Intelligent and Distributed Solving of Multiphysics Problems Coordinated by Software Agents
An Intelligent Approach for Decentralized Simulations

Desirée Vögeli[1], Sebastian Grabmaier[2], Matthias Jüttner[2], Michael Weyrich[1], Peter Göhner[1] and Wolfgang M. Rucker[2]
[1]*Institute of Industrial Automation and Software Engineering, University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany*
[2]*Institute for Theory of Electrical Engineering, University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany*

Keywords: Multiphysics Simulation, Software Agent, Distributed System, Case-based Reasoning.

Abstract: This paper presents an intelligent approach to support engineers with performing computational simulation of new developments and prototypes. With multiple interacting physical effects and large three dimensional models the choice of the right solution strategy is crucial for a correct solution and an acceptable calculation time. The presented multi-agent system can solve these simulation tasks using distributed heterogeneous computation resources with the objective to reduce the calculation time. An important factor for the criterion time is the choice of the linear solver. Here a case-based reasoning concept is introduced to improve the decisions in the multi-agent system. Allowing each agent to solve its problem part by using appropriate solution methods, a decentralized architecture with autonomous software agents is provided.

1 INTRODUCTION

Nowadays, our daily life becomes more and more affected by intelligent assistance systems. Industrial automation systems interact with each other to build individual products (Jazdi, 2014), kitchen devices are communicating with each other (Blasco et al., 2014), and cars are driving autonomously (Zhang et al., 2016). These things assist their users based on improving automation technology. Even if the things are built and run automatically, their development must be done by an engineer.

In times of workforce deficits in the engineering domain and an ever shorter time to market, engineers have to manage many tasks. When developing a new system, they have to check different approaches. Due to missing time decide must be based on their experiential knowledge. This results in missing better solutions. Therefore, they need support in form of software tools and assistance systems. Up to now there are only a few assistance systems to support the development processes of systems by creating new solutions on their own. One of these is Depias (Beyer et al., 2016). It allows an engineer to plan a logistic system, finding different possible compositions, and comparing them. Other concepts deliver methods to generate code from models (Mozumdar et al., 2008) or to automatically manage system requirements (Lambersky, 2012). None of these approaches can be used to develop a whole system yet, but they support partially the engineering process. Another part of the development process that is becoming more and more important are simulations (Clement et al., 2017). Simulations are used to analyze, optimize, and understand system behaviors. They are also able to assist at virtual commissioning and test verification.

An important type of simulations are so-called multiphysics simulations for which different tools exist. However most simulation tools are made for single physics problems. All these tools need a huge amount of computing power and time plus expertise. These multiphysics problems become even bigger due to the necessity of more detailed simulations. Hence the systems become more complex and the environment has to be considered (Tolk, 2016). That is why this paper shows an approach of how to solve

Vögeli, D., Grabmaier, S., Jüttner, M., Weyrich, M., Göhner, P. and Rucker, W.
Intelligent and Distributed Solving of Multiphysics Problems Coordinated by Software Agents - An Intelligent Approach for Decentralized Simulations.
DOI: 10.5220/0006590402000207
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 1*, pages 200-207
ISBN: 978-989-758-275-2

multiphysics problems in parallel using software agents inspired by the contract net protocol.

After the motivation and the introduction, chapter 2 describes multiphysics simulation and existing approaches for parallelization. Afterwards possibilities to decompose multiphysics problems are given. The last part of chapter 2 introduces the software agents and the concept to simulate the parts in parallel. The procedure of this agent-based system is shown in chapter 3 using an example. In chapter 4 a concept to improve the effectivity of the decisions within the multi-agent system using case-based reasoning is presented. Finally a conclusion and an outlook are given in chapter 5.

2 DECOMPOSITION OF MULTIPHYSICS PROBLEMS FOR PARALLEL SIMULATION

To simulate multiphysics problems in a decentralized way an approach to decompose the problem is introduced. Therefore an introduction to multiphysics simulation is given before the flexible and self-organizing agent concept is presented.

