Agent-based Resource Management in Tsunami Modeling

Alexander Vazhenin, Yutaka Watanobe, Kensaku Hayashi
Graduate School Department, University of Aizu,
Aizu-Wakamatsu, 965-8580,Japan
Email: {vazhenin, yutaka, m5161111}@u-aizu.ac.jp

Michał Drozdowicz, Maria Ganzha, Marcin Paprzycki, Katarzyna Wasielewska
IBS PAN, Newelska 6, 01-447
Warszawa, Poland,
Email:{firstname.lastname}@ibspan.waw.pl

Paweł Gepner
Intel Corporation
Pipers Way
Swindon Wiltshire SN3 1RJ
United Kingdom
pawel.gepner@intel.com

***Abstract*—Complexity of tsunami modeling requires designing software system with high level of reusability and interoperability of its components, and flexible resource management. In this paper we investigate how to integrate the tsunami modeling software with an agent-based resource management infrastructure.**

I. Introduction

REFLECTIONS brought about by the Great Japanese Earthquake and Tsunami raise questions how to mitigate the impact of such events at different time scales, from real time tsunami warning, to long-term hazard assessment. This makes the Tsunami Modeling Problem even more important, and requires applying modern software design approaches, including collaboration among distributed clients and computational services. Another challenge is scalability of the approach, which should allow an arbitrary number of users and computational resources to interact in a customizable working environment. Finally, since separate applications and services may be developed for heterogeneous technologies and platforms, reusability and interoperability are important aspects of exposing and combining computational resources and services [1], [2].

Increasing power of personal computers allows their volunteer participation in high-performance computing, by granting computer time for public calculations. For example, the Folding@home project involves distributed computationally intensive simulations of protein folding and other molecular dynamics simulations [3], [4]. Each user can participate in these computations by calculating a part of a problem. Note that volunteer computing does not allow private use of results obtained by a community of users. Furthermore, this project (and many others) is based on applying a single computational kernel, using a BOINC-like client. Unfortunately, tsunami modeling requires more human interactions during the process, to understand the results and on their basis to specify the next round of experiments. Furthermore, as mentioned below, there exists a number of different tsunami models, that often need to be combined to model various phases of the tsunami phenomenon. This makes simple application of the BOIC-like volunteer approach unfeasible. For different reasons, it may not always be possible to use grid-like infrastructures. For instance, in a University with multiple laboratories "belonging to" separate administrators, it may not be easy to combine them into a single grid. This is especially the case when when different resources have different availability schedules.

In response to these challenges, the aim of this note is to outline how to combine tsunami modeling software with an agent-based resource management infrastructure. Here, agents' flexibility and ability for negotiations will allow malevolent use of resources belonging to different administrative domains.

II. State-of-the-art in Tsunami modeling

Let us start from a brief overview of the state-of-the-art in tsunami modeling. In [5], authors suggested that complex mathematical models and high mesh resolution should be used only when and where necessary. They proposed a "parallel hybrid tsunami simulator," based on mixing different models, methods and meshes. Their system was implemented implemented using object-oriented techniques, allowing for easy (re)use of existing codes and adding new ones. Note that the goal was to combine existing approaches to develop high quality hybrid models (rather than high performance).

Authors of [6] experimented with eight parallel tsunami simulators. They applied different programming models; e.g. thread based shared memory, distributed memory, and virtual shared memory. As a result of experiments it was shown that the threading-based approach does not scale well, especially if sufficient "node memory" is not available.

The TsunamiClaw package was developed on the basis of the finite volume method [7]. Here, the solution is represented as piecewise constant and is approximated in discrete computational grid cells. The obtained solution is represented in the form of water depth and momentum. Currently, this project is no longer being actively developed. Instead it has been generalized into the GeoClaw software [24].

The TUNAMI–N2 software [8], uses a hybrid tsunami model with different approaches to deep sea and shallow water / dry land modeling. However, it applies constant grid size in the entire domain. The TUNAMI was developed by Imamura in 1993 by Imamura. The package was written in FORTRAN and had a standard GUI for interaction.

