TOWARDS AUTOMATED SIMULATION OF MULTI AGENT BASED SYSTEMS

Ante Vilenica and Winfried Lamersdorf
Distributed Systems and Information Systems, Computer Science Department, University of Hamburg, Hamburg, Germany

Keywords: Multi agent based simulation, Optimization, Automatization, Declarative description language.

Abstract: The simulation of systems offers a viable approach to find the optimal configuration of a system with respect to time, costs or any other utility function. In order to speed up the development process and to relieve developers from doing cumbersome work that is related to the execution of simulation runs, i.e. doing the simulation management manually, it is desirable to have a framework that provides tools which perform the simulation management automatically. Therefore, this work addresses this issue and presents an approach that reduces the effort to manage simulations, i.e. it eases and automates the execution, observation, optimization and evaluation. The approach consists of a declarative simulation description language and a framework that is capable of automatically managing simulations. Thereby, this approach reduces the costs, i.e. with respect to time and money, to perform simulations. Furthermore, this work targets a special subset of simulations, i.e. multi agent based simulation, that has grown large attention in many areas of science as well as in commercial applications in the last decade. The applicability of the approach is proven by a case study called ”mining ore resources” that has been conducted.

1 INTRODUCTION

Multi-Agent Systems (MAS) are a well established approach to model and build complex distributed systems. Due to the inherent distribution and separation of functionality as well as the ability to act autonomous software agents are a suited paradigm to build applications that consist of many single entities that cooperate in order to achieve a certain goal. The purposeful development of MAS is challenged by the dynamics that these systems exhibit and it leads sometimes to emergent phenomena (Serugendo et al., 2006). Therefore, it is a real challenge for application developers to equip the vast number of parameters that agents may have with appropriate values in order to ensure an appropriate behaviour of the MAS application. In order to handle this challenge (Edmonds and Bryson, 2004) have proposed a *simulation-based* development process that validates parameter settings in order to understand the behaviour of the system.

Simulation itself has a long tradition in science and it is sometimes called as the ”the third way of doing science” (Macal and North, 2007, page 95) besides observation and experimentation (Kelly, 1998). Thereby, simulation is used for many reasons. Mostly, it is used to gather information about systems that are not accessible due to undesired perturbations. Further, it is applied when the behaviour of the system has a time scale that is too small or to large in order to be observed (Banks, 1998).

This work targets *multi-agent based simulation (MABS)* (Drogoul et al., 2002) which has progressively replaced simulation techniques like object-oriented (Troitzsch, 1997) or micro-simulation (Orcutt, 1957) in many areas. MABS is therefore mostly of interest for areas that benefit from the opportunity to model and simulate different types of individuals at the same time. In the last ten years MABS has been successfully used in sociology (Pietrula et al., 1998), chemistry (Resnick, 1995), physics (Schweitzer and Zimmermann, 2001), ecology (Huberman and Glance, 1993) and economy (Said et al., 2002). In economy, for instance, MABS has been used to gather information and knowledge about stock markets, self organising markets, trade networks, the management of supply chains and the behaviour of consumers. One prominent example of the link between MABS and economy is the *Eurace*[1] project that aims at simulating the whole european economic space (Groetker, 2009). Thereby, information taken

[1]http://www.eurace.org/

from MABS can be used to reduce costs, optimize the efficiency of plants or reduce the risk of failure of an investment to be done.

The work presented here aims at easing the conduction of MABS by managing them automatically. Therefore, a declarative description language for simulations is presented in connection with a framework that is capable of computing simulations automatically. This approach enables developers to run different simulations with different settings without the need to manually adjust the simulation after each run. Simulations can be conducted as a background process or even over night. This reduces the time and attention needed to run simulations for the developers. Furthermore, the approach presented here can also be used to search for local maxima (minima) or a global maximum (minimum) since it contains an optimization component.

This paper is structured as follows. The next section introduces the declarative simulation language whereas section 3 describes in detail the architecture of the framework that enables the automated execution of simulations. Section 4 presents a case study from the area of economy that has been conducted using the simulation description language and the framework. Related work is discussed in section 5 before section 6 gives a conclusion and talks about future work.

2 DESCRIBING SIMULATIONS USING A DECLARATIVE LANGUAGE

This section will present a declarative simulation language that is designed in order to describe all information related to the execution and evaluation of a simulation.

2.1 DSDL: Declarative Simulation Description Language

The aim of the declarative simulation description language (DSDL) is to offer application developers an easy to use but still powerful and flexible language to describe MABS that can be conducted automatically. Therefore, the benefit of automatically executing the simulation and evaluating the results has to be much higher than the effort needed to model the simulation with the description language. Figure 1 depicts the most important aspects of the simulation language on an abstract level whereas figure 2 shows a code listing using DSDL to define an automated simulation. DSDL itself has been defined using XML Schema. Therefore, it bases on a widespread standard that is supported by various software tools which again enable a convenient way to create such files. The next subsections will explain the elements of DSDL in detail.