2.1 Multiphysics Simulation

A multiphysics simulation analyzes relations between different domains and their effects among themselves. This is getting more and more important, since model driven engineering and virtual commissioning are significant topics today (Boschert and Rosen, 2016). This simulations are used to predict and to understand the behavior of a system, often before building it. Thus a geometrical model of the system is built. The included physical laws are represented by partial differential equations or integral equations. To simulate physical effects, the model is discretized using e.g. the finite element method (FEM) or the boundary element method (BEM) (Gupta, 2002). Software tools such as COMSOL Multiphysics (Dickinson et al., 2014) are used. For solving the resulting linear equation system different solvers exists. Some use direct methods and others iterative ones. Direct methods mount the matrix by a sequence of calculation steps. In iterative methods, the results will be approximated until the convergence criterion is reached.

To estimate which configuration is better suited for a simulation task, experiential knowledge is needed. And even then it is often impossible to predict the best configuration before the simulation. There are solvers that deliver for nearly every task a simulation result and others that are good for only a few ones but do these simulation in just a fraction of time with less computational effort.

In order to meet the increasingly larger and more complex models, the computational effort can be spread over several distributed computers. Next to classical high performance computing on a supercomputer, there are other approaches to run simulations in parallel on distributed heterogeneous computing resources. In many approaches there is a static decomposition before the simulation or the simulation tool is made for exactly this use case. So each one can just be used for very specific problem setups. To solve different kinds of multiphysics simulations in parallel, there must be a more general attempt to decompose a multiphysics problem. A more general approach is presented in (Vázquez et al., 2016). The code Alya works on already discretized models. But here, every partial problem must have the same discretization and is calculated with the same code. As computing resource a super computer is used, but super computers are expensive.

However, if a model is updated in the presented approaches, for example by adding a new physical domain, everything must be simulated again and the configuration must be updated, too. That is why in this paper a more flexible approach is introduced, which uses idle and already existing personal computers (PC) and servers.

2.2 Decomposition of Multiphysics Problems

With the decomposition of multiphysics problems, there are two goals that should be reached. The first goal is to enable calculation resources with less calculation power to assist in the distributed simulation process, for example if they don’t have enough memory to load the problem or if they just have the ability to simulate parts like the heat transfer but not electromagnetic waves. The second goal is to use different methods to get a better result in shorter time. In some cases it can be an opportunity to simulate different areas or different physical effects with different methods and configurations (Buchau et al., 2003; Fetzer et al., 1999). Some approaches to decompose a problem for distributed calculation is shown in Figure 1.

To reduce the computational effort for one resource, there is the possibility to simulate the physical effects separately. This works on most weakly coupled problems. Another possibility to reduce the computational effort is to cut the model in parts. To reduce the calculation time various simulation configuration like the solver or the method (FEM, BEM) can be evaluated.

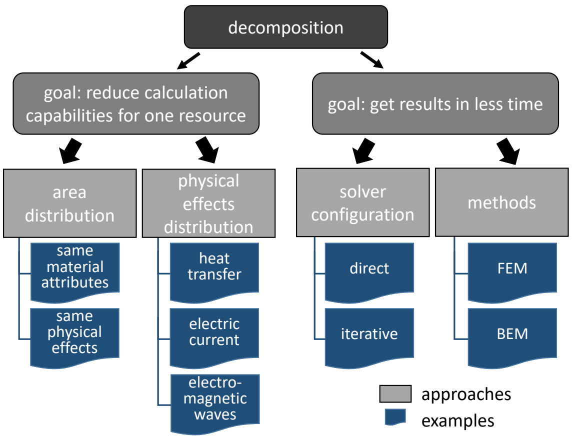

Figure 1: Objectives and approaches for decomposition of multiphysics simulation problems.

When no coupling between the physical effects exists, parallel calculation is trivial. Considering uni- or bidirectional coupling between the physics, information must be exchanged during the simulation. For this exchange an intelligent resource management is needed. One approach for this are agent-based systems.

2.3 Agent-based Concept for a Distributed System

To simulate the coupled problem parts on distributed calculation resources software agents are used to manage the solution process. Software agents are autonomous software units able to interact with the environment and communicate with each other (Jennings and Wooldridge, 1996). The ability to communicate is used to exchange information between coupled problem parts. Therefore, the agents use an agent communication language (ACL) based on the guidelines of the Foundation for Intelligent Physical Agents (FIPA, 2002). The ACL is based on the human speech act and allows to structure conversations. So, there is the possibility to add in addition to an identifier a performative to a message, such as INFORM, PROPOSE, AGREE, and many others.

