Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds

DISCRETE EVENT FORMALISM TO CALCULATE ACCEPTABLE SAFETY DISTANCE

Paul-Antoine Bisgambiglia
Romain Franceschini
François-Joseph Chatelon
Jean-Louis Rossi
Paul Antoine Bisgambiglia

University of Corsica UCPP, CNRS UMR SPE 6134
Campus Grimaldi, bat. D. Alfonsi
Haute Corse, 20250 CORTE, FRANCE

ABSTRACT

The aim of this paper is to present a dimensioning tool for fuelbreaks. It focus on the overall approach and specifically mapping a physical model to a DEVS model, mapping a DEVS model to a DEVS service, and the client that communicates with the server. In order to assist the firefighters, we focus on a Web Service based on different software tools that can be used by firefighters to forecast fuelbreak safety zone sizes. This Web Service uses a simulation framework based on DEVS formalism, a theoretical fire spreading model developed at the University of Corsica and to display the results on a Google Map SDK. The SDK is embedded in a mobile application for touchscreen tablet. The application sends a request to our DEVS Web Service, with its geolocation, and in response receives data sets that allow to draw the safety distance.

1 INTRODUCTION

Mediterranean territories are at high risk of forest fire. During the 2003 and 2004 summers, very large fires have developed in five countries of the Mediterranean coast of Europe (France, Italy, Spain, Portugal, Greece). They caused major damage and many human victims. The management of forests through fuelbreaks (firewall) is a preventive means used to limit the development of large fires.

Implanted in a strategic zone, the fuelbreak ensures the compartmentalization of forests, to limit the spread of fire and decrease the intensity. The purpose of these fuelbreaks is to reduce the risk of fire outbreak, provide a bearing zone to the fight (to secure the interventions), and reduce the power of the fire front. Born of collaboration between physicists, chemists and computer scientists, the tool that we present aims to help firefighters for the fuelbreaks dimensioning. This tool is based on several physical models, and an online computational framework based on the DEVS formalism.

DEVS for Discrete EVent system Specification (Zeigler, Praehofer, and Kim 2000) is a formalism based on the development of time according to events. It allows the composition of models from components stored in libraries, thus avoiding the redevelopment of existing models. It is an open, flexible formalism with a great capacity for extension. Recent works (Zeigler 2003; Vangheluwe 2001) have shown that DEVS formalism may be called multi-formalism because, due to its open nature, DEVS is particularly suitable to be extended towards other formalisms. This capacity is very interesting, as the representation of the various entities which constitute a complex system can be accomplished by using the most appropriate formalism.

A Heat transfer by radiation is the main thermal impact on the people who fight against fires, or on the structures, such as fire truck. The estimated radiation is therefore of paramount importance. To calculate the radiation, we use several physical models. These models will allow us to estimate a safety distance or ASD for *Acceptable Safety Distance* (Rossi et al. 2011; Morandini, Santoni, and Balbi 2001).

The proposed tool must be used in the field. To calculate the ASD, we propose to host the application on a web server accessible from the thin clients as a touchscreen tablet (tablet). A Web Service is a computer program accepting the communication and data exchange between heterogeneous applications and systems in distributed environments. To achieve this goal we use a forest fire spread model fast enough to meet to client requests. The model must then be described using a modeling approach and simulated in an open environment to allow the data exchange on the network.

The paper is organized as follows. The next section provides information about the related work. In the first part, we present the physical model the basis of our work. This model allows to obtain a safety distance depends on many parameters. Then, we detail the DEVS formalism and some works aimed at transforming a DEVS model in Web Services. This works are based on architectures like Web Services. Section 3 provides basic information about the Web Service architecture with both server and client designs. We present our architecture and our DEVS models. We begin by describing the transformation of the physical model into DEVS model. We also present results, results that are identical to those obtained from a scientific computing platform. These results allow us to draw some conclusions: as to identify the most influential parameters on the ASD calculation, and also noted that the DEVS formalism can be used as a very efficient scientific computing tool. Finally, we describe the technological choices that we have made to transform our DEVS models into Web Service. Before concluding, in Section 4, we present the GUI and results displaying in our mobile application.

2 RELATED WORKS

Fire evolution is a complex phenomenon that requires a mathematical and physical analysis. Simulation software has highly enhanced the understanding of the phenomenon, as facing simulation results to experimental data improves the model definition and helps understanding the fire behavior (Harzallah et al. 2008). Simulation software are often used to study complex systems, such as to model fire evolution. For example, the DEVS formalism has been used to model physical equations describing fire evolution; we can quote (Harzallah et al. 2008; Muzy et al. 2002; Bisgambiglia, Filippi, and Gentili 2006; Nader, Filippi, and Bisgambiglia 2011). This work focuses on the spreading aspect; our application aims to provide a safety distance.

