Agent-Based Simulation of Data-Driven Fire Propagation Dynamics

John Michopoulos[1], Panagiota Tsompanopoulou[2], Elias Houstis[3,2], and Anupam Joshi[4]

[1] Special Projects Group, Code 6303, U.S. Naval Research Laboratory, U.S.A.
john.michopoulos@nrl.navy.mil
[2] Dept. of Comp. Eng. and Telecommunications, University of Thessaly, Greece
yota@inf.uth.gr
[3] Computer Sciences Department, Purdue University, U.S.A.
enh@cs.purdue.edu
[4] Dept. of Comp. Sci. and Electr. Eng., U. of Maryland Baltimore County, U.S.A.
joshi@cs.umbc.edu

Abstract. Real world problems such as fire propagation prediction, can often be considered as a compositional combination of multiple, simple but coupled subproblems corresponding to analytical and computational behavior models of the systems involved in multiple domains of action. The existence of various computational resources (legacy codes, middleware, libraries, etc.) that solve and simulate the subproblems successfully, the coupling methodologies, the increasing and distributed computer power (GRID etc.) and the polymorphism and plurality of the available technologies for distributed mobile computing, such as Agent Platforms, motivated the implementation of multidisciplinary problem solving environments (MPSE) to overcome the difficulties of their integration and utilization. In this paper we present the onset of the development of computational infrastructure for the simulation of fire propagation in multiple domains using agent platforms as an informal validation of our data-driven environment for multi-physics applications (DDEMA).

1 Introduction

The main objective of this effort is to provide informal validation to the general framework of DDEMA [1,2,3,4], for the area of fire-environment qualitative assessment and evolutionary fire propagation prediction based on dynamic sensor data. To achieve this objective we have targeted the development of a system that will be a multidisciplinary problem solving environment (MPSE), and will provide simulation based behavior prediction for monitoring situational assessment and decision support required by fire fighting teams. We will be referring to it as the Data-Driven Fire Hazard Simulator (DDFHS).

The context, general and specific requirements, architecture and meta-archite-cture of the DDEMA framework has been described already elsewhere [1,2,3,4].

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 732–739, 2004.

Development of DDFHS in the context of the above referenced main objective requires that it should be producible by utilizing the resources of the DDEMA framework as a meta-meta-system and that it will demonstrate two main features. The first feature is the software encapsulation of multiphysics data-driven model selection. The second feature is heavy- and light-weight computational solution implementations and comparison for solving of multi domain coupling problems (i.e. multiple rooms building with embedded sensor network under fire conditions).

The rest of the paper is organized as follows. In Section 2, we describe the fire dynamics in terms of two models of various complexities. In Section 3, the description of DDEMA's multilayered implementation for creating DDFHS is presented. Finally, conclusions follow in Section 4.

2 Modelling of Fire Dynamics

For the sake of demonstrating dynamic model selection capability we are employing two modelling formulations of significantly different complexity. They are the partial and the complete fire propagation models referring them with the acronyms (PFPM) and (CFPM) respectively. The system should be able to select which of the two models will actually be utilized for the simulation based on user specification. Our selection of these two models is by no means unique and many other formulations could be used in their place.

2.1 Partial Fire Propagation Model

For this case we are considering a time shifted term-selective sequential application of the convention-conduction partial differential equation (PDE):

$$\frac{\partial T}{\partial t} = -\beta_i \frac{\partial T}{\partial x_i} + \frac{1}{\rho C} \nabla (k_i \frac{\partial T}{\partial x_i}) - \frac{\alpha}{\rho C}(T_0 - T) + f, \tag{1}$$

where T is the temperature, β_i are convection coefficients, ρ is the density of the medium, C is the heat capacity, k_i are conductivity coefficients, α is the thermal expansion coefficient, T_0 is the initial temperature and f is the heat production. The second term, in the right-hand side, is the convective term, while the third term is the conductive term. When the convective term is missing the obtained PDE is elliptic, while when the conductive term is missing the resulting PDE is hyberbolic, and this suggests different solution approaches.