To manage the solving process next to an agent management system (AMS) that cares for the message transport and the directory facilitator (DF) as yellow pages, three kinds of agents are developed: a coordination, a calculation and a report agent. The report agent that provides state news from all agents received by messages isn't regarded in the further paper because it isn't necessary for the calculations.

2.3.1 Coordination Agent

In the multi-agent system (MAS) there is one coordination agent that manages the simulation process. It gets the simulation task from the user and informs the other agents about the new model and the decomposition possibilities. It also supervises the distribution process among the agents and decides about the offers made by the calculation agents. With the coordination agent's graphical user interface, it is possible to stop simulations or to update the model. This can be used to integrate another physical effect.

2.3.2 Calculation Agent

The MAS contains many calculation agents. The calculation agents have the task to do the partial simulations. Thus they have the knowledge of the necessary steps to simulate a model. A calculation agent can have different abilities depending on the calculation resource on which it runs and the simulation software that is installed. Based on these abilities they bid for the part problems they want to simulate. Additional and redundant calculations are also supported to speed up the process. As soon as a calculation agent received the confirmation to a partial problem it starts the calculation using various software packages. To guarantee global convergence, which means the fulfillment of all constraints rising from couplings between partial problems, it exchanges results with other agents involved. Relevant results from other agents are considered in their own solutions. If the local convergence criterion is reached and no changes in the relevant results occur, the partial problem is terminated. When all partial problems are terminated, the global solution is completed. The calculation agents also provide interim results and the result to the user.

3 AN EXAMPLE ON DISTRIBUTED AGENT-BASED SIMULATION

In this chapter the solving procedure of the prototype is illustrated using a realistic example. The prototype implements the agents using the framework JADE (Java Agent Developing framework) (Bellifemine et al., 2007). Next to the presented agents in chapter 2.4, an AMS and a DF are used. As bidirectional coupled multiphysics problem microwave and dielectric heating is used for demonstration. Here a lossy dielectric medium (water) in a waveguide is considered. Hence the time harmonic electromagnetic wave equation (EMW) and the heat transfer (HT) are regarded. The underlying system of differential equations is given by

$$\nabla \times \nabla \times \vec{E} - k^2\vec{E} = i\omega\mu_0\vec{J}$$

$$-\nabla \cdot \kappa\nabla T = \sigma\left\|\vec{E}\right\|^2,$$

with the electric field $\vec{E}$, the electric current density $\vec{J}$, the temperature field T, the angular frequency ω, and the thermal conductivity κ. Taking temperature dependent conductivity σ and permittivity ϵ_r into account the wave number k reads

$$k^2 = -i\omega\mu_0\big(\sigma(T) + i\omega\epsilon_0\epsilon_r(T)\big),$$

With the vacuum permeability μ_0 and the vacuum permittivity ϵ_0.

To simulate the problem three calculation resources are used, listed in table 1. The data exchange between the calculation resources uses LAN technology. The access to the simulation tools is realized by a Java interface.

Table 1: Available calculation resources.

PC 1	**PC 2**	**PC 3**
16 GB RAM	8 GB RAM	8 GB RAM
Intel® Core™ i7-3520M CPU @ 2.90 GHz	Intel® Core™ i5 CPU 650 @ 3.20 GHz	Intel® Core™ i5 CPU 650 @ 3.20 GHz
tool for EMW	tool for EMW	tool for HT

There are three phases for the MAS that are explained in the next sub-sections: initialization, decomposition, and solving.

3.1 Initialization Phase

First, all agents must be started. Each PC runs one calculation agent (CalcA). PC 3 also runs the AMS, the DF, and the coordination agent (CoordA). After starting, the CoordA connects with the software tool. The CoordA registers at the DF and starts the GUI, where the user can load a model. The CalcAs CA1, CA2, and CA3 on PC 1, PC 2, and PC3 connect with the software tools. Then they check the performance of their resource and register at the DF. Last the CalcAs look for the CoordA using the DF and subscribe, so that they will be informed when a new model is available. After this initialization phase is done, the MAS waits for the user to load a model.