The Method of Splitting Tsunami (MOST; [9], [10]) was developed at the NOAA (Seattle, USA). It allows real-time tsunami effect forecasting, as it can incorporate data from detection buoys. Furthermore, in the US, the MOST model is used to create inundation maps [11]. Recently, MOST software was combined into a SOA system [1] available through a web enabled interface (ComMIT; [25]).

Summarizing, a number of models exist for tsunami risk mitigation. Typically, they involve: origins of tsunamigenic earthquakes (estimation of magnitude and epicenter location), determination of the initial displacement of the tsunami source, wave propagation, inundation to the dry land, etc. Typical tsunami modeling environment simulates three processes: (1) estimation of residual displacement area, resulting from an earthquake and causing the tsunami, (2) transoceanic propagation of the tsunami through the deep water zones, and (3) contact with the land (run-up and inundation).

III. University of Aizu Tsunami Model

A. Basic Model and Software Tools

Currently, a novel tsunami modeling system is being developed at the University of Aizu. It is based on the principles of the Service-Oriented Architecture (SOA) and follows a Virtual Model-View-Controller pattern (VMVC). The VMVC is an adaptation of the traditional MVC to the SOA ([13]). The starting point for this work was the MOST package ([9], [10]). This method was initially developed in the Tsunami Laboratory of the Computing Center of the USSR Academy of Sciences in Novosibirsk. Subsequently, it was updated in the National Center for Tsunami Research (NCTR, Seattle, USA), and adapted to the models and standards used by the tsunami watch services in the US (and other countries). Here, the propagation of the long wave in the ocean is governed by shallow-water differential equations:

$$\begin{aligned} H_t + (uH)_x + (vH)_y &= 0, \\ u_t + uu_x + vu_y + gH_x &= gD_x, \\ v_t + uv_x + vv_y + gH_y &= gD_y, \end{aligned} \tag{1}$$

where $H(x, y, t) = h(x, y, t) + D(x, y, t)$; h represents water surface displacement, D depth, g gravity, $u(x, y, t)$ and $v(x, y, t)$ are the velocity components along the x and the y axis. Initial conditions should confirm the presence of water in all grid points, except for the tsunami source, where the surface displacement is not equal to zero.

The numerical algorithm is based on splitting in spatial directions the difference scheme, which approximates equations (1). This transforms solution of equations with two space variables to the solution of two one-dimensional equations and allows use of effective finite-difference schemes developed for one-dimensional problems. Moreover, this method permits to set boundary conditions for a finite-difference boundary value problem using a characteristic line method.

B. General Calculation Process

Figure 1 presents the block-diagram summarizing the calculation process. The input data consists of:

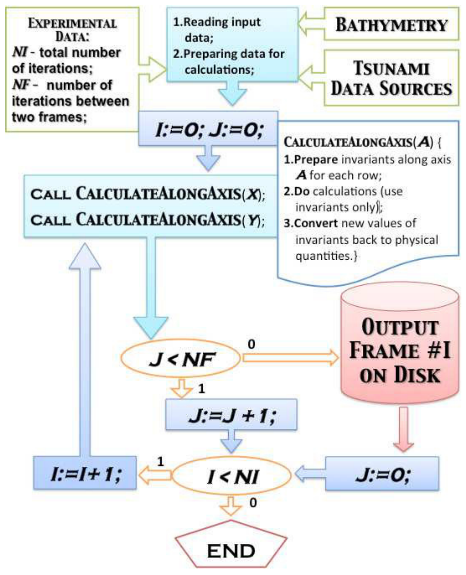
EXPERIMENTAL DATA:
NI - total number of iterations;
NF - number of iterations between two frames;
1.Reading input data;
2.Preparing data for calculations;
BATHYMETRY
TSUNAMI DATA SOURCES
I:=0; J:=0;
CALCULATEALONGAXIS(A) {
1.Prepare invariants along axis A for each row;
2.Do calculations (use invariants only);
3.Convert new values of invariants back to physical quantities.}
CALL CALCULATEALONGAXIS(X);
CALL CALCULATEALONGAXIS(Y);
J < NF
0
1
OUTPUT FRAME #I ON DISK
J:=J + 1;
I:=I+ 1;
1
I < NI
J:=0;
0
END

Fig. 1. General Computational Scheme

- bottom topography or bathymetry data,
- initial and boundary conditions,
- other parameters, e.g. time-steps and length of model run.

