Iterative Solution of Multiphysics Problems with Software Agents Designed as Physics Experts

[1)]M. Jüttner, [1)]A. Buchau, [2)]M. Rauscher, [1)]W. M. Rucker, and [2)]P. Göhner
[1)]Institute for Theory of Electrical Engineering, Pfaffenwaldring 47, D-70569 Stuttgart, Germany
[2)]Institute of Industrial Automation and Software Engineering, Pfaffenwaldring 47, D-70569 Stuttgart, Germany

E-mail: matthias.juettner@ite.uni-stuttgart.de

***Abstract*—An iterative approach for the solution of multiphysics problems based on software agents is presented. The usage of optimized numerical methods for each physical problem as well as the handling of physics-optimized meshes is enabled. To realize the coupling of physics, a boundary condition mapping algorithm is described including remarks on implementation. Finally, the realization of the software agent system is demonstrated for a heat transfer problem that is strongly coupled to an electric current flow field problem.**

Keywords— boundary conditions mapping; iterative solver; multiphysics simulations; software agents;

I. INTRODUCTION

Due to the increasing performance of modern computer systems, a growing number of unknowns can be calculated within also growing equation systems. Therefore, parallelisation and optimisation has been a topic for researchers for the last decades. As results, a large number of numerical solvers and methods were introduced. These methods where deeply analysed and offer solutions for lots of problems. Nowadays a combination of different problems called multiphysics problem is getting more interesting for practical applications. This leads to the question of automatically finding a flexible and well performing combination of existing methods to solve these problems.

An iterative strategy for solving multiphysics problems has been introduced in [1]. A theoretical approach for a flexible combination of different methods was given in [2]. The necessary flexibility for independent calculation units is reached by software agents. A combination of both approaches to an operational software is new and the main topic of this paper. This enables the calculation of multiphysics simulations with physics optimized meshes and physics optimized numerical methods. In addition, more accurate results and further parallelization for multiphysics simulations gets possible.

As an example, the iterative solution of a coupled heat transfer problem and electromagnetic field problem is considered. Therefore, two software agents are used based on a single implementation.

II. AGENT DESIGN

A. Behaviours

Agents work and interwork according to predefined tasks [3]. Each task of an agent is called behaviour. The sequence of behaviours an agent does process depends on the received input or the system states the agent is currently in. As example, one implemented behaviour cares about the communication between the agents working on a problem. The solution for a problem connected to multiple agents is built dynamically. It can be easily and well adapted to new information [3]. To avoid the implementation of error correction, character sets or ontology problems for communication [4] the Java Agent Development Framework (JADE) [5] is used.

Another important behaviour coordinates the local numerical method. It can be chosen dynamically and independently. So an adaption to a single physics problem and the specific partial problem, handed over to the agent, gets possible. Due to the concept of agents, agents can be distributed over a computer network. This creates an environment where expensive simulation time, resources, or licences are only used when they are needed. Smaller problems can easily be handed over to systems or further specialized software for the different partial problem. As drawback, additional overhead for communication and data matching has to be accepted.

B. Solving Strategy

Initially the multiphysics problem is split into multiple single physics problems. Here, different meshes depending on physics-based challenges can be used for the partial problems as well as different methods like finite element method (FEM) or boundary element method (BEM). In a second step, the coupling has to be realized. Therefore, coupling sources are integrated as boundary conditions in the corresponding physics. Their values will be evaluated as request for results from the software agent. In the next step, the initial values for an initial partial problem are specified. This initial problem is solved by the behaviour controlling a physics expert method within an agent (*agent a*). To do so, an interface to a problem specific method is realized. Here, the agent does not care about the actual implementation of the method; it just needs to know how to handle the interface. The agent hands over the problem to the expert method, defined in the problem description. Then a solution is calculated. The solution is kept locally within the solver. If any access to the solution gets necessary it is realised via the solver interface. The agent (*agent a*) finally informs the iteratively coupled agent (*agent b*) about the available result.

Agent b checks whether the calculated results can be integrated. This information can be considered as constant for every possible simulation software and needs to be evaluated before the agent connects itself to the solver. In the case of a non-understanding of the previous results, the results has to be considered as independent and the next partial problem can be calculated. In the case of a possible integration, *agent b* requests the solutions from *agent a* and integrates them as new boundary conditions. Due to different meshes, a mapping algorithm of values is needed. This algorithm is explained in detail in section II.C. Evaluating values in a BEM area for a FEM mesh gets possible via post-processing explained in [6]. At this point, a counter is implemented to represent the maximum number of loops for the iterative

calculation cycle. It is incremented if results of other agents are integrated and the deviation of the integrated values differs in sum from a constant. The counter is decremented after every successful calculation cycle. The calculation finishes if this counter is zero. Note the overall sum does not lead to conceptional errors because of the possibility of different meshes for the partial problems. Now the calculation of the next partial problem can be started.

To handle and prioritise all agents' behaviours necessary for solving a problem, a schedule is implemented within each agent. The behaviours representing the interface to the simulation software can be managed by the agents graphical user interface (GUI).