The solution of this heat evolution PDE will produce the approximate temperature distribution field in the solution domain(s) and will become the input of the embedded sensor network. This formulation represents a gross-simplification of the actual case since most conservation laws have been ignored from a thermodynamic modelling perspective.

2.2 Complete Fire Propagation Model

For this case we are considering one of the most complete formulations for reactive flaw dynamics as they are relevant to fire propagation dynamics. All thermodynamic and conservation laws have been incorporated and they yield the following set of PDEs [5]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0, \tag{2}$$

$$\frac{\partial}{\partial t}(\rho Y_l) + \nabla \cdot \rho Y_l \mathbf{u} = \nabla \cdot \rho D_l \nabla Y_l + \dot{m}_l''', \tag{3}$$

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) + \nabla p = \rho \mathbf{g} + \mathbf{f} + \nabla \cdot \tau, \tag{4}$$

$$\frac{\partial}{\partial t}(\rho h) + \nabla \cdot \rho h \mathbf{u} = \frac{Dp}{Dt} - \nabla \cdot \mathbf{q_r} + \nabla \cdot k \nabla T + \sum_l \nabla \cdot h_l \rho D_l \nabla Y_l, \tag{5}$$

where the most important variables are $\rho, \mathbf{u}, T, \mathbf{q_r}, Y_l, M_l, p$ and p_0 and are density of the fluid, velocity vector, temperature, radiative heat flux vector, mass fraction of each species, molecular weight of each gas species, pressure and background pressure respectively, while $\mathbf{f}, \mathbf{g}$ and τ are external forces, acceleration of gravity and viscous stress tensor respectively.

Equation (2), expresses the conservation of total mass, Eq. (3) expresses the conservation of mass for each species participating in the combustion process, Eq. (4) expresses the conservation of momentum, and finally Eq. (5) expresses the conservation of energy according to [5].

This coupled system of PDEs yields an approximate form of the generalized Navier-Stokes equations for reactive flow application with low Mach number. This approximation involves exclusion of acoustic waves while it allows for large variations of temperature and density [6]. This endows the equations with an elliptic character that is consistent with low speed, thermal convective reactive processes valid for our interests.

3 DDEMA Implementation for the Case of DDFHS

3.1 Fire Scenario Specification

To avoid providing a lengthy specification of user requirements that falls outside the sizing scope of this paper, we present a common scenario for various fire-fighting situations. This scenario will be equivalent to a qualitative specification of the involved user (fire-fighter) requirements. The development of DDFHS will be addressing the needs of all participants of this scenario over the selected represented hardware/network topology. Figure 1 depicts a schematic diagram of this scenario.

According to the scenario a room is assumed as the space defining the area of interest where a fire may brake out. The room is equipped with a stationary or moving network of sensors with various degrees of clustering aggregation [7]. There is a base-station farm that is capable of collecting the sensor

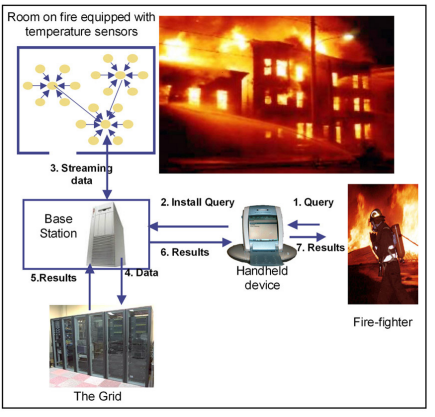

Fig. 1. Fire scenario for micro-future behavioral prediction of fire dynamic propagation upon actions and conditions specified by the fire fighter.

