2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Simulation Processes In The Cloud For Emergency Planning

Judicaël Ribault, Gabriel Wainer
Department of Systems and Computer Engineering
Carleton University Centre for Visualization and Simulation (V-Sim)
Ottawa, ON K1S-5B6
jribault@gmail.com, Gabriel.Wainer@sce.carleton.ca

Abstract— **In recent years, various environmental disasters (tsunamis, earthquakes, forest fires and nuclear incidents) have caused numerous losses in lives and infrastructure. In such emergency, remote access to the simulation resources can increase the emergency response success. We propose a new architecture based on the RISE simulation services middleware and on Taverna workflow to execute the entire simulation process into the Cloud. We present its use through an emergency flooding scenario use case in which emergency crew can order a new simulation and visualize result on Google Earth.**

Keywords: Modeling; simulation; workflow; emergency planning; RISE; web-service; REST; DEVS

I. INTRODUCTION

In recent years, various environmental disasters (tsunamis, earthquakes, forest fires and nuclear incidents) have caused numerous losses in lives and infrastructure. Studying the behavior of such environmental systems is a key to minimize the impact of such disasters, and in providing better aids for emergency management. Studying such environmental systems is difficult, because the environment is complex, involve interactions between several processes and require multi-disciplinary expertise. To model such complex systems, we cannot rely on analytical solutions, which usually need simplifications and deals with the problem at a high level of abstraction. Instead, Modeling and Simulation (M&S) have been successful in studying study such detailed phenomena by providing a constrained experimentation environment for better emergency management [1][2].

M&S of environmental systems has existed since the early days of computer simulation [3], and since the 1980's, these simulations have been combined with Geographic Information Systems (GIS) [4][5][6][7][8][9]. In recent years, the use of GIS has become popular for the input, storage, manipulation, and output of geographically referenced data due to the numerous online engines (Google Earth, Google Maps, Bing Maps, MapQuest, and others).

Building and running advanced environmental simulation models is complex, and many methods have been proposed for M&S of these applications. In our case, we are interested in efforts based on the DEVS formalism [10], which has shown potential for managing the complexity of environmental systems simulation [11][12][13][14] (in terms of quality and performance). DEVS has been successfully used in this area due to its ease of modeling, the varied mechanisms to combine existing models, and the efficiency of the simulation engines. Also, several studies showed interesting results when mixing GIS and DEVS M&S [12][15][16].

Most of these methods run on single-user workstations; nevertheless, in case of actual emergencies (where there are timing and spatial constraints), it is better to have remote access to the simulation resources. In this way, results can be visualized by the emergency crews on site; the emergency crews can provide new data and run new simulations on demand (when conditions change), and such arrangement can increase the emergency response success.

In order to deal with these issues, we propose to deploy all the simulation software, files and process in the Cloud. The simulation engine can be controlled by Web Services (WS) and the simulation files are archived on a server. The simulation experiment formalizes the whole simulation process from the creation of the simulation model to the execution and the processing of simulation results. The simulation experiments are stored on a server and can be run remotely to execute all the simulations tasks describe above in order to show simulation results to the emergency crews on site. This provides other advantages. From an M&S methodology perspective, previous solutions are deployed on one machine and are self-contained. This helps reproducing simulation results, which is usually complicated, as it needs the installation and configuration of all the software and dependencies needed by the simulation.

To demonstrate this concept, section III presents a lifecycle of a flooding emergency simulation scenario and propose architecture to deploy and control the simulation environment and the simulation process of such simulation in the Cloud using software and tools described section II. Section IV relates on the work we did to control and manage simulation in the Cloud, from the creation of the simulation model based on GIS data and real-time weather forecast to the execution and the processing of simulation results and their visualizations on Google Earth. Finally, we apply our approach through the simulation of a flooding emergency and discuss the benefits of our approach.