2.1.1 Run Configuration

First of all, DSDL introduces the concept of ensembles and single experiments. A simulation consists of *i* ensembles that again consist of *j* single experiments with $i, j \in 1...n$. Thereby, one ensemble represents one possible setting of parameters which is simulated *j-times*. Such a hierarchy is especially needed for simulations that have a non-deterministic behaviour. Thus, in order to get a significant result several single experiments have to be conducted. The settings for the ensembles and single experiments are done in the *run configuration* part of DSDL. Also, this part handles two other important aspects: the termination condition for the experiments as well as the start time of the simulation.

Termination conditions can be either a time expression, i.e. the duration an experiment should run, or a semantic expression, i.e. denoting a criterion that has to be true in order to terminate the experiment. Semantic expressions can be formulated using the *Java Condition Language (JCL)*[2]. This language roughly supports expressions that can be formulated within an ”if” statement in Java.

The other aspect, i.e. the start time, works as a timer for the simulation. It allows to postpone the start of the simulation by specifying either a relative or absolute time. Such an option is especially interesting for simulations that need many resources and that should therefore be conducted at a time when the computer is mostly idle.

2.1.2 Data Observer

Data observers allow to specify those elements of the simulation that should be observed and that in conclusion help to determine the fitness of a certain parameter setting. Such observed elements are mostly certain agent types or objects in the environment. Data observers may also contain certain filters that determine whether all elements of a type should be observed or only certain ones. Therefore, the observation can be customized to the needs of every simulation setting. Also, it can be defined whether the observation should pull data from the observed elements periodically, i.e. every *n* seconds, or on change, i.e. a certain

[2] http://jadex-rules.informatik.uni-hamburg.de/xwiki/bin/view/Resources/Rule+Languages

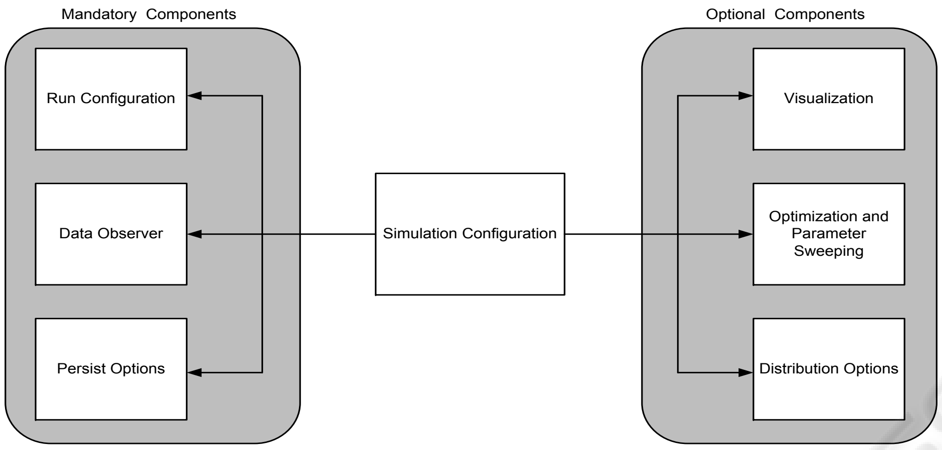


Figure 1: Most important components of DSDL.

trigger is activated that denotes an interesting event. Thus, the observed events have an id, value and timestamp. These raw data can then be further processed by other components, i.e. especially the optimization and parameter sweeping part as well as the visualization unit.

2.1.3 Optimization and Parameter Sweeping

The optimization of a MAS application with respect to a given utility function is the main motivation for automatically executing simulations. Thereby, various methods and algorithms can be applied while sweeping through a parameter space in order to find local optima (minima) or a global optimum (minimum).

Generally, algorithms for optimization can be divided into two subclasses: deterministic and probabilistic approaches. The first ones ”are most often used if a clear relation between the characteristics of the possible solutions and their utility for a given problem exists” (Weise, 2008, page 22). In contrast, the latter ones are more feasible if ”the relation between a solution candidate and its ’fitness’ are not so obvious or too complicated, or the dimensionality of the search space is very high” (Weise, 2008, page 22). Additionally, the same author has proposed a taxonomy of global optimization algorithms that shows the different approaches and techniques that are applied in this area.