While running the program implements stores results as a series of frames representing the tsunami propagation process. Parameter NF defines the time interval, during which results are persisted in memory. After this time expires, results of current iteration, containing wave parameters, are stored on the secondary storage devices in the NetCDF format ([11]). The NetCDF format supports the creation, access, and sharing of array-oriented scientific data, while special programs allow its analysis and visualization.

While the original MOST software was implemented in Fortran 90, it was later ported to C/C++. Now, it takes about 3.00 seconds for a single time step on a 4 dual-core (Intel Xeon 2.8GHz) CPUs computer ([13], [14]). Observe that, a typical simulation, consists of about 10000 time steps (8 hours to complete). Therefore, the tsunami modeling needs to be significantly accelerated; especially for real-time tsunami warning guidance. However, speeding up modeling is also crucial for repetitive tsunami simulations.

IV. Service-oriented Architecture and Application Engines

A. Virtual MVC-design Patterns

Variety of methods for tsunami modeling require effective use of heterogeneous components on a variety of platforms and architectures. Furthermore, to achieve reusability, it is more cost effective to integrate applications rather than to rebuild them. Therefore the Virtual MVC design pattern (VMVC) was applied (Figure2). The demarcation of a Functional (View) and an Implementation (Model) task can be achieved by inducing an Integrator (Controller). The Controller can be enriched by encapsulating certain Non-Functional activities

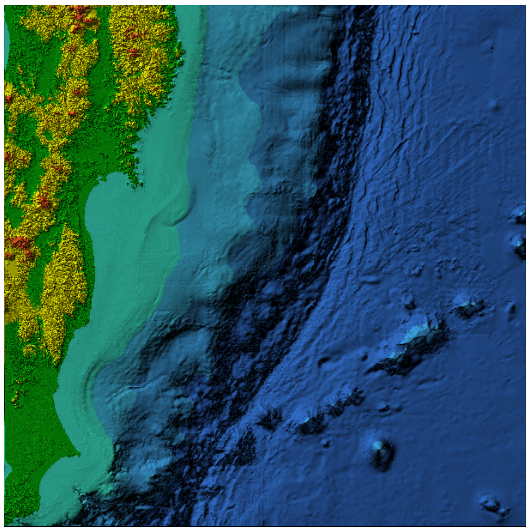
Fig. 3. Visualization of the 2413x2405 gridded relief around NE coast of the Honshu island.

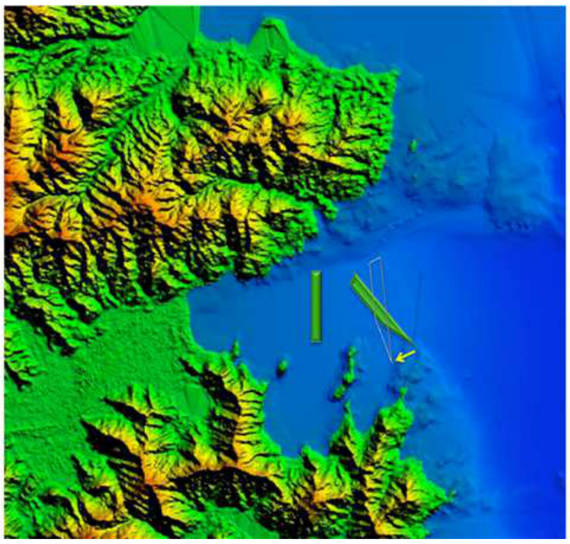
Fig. 4. Bathymetry with artificial objects.

such as security, reliability, scalability, and routing. This enables the separation of Integration Logic from that of Functional Logic (Client Application) and Implementation Logic (Service). The extendable set of application-oriented systems can be realized based on this integration environment including Service-Oriented Multistage Tsunami Modeling [13]. Figure2 shows main elements of this architecture, where the Tsunami Modeling database contains Tsunami Scenarios (input parameters), Results of Modeling, and the Bottom Topography (or Bathymetry) data. The scenarios can be created/edited or downloaded from the Internet. The Tsunami Scenario Loader converts data to the Database format. Similar operations can be provided with the Tsunami Modeling Result Data obtained by external modeling components and published on the Internet.