II. BACKGROUND

A flood is an overflow of an expanse of water that submerges land not normally covered by water. According to the Stockholm International Water Institute (SIWI), these are the natural disasters producing the most damage for human lives and property [17]. Flooding can occur suddenly after excessive rainfall, a dam or levee failure, or a sudden release of water held by an accumulation of broken river ice caught in a narrow channel. In such cases, any emergency team on site needs accurate information about the flood in the next minutes (or hours) in order to help people and preventing serious damages. A simulation can predict and give useful in-

978-0-7695-4691-9/12 $26.00 © 2012 IEEE
DOI 10.1109/CCGrid.2012.88

formation to the emergency team, such as the next zone to be underwater, the water level, the safest way to evacuate the zone, etc. Different authors have discussed how to simulate flooding. In [2], the authors defined a two-dimensional simulation model based on a 2D grid that that allows identifying area of inundation. More recently, LISFLOOD [7][23] introduced a simulation framework based on GIS data, which allows obtaining accurate land and soil information for grid models, and it allows identifying the flooded area. Other studies have focused on how to simulate flooding during an emergency scenario, using real-time data. In [9], the author's presents Shyska, a solution using GIS to support real-time decision based on simulation. Similarly, in [18] the authors use a GIS-based model and access the resulting data online using a WWW interface, a GIS and numerical functions. In [20], the authors proposed an interaction and visualization system for flooding emergency based on a 3D physical engine. The 3D visualization communicates with the simulation using a Web Services layer. Most of these solutions are based on GIS, and some of them ([18][20]) use Web Services to improve the interoperability or the user experience. However, none of them distributes the simulation experimentation in the Cloud, and the simulation environment, the models, and the simulation lifecycle are run locally.

Many of the flooding simulation models use Geographic Information Systems (GIS), which allow manipulating georeferenced information and performing different operations with maps [30]. GIS are usually organized in multiple data layers, centralizing all the environmental data available and making it accessible in several forms (maps, digital maps or raw data files). Whatever format is chosen, it is necessary to transform GIS data into a format compatible with the simulation software. As discussed in the Introduction, we are interested in using DEVS [10], which is a formal specification for discrete event systems that has been successfully applied to this kind of problems in the past [11][12][13][14][15][16]. One of the advantages of using DEVS is that it is a universal abstract formalism, independent from the implementation, which makes it perfect for deployment in the Cloud. Numerous tools have been used for DEVS M&S of environmental systems such as JDEVS [32], DEVSimPy [31], or CD++ [11]. In particular, in [13], we showed how to simulate environmental systems efficiently using DEVS and Cell-DEVS. Those studies used the CD++ M&S environment [11], and this software stack was recently expanded to allow GIS data to be transformed into CD++ simulation engine [21]. This method relies on the GeoTIFF standard file format, which is supported by most GIS. In [21] we used GRASS [22], an open source GIS, and Google Earth as the Geospatial Visualization System (GVS). Google Earth uses KML (Keyhole Markup Language), an XML-based language focused on geographic visualization that includes annotation of maps and images [33]. Its file format can be used to display geographic data in Earth browsers (such as Google Earth), using a tag-based structure with nested elements and attributes. The geographic visualization needs to include not only the presentation of graphical data, but also the control of the user's navigation (e.g., where to go, where to look). Once A KML file is created, it can be imported into Google Earth, allowing the visualization of the simulated results a layer impressed over the standard layers (e.g. satellite views, street maps, etc.). Google Earth provides mechanisms to make layers evolve forward and backward in time, which is useful to analyze the progress of a simulation interactively (e.g., a cell temperature in a fire spreading model).