Nevertheless, DSDL does support the use of any optimization algorithm due to its black box approach. Therefore, it relies on the Jadex XML data binding framework[3] which allows easily to specify the optimization library that will be loaded at runtime to perform the optimization. The developer specifies the needed input parameter settings in DSDL and those parameters a passed to the referenced optimization library which again returns the output parameter settings. Even more, DSDL does support parameter sweeping without the use of dedicated optimization methods. Thereby, it is inspired by the specification of batch parameters used in Repast Symphony[4]. It allows to define a parameter range and the step size that should be applied in order to sweep through this range. Also, a simple list of parameters can be specified that should be iterated through. Moreover, parameter sweeps can also be nested, i.e. the sweeping through ranges of different parameters can be easily combined.

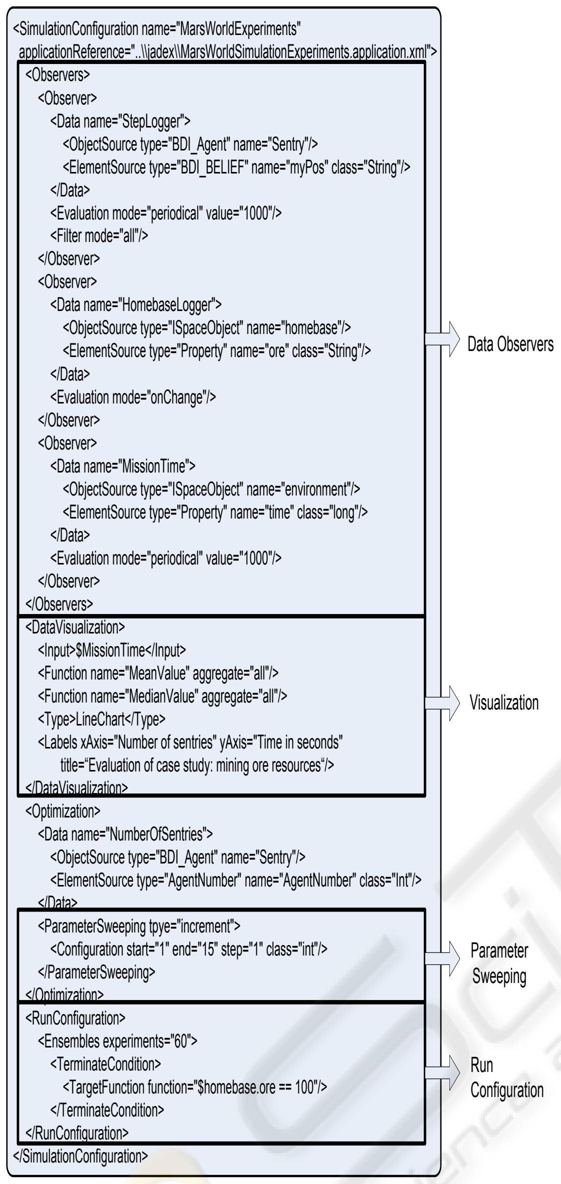
2.1.4 Visualization

The visualization section of DSDL is an optional component. It does not have to be specified in order to run a simulation but it can help to understand the results by not only having the pure facts but also a visual representation. Thereby, this component is closely related to the events received from the data observer. It can take these events and compute different (statistical) functions that support the evaluation of a certain parameter setting. Also, the visualization component can cake events from different observers and visualize them within the same chart. Thereby, different types of charts like pie, area, line, bar, histogram etc. are supported.

Now, that the features of DSDL have been presented the next section will introduce the architecture of a system that is capable of automatically processing simulations specified in this language.

[3] http://jadex-xml.informatik.uni-hamburg.de/xwiki/bin/view/About/Overview

[4] http://repast.sourceforge.net/docs/reference/SIM/Batch Parameters.html



```xml
<SimulationConfiguration name="MarsWorldExperiments"
 applicationReference="..\\jadex\\MarsWorldSimulationExperiments.application.xml">
 <Observers>
   <Observer>
     <Data name="StepLogger">
       <ObjectSource type="BDI_Agent" name="Sentry"/>
       <ElementSource type="BDI_BELIEF" name="myPos" class="String"/>
     </Data>
     <Evaluation mode="periodical" value="1000"/>
     <Filter mode="all"/>
   </Observer>
   <Observer>
     <Data name="HomebaseLogger">
       <ObjectSource type="ISpaceObject" name="homebase"/>
       <ElementSource type="Property" name="ore" class="String"/>
     </Data>
     <Evaluation mode="onChange"/>
   </Observer>
   <Observer>
     <Data name="MissionTime">
       <ObjectSource type="ISpaceObject" name="environment"/>
       <ElementSource type="Property" name="time" class="long"/>
     </Data>
     <Evaluation mode="periodical" value="1000"/>
   </Observer>
 </Observers>
 <DataVisualization>
   <Input>$MissionTime</Input>
   <Function name="MeanValue" aggregate="all"/>
   <Function name="MedianValue" aggregate="all"/>
   <Type>LineChart</Type>
   <Labels xAxis="Number of sentries" yAxis="Time in seconds"
       title="Evaluation of case study: mining ore resources"/>
 </DataVisualization>
 <Optimization>
   <Data name="NumberOfSentries">
     <ObjectSource type="BDI_Agent" name="Sentry"/>
     <ElementSource type="AgentNumber" name="AgentNumber" class="Int"/>
   </Data>
   <ParameterSweeping tpye="increment">
     <Configuration start="1" end="15" step="1" class="int"/>
   </ParameterSweeping>
 </Optimization>
 <RunConfiguration>
   <Ensembles experiments="60">
     <TerminateCondition>
       <TargetFunction function="$homebase.ore == 100"/>
     </TerminateCondition>
 </RunConfiguration>
</SimulationConfiguration>
```