3.2 Decomposition Phase

The goal of the decomposition phase is to manage the decomposition of a model and to distribute the parts to the system resources. The necessary communication between the agents for the example based on a dynamic allocation negotiation is shown in Figure 2. As soon as the user loads a new model, the coordination agent analyses it on splitting possibilities and informs the CalcAs that have subscribed. Now all agents aim to find the best allocation of the problem parts. To get an overview of the different problem parts, the CalcAs request a price list from the CoordA that contains the parts and the current biddings. Next the CalcAs have to make their decision on which part to bid depending on their abilities. The better the calculation power of a resource, the more money the agent has to bid on parts. Also dynamic abilities like utilization are considered. The CoordA checks the biddings and updates the price list if there is a higher bidding. It also informs the formerly highest bidder and the new highest bidder about the new list. As soon as an agent has its problem part it starts calculating.

In this example the CA3 bids on the HT problem part. The other two CalcAs both bid on the EMW problem part. Therefore, they are overbidding each other as long as the one with the better resource gets the problem. However, the other agent is simulating the EMW part with other simulation configuration and tries to provide the result faster.

With this procedure the most promising variants are computed on the best calculation resources. The phase is finished when all problem parts are allocated. If a model becomes updated, the new part is allocated the same way to the CalcAs.

3.3 Solving Phase

During the solving phase the agents simulate their partial problems. To take care of dependencies, the agents cooperate with each other. So the first step of this phase is, that every calculation agent informs all others about the start of its task. Then the calculation agents analyze which other problem part depends on their own one. Next they subscribe to the calculation agents that simulate the related parts. Here, the agent with the HT problem part is interested in all results about the EMW and the agents with the EMW problem parts are interested in the HT part. After the subscription, the agents start the calculations.

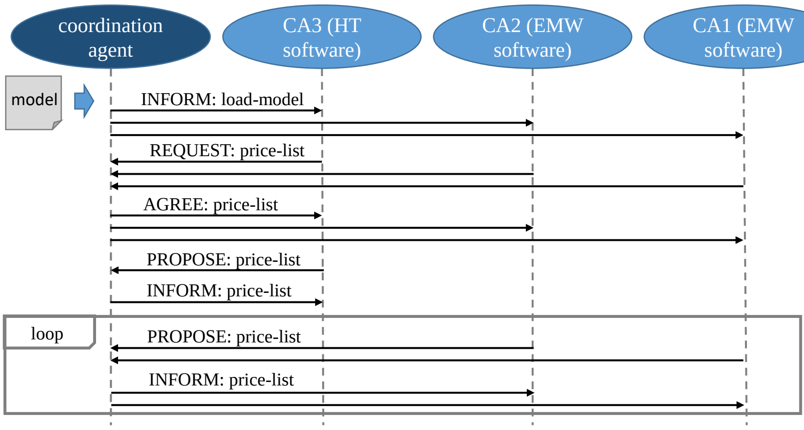

Figure 2: Communication during the decomposition phase.

Because of the lack of a stationary heat source in the HT model, CA3 finishes soon and waits for results from the EMW part to incorporate them as Joule's losses. When the first agent with the EMW problem finds a result, it checks if the tolerances are within the limits and if the result is already published. Here, it isn't published yet, so it sends a message with the result information to the other agents. The other agents check if their partial problem depends on the results. Here CA3 requests the solution, since the temperature depends on the heat sources. Before publishing results that correspond to the convergence criterion, the agents check, if they aren't already published. If so, they don't publish them and just wait for further results that may change something within their own simulation. Once the convergence criterion is reached and no agent is calculating anymore, the simulation process is finished.

During the calculations, the user is able to add further physical effects or couplings between them into the simulation. When doing so the existing solutions are reused as initial values.

3.4 The Solution

The graphic diagrams in figure 3 shows the simulation results for the electromagnetic wave and the heat transfer in the water. The solutions are presented by the agents, using a browser as graphic interface.

Besides the decomposition based on physical domains, the introduced MAS is able to decompose the simulation task based on geometries. It is also able to use different software tools and numerical methods to get a good result (Grabmaier et al., 2016; Jüttner et al., 2017). Further decomposition in space and time is possible.

The decision, which solver configuration has to be chosen if there is more than one resource capable to simulate it, is made by the calculation agents based on heuristics. Experienced engineers are able to choose a suitable solver configuration by regarding the model and the study. To imitate this human behavior, our MAS is expanded to learn from solved cases. Therefore a case-based reasoning approach is used.