As discussed earlier (and proposed by other studies [18] [20]), Web Services can be used for remote simulation of environmental systems, improving data accessibility, interoperability and user experience through Cloud computing. Cloud computing allows scaling the number of resources as required sand is fault-tolerance. We can identify two classes of Web Services: REST-compliant WSs [24] (whose primary purpose is to manipulate XML representations of Web resources using a uniform set of "stateless" operations), and arbitrary WS, in which the service may expose an arbitrary set of operations [23]. Access to RESTful WS is done through Web resources (URIs) and XML messages using HTTP methods (GET, PUT, POST and DELETE). Due to this simple protocol, RESTful WS are simple, efficient and scalable as evidenced by the WWW. In [27], we identified the benefits of distributing the simulation using WS while most existing distributed simulation systems use SOAP-based WS and other approaches (see [27] for a more comprehensive list). As discussed in [25], most of these simulation middleware lack plug-and-play interoperability, dynamicity, and composition scalability. Instead, RESTful WS can solve these issues by imitating the Web interoperability style. Based on these ideas, in [25], we presented the first existing RESTful Interoperability Simulation Environment (RISE) middleware. The main goal of RISE is to provide simulation interoperability and mash-ups regardless of the base formalism, theory or implementation. RISE allows modelers to run experiment instances, whose settings and resources (URIs) are persistent and repeatable (unless deliberately removed or updated). An interface between RISE and CD++ allows running distributed simulations. Web Services also enable the integration of application or data from heterogeneous sources (mash-up) within the simulation. For instance, Taverna [28] eases the use and integration of the growing number of tools and databases available on the Web, especially Web Services. Users that are not necessarily expert programmers, can design, execute and share workflows of Web Services, allowing them to integrate many resources into a single analysis. Taverna workflows can be also shared in myExperiment [29], an online social networking environment to find, share and reuse workflows. MyExperiment allows anybody who finds workflows relevant to their research to reuse and repurpose them to their specific requirements. At the same time, developers who build their own workflows can submit them to myExperiment to contribute to the scientific community.

The approach we present in the next section III is a new simulation method that takes benefit of the tools and software described above to manage emergency simulation in the Cloud. Our approach whose implementation is detailed section IV uses RISE to provide a distributed simulation environment, accessible via RESTful WS, and using the CD++ simulation engine. Taverna is used to organize all Web Ser-

vices involved in the simulation process, such as the creation of the simulation model using GIS data and weather services, simulation experimentation, execution, or processing the result. Our approach contribute to improve the emergency planning simulation methodology by offering to the end-users a new way to manage and execute simulation remotely, from the creation of the simulation model to the processing of results, what is useful when one needs simulation in emergency areas.

III. EMERGENCY SIMULATION IN THE CLOUD

As discussed in section I, in the case of an emergency scenario, the use of online access to remote simulation resources can improve the management of the emergency situation. In the case of emergency planning simulations, the following tasks are typically needed:

1. Creation of the simulation model; with preference, a GIS should be used to obtain information about the different data layers of the dedicated area to study.
2. Creation of a simulation experiment; based on the simulation model and specifying simulation inputs.
3. Execution of the simulation.
4. Processing of the simulation results.
5. Visualization of the results.

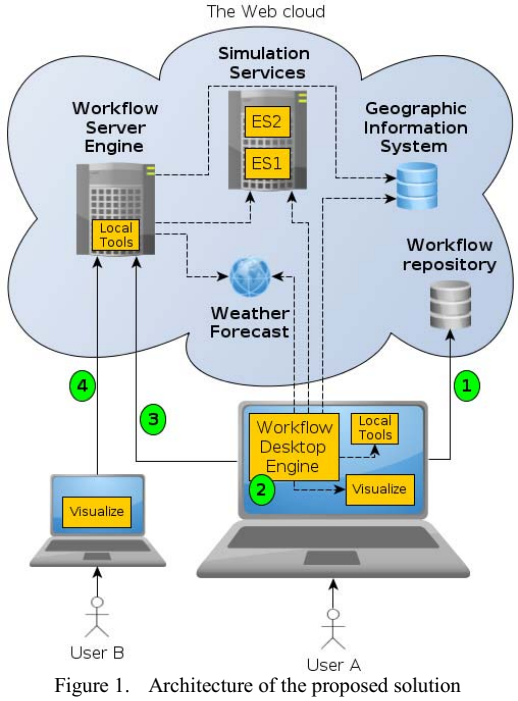

Figure 1. Architecture of the proposed solution

Unless you plan to build a specific simulation for a dedicated zone with predefined inputs, these tasks should be done online during the emergency. If we can formalize and automate the tasks described above and host the information on the Cloud, we can obtain real-time experimentation and response, improving whole process. To do this, we propose the conceptual architecture presented in figure 1. The simulation process is run in the Cloud, using the techniques discussed in section II. The idea is to use RISE, simulation services, and to ensure repeatability using workflow of services to formalize and automate the simulation process. This architecture also facilitates mashing up services available in the Cloud.