The quality of the bathymetry data is one of the key parameters defining the accuracy of the model. This data is updated periodically. Here, a special bathymetry was developed, covering the area of the Pacific Ocean adjacent to the northwest parts of the Honshu island (Japan). The gridded digital bathymetry for the numerical modeling was prepared using 500 m resolution bathymetry around Japan [14], and 1 arc sec ASTER Global digital elevation model [15]. A computational rectangular grid of 2413x2405 points includes knots of pre-setup depth values. Length of a spatial step in both directions made 0.0024844 geographical degrees that is about 277 m in a North-South direction and about 221 m in the West-East direction. The bottom relief of the domain is stretching from 34 to 40 degrees of North Latitude and from 140 to 146 degrees of East Longitude, and is shown in Figure 3.

Modeling used tsunami data generated from the Great Japanese Earthquake (38.322°, 142.369°E, Mw = 8.9 at 5:46:23 UTC) on March 11, 2011 [16]. The fault length and width were 400 km $\times$ 150 km. Numerical experiments confirmed the reliability of this technique, and a good fine-grained CUDA acceleration of modeling process [14].

B. Hybrid Tsunami Modeling Combining Natural and Artificial Bathymetry Objects

Lessons from the Great Japanese Tsunami stress importance of reducing impact of tsunamis on different time scales. First, it is necessary to provide real-time tsunami warning. Second, tsunami modeling can be used for long-term hazard assessment (e.g. detailed inundation models across the Japanese sea-line). Third, the well–known "Matsushima effect," the considerable influence of natural geographical objects, like islands and bathymetry, on the wave height and speed of tsunamis, could be considered. Such effect exists in the Matsushima area where presence of islands mitigates effects of tsunamis (while they are absent on the Fukushima coast).

This conjecture is based on [17], where results of a simulation of effects of submarine barriers on tsunami wave propagation were presented. The experiments were conducted in a basin 5 m in length and 10.5 cm in depth. Single and double barriers were used in variable arrangements. Experiments indicated capability of reducing tsunami run-up. It was also shown that parameters of simulations can be translated to natural conditions. Therefore, similar computer simulations could be completed for crucial coastal areas. In this case, aim of the simulation would be to study effects of objects of different shapes, sizes and placement configurations (see, Figure 4). In this way it may be possible to design and build a set of artificial objects (islands) that can be used to protect the coastal areas. In particular, such protection could be of extreme value in highly populated areas, as well as in industrial areas (e.g. nuclear plants, factories, airports, etc.).

Currently, we are enhancing the modeling system by adding the Interactive Bathymetry Editor, supporting object-oriented GUI-editing on bathymetric data, as well as including/removing artificial objects of variable placement, shapes and size, and a plan of changing these parameters during the simulations (see, Figure 5). Here, the modeling process is

