contains a list of taken assumptions.

B. KIDS: "Keep it Descriptive, Stupid"

In 2004, B. Edmonds and S. Moss published a plea against in their eyes over-simplified models in social science (see [16]); Their major argument was: "The point is that it is simply not appropriate to make simplifications *before* one knows what is relevant and what not." (italics in the original). They therefore suggest to initially construct a model with agent behavior that is understandable and directly deducible from the observed behavior, but not necessary simple.

The iterative algorithm based on this KIDS-principle based strategy can formulated as in the following.

```
1) Repeat until a valid model $M_s$ is constructed
    a) Define a model $M$ that contains all
       apparently relevant aspects of agent
       behavior.
    b) Identify all assumptions and make explicit
       all parameter in $M_i$.
    c) Execute a sensitivity analysis for all
       parameter of $M$ and eliminate all blocks
       of behavior that are controlled by a
       parameter without effect on
       the overall outcome. $M_s$ is the model $M$
       after sensitivity analysis
    d) Test $M_s$ for credibility and validity
```

At first sight, this procedure basically resembles the usual try-and-error strategy. It does not give any hint what to do if the model is not sufficiently valid in terms of aspects that are not reproduced; As it is less systematic than the KISS strategy, the KIDS strategy is more apt for experienced modelers that know with what model to begin and how to operate if the outputs are not as intended.

C. TAPAS: "Take a previous model, add something"

A third strategy for model design is pragmatic and focusses on reuse of models. In the agent-based simulation area, it has been coined by [17] without further discussing the term. It is related to the KIDS-based strategy but takes an existing model as starting point.

Reuse of models becomes the more inevitable the more expensive investments have been already made to produce the model. As construction and testing of an agent-based simulation is more expensive than for traditional analytical or simpler microscopic models, reusing a fully validated model should be especially interesting for agent-based simulations. Put into a more pseudo-code way of presentation, the TAPAS strategy might look like the following procedure.

```
1) Select an *appropriate* existing model $M$
2) if $M$ is not implemented, implement it and
   validate it using model alignment
   with respect to published data about $M$.
3) Add new, additional aspects to produce $M_{add}$
4) Test and Validate $M_{add}$,
   if sufficient, ready, else go back to 3 or - if
   necessary to 1.
```

Step 3 and 4 are similar to the KIDS methodology. The critical step is the selection (and existence) of the reusable model. A sufficient documentation of the original model is essential and can be seen as a major prerequisite for reuse [18].

There are also some perils: it is not known a priori whether the model with modification is minimal, valid, or possesses the intended properties. The advantages of having a starting point have to be traded off the effort that it means to adapt the given model to the new ideas. This is very risky when the model for reuse is not fully understood by people that want to reuse it.

Given appropriate tool support, also partial models, that means single agents, groups of agents or the complete environmental structure, may be used as a starting point for new model development.

D. Candidate-based Modeling

In his book about biological modeling in general Haefner [19] suggests a procedure for modeling and simulation in research. Basically, it consists of the construction of a set of alternative models. They might differ in parameter settings, but may also use different architectures, etc. Each of the model candidates is calibrated and evaluated by validation. The "best" model is selected and used as a basis for future research. This procedure is hardly about iterative changes, but involves the treatment of several models which has to be done when for example conflicting hypotheses have to be evaluated for possible rejection. The candidate-based strategy is generally applicable, but does not state how one may come to one candidate or from one candidate to another.

E. Discussion

In this section we surveyed iterative model design strategies. We did not tackle particular design strategies for one model, but general procedures how to proceed from a first prototype to the model ready for deployment. Which of these strategy is appropriate for a particular simulation study is depending on the personal style of the modeler, on his experience, on the application domain, on the available data, etc. The table II sums up and contrasts properties of these strategies. First, the appropriateness of the strategies for different types of multi-agent simulation models is estimated. A second block refers to directed-ness of the model and minimality of the outcome of the modeling process. The third block of rows compares aspects of necessary data availability and other properties of the modeling process - for example how well this procedure

Strategy	KISS	KIDS	TAPAS	Cand.-based
Apt for linear models	high	high	high	high
Apt for emergent phenomena	low	high	mid	high
for shared-environment actors	mid	high	mid	high
Objective-orientedness	high	mid	mid	mid
Resulting minimality	high	mid	low	mid
Share of try & error	low	mid	high	high
Empirical data requirement	mid	mid	low	high
Integration of knowledge	mid	high	high	mid
Communication support	mid	high	mid	mid
Modeling overhead	mid	mid	low	high
Expertise in macro models	high	low	low	low
Expertise in micro models	high	high	low	high
Required tool knowledge	high	mid	mid	high

TABLE II
COMPARISON OF ITERATIVE MODELING PROCEDURES

supports the communication of the current model status. The last block refers to whether the modeler needs expertise, whether he has also to tackle macroscopic relations within the model – usually expressed in complex formulas, or whether this iterative procedure is also apt for beginners in agent-based modeling. We assumed that tools are available that support each of the strategies.