ES1 and **ES2** represent two different simulation environments in the Cloud, managed by **simulation services**. The execution of the simulation in the Cloud allows not installing a simulation tool locally, sharing and reusing the same software among all practitioners, thus improving repeatability. The **Workflow Desktop Engine** can communicate with the simulation Cloud services, and with other Web Services (such as a **Weather Forecast service** or a **Geographic Information System)**. Indeed, the simulation workflow can be seen as a blueprint describing the tasks represented by Web Services within the simulation Cloud for the creation and execution of the simulation or within other Web resources, such as weather services for the input of the simulation. In that way, the workflow formalizes the simulation process, and enhances repeatability and ease of use by automating execution. The simulation experiment workflow can be executed by a **Workflow Server Engine** also in the Cloud (avoid the need to install the workflow engine locally). The **Workflow Repository** contains existing simulation experiment workflows, and it allows finding, downloading existing workflows and uploading new workflows.

User A in the figure represents a developer building a new simulation workflow. *User B* represents an emergency crew using the simulation workflow developed by *A*. *A* can get existing workflow resources from a repository (1), which allows reusing related workflows (instead starting from scratch). Then, *A* builds its own simulation workflow locally (2). This simulation workflow formalizes and automates the process described at the beginning of this section: it communicates with a GIS server to obtain geographic information, and with a weather service to obtain meteorological information on the zone to study. The workflow also communicates with the simulation Cloud services to create, execute, and get simulation results. When the experiment workflow is finished, *A* uploads the workflow to the repository (1), and other practitioners can find and download it to reuse it (as is, or as modifying it for another emergency planning simulation workflow). Finally, *A* uploads and setups its simulation workflow on the Workflow Server Engine (3). In that way, *B* (the emergency crew on site) can execute the simulation workflow developed by user *A* (in the Cloud). *B* inputs any parameters defined by *A* in the workflow (4), and visualizes the simulation results, without being involved in any tasks of the simulation process.

In the following section, we show how we can use the tools and software described in the background section III to implement the conceptual architecture of figure 1. RISE will be used as the Simulation Cloud Service, CD++ as a Simulation Environment (in the Cloud), myExperiment as a Workflow Repository, and Taverna as a Workflow Engine. Other tools will be used to transform GIS data to CD++ and CD++ simulation results into KML (for Google Earth).

IV. IMPLEMENTATION

A typical emergency planning simulation involves several tasks. We first need to build a simulation model (like *User A* in figure 1), and we do that using Taverna workflows.

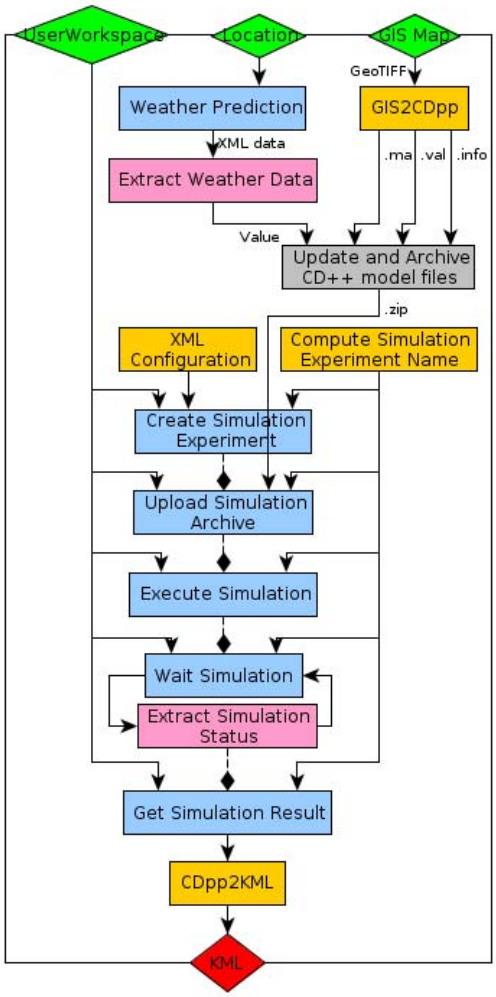

Figure 2. Simplified view of the Taverna simulation workflow.