Figure 2: Exemplarily simulation description using DSDL.

3 SYSTEM ARCHITECTURE OF THE FRAMEWORK

The basic idea of this framework is to relieve application developers from further work once the simulation has been modelled using DSDL. The simulation is automatically managed by the framework and the developer can continue working on other things until the simulation terminates.

Figure 3 depicts the architecture of the framework that is capable of automatically executing simulation experiments. It shows that the framework itself is composed as a MAS and that it consists mainly of two types of agents that handle the simulation.

The *master simulation agent* encapsulates the main functionality. It processes the declarative simulation description (cf. section 2) that contains all information and parameters that are needed to perform the simulation experiments and returns the results of the simulation to the developer. Thereby, the master simulation agent does not perform the single simulation experiments itself. Moreover, it delegates those to a client simulation agent that is in charge of performing a single experiment. Basically, the master simulation agent consists of four components: a *simulation run manager*, a *visualization component*,an *optimization component* and a *distribution component*.

Thereby, the simulation run manager is the most important component. It is instantiated on agent creation and it handles the progress of the whole simulation by delegating tasks to other subcomponents and agents. Figure 4 shows the workflow of the master simulation agent. After the agent has parsed the simulation description it creates a task for the execution of the first ensemble. The experiments of this ensemble are performed by the client simulation agents. Once the master agent has received these results it evaluates the observed data accordingly to the simulation description. Furthermore, the agent persists the raw data results in a database in order to make sure that they can be later reprocessed in a different manner than described in the description. Therefore, this approach makes sure that the single simulation results can be analyzed easily after the simulation has been terminated. As the master simulation agent has received all the results of the experiments of an ensemble it has to compute the target function of the simulation. This function may be defined as a simple parameter sweeping with predefined values or it may be a utility function. Depending on the result of the evaluation of the target function the simulation master agent terminates the simulation or it starts to prepare the execution of a new ensemble with new parameters. In the case of a simple parameter sweeping those parameters are predefined in the simulation description. In the case of a utility function the master simulation agent creates a task for the optimization component. This component is then in charge of computing the new values for the parameters. Finally, the master simulation agent creates a new ensemble and delegates again the execution of the single experiments to the client agents.

Therefore, it is obvious that the master simulation agent uses basically four components to perform the simulation and that the simulation run component manages the execution of the components. Due to the clear separation of functionality and clear interfaces

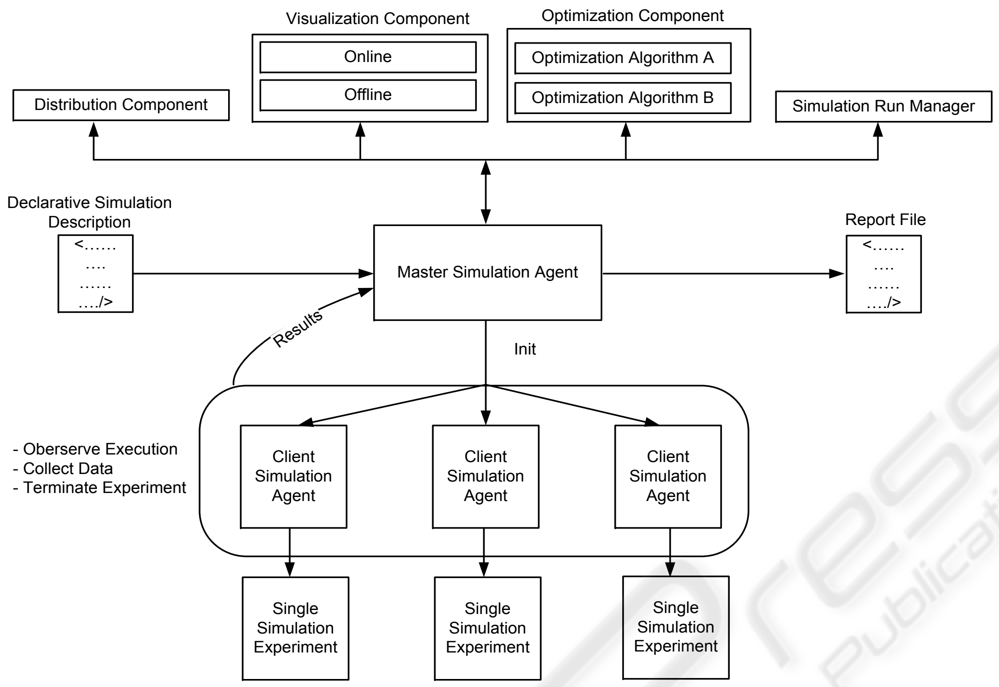


Figure 3: System architecture of the framework for automatically managing simulations.