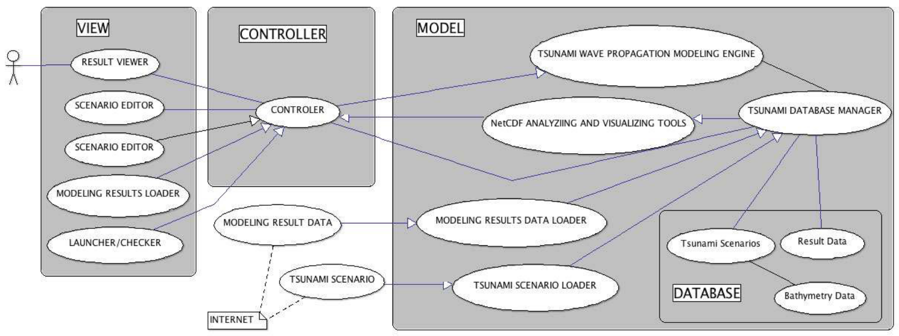

Fig. 2. VMVC-SOA Decomposition of the Tsunami Modeling Environment

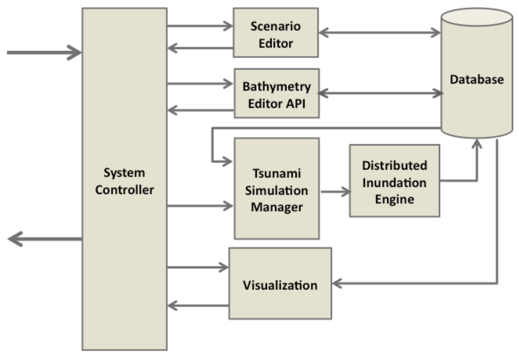

Fig. 5. Components for Hybrid Tsunami Modeling

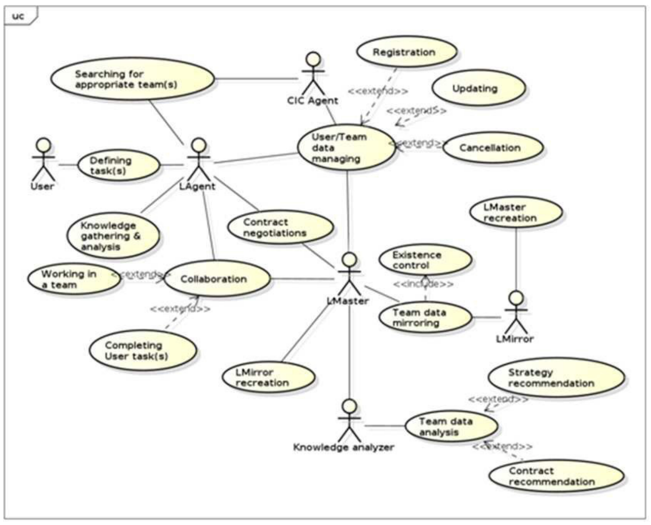

Fig. 6. Use Case diagram of the AiG system

controlled by the Tsunami Simulation Manager, and has as its goal finding suitable number, sizes, and placement of artificial islands and submarine bathymetry objects that minimize the dangerous tsunami wave parameters (height and speed). This problem requires multiple runs of identical tasks with different parameters. While such computing scenario could be realized in a BOINC-like environment, for reasons outlined above, we have decided to attempt at their realization using the agent-based infrastructure.

V. AiG for Tsunami Modeling

The Agents in Grid (AiG) project aims at providing a flexible agent-based infrastructure for managing resources in the Grid ([18], [19]). Application of software agents and semantic technologies makes it well-suited for open, dynamic and heterogeneous environments. The AiG architecture is based on the concept of an open Grid – a network of heterogeneous resources, owned and managed by different organizations. It allows for users to either provide a new resource to the Grid in order to earn money, or to use the Grid to execute a task. Note, that this design nicely fits into the VMVC design pattern, since the Client Application is separated from reusable Services (available resources), while agents representing user and resource manager share characteristics of the Controller. Hence, we can distinguish component responsible for Integration, Functional and Implementation Logic.

The elements of the system have been modeled as software agents: each resource is governed by (and each user is represented by) an *LAgent* and performs its tasks as part of a team, managed by an *LMaster* agent. Teams are registered in a directory service, represented by the *Client Information Center* (CIC) agent, which handles matchmaking of users to teams. The decision, which team to choose to execute the job is a result of autonomous negotiations between the *LAgent* and the *LMaster*s of the appropriate teams. In a similar way, adding a resource to the system involves negotiations with teams looking for new team members. The main features of the system are shown in Figure 6.