To automate the simulation process, we need to build a simulation workflow (using Taverna), as described in the rest of the section (due to space limitation, we use a simplified view of the final workflow. The complete Taverna workflow can be reviewed on myExperiment at this URL: http://www.myexperiment.org/workflows/2739.html). The workflow, described in figure 2, starts by taking a *User Workspace*, a *Location*, and a *GIS Map* as inputs, and produces, as output, a KML file to visualize the simulation results (which can be seen using Google Earth).

To build this workflow, we first implemented the definition of the flooding models (using CD++ language), using GIS and weather data to initialize the model. A Taverna workflow takes the GIS map, its location and produces the files needed by CD++ to start a simulation. As we can see on figure 2, the Taverna Local Tool Service (in orange) takes in as a parameter a GIS map given (input workflow parameter). The GIS2CDpp executable is called, and after completed, the application transforms the GeoTIFF maps into the simulator's format, in this case, initial data values (*val*), an updated Cell-DEVS (*ma*), and an information file (*info*) to keep global geographical references. The Local Tool Service generates three outputs provided by GIS2CDpp, as well as the standard output (*stdout*) and error (*stderr*).

The models resulting from this phase need to be updated using the weather data. This weather prediction can be accessed using a Taverna RESTful WS (in blue) which queries the online weather service. A Taverna REST WS can take many inputs, and it provides 2 outputs: a *status* and a *responseBody*. The *status* contain the HTTP response to indicate the invocation status (200, 404, etc.) and the *responseBody* contains the result. The Taverna RESTful WS we added to the simulation workflow takes in parameter the location given in workflow parameter (*Location*), and it is configured with the URL template of the weather forecast service *http://weather.yahooapis.com/forecastrss?w={location}* and the HTTP method GET. The Taverna REST WS generates the content of the weather forecast response, which needs to be processed in order to extract the information we need. This processing is done using the Taverna XPath Service (in pink), which can be used to extract data from an XML file. Finally a Taverna Beanshell Service (in grey) takes the model files and the weather information as inputs, and it produces the final CD++ model file ready to be used by the RISE simulation environment.

The second step of the simulation process is to create a simulation experiment in the Cloud. As described in the Background section, RISE is a simulation middleware that allows CD++ simulation to be executed in the Cloud. Access to RISE is done through Web resources (URIs) and XML messages using HTTP methods: GET (to read a resource), PUT (to create new resource or update existing data), POST (to append new data to a resource), and DELETE (to remove a resource). The RISE API looks like a classic website URL such as: http://www.example.com/cdpp/sim/*{userworkspace}/{servicetype}/{framework}*. URLs respond to HTTP methods, for example with the URL above, to submit files to a framework we use the POST method. PUT is used to create a framework or update simulation configuration settings. DELETE is used to remove a framework. GET is used to retrieve the framework's state.

We can identify four steps to manage a new simulation using the Web Services offered by RISE: (1) Create a new empty experiment based on 4 parameters (the user workspace, the service type, the framework name, and a XML configuration file); (2) Upload the simulation model files we created earlier; (3) Start the simulation and wait for its end; (4) Retrieve the simulation results. To automate these steps, we first added a Taverna REST WS that takes as parameters the emergency crew user workspace, the name of the new simulation experiment, and an XML file to configure the RISE experiment. The Taverna REST WS configuration contains the URL template *http://www.example.com/cdpp/sim/workspaces/{userworkspace}/dcdpp/{framework},* and uses the HTTP method PUT to send the XML file to create a new simulation experiment on RISE. Taverna invokes the REST WS according to the parameters given prior to the simulation workflow execution (for example, if the user workspace given in parameter is "Bob" and the simulation expe-

riment is "flood13022012", Taverna will invoke *http://www.example.com/cdpp/sim/workspaces/Bob/dcdpp/flood13022012*). The user workspace will be set by the emergency crew, and it must be part of the workflow parameters (in addition to the GIS map and the location); while the framework name and the RISE XML configuration file will be computed automatically by a Taverna Beanshell Script.