and/or cluster-head output data dynamically. In the beginningof this project we are considering these data to be temperature values at the associated sensor locations. They can also be coefficients corresponding to the interpolation polynomials representing the temperature distribution fields in the domain of the room. This implies that the sensors may extend from non-intelligent thermocouples, to very intelligent sensors with associated embedded microprocessors for local computation of the necessary coefficients. A farm of high performance computing (HPC) systems can be considered as the appropriate GRID replication resource. When a fire brakes out in the room and the fire-fighter arrives, then he/she uses his portable digital assistant (PDA) or PDA-enabled mobile phone or other handheld device to issue queries to the system to help him assess the current situation (based on the sensor outputs) and more importantly to ask 'what-if' questions of the type: *Is it safe to open a particular door in the next 1-2 minutes, to gain access in the room or not?*, or, *What are the risk and the confidence levels that the fire will not propagate through the door when the door is opened in the next 1-2 minutes?* Clearly these type of questions can only be answered when a sensor data-driven simulation of the fire dynamics is continually adjusted and steered by the fire-fighter's queries (usually actions translate to boundary conditions modifications) and the sensor data.

The flow of data is denoted by the vectors connecting the various resources in Fig. 1, and their sequential order by their numerical labels.

The software implementation as described below, is capable of responding to all requirements associated with the above mentioned scenario.

Since the design of DDEMA has been discussed from various perspectives [3, 4] here we will not repeat the detailed description of three-layered architecture. Only a short description will be provided for the scope present paper.

On the top (closer to the user and developer) is the *surfaceware* layer that implements the user interface functionality of DDEMA at the stage solution development time. The second layer is the *middleware* and serves as an interface between the surfaceware and the layer below it. Its most important activity is to allow process migration within multiple virtual machines for increased redundancy, control of the processes at the layer below and very lightweight hardware integration capability. The layer at the bottom, *deepware*, contains the legacy codes, the symbolic algebra modules, and specific codes for coupling multi-domain PDE written for DDEMA and execute them on the available resources. As an instantiation of DDEMA, DDFHS's implementation will inherit this architecture from DDEMA. Separate descriptions of these layers are going to be provided below as detailed as they are relevant to the DDFHS.

3.2 Implementation of Surfaceware

The top level of DDEMA and its validation instantiations (such as the DDFHS) is based on two modes, the application design mode and the application use mode. In the first mode the designer will describe the actual application architecture in terms of data flow and message passing diagrams. The use of Ptolemy-II from UC Berkeley [8] enable us to convert the visual representation of the application to components suitable for the meta-meta-system presented in [4]. In the application use mode, the user initiates the procedure for the automatic generation of necessary java source code or bytecode that implements the described application at runtime. The execution of the code will provide the user a stand alone application, which will perform the activities associated with the corresponding requirements. In most of the cases the actors (by Ptolemy-II) encapsulate existing code or agent-wrapped code that calls the legacy codes which simulate the specific application.

3.3 Implementation of Middleware

To select a suitable agent platform for our middleware we have conducted a comparison study based on knockout and performance criteria that will be published independently. The systems that surfaced as the three leaders of this study are the Java Agent DEvelopment (JADE) framework [9], GRASSHOPPER [10] and ProActive [11]. The first two of them are FIPA [12] compatible and therefore they can be integrated to a common but heterogeneous, integration.

For our initial implementation we exclusively selected JADE, driven mainly, by the fact that it has been ported to a very wide variety of hardware platforms extending from handheld devices to HPC clusters.

A typical surfaceware view of the middleware implementation of a multi-domain (corresponding to multiple rooms) fire propagation simulator is expressed by the actor-based data-flaw network in Fig. 2 where Ptolemy-II Vergil editor has been utilized. The actors are wrappers for agents that are wrappers for legacy codes that implement the Finite Element Tear and Interconnect method. Other domain compositions methods will also be considered.