In the AiG project all information is represented in ontological format, using the OWL language. The usage of ontologies enables to describe jobs, resources and their relationships in a structured, yet flexible way (for more details, see [20]). Support previously unhandled hardware and software (such as new hardware devices, software libraries or programs) involves only modification of the ontology and does not require any additional customization. As the job descriptions are also defined in OWL, it is possible to specify different parameter

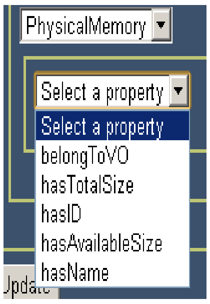

Fig. 7. Example of choosing a property for constraining

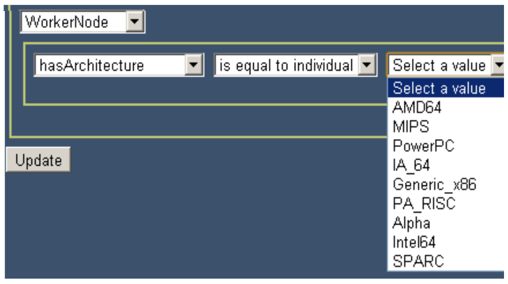

Fig. 8. Example of choosing an individual

sets and required computing resources depending on the type of the task to be performed. This is especially important for the applications described above, where the hardware and software configurations are highly heterogeneous and tools used to perform the experiments may differ from machine to machine.

Attempting at using the AiG architecture at the University of Aizu, one can observe that it is not an open environment (with resources joining and leaving the Grid dynamically). Moreover, there is no economic aspect – if a resource matches the requirements of the job and is available for use, there is no need for negotiations of terms of usage. These differences allow us to simplify its structure and use. As a result, we resign from the notion of resource teams, and place an *LAgent* (playing the role of the *LMaster*) on each computing node. We can also eliminate the scenario of the *LAgent* joining a team. Instead, adding a new resource will mean registering it with the *CIC* as a standalone node (one member team). Finally, we reduce (and re–focus) the negotiations. Their role will be to provide information about current and planned utilization of the resource. This will allow the *LAgent* representing user to decide where to run which job and when.

Let us now describe how the system will allow tsunami modeling using multiple resources in the university laboratory. The user starts by accessing a web based interface, which allows communication with her *LAgent*. Next, she specifies the hardware and software requirements of the job (as constraints on the ontological terms describing needed resources). This task is done using the interface based on the OntoPlay module [21] (its Condition Builder component), giving the user freedom in describing the needed resources, while guiding her through the contents of the ontology without deep knowledge of its structure (of semantic technologies in general). As shown in Figure 7, the Condition Builder is composed of a series of condition boxes used to create constraints on class-property relationships. Depending on the chosen class, the user can select, which class property she wishes to restrict. For example, having selected the *PhysicalMemory* class, the expanded property box will contain properties such as *hasTotalSize* and *hasAvailableSize* (see, Figure 7).

After selecting the class and property the user can choose the required operator and value. Here, she sees only operators applicable to the type of the property. Specifically, this means that for value properties it would be operators such as *equalTo*, *lessThan* or *greaterThan*, while for object, properties the user would be allowed to select, e.g. is equal to individual or is constrained by. Should a user wish to restrict the value of a particular property to a fixed individual from the ontology, Condition Builder lists all individuals that can be used in the context (see, Figure 8).

To illustrate the process, let us assume that the user wishes to find all resources that are running the Linux operating system, have at least 8 GB of memory and a 4-core processor. In this case the user starts with an empty *ComputingElement* specification. First, he would constrain the property isRunningOS of the class *ComputingElement* to any individual belonging to the class Linux. We assume it does not need to be any particular flavor of Linux, so we do not add additional conditions on this class (however, when installing the AiG system at the University of Aizu, we will represent each machine as an individual in an ontology). Second, the user adds condition on the property *hasMemory*, constraining it to the *PhysicalMemory* subclass and adding a nested condition specifying that the *hasTotalSize* property should have a value greater than 8000 MB. Similarly, the user constraints the *hasCPU* to an instance of the CPU class with a value of the *hasCores* property equal to 4.

After user submits the resource requirements, the *LAgent* passes this description to the *CIC*, which performs semantic reasoning on its knowledge base, to find resources satisfying the given criteria and returns a list of matching nodes, including the information how to contact the *LMaster*s at each node.