The next Taverna REST WS is invoked to upload the archived CD++ model files. This Service takes in a CD++ model (in a ZIP file), the user workspace, and the framework name. Its configuration contains the template *http://www.example.com/cdpp/sim/workspaces/{userworkspace}/dcdpp/{framework}* and use the HTTP method POST to send the ZIP file to the RISE server. RISE extracts the simulation files into the simulation experiment. We also added a Taverna REST WS into the simulation workflow to run the simulation, which takes the user workspace, the framework name, and its configuration in the URL template *http://www.example.com/cdpp/sim/workspaces/{userworkspace}/dcdpp/{framework}/simulation.* Then, the HTTP method PUT executes the simulation.

We then need to wait for the simulation to finish. This is done using a REST WS which invokes the GET HTTP method. RISE WS results are XML-based, so we used Taverna's XPath service to process the XML result and extract their data. These two Taverna services are re-executed every second until the DONE message is found.

In order to retrieve the simulation results, we added another Taverna REST WS that takes as parameters the user workspace and the framework name. Its configuration contains the URL template *http://www.example.com/cdpp/sim/workspaces/{userworkspace}/dcdpp/{framework}/results,*and it uses the HTTP method GET. The result of the simulation is provided on the Taverna REST WS output *ResponseBody*.

As discussed in the Background, we want to process the simulation results and generate a KML file to visualize the simulation results (using, for instance, Google Earth). To automate this last step of the simulation process, we added to a Taverna Local Tool Service call to invoke the tool CDpp2KML [21], which parses the data generated by the simulator (representing the model's state changes) into the desired output visualization format. It then generates a KML file with geo-referenced and timed representation of each simulated cell state change. The local tool takes as parameter the simulation result from the previous Taverna REST WS output (*ResponseBody*), calls CDpp2KML, and provides a KML file ready to be used. The output of this last Taverna service is connected to the output of the simulation workflow seen on figure 2.

In summary, the simulation workflow in figure 2 takes as parameters the GIS map file, the location, and the emergency crew user workspace, and it provides the KML file to visualize the simulation result (on Google Earth). The workflow can be shared on myExperiment to be used, rated, commented, modified and uploaded as a new version by other users. The workflow on myExperiment can be found at this URL: http://www.myexperiment.org/workflows/2739.html).

In order to be used by the emergency crew, this Taverna simulation workflow needs to be uploaded on a Taverna Server to be executed directly in the Cloud. As this workflow depend on local tools (GIS2CDpp and CDpp2KML), the developer (*User A*) need to set up the Taverna Server and install on the same computer in the Cloud the necessary tool GIS2CDpp and CDpp2KML. Once the simulation workflow is on the Taverna Server, the emergency crew (*User B*) can specify the workflow input and execute the workflow using an Internet Browser. The next section presents how an emergency crew on site can make use of a simulation workflow in the cloud in case of a flooding emergency scenario.

V. CASE STUDY: FLOODING EMERGENCY SCENARIO

Section III presented the architecture we used to define this simulation process in the Cloud, and Section IV explained how to implement the simulation workflow using Taverna. In this section we show simulation results obtained when running this simulation workflow to create and execute new simulation in the cloud in case of a flooding emergency.

An emergency scenario, as may be a flash flood, can generate damages, and worse, an attempt on human life. Under these conditions, emergency teams must act quickly to save lives and reduce damages to infrastructure. Consider the case of an emergency crew to the scene of the flooding. The rain intensifies and new areas may be submerged and must be evacuated. The emergency crew must choose which areas should be evacuated in priority. The emergency crew (represented by *User B* on Figure 1) must locate the Taverna simulation workflow to manage flooding simulation (http://vs3.sce.carleton.ca/taverna-server/). According to the implementation described section IV, the emergency crew needs to give in parameters prior to the workflow execution the User Workspace (user ID), the GIS map of the region to study, and the location code of the region. In our flooding scenario case, the emergency crew will give in parameter the emergency team ID (for example *EC001*), the GIS map file of the flooding area, and the WOEID location code of the region (for example, *3369)*. The emergency crew can then submit the form and wait for the Taverna workflow to be executed on the server side. According to the implementation, the Taverna simulation workflow will first create the simulation model. The model we choose simulate natural region that acts as the water-receiving area of a drainage basin. The model considers water from different origins: rain, rivers and snow and the type of ground to compute the level of water for each cell. Once the model is created based on the GIS map and the weather forecast, the Taverna simulation workflow deploys it on the RISE server. Then the Taverna simulation workflow order RISE to execute the workflow, wait for the simulation to be finish and get the simulation results available on RISE. The last step of the Taverna simulation workflow is to process the results into a KML file viewable with Google Earth.