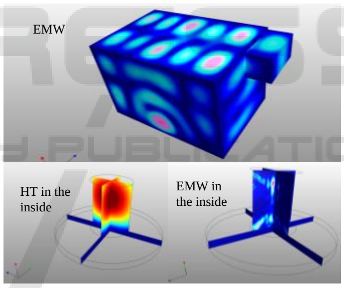

Figure 3: Simulation results of the calculation agents: Top electric field strength on the surface of the waveguide, Bottom left temperature inside the dielectric media, Bottom right Joules losses inside the dielectric media.

4 USING CASE-BASED REASONING TO IMPROVE THE EFFICIENCY

Case-based reasoning (de Mantaras, 2001) is used in the field of automated reasoning to use similar cases that are already solved to adapt the solution instead of building a new one from scratch. The case-based reasoning process is divided into four steps:

- Retrieve: the new task is compared with the already solved tasks to find a similar one

- Reuse: the solution of the similar already solved task is adapted for the current one
- Revise: the new solution based on the adaption is valuated
- Retain: the new case with its task and solution is saved to reuse it in the future

The advantage of this method compared with neural networks is, that there is no need for an initial training. Solutions can be created using the conventional way, if there is no similar case that can be adapted. Neural networks are set up by a training data set while case-based reasoning approaches learn from case to case due to the increasing database. Thus, the algorithm used to compare the similarity must be created by an expert. Here this method is used to optimize the choice of the linear solver. Thus, the cases are the simulation models with their studies and the used solver configuration.

4.1 Cases

To learn from cases, these must be saved in a data base. Therefore, a structure is needed. Figure 4 shows the structure of the cases for multiphysics simulations, consisting of model properties, approach parameters, and a grading.

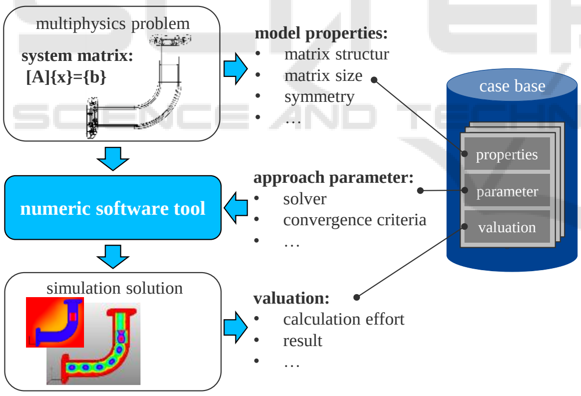

Figure 4: Structure of multiphysics simulation cases.

4.1.1 Model Properties for Comparison

To reuse information about the solving process from already solved cases, there must be a possibility to compare them. This is made by the model properties. An important property is the discipline, like HT or EMW in the previous example. Considering EMW simulation many properties are evaluated, shown in table 2. The parameters are grouped in boundary conditions (BC), material properties (M), discretization and geometry (G). The parameters are represented by booleans, by integers or floats, and others that contain lists. All information are a priori available using the model description.

Table 2: Model properties from electromagnetic waves problems.

	property name	type
BC	hasPortIntegral	boolean
	hasPerfectlyMatchedLayers	boolean
	isHermitian	boolean
G	maxGeometricalDistance	float
	minGeometricalDistance	float
discretization	shapeOrder	integer
	minSizeOfElements	float
	maxSizeOfElements	float
	minSkewness	float
	averageSkewness	float
	minQualityOfElementAngle	float
	averageQualityOfElementAngle	float
	averageElementsPerPenetrationDepth	float
	minElementsPerPenetrationDepth	float
	numberOfDegreeOfFredoms	integer
M	maxConductFact	float
	waveNumber	array
	maxImagWaveNumber	float

4.1.2 Approaches

To reuse the simulation configuration of already calculated models, the information about how this old cases were simulated must be saved. Here, the used method, the software tool, and the solver configuration are stored. The solvers have different advantages and disadvantages, so there isn't one that is the best in any case (Meister, 2015). Some are slow but solve the task in nearly any case, some are fast but need much memory and some cannot even guarantee to always find a solution. For the iterative applied solvers there is also the choice of a suitable preconditioner. The considered solver possibilities for the electromagnetic wave physic are shown in table 3.

Table 3: Considered solver configurations for simulation.

	Solver
direct	MUMPS
	Pardiso
	Spooles
iterative	BiCGStab + left preconditioning
	BiCGStab + right preconditioning
	GMRES + left preconditioning
	GMRES + right preconditioning
	preconditioned CG

Since the solver choice doesn't depend only on the model properties but also on the computation abilities of the resource, this is also considered and saved together with the cases and their evaluations.