components can be easily interchanged. This is especially important for the optimization component and the visualization component. Since there are many approaches towards optimization (cf. section 2.1.3) it is crucial to have an approach that supports the use of different mechanisms and libraries. Therefore, existing libraries like the Java Genetic Algorithms Package (JGAP)[5] can be used within the framework to compute the parameters for a new ensemble.

Also, the visualization of simulation results can be customized according to the requirements of a certain application domain. Results may be visualized on the local computer that is running the simulation by using existing libraries like JFreeChart[6]. On the other hand, the results may also be accessed remotely using a web server and browser to visualize them. Furthermore, the type of result visualization can not only be distinguished spatially but also temporarily, i.e. online or offline visualization. Whereas online visualization means access to data of currently running simulation experiments offline visualization denotes the processing of data from simulation runs that have terminated. Thus, the framework presented here, supports both approaches at the same time.

Finally, the master simulation agent contains a distribution component. The aim of this component is twofold: to speed up the simulation process and to use the computation power of several nodes in a computer network. By accessing shared computational resources across several nodes the simulation framework is even able to run massive agent based simulations (Yamamoto et al., 2007) which can not be performed on a single computer. On the other hand, the simulation framework can use the nodes to parallelize, e.g. to speed up, the simulation by running the single experiments of an ensemble at the same time on different computers.

The framework, described in this section, has been implemented using the Jadex Agent Framework (Braubach et al., 2005). The next section will describe a case study that has been done using this framework implementation.

4 CASE STUDY: MINING ORE RESOURCES

This section will present a case study called *mining ore ressources* that was conducted in order to prove the applicability of the framework introduced in the section before. First, the setting of the case study will be present. Second, the processing of the case study will be described as well as the evaluation of the results.

[5]http://jgap.sourceforge.net/

[6]http://www.jfree.org/jfreechart/

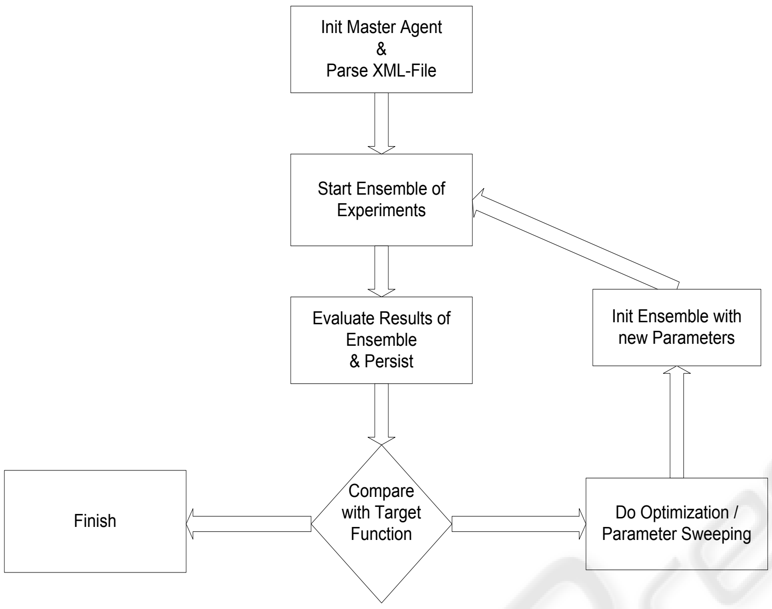

Init Master Agent
&
Parse XML-File
Start Ensemble of Experiments
Evaluate Results of Ensemble & Persist
Init Ensemble with new Parameters
Finish
Compare with Target Function
Do Optimization / Parameter Sweeping

Figure 4: Workflow of the master simulation agent.

4.1 The Setting

The setting of the case study is inspired by an example presented in (Ferber, 1995) and the basic implementation of this example was taken from the Jadex examples library. The case study deals with a MAS that was implemented in order to perform and coordinate the mining of ore resources on a non-explored field. As this setting was initially introduced in connection with the NASA (National Aeronautics and Space Administration)[7] it is sometimes also called *marsworld*. Nevertheless, this case study is a representative for a whole class of problems from the economy. Basically it targets the question: how many (different) resources do I need in order to accomplish a goal in time and budget? Therefore, this is a classic optimization function that targets a trade-off often found in the economy: time vs. money/costs.