Once the workflow execution is finish, the web portal offers the possibility to download the KML file. The emergency crew imports this KML file into the locally installed Google Earth and visualizes the result. Thanks to the Google Earth feature to go forward and back we can navigate through the simulation results

VI. CONCLUSION

We introduced new software architecture for emergency crews to execute remote simulations. The various steps of the simulation process are all executed in the Cloud, and a workflow manages the different steps of the simulation process automatically. This allows emergency units to use thin clients. The simulation workflow execution is requested through a Web portal which makes the models accessible.

We showed how to use this architecture for emergency scenarios in which an emergency crew can run flooding simulations. In order to run this simulation, 3 inputs are needed: the emergency crew ID, a GIS Map file, and the location code of the area to study. The emergency crew obtains, in response, a KML file to visualize simulation result with Google Earth.

The architecture we presented facilitates access to the simulation by reducing the infrastructure costs to a minimum: it only requires access to the Internet to execute a simulation workflow. It also increases the credibility of a study, by providing an automated simulation workflow that will guarantee the repeatability. It also permits mixing data from simulations and from the real world using mash-up to use simulation as a prediction and decision-making tool.

REFERENCES

[1] Sheffi, Y., Mahmassani, H. and Powell, W.B. 1982. A transportation network evacuation model. *Transportation Research Part A: General*. 16, 3 (1982), 209–218.

[2] O'brien, J., Julien, P. and Fullerton, W. 1993. Two-dimensional water flood and mudflow simulation. *Journal of Hydraulic Engineering*. 119, 2 (1993), 244–261.

[3] Botkin, D.B., Janak, J.F. and Wallis, J.R. 1972. Some ecological consequences of a computer model of forest growth. *The Journal of Ecology*. 60, 3 (1972), 849–872.

[4] Band, L. 1986. Topographic partition of watersheds with digital elevation models. *Water resources research*. 22, 1 (1986), 15-24.

[5] Desmet, P.J.J. and Govers, G. 1995. GIS-based simulation of erosion and deposition patterns in an agricultural landscape: a comparison of model results with soil map information. *Catena*. 25, 1-4 (Jun. 1995), 389-401.

[6] Wang, X. 2005. Integrating GIS, simulation models, and visualization in traffic impact analysis. *Computers, Environment and Urban Systems*. 29, 4 (Jul. 2005), 471-496.

[7] Van Der Knijff, J.M., Younis, J. and De Roo, a. P.J. 2010. LISFLOOD: a GIS-based distributed model for river basin scale water balance and flood simulation. *International Journal of Geographical Information Science*. 24, 2 (Feb. 2010), 189-212.

[8] Chapman, L. and Thornes, J.E. 2003. The use of geographical information systems in climatology and meteorology. Progress in physical geography. 27, 3 (2003), 313–330.

[9] García, S.G. 2004. Technical Note GRASS GIS-embedded Decision Support Framework for Flood Simulation and Forecasting. Transactions in GIS. 8, 2 (2004), 245-254.

[10] Zeigler, B.P., Praehofer, H., and Kim, T.G. 2000. Theory of modeling and simulation, 2nd Edition, Academic Press.

[11] Wainer, G.A. 2009. *Discrete-Event Modeling and Simulation*. CRC.

[12] Hu, X., Sun, Y. and Ntaimo, L. 2011. DEVS-FIRE: design and application of formal discrete event wildfire spread and suppression models. *Simulation*. October (Oct. 2011).