4.1.3 Result Evaluation

The evaluation of a case must be done after the simulation is finished. Even bad tries must be saved, so the configurations won't be used again on a similar problem. The most important evaluation criterion is if the simulation was solved or not. Other criteria like the computation time must be compared relative to the computation abilities, like the random access memory of the resource. The best evaluation can be obtained by comparing different solver configurations and their results.

4.2 Similarity to Other Cases

Often examples on case-based reasoning are using only one attribute to compare cases. For the solver configuration in simulations, cases are more complex and can't be described or compared by one parameter. As described in chapter 4.1.1 the properties additionally contain different kind of parameters. There are parameters that have to match exactly and parameters that should be in the same range to use the same solver configuration. To measure the similarity between a new case and the cases saved in the database, the algorithm must take this into account. A common function to name the similarity sim between two cases c1 and c2 with n different parameter and with the weights w_i is given by

$$\mathrm{sim}(c_1, c_2) = \frac{\sum_{i=1}^{n} w_i \mathrm{sim}_i(c_{1i}, c_{2i})}{\sum_{i=1}^{n} w_i}.$$

sim_i is the similarity between the cases in one parameter. So multi-conditional cases are compared considering different kind of parameters. To create such an algorithm with appropriate weights a set of data is analyzed by an expert.

Due to the decentralized architecture of the MAS, the data is distributed and must be collected as well as exchanged. This is done by our incooperation concept.

4.3 Incooperation Concept

The cases and the database must be integrated to the existing MAS. Thus, the database is cared for by a new agent, the case base agent. It manages the old cases and is able to present similar cases for received properties. Thus a message with the properties as content is send. To save new cases into the database, it can receive the cases by a new ACL message with the id new_case. Figure 5 shows the new structure of the MAS and the additionally needed communication ways.

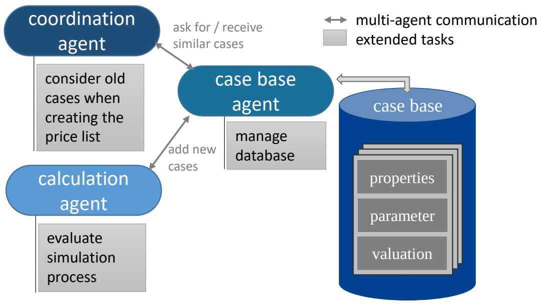

Figure 5: MAS with integrated case-based reasoning approach.

The first step when using case-based reasoning for a new model is to find the model properties that are needed for the similarity measurement. Here, the coordination agent that receives the model from a user is responsible. It analyzes the model and sends the properties to the case base agent. If there are similar cases, it uses them for creating a prioritized price list as a basis for the bidding process. If there are no similar cases the previous concept is used for creating the price list. If there are just one or two cases yet, another configuration can be tried in the case that there are more calculation resources left.

The calculation agents bid on the parts. Thus they check if they match the computation abilities for the parts and configurations in the price list. With this method the agents can decide on their own whether the case can be adapted for their current simulation tasks or not. After each simulation, the calculation agents evaluate the results by comparing their solutions and send the cases to the case base agent.

With the presented incooperation concept, the data for each physical domain is collected and saved centrally but the agents keep their autonomy. The case-based reasoning assists them by the choice on which partial problem to bid.

5 CONCLUSIONS

In this paper a multi-agent system is presented, which is able to solve different multiphysics simulations on distributed computation resources. The basic functionality is shown on a microwave oven example, considering electromagnetic waves and the heat transfer. The MAS has also been successfully tested for further models. Because of the difficulties to choose the right solver configuration, a concept to incorporate a case-based reasoning approach is given. The case-based reasoning is used to support the decisions made by the agents in concern of the solver configuration. This extension allows the MAS to learn like a human expert from model to model. The selection of an appropriate linear solver is only one approach where the proposed MAS collects knowledge. Further this approach can be extended to the more complicated case of nonlinear and/or time dependent problems. The MAS can be regarded as intelligent assistant system for multiphysics simulations. It enables inexperienced users to simulate complex problems on distributed, already available resources using proven software tools.

ACKNOWLEDGEMENTS

The authors would like to thank the Deutsche Forschungsgemeinschaft (DFG) for supporting the project GekoProAg (RU 720/11-2 & WE 5312/8-2).