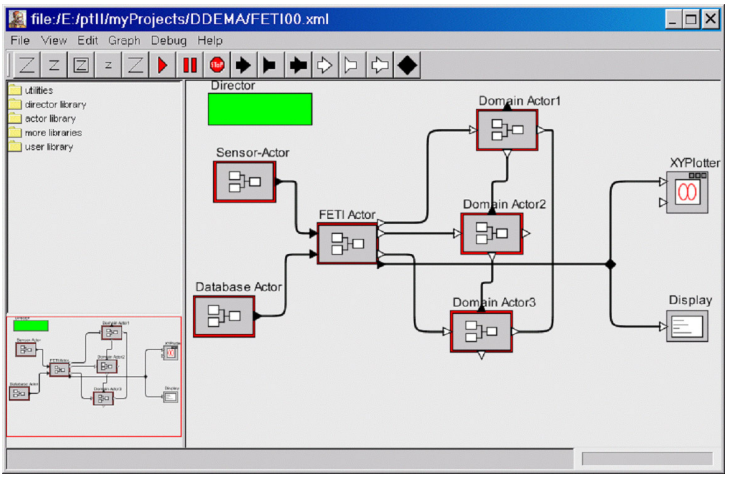

Fig. 2. Ptolemy-II surfaceware view of JADE middleware agents as a data-flaw diagram implementing a multi-domain FETI-based simulator

3.4 Implementation of Deepware

The deepware of DDEMA consists of two major components:

(1) legacy codes capable of implementing the solution of PDEs that express behavioral models of fire propagation dynamics such as those expressed by in Eqs. (1) or (2)-(5).

(2) intermediate codes and scripts responsible for data preparation, conversion and integration.

There are many codes capable of handling the numerical solution of the PDE encapsulated models given in Section 2. We have decided to use the general purpose code freeFEM+ [13] for solving Eq. (1) for the case of PFPM and NISTS's FDS [5] for the system of Eqs. (2)-(5) that corresponds to the CFPM. Preliminary results from our use of freeFEM+ are shown in Fig 3.

The major issue that has developed for the case FDS are the legacy code conversion and wrapping difficulties. Working with the main module of FDS we faced the well-known problem of calling a FORTRAN routines from a JAVA agent, because FDS is implemented in FORTRAN 90 and the agent platform we use is written in JAVA. This procedure could have been done *automatically* by converting the FORTRAN code to JAVA directly using tools like `f2j, f2c` and `f90toC`, however, various expressions of problematic behavior in each case prohibited the conversion. The tools `f2j` and `f2c` work for FORTRAN 77, and FDS code is in FORTRAN 90, while even in the latest version of `f90toC` important modules (e.g. I/O, vector subscripts, memory allocation) are not yet implemented. Since the code of FDS contains approximately 35 Klines, manual conversion to JAVA is as very tedious endeavor and has therefore been excluded. Thus, we use our experience from the implementation of GasTurbnLab [14] and we convert the main FORTRAN program to a subroutine which we call from a

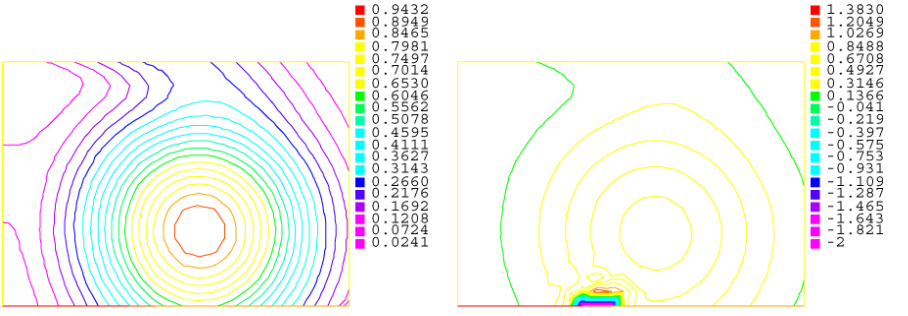

Fig. 3. Typical freeFEM+ produced simulation of 2D temperature distribution due to fire propagation before (left) and after (right) opening the door of the modelled room.

C/++ function. These C/++ functions are wrapped using the automated Java wrapping technology of JACAW [16](which utilizes Java Native Interface [15]). This ensures the capability of our legacy code functionality to be available for JADE agents and Ptolemy-II actors.

4 Conclusions

In this paper we presented the first steps towards utilizing DDEMA's infrastructure to develop DDFHS. We presented two potential modelling approaches for the fire propagation dynamics. A fire-fighting scenario was used to explain certain aspects of the software's implementation to be used for this effort. Efficient and accurate legacy codes are used to simulate the fire dynamics, while new code has to be written for the agentization of them and their coupling with surfaceware actors, based on previously used techniques.