[13] Wainer, G. 2006. Applying Cell-DEVS Methodology for Modeling the Environment. *Simulation*. 82, 10 (Oct. 2006), 635-660.

[14] Filippi, J.-B., Morandini, F., Balbi, J.H. and Hill, D.R. 2009. Discrete Event Front-tracking Simulation of a Physical Fire-spread Model. *Simulation*. 86, 10 (Aug. 2009), 629-646.

[15] Chiari, F., Delhom, M., Filippi, J.-B. and Santucci, J.-F. 2000.A GIS based methodology for the modeling and the simulation of watersheds. In Proceedings of the Advanced Technology Workshop (ATW) 2000 Conference, Corsica, France.

[16] Hill, D., Thibault, T., and Coquillard, P. 2002. Predicting invasive species expansion using GIS and simulation coupling. Modeling and Simulation 1(1):30–5

[17] http://www.siwi.org/statistics SIWI. Retrieved 2012-02-06

[18] Alsabhan, W., Mulligan, M. and Blackburn, G. 2003. A real-time hydrological model for flood prediction using GIS and the WWW. Computers, Environment and Urban Systems. 27, 1 (Jan. 2003), 9-32.

[19] Bates, P.. and De Roo, a. P.. 2000. A simple raster-based model for flood inundation simulation. Journal of Hydrology. 236, 1-2 (Sep. 2000), 54-77.

[20] Nóbrega, R., Sabino, A., Rodrigues, A. and Correia, N. 2008. Flood emergency interaction and visualization system. Visual Information Systems. Web-Based Visual Information Search and Management. (2008), 68–79.

[21] Zapatero, M., Castro, R., Wainer, G. and Hussein, M. 2011. Architecture For Integrated Modeling, Simulation And Visualization Of Environmental Systems Using GIS And Cell-DEVS. *Winter Simulation Conference 2011* (2011).

[22] Neteler, M. and Mitasova, H. 2010. Open Source GIS: A GRASS GIS Approach, 3rd. Edition. Springer Publishing Company, Incorporated.

[23] "Relationship to the World Wide Web and REST Architectures". http://www.w3.org/TR/ws-arch/#relwwwrest Web Services Architecture. W3C. Retrieved 2012-02-06.

[24] Richardson, L. and Ruby, S. 2007. *RESTful Web Services*.

[25] K. Al-Zoubi and G. Wainer. Distributed simulation using restful interoperability simulation environment (rise) middleware. Intelligence-Based Systems Engineering, pages 129–157, 2010.

[26] F. Kuhl, R. Weatherly, and J. Dahmann. Creating computer simulation systems: an introduction to the high level architecture. Prentice Hall PTR, 1999.

[27] G.A. Wainer, R. Madhoun, and K. Al-Zoubi. Distributed simulation of devs and cell-devs models in cd++ using Web Services. Simulation Modelling Practice and Theory, 16(9):1266–1292, 2008.

[28] Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P. and Oinn, T. 2006. Taverna: a tool for building and running workflows of services. *Nucleic acids research*. 34, Web Server issue (Jul. 2006), W729-32.

[29] Goble, C. a, Bhagat, J., Aleksejevs, S., Cruickshank, D., Michaelides, D., Newman, D., Borkum, M., Bechhofer, S., Roos, M., Li, P. and De Roure, D. 2010. myExperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic acids research*. 38, Web Server issue (Jul. 2010), W677-82.

[30] Longley, P.A, Goodchild, M.F., Maguire, D.J., and Rhind, D.W. 2005. Geographic Information Systems and Science. Wiley.

[31] Capocchi, L., Santucci, J.F., Poggi, B. and Nicolai, C. 2011. DEVSimPy: A Collaborative Python Software for Modeling and Simulation of DEVS Systems. 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (Jun. 2011), 170-175.

[32] Filippi, J.B. and Bisgambiglia, P. 2004. JDEVS: an implementation of a DEVS based formal framework for environmental modelling. Environmental Modelling & Software. 19, 3 (2004), 261–274.

[33] OGC. 2011. "KML." Accessed June 2011. http://www.opengeospatial.org/standards/kml