C. Mapping Algorithm

For exchanging results between different agents, the following algorithm is designed. A goal is to guarantee as much flexibility as possible for the integration of results as new boundary conditions in different calculation software. Hence, the values are integrated as three-dimensional and location depending source. This allows the agent to modify the simulation without the need of a direct access to the (in most simulation programs hidden) equation system. So only public interfaces are used. Due to different meshes that can be used for the same model and their relation to the same model a global coordination system for all agents is used. To set up a three-dimensional boundary condition, every node needs its corresponding values from the previously calculated result that has to be validated. To preserve the specification of physics based expert agents, the agent integrating the results (*agent a*) has to send a request for each value of interest to the agent offering the results (*agent b*) including the coordinates of the points. To find the corresponding value of all nodes, *agent b* needs to know the complete mesh of *agent a*. In case of an identic mesh for *agent a* and *agent b* a modified request is sent. It is recommended to store a local copy of the received mesh to avoid multiple transmissions of the mesh form *agent a* to *agent b*. Data compression also saves transmission time. Now the requested values will be evaluated. This is done by the expert agent by interpolation or a recalculation of a tiny area of the mesh. The answer to the request contains all evaluated values for the requested points. Here compression can also be useful for larger data transmissions.

To map the values at *agent a* to the three-dimensional boundary condition, a geometric minimal distance mapping function is created. Input parameters are the coordinates of the vertexes ($\boldsymbol{tp}$) to be evaluated and the list of all transmitted nodes ($\boldsymbol{np}$) including the evaluated values for the boundary conditions. Now vertex with the minimal distance to the evaluation point needs to be found. If this is known at *agent a*, the corresponding boundary condition can be mapped. Therefore a distance vector $\boldsymbol{d}$ is defined according to (1).

$$\boldsymbol{d} = \boldsymbol{tp} - \boldsymbol{np} \qquad (1)$$

For a parallel evaluation of all possible test points n for a model with i dimensions a matrix is build (2).

$$[\,|d_1|^2, |d_2|^2, \dots, |d_n|^2] = |\boldsymbol{tp_i} \cdot \boldsymbol{O_n^T} - \boldsymbol{np_i}|^2 \qquad (2)$$

Here, $\boldsymbol{O}$ represents a vector filled with n ones. Within the matrix the index of the value with the smallest distance is searched. Therefore we define the search radius r and search the minimum of eq. (3)

$$[\mathrm{q}_1, \mathrm{q}_2, \dots, \mathrm{q}_n] = [\,|d_1|^2, |d_2|^2, \dots, |d_n|^2] - \boldsymbol{O}_{1,n} \cdot r \qquad (3)$$

The radius r needs to be smaller then the minimal distance between two evaluation points to find a valid mapping. Because only one node is inside the sphere, the value is mapped to its correct destination. Due to the mesh exchange, it can be chosen close to zero to consider numerical processing tolerances. The boundary condition of value is then chosen as shown in eq. (4) and returned.

$$q_p < 0 \;\forall\; p = 1 \dots n \qquad (4)$$

For a fast computation, this algorithm can be highly parallelised [7].

III. Numerical Results

Here, a field effect transistor (FET) mounted on a circuit board is simulated. Boundary conditions are the convective cooling at the surface of the model and the conductive heat transfer of the FET within the solids as well as electromagnetic losses. The necessary number of iterations for the given problem as well as a detailed signaling diagram and the processing time of the matching algorithm will be shown in the full paper. Additional results will be the solution for each step of the iterative solution process including the overall result. A successful run of the iterative calculation of the described model is shown in Fig.1. To do so, an electromagnetic agent (*agent a)* and a heat transfer agent (*agent b)* were used.

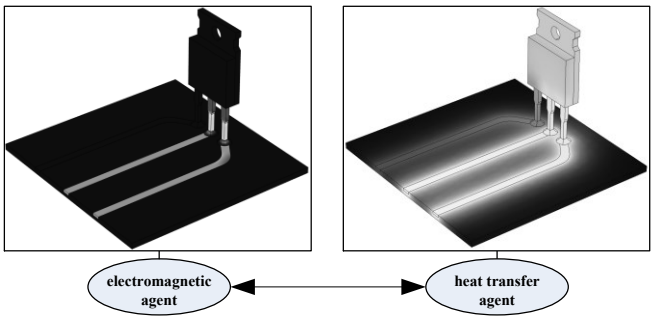

Fig. 1: Locally (left) and iteratively (right) computed results

References

[1] H. Liu and J. Rao, "Coupled Modeling of Electromagnetic-Thermal Problem in Induction Heating Process Considering Material Properties," International Conference on Information Engineering and Computer Science, pp. 1-4, 2009.

[2] M. Jüttner, A. Buchau, M. Rauscher, W. M. Rucker and P. Göhner, "Software Agent Based Domain Decomposition Method," Proceedings IGTE'12, pp. 89-94, 2012.

[3] N. R. Jennings, "On Agent-Based Software Engineering," Artificial Intelligence, vol. 117, no. 2, pp. 277-296, 2000.

[4] "FIPA ACL Message Structure Specification," Foundation for Intelligent Physical Agents, 2002.

[5] F. L. Bellifemine, G. Caire and D. Greenwood, "Developing Multi-Agent Systems with JADE," John Wiley & Sons, UK, 2007.

[6] A. Buchau, M. Jüttner and W. Rucker, "Automatic domain detection for a meshfree post-processing in boundary element methods," Proceedings IGTE'12, pp. 386-391, 2012.

[7] P. Pratt-Szeliga, J. Fawcett and R. Welch, "Rootbeer: Seamlessly using GPUs from Java," IEEE 14th International Conference on High Performance Computing and Communication, pp. 375-380, 2012.