Note that even in a semi-dynamic environment, such as a university laboratory, there is no guarantee that the resources found by the CIC are, at the moment, available for use. The machine may be offline, used for other purposes, or the agent process may not be running. Therefore, there is a need for additional verification of the availability of resources. This is handled through agent negotiations, albeit in a very simplified form. When the *LAgent* receives the list of *LMaster* addresses, it issues a *Call For Proposal* message to gain confirmation of whether the resources are able to perform the task. The

LMasters confirms (or rejects the proposal), and provides information when it could start executing the job. This helps to handle the case of temporarily occupied / not available nodes. Once the *LAgent* receives offers from the (benevolent) agents, it presents the list to the user, who can choose the nodes on the basis of their availability and parameters.

The final step of submitting jobs is the specification of task parameters. As described in the previous sections, for the tsunami simulations it is necessary to run different algorithms on different data sets and parameters to come up with multiple results. Consequently, user is required to provide multiple job descriptions (one for each model / parameter). The job description is going to be provided using the same Condition Builder mechanism, although using a different part of the ontology and will contain information such as the command or script for running the computation, together with any additional parameters required by the algorithm, such as location of the necessary data and the place for storing the output(s).

Completed job description is sent by the *LAgent* to the respective *LMaster*, which then executes the task. Once the computation is finished, the *LMaster* creates a *JobResult* message, which contains information about the job execution, the outcome and links to the result data and any resources created by the simulation algorithm. The *LAgent*, on the other hand, is responsible for gathering all responses from the nodes taking part in the experiment and presenting them to the user, thus ending the scenario.

VI. Concluding remarks

The aim of this paper was two-fold. First, to introduce important issues that arise in the tsunami modeling and that require a robust resource management to run the needed simulations. Here, the "Matsushima effect" was used as the main grounding scenario. Second, we have argued that the agent-semantic infrastructure, developed within the Agents in Grid, project can provide the needed solution. This is especially in the situation when multiple resources are under administration of separate authorities and may be availale according to their own schedules. Next, we have discussed changes (simplifications) that need to be introduced to the AiG infrastructure to adapt it to the needs of the tsunami modeling, as a task completed using machines available within the University of Aizu. Finally, we have presented a detailed scenario how the user would use the AiG infrastructure to run a simulation. Goal of our future research is to complete the changes outlined in this paper and install the AiG infrastructure on the machines in the laboratory of Prof. Vazhenin and Prof. Watanobe, and run experiments with tsunami modeling.

Acknowledgment

Work of Marcin Paprzycki was completed while visiting the University of Aizu.

References

[1] Th. Erl, *SOA Design Patterns,* Prentice Hall, 2010

[2] M. Kuniavsky, *Smart Things: Ubiquitous Computing User Experience Design,* Elsevier, 2009

[3] Folding@home Distributed Computing, http://folding.stanford.edu/

[4] A. Beberg, D. Ensign, G. Jayachandran, S. Khaliq, "Folding@home: Lessons from eight years of volunteer distributed computing", *in Proc. of IEEE International Symposium on Parallel & Distributed Processing (IPDPS),* Rome, Italy, 2009, pp. 1–8

[5] X.Caiand, and P.Langtangen, "Making Hybrid Tsunami Simulators in a Parallel Software Framework", *LNCS,* vol. 4699, pp. 686–693, Springer–Verlag. 2008

[6] K.Ganeshamoorthy, D. Ranasinghe, K.Silva, and R.Wait, "Performance of Shallow Water Equations Model on the Computational Grid with Overlay Memory Architectures", *Proc. of the Second International Conference on Industrial and Information Systems (ICIIS 2007), IEEE Press,* Sri Lanka, 2007, pp. 415–420

[7] D. George, TsunamiClaw User's Guide, http://faculty.washington.edu/rjl/pubs/icm06/TsunamiClawDoc.pdf/pubs/icm06/TsunamiClawDoc.pdf

[8] N. Shuto, F. Imamura, A. C. Yalciner, G. Ozyurt, TUNAMI N2; Tsunami modelling manual, http://tunamin2.ce.metu.edu.tr/