Using table II a modeler may select a specific iterative strategy for finding the best model in accordance with features of the simulation problem. Selection may also rely on common sense heuristics: if a good previous model is accessible, then it is a good idea to reuse that model. If the modeler knows the system that has to be simulated very well, then the KIDS approach should be used, whereas large holes in the system knowledge advise more candidate-based or KISS modeling. All these iterative procedures may be combined with the particular design strategies identified above.

V. CONCLUSION

Model design is the phase in the development of a simulation model, that requires most experience and skills[2]. This is true for all forms of simulation modeling, in particular for the development of multi-agent simulation models. In this contribution we have identified and discussed different strategies for model design – one shot or iterative. The latter can be combined with the former for really developing a well designed model. Even, if the resulting model is not significantly "better" than a model developed without these strategies, their application has the great advantage that they support a systematic development and at least partially guide the development. Nevertheless, the different strategies have to be further tested and elaborated in various simulation studies with different degrees of necessary model complexity.

The vision behind our research is developing a methodology for successful development of multi-agent simulation studies. The idea is to develop a methodology similar to *System Dynamics* [20] that allows the development of difference equation models starting from causal loop graphs to systems of formulas in a guided and systematic way. In agent-based simulation, systematic model design has to be accompanied with equally thoughtful implementation, calibration, documentation and especially validation. We already tackled these phases in an insolated way that leaves a lot of research open for combining the different solutions. This – and the application and test of the identified design strategies – form the next steps in the future.

REFERENCES

[1] F. Klügl, "Sesam: Visual programming and participatory simulation for agent-based models," in *Multi-Agent Systems: Simulation and Applications*, A. M. Uhrmacher and D. Weyns, Eds. Taylor & Francis, 2009.
[2] R. E. Shannon, "Introduction to the art and science of simulation," in *Winter Simulation Conference 1998*, 1998, pp. 7–14.
[3] N. Gilberg and K. G. Troitzsch, *Simulation for the social scientist*, 2nd ed. Open University Press, 2005.
[4] A. Drogoul, D. Vanbergue, and T. Meurisse, "Multi-agent based simulation: Where are the agents?" in *MABS 2002*, 2002, pp. 1–15.
[5] M. Richiardi, R. Leombruni, N. Saam, and M. Sonnessa, "A common protocol for agent-based social simulation," *Journal of Artificial Societies and Social Simulation*, vol. 9, no. 1, 2006.
[6] A. M. Law, *Simulation Modeling and Analysis*, 4th ed. McGraw-Hill, 2007.
[7] O. Balci, "Validation, verification and testing techniques troughout the life cycle of a simulation study," *Annals of Operations Research*, vol. 53, pp. 121–173, 1994.
[8] F. Klügl, "Towards a formal framework for multi-agent simulation models," Institut für Informatik, Universität Würzburg, Tech. Rep. 412, 2007.
[9] B. Henderson-Sellers and P. Giorgini, Eds., *Agent-Oriented Methodologies*. IDEA Group Publishing, 2005.
[10] F. Bergenti, M.-P. Gleizes, and F. Zambonelli, Eds., *Methodologies and Software Engineering for Agent Ssytems: The Agent-oriented Software Engineering Handbook*, ser. Multiagent Systems, Artificial Societies and Simulated Organizations. Boston, London: Springer, 2004.
[11] G. Weiss and R. Jakob, Eds., *Agentenorientierte Softwareentwicklung*. Springer, 2004.
[12] Q.-N. N. Tran and G. C. Low, "Comparison of ten agent-oriented methodologies," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds. IDEA Group Publishing, 2005, ch. XII, pp. 341–367.
[13] C. Bernon, M.-P. Gleizes, and G. Picard, "Enhancing self-organising emergent systems design with simulation," in *Engineering Societies in the Agents World VII, 7th International Workshop, ESAW 2006, Dublin, Ireland, September 6-8, 2006 Revised Selected and Invited Papers*, ser. Lecture Notes in Computer Science, G. M. P. O'Hare, A. Ricci, M. J. O'Grady, and O. Dikenelli, Eds., vol. 4457. Springer, 2007, pp. 284–299.
[14] A. Dornhaus, F. Klügl, C. Oechslein, F. Puppe, and L. Chittka, "Benefits of recruitment in honey bees: effects of ecology and colony size in an individual-based model," *Behavioral Ecology*, vol. 17, no. 3, pp. 334–344, 2006.
[15] V. Grimm and S. F. Railsback, *Individual-Based Modeling and Ecology*. Princeton University Press, 2005.
[16] B. Edmonds and S. Moss, "From kiss to kids - an 'anti-simplistic' modelling approach," in *Multi-Agent Based Simulation*, ser. LNAI, P. D. et al., Ed., no. 3415. Springer, 2004, pp. 130–144.
[17] A. Pyka and G. Fagiolo, "Agent-based modelling: A methodology for neo-schumpeterian economics," Universität Augsburg, Volkswirtschaftliche Diskussionsreihe, Tech. Rep. 272, 2005.
[18] C. Triebig and F. Klügl, "Elements of a documentation framework for agent-based simulation models," *Cybernetics and Men*, accepted 2009.
[19] J. W. Haefner, *Modeling Biological Systems – Principles and Applications*, 2nd ed. New York: Springer, 2005.
[20] J. D. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Boston: McGraw Hill, 2000.