Hence, the case study conducted targets the mining of ore resources (cf. figure 5). It consists of three types of components (i.e. agents): sentries, producers and carriers. These three components cooperate in order to achieve the goal, i.e. to find ore resources, inspect whether they can be explored and finally to bring the ore to the home base. At the beginning, all components explore the environment in order to find ore resources. If an agents finds a resource it reports the position to the sentry. The sentry is in charge of inspecting this resource in order to determine its capacity. If it can be exploited the sentry sends a message to the producer that is in charge of producing ore as much as the capacity permits. As the producer has finished its job it informs the carrier. Then, the carrier brings the ore to the home base. The agents have accomplished their goal when all available ore of the environment has been brought to the home base.

[7]http://www.nasa.gov/home/index.html

4.2 The Evaluation

Now, the afore described scenario has been taken to prove the benefit of DSDL and the framework for automated simulation. In fact, the framework has been taken in order to answer an important question that rises from the description of the scenario: What is the optimal number of sentries, producers and carriers for a given environment? As every component costs a certain amount of money it is obvious that the utility function has to take into account the acquisition cost and operation costs of the agents. Therefore, the framework has been used to investigate the relation between the number of operating agents and the time they need to complete the task.

Table 1 and figure 6 show the results from the simulation, that focuses on the relation between the number of sentries and the time needed to accomplish the goal, i.e. to collect all ore found in the environment. The results reveal that in the range from 1 to 10 sentries the time decreases continuously as the number of sentries rises. At the same time it can also be seen that this relationship does not hold for simulation experiments that contained more than 10 sentries. Two things can be concluded from these results. First, it seems that in order to minimize the time needed to

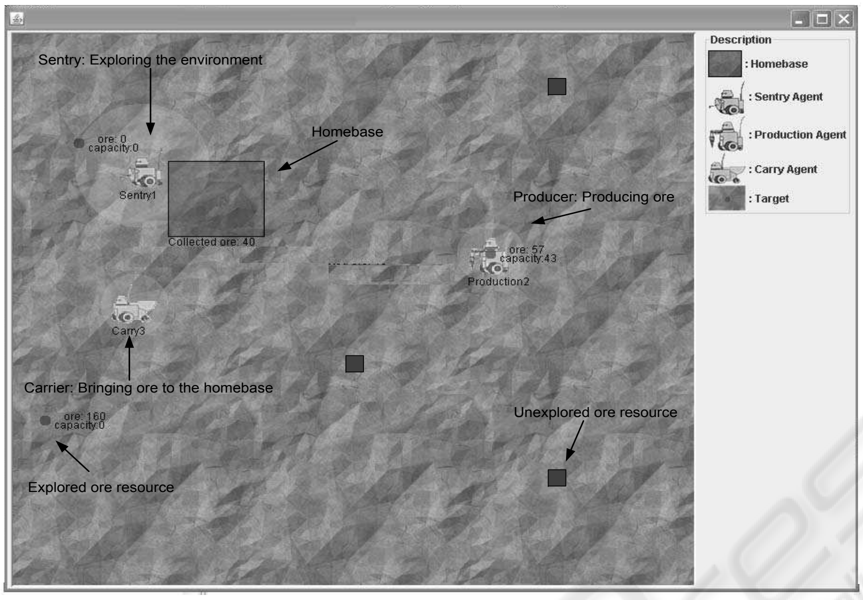


Figure 5: Screenshot depicting the scenario of the case study ”mining ore resources”.

accomplish the goal more than 10 sentries do not provide a significant speed up. Therefore, other components than the sentry seem to be a bottleneck which prevent a better performance. With respect to the setting of the case study these bottlenecks may be the carrier and the producer since they are not able to analyze and carry all the ore resources in time that have been discovered by the many sentries. Second, it can be concluded from the results of the settings with less than 10 sentries that there is a trade-off between time and money. Therefore, it depends on the costs of a sentry whether an additional sentry, that increases the costs but speeds up the time, pays off.

The case study has proven the applicability of the framework as well of DSDL. It shows the benefit of an automated simulation management that allows application developers easily to run experiments with different parameter settings without the need to manually edit, observe or evaluate the simulation runs. All this work is performed by the framework.

Table 1: Results of the evaluation of the relationship between the number of sentries and the time needed to collect all ore. Every setting was simulated sixty times.