[9] V. Titov, "Numerical Modeling of Tsunami Propagation by using Variable Grid",*Proc. of the IUGG/IOC International Tsunami Symposium, Computing Center Siberian Division USSR Academy of Sciences, Novosibirsk,* USSR, pp. 46–51, 1989

[10] V. Titov and F. Gonzalez, "Implementation and Testing of the Method of Splitting Tsunami (MOST)",*Technical Memorandum ERL PMEL-112,* National Oceanic and Atmospheric Administration, Washington DC, 1997

[11] J.C. Borrero, K. Sieh, M. Chlieh, and C.E. Synolakis, "Tsunami Inundation Modeling for Western Sumatra",*Proc. of the National Academy of Sciences of the USA,* Vol. 103, N 52, http://www.pnas.org/content/103/52/19673.full, 2006

[12] R. Cortez, A. Vazhenin, "Developing Re-usable Components Based on the Virtual-MVC Design Pattern",*LNCS,* vol. 7813, pp. 132–149, Springer-Verlag, 2013

[13] A. Vazhenin, K. Hayashi, Al. Romanenko, "Service-oriented tsunami wave propagation modeling tools",*in Proc. of the Joint International Conference on Human-Centered Computer Environments (HCCE '12),* Aizu-Wakamatsu, Japan, ACM Publisher, 2012, pp. 131–136.

[14] A. Vazhenin, M. Lavrentiev, A. Romanenko, An. Marchuk, "Acceleration of Tsunami Wave Propagation Modeling based on Re-engineering of Computational Components",*International Journal of Computer Science and Network Security,* vol.13, N 3, pp. 24–31, 2013

[15] http://jdoss1.jodc.go.jp/cgi-bin/1997/depth500_file

[16] http://www.gdem.aster.ersdac.or.jp/search.jsp

[17] http://iisee.kenken.go.jp/staff/fujii/OffTohokuPacific2011/tsunami.html

[18] Katarzyna Wasielewska, Micha l Drozdowicz, Maria Ganzha, Marcin Paprzycki, Naoual Attaui, Dana Petcu, Costin Badica, Richard Olejnik, and Ivan Lirkov. Trends in Parallel, Distributed, Grid and Cloud Computing for engineering. chapter Negotiations in an Agent-based Grid Resource Brokering Systems. Saxe-Coburg Publications, Stirlingshire, UK, 2011

[19] Wojciech Kuranowski, Maria Ganzha, Maciej Gawinecki, Marcin Paprzycki, Ivan Lirkov, and Svetozar Margenov. Forming and managing agent teams acting as resource brokers in the grid–preliminary considerations. International Journal of Computational Intelligence Research, 4(1):9–16, 2008

[20] Micha l Drozdowicz, Maria Ganzha, Katarzyna Wasielewska, Marcin Paprzycki, and Pawe l Szmeja. Using ontologies to manage resources in grid computing: Practical aspects. In Sascha Ossowski, editor, Agreement Technologies, volume 8 of Law,Governance and Technology Series, pages 149–168. Springer Netherlands, 2013

[21] M. Drozdowicz, M. Ganzha, M. Paprzycki, P. Szmeja, K. Wasielewska, OntoPlay – a flexible user-interface for ontology-based systems, http://ceur-ws.org/Vol-918/111110086.pdf

[22] A. M. Fridman, L. S. Alperovich, L. Shemer, L. A. Pustil'nik, D. Shtivelman, An. G. Marchuk,and D. Liberzon, "Tsunami wave suppression using submarine barriers," Physics-Uspekhi, vol. 53, no. 8, pp. 809–816, Aug. 2010

[23] Ian Foster, Nicholas R. Jennings, and Carl Kesselman. Brain meets brawn: Why grid and agents need each other. Autonomous Agents and Multiagent Systems, International Joint Conference on, 1::8–15, 2004

[24] http://depts.washington.edu/clawpack/geoclaw/

[25] http://nctr.pmel.noaa.gov/ComMIT/