In the subsequent steps of the implementation, we will continue with the legacy code wrapping and implementation of compositional solutions. Finally, we shall remain aware of new technologies relative to our project and we will adapt our approach as needed.

Acknowledgements. The authors acknowledge the support by the National Science Foundation under grants ITR-0205663 and EIA-0203958.

References

1. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Rice, J., Farhat, C., Lesoinne, M., Lechenault, F., DDEMA: A Data Driven Environment for Multiphysics Applications,in: Proceedings of International Conference of Computational Science - ICCS'03, Sloot, P.M.A., et al. (Eds.) Melbourne Australia, June 2-4, LNCS 2660, Part IV, Springer-Verlag, Haidelberg, (2003) 309-318.

2. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Rice, J., Farhat, C., Lesoinne, M., Lechenault, F., Design Architecture of a Data Driven Environment for Multiphysics Applications, in: Proceedings of DETC'03, ASME DETC2003/CIE Chicago IL, Sept. 2-6 2003, Paper No DETC2003/CIE-48268, (2003).
3. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Farhat, C., Lesoinne, M., Rice, J., Joshi, A., On a Data Driven Environment for Multiphysics Applications, Future Generation Computer Systems, in-print (2004).
4. J. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Farhat, C., Lesoinne, M., Rice, Design of a Data-Driven Environment for Multi-field and Multi-Domain Applications, book chapter in Darema, F. (ed.), Dynamic Data Driven Applications Systems, Kluwer Academic Publishers, Netherlands, in-print (2004).
5. McGrattan, K.B., Baum, H.R., Rehm, R.G., Hamins, A., Forney, G.P., Fire Dynamics Simulator - Technical Reference Guide, National Institute of Standards and Technology, Gaithersburg, MD., NISTIR 6467, January 2000.
6. Rehm, R.G., Baum, H.R., The Equations of Motion for Thermally Driven, Buoyant Flows. J. of Research of the NBS, 83:297–308, (1978).
7. Hingne, V., Joshi, A., Houstis, E., Michopoulos, J., On the Grid and Sensor Networks, Fourth International Workshop on Grid Computing November 17–17, 2003 Phoenix, Arizona (2003).
8. Davis II, J., Hylands, C., Kienhuis, B., Lee, E.A., Liu, J., Liu, X., Muliadi, L., Neuendorffer, S., Tsay, J., Vogel, B., Xiong, Y.: Heterogeneous Concurrent Modeling and Design in Java, Memorandum UCB/ERL M01/12, EECS, University of California, Berkeley, CA USA 94720 March 15, 2001, http://ptolemy.eecs.berkeley.edu/ptolemyII/ .
9. The JADE Project Home Page. http://jade.cselt.it
10. The Grasshopper Agent Platform, IKV++ GmbH, Kurfurstendamm 173-174, D-10707 Berlin, Germany. http://www.ikv.de
11. Caromel, D., Klauser, W., Vayssiere, J.: Towards Seamless Computing and Metacomputing in Java, pp. 1043–1061 in Concurrency Practice and Experience, September-November 1998, 10(11–13). Editor Fox, G.C., Published by Wiley & Sons, Ltd., http://www-sop.inria.fr/oasis/proactive/ .
12. FIPA 2000 Specifications. Available at http://www.fipa.org .
13. Bernardi, D., Hecht, F., Ohtsuka, K., Pironneau, O., freeFEM+, a finite element software to handle several meshes. (1999), downloadable from http://www.frefem.org .
14. Fleeter, S., Houstis, E.N., Rice, J.R., Zhou, C., Catlin, A.: GasTurbnLab: A Problem Solving Environment for Simulating Gas Turbines. Proc. 16th IMACS World Congress, (2000) No 104-5.
15. Java Native Interface Specification. http://web2.java.sun.com/products/jdk/1.1/docs/guide/jni.
16. JACAW: A Java-C Automatic Wrapper Tool, http://www.wesc.ac.uk/projects/jacaw .