Number of Sentries	Time: Mean Value (sec.)	Time: Median Value (sec.)
1	117.94	101.52
2	99.74	87.51
3	84.44	73.75
4	72.13	66.17
5	71.87	65.01
6	62.67	58.51
7	61.45	56.60
8	56.49	54.52
9	55.94	51.04
10	53.95	51.02
11	54.34	52.06
12	52.74	50.19
13	53.92	51.04
14	50.85	49.28
15	52.13	50.55

5 RELATED WORK

Most of related work is linked to the interference between simulation and optimization performed by MAS. Therefore, existing approaches use simulation runs in order to optimize the performance of an application with respect to some utility function. These approaches usually use the phrase *simulation based optimization* to characterize their aim and have basically an infrastructure as depicted in figure 7. Thereby, these types of approaches often focus on different optimization algorithms and implementations that use these mechanisms. Often, these approaches aim to solve problems of a certain application domain. Whereas (Fu, 2002) has in general investigated how different optimization techniques are used in the field of simulation (April et al., 2003; April et al., 2004) present a software tool that has been used to solve different real-world problems. These approaches use a mathematical model as input and do therefore not take advantage of the possibilities offered by simulation models that base on agents (Macal and North,

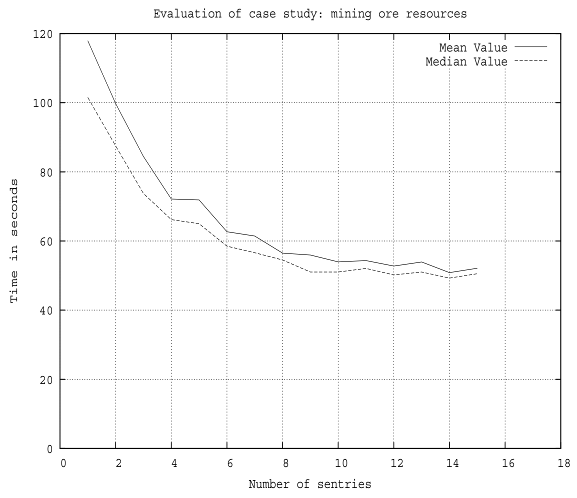


Figure 6: Evaluating the relation between the number of sentries and the time needed to collect all ore. Every setting was simulated sixty times.

2007). Also, they are designed as closed systems that can not be extended by users in order to customize the software according to their need.

(Brueckner and Van Dyke Parunak, 2003) have presented an approach that goes one step further than those described above and that is therefore not only limited to different optimization techniques. Moreover, this work describes an infrastructure that targets the evaluation of the properties of dynamic systems. Thereby, optimization is only one aspect among others like visualization or persistence. Unfortunately, the work of Brueckner et al. lacks of the description of a simulation modelling language as well as certain details of the infrastructure that is mostly described on an abstract level. Therefore, it is difficult to value this work with respect to reusability and extensibility.

In conclusion, existing work does not offer the connection of a declarative simulation description language and an open framework for automatically conducting simulations that is capable of more than just doing optimization. Nevertheless, existing optimization techniques and tools can be easily integrated into this open framework.

6 CONCLUSIONS AND FUTURE WORK

This work targets the development of complex and dynamic applications. It advocates, that simulation is a viable approach to gain information about the dynamic behaviour of such systems. Therefore, this work has presented an approach that automates the management of simulations and relieves thereby

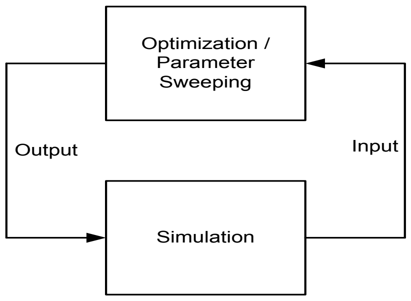


Figure 7: Simulation-based optimization.

the application developer from managing simulations manually.

The approach consists of a declarative simulation description language (DSDL) that offers the possibility to define all aspects related to the execution of a simulation. In detail, DSDL has components that define which data has to be observed, how data can be visualized and persisted. Moreover, DSDL can be used to define the sweep through a parameter space and this sweeping can be linked with an optimization mechanism that supports and speeds up the parameter sweeping process.

Additionally, this work has presented the architecture and implementation of a framework that is capable of automatically performing simulations that have been modelled using DSDL. This framework eases the developer from manually starting simulations with certain parameter settings and evaluating them. It supports the whole life cycle of a simulation with built-in components that can easily be extended and customized towards the needs of a certain application domain.

As a proof of concept, a case study has been conducted that shows the benefit of automatically managing simulations. The case study shows further, how DSDL and the framework can be used to easily optimize the setting of an application with respect to a certain utility function. Although this work targets the multi-agent based simulation the approach presented can be adopted and extended to other simulation techniques.