REFERENCES

Bellifemine, F., Caire, G., Greenwood, D., 2007. *Developing multi-agent systems with JADE*. Wiley.

Beyer, T., Yousefifar, R., Göhner, P., Wehking, K.-H., 2016. *Agent-Based Dimensioning to Support the Planning of Intra-Logistics Systems.* IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA).

Blasco, R., Marco, Á., Casas, R., Cirujano, D., Picking, R., 2014. *A smart kitchen for ambient assisted living*. Sensors, 14, pp. 1629-1653.

Boschert, S., Rosen, R., 2016. *Digital twin—the simulation aspect*. In Mechatronic Futures, pp. 59-74, Springer International Publishing.

Buchau, A., Rucker, W. M., Rain, O., Rischmüller, V., Kurz, S., Rjasanow, S., 2003. *Comparison between different approaches for fast and efficient 3-D BEM computations*. In: IEEE Transactions on Magnetics, vol. 39, no. 3, pp. 1107-1110.

Clement, S. J., McKee, D. W., Romano, R., Xu, J., Lopez, J. M., Battersby, D., 2017. *The Internet of Simulation: Enabling agile model based systems engineering for cyber-physical systems*. In: System of Systems Engineering Conference (SoSE), pp. 1-6.

de Mantaras, R. L., 2001. *Case-based reasoning*. In: Machine Learning and Its Applications, pp. 127-145, Springer Berlin Heidelberg.

Dickinson, E. J., Ekström, H., & Fontes, E., 2014. *COMSOL Multiphysics®: Finite element software for electrochemical analysis. A mini-review*. Electrochemistry communications, 40, pp. 71-74.

Fetzer, J., Kurz, S., Lehner, G., Rucker, W. M., Henninger, P., Röckelein, R., 1999. *Analysis of an actuator with eddy currents and iron saturation: Comparison between a FEM and BEM-FEM coupling Approach*. In: IEEE Transactions on Magnetics, vol. 35, no. 3, pp. 1793-1796.

Fipa, A. C. L., 2002. *Fipa acl message structure specification*. Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00061/SC00061G.html (18.8. 2017).

Grabmaier, S., Jüttner, M., Vögeli, D., Rucker, W. M., Göhner, P., 2016. *Numerical framework for the simulation of dielectric heating using finite and boundary element method*. In: International Journal of Numerical Modelling: Electronic Networks, Devices and Fields.

Gupta, O. P., 2002. *Finite and Boundary Element Methods in Engineering*. Balkema Publishers.

Jazdi, N., 2014. *Cyber physical systems in the context of Industry 4.0*. In: IEEE International Conference on Automation, Quality and Testing, Robotics, pp. 1-4.

Jennings, N., Wooldridge, M., 1996. *Software agents*. In IEE review, 42(1), pp. 17-20.

Jüttner, M., Grabmaier, S., Vögeli, D., Rucker, W. M., Göhner, P., 2017. *Coupled Multiphysics Problems as Market Place for Competing Autonomous Software Agents*, In: IEEE Transactions on Magnetics, Vol. 53, Issue 6.

Lambersky, V., 2012. *Model based design and automated code generation from Simulink targeted for TMS570 MCU*. In: Education and Research Conference (EDERC), pp. 225-228.

Meister, A., 2015. *Numerik linearer Gleichungssysteme*. Springer Spektrum, Vol 5.

Mozumdar, M. M. R., Gregoretti, F., Lavagno, L., Vanzago, L., Olivieri, S., 2008. *A framework for modeling, simulation and automatic code generation of sensor network application*. In: 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08, pp. 515-522.

Tolk, A., 2016. *Tutorial on the engineering principles of combat modeling and distributed simulation*. In: Proceedings of the 2016 Winter Simulation Conference, pp. 255-269.

Vázquez, M., Houzeaux, G., Koric, S., Artigues, A., Aguado-Sierra, J., Arís, R., Taha, A., 2016. *Alya: Multiphysics engineering simulation toward exascale*. In: Journal of Computational Science, 14, pp. 15-27.

Zhang, X., Gao, H., Guo, M., Li, G., Liu, Y., Li, D., 2016. *A study on key technologies of unmanned driving*. CAAI Transactions on Intelligence Technology, 1, pp. 4-13.