Future work will strive towards further extending the functionality of the framework. It is envisioned to add a component that supports the validation of system dynamics by offering the possibility to define hypotheses that target the causal structure of the system as introduced by (Sterman, 2000). Therefore, DSDL will be extended in order to be able to specify hypotheses and the simulation framework will be able to automatically validate these hypotheses while performing the simulations. Also, it is envisioned to built up a library that contains the results of all system validations that have been conducted. The aim of this library is to gain information about the qualitative behaviour of systems that exhibit a certain causal structure. Therefore, it will be able to predict the behaviour of a system by analyzing its causal structure and comparing it with the results from the library.

ACKNOWLEDGEMENTS

The authors would like to thank the Deutsche Forschungsgemeinschaft (DFG) for supporting this work in a project on ”Self-organisation based on decentralized Co-ordination in Distributed Systems” (SodekoVS). Furthermore, the authors would like to thank the Distributed Systems and Information Systems (VSIS) group at Hamburg University, particularly Lars Braubach and Alexander Pokahr, as well as the Multimedia Systems Laboratory (MMLab) at Hamburg University of Applied Sciences, particularly Wolfgang Renz and Jan Sudeikat, for inspiring discussion and encouragement.

REFERENCES

April, J., Better, M., Glover, F., and Kelly, J. (2004). New advances and applications for marrying simulation and optimization. In *WSC ’04: Proceedings of the 36th conference on Winter simulation*, pages 80–86. Winter Simulation Conference.

April, J., Glover, F., Kelly, J., and Laguna, M. (2003). Simulation-based optimization: practical introduction to simulation optimization. In *WSC ’03: Proceedings of the 35th conference on Winter simulation*, pages 71–78. Winter Simulation Conference.

Banks, J., editor (1998). *Handbook of Simulation. Principles, Methodology, Advances, Applications, and Practice*. Wiley.

Braubach, L., Pokahr, A., and Lamersdorf, W. (2005). Jadex: A bdi agent system combining middleware and reasoning. In *Software Agent-Based Applications, Platforms and Development Kits*, pages 143–168. Birkhaeuser-Verlag.

Brueckner, S. and Van Dyke Parunak, H. (2003). Resource-aware exploration of the emergent dynamics of simulated systems. In *AAMAS ’03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 781–788. ACM.

Drogoul, A., Vanbergue, D., and Meurisse, T. (2002). Multi-agent based simulation: Where are the agents? In *MABS*, pages 1–15. Springer.

Edmonds, B. and Bryson, J. (2004). The insufficiency of formal design methods - the necessity of an experimental approach for the understanding and control of complex mas. In *AAMAS ’04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 938–945. IEEE Computer Society.

Ferber, J. (1995). *Les systmes multi-agents. Vers une intelligence collective*. InterEditions.

Fu, M. (2002). Feature article: Optimization for simulation: Theory vs. practice. *INFORMS: Journal on Computing*, 14(3):192–215.

Groetker, R. (2009). Europa reloaded. *Technology Review*, (2):52–57.

Huberman, B. and Glance, N. (1993). Evolutionary games and computer simulations. *Proceedings of the National Academy of Sciences of the United States of America*, 90(16):7716–7718.

Kelly, K. (1998). The third culture. *Science*, 279(5353):992–993.

Macal, C. and North, M. (2007). Agent-based modeling and simulation: desktop abms. In *WSC ’07: Proceedings of the 39th conference on Winter simulation*, pages 95–106. IEEE Press.

Orcutt, G. (1957). A new type of socio-economic system. *The Review of Economics and Statistics*, 39(2):116–123.

Pietrula, M., Carley, K., and Gasser, L. (1998). *Simulating Organizations*. M.I.T. Press.

Resnick, M. (1995). *Turtles, Termites and Traffic Jams*. M.I.T. Press.

Said, L., Bouron, T., and Drogoul, A. (2002). Agent-based interaction analysis of consumer behavior. In *AAMAS ’02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 184–190. ACM.

Schweitzer, F. and Zimmermann, J. (2001). *Communication and self-organization in complex systems: A basic approach. In: Knowledge, complexity and innovation systems*, pages 275–296. Springer.

Serugendo, G., Gleizes, M., and Karageorgos, A. (2006). Self-organisation and emergence in mas: An overview. *Informatica (Slovenia)*, 30(1):45–54.

Sterman, J. (2000). *Business Dynamics - Systems Thinking and Modeling for a Complex World*. McGraw–Hill.

Troitzsch, K. G. (1997). Social simulation – origins, prospects, purposes. In Conte, R., Hegselmann, R., and Terna, P., editors, *Simulating Social Phenomena*, volume 456 of *Lecture Notes in Economics and Mathematical System*, pages 41–54. Springer.

Weise, T. (2008). Global optimization algorithms - theory and application. E-Book, available at: http://www.it-weise.de/projects/book.pdf, accessed 2010-04-12.

Yamamoto, G., Tai, H., and Mizuta, H. (2007). A platform for massive agent-based simulation and its evaluation. In *AAMAS ’07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–3. ACM.