2.1 Physical models

Fire model of interest in this paper is described in (Rossi et al. 2011). It is an analytical model based on University of Corsica's forest-fire propagation rules (Balbi et al. 2010).

In (Rossi et al. 2011) a model to calculate a safety distance is presented, it is called ASD for *Acceptable Safety Distance*. This system has been validated as an operational model. It is used to place the firefighters on the ground and assess the radiation rate to which they are exposed. The role of this system is twofold: (1) it calculates a safe distance for the prevention of forest fires. This distance is used to realize a fuelbreaks by vegetation clearing; (2) it can also calculate a safety distance during the struggle. This distance informs the firefighters on the degree of heat in the vicinity of the fire front. This is an analytical model based on radiative heating, and a whole set of parameters, such as vegetation, meteorology, and a combustion model (Balbi et al. 2010). The flame model adopted is based on the radiant surface approach (Zárate, Arnaldos, and Casal 2008), it is generalized to take into account the effect of the fire front width (Figure 1). So far this model prediction has been compared against measured flame length of several experimental fires conduced at the field scale through a variety of natural vegetation in Corsican mountain region. We used simplified surface fire spread model developed until 2007 at the University of Corsica (Rossi et al. 2011). This model solves nonlinear differential equations using the fixed-point method. This

method requires setting up an iterative scheme that induces the creation of the loop and causes a lot of data exchange.

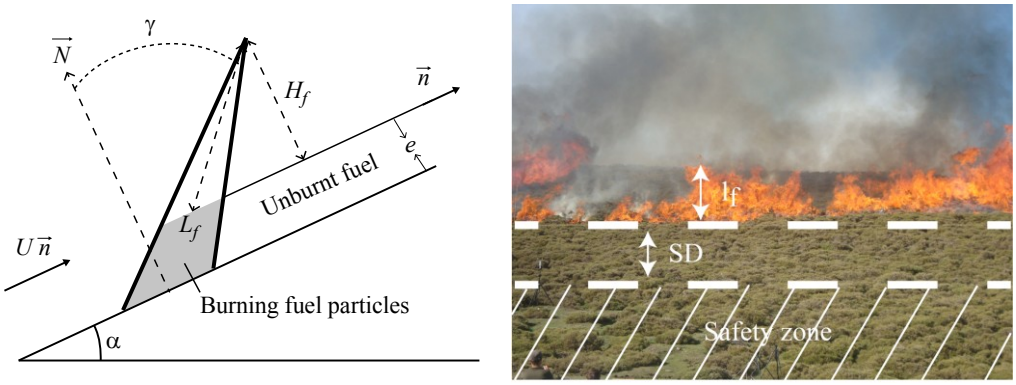


Figure 1: Flame front

In addition to this two models eighteen sub-models complement the global physical model and used to calculate internal variables as the temperature of ignition, the ratio energy radiated, the rate of climb of gas, the flame height H_f, and several other parameters. Details of these equations, as well as a comprehensive look at the models can be found in (Balbi et al. 2010).

2.2 DEVS formalism

Since the 1970s, formal tasks have been performed to develop the theoretical foundations of modeling and simulating of discrete event systems. Our interest focuses on the DEVS formalism. It may be defined as a universal, general methodology which provides tools to model and simulate complex systems. Major efforts have been made to adapt this formalism to various domains and situations as to study forest fire spreading. DEVS permits the modeling of causal and deterministic systems with two types of components. A DEVS atomic model is based on continuous time, inputs, outputs, states and functions (output, transition and lifetime of states). More complex models are constructed by connecting several atomic models in a hierarchical way. The interactions are created via the models' input and output ports, which favors modularity.

The atomic model provides an independent description of the system's behavior, defined by the states and functions of the inputs/outputs and by the model's internal transitions. An atomic model is described by the following formula: AM: < X; Y; S; t_a; δ_{ext} ; δ_{int} ; λ>. Where, X = {(p_{in}, v) | p_{in} ∈ Input ports, v ∈ Xp_{in}}: is the list of the model inputs; Y = {(p_{out}, v) | p_{out} ∈ Output ports, v ∈ Yp_{out}}: is the list of the model outputs; S: is the list of the states or state variables; t_a: S⟶R^+: is the time advancement function or the S state's lifetime; δ_{ext}: QxX⟶S : is the external transition function; δ_{int}:S⟶S: is the internal transition function; and, λ:S⟶Y: is the output function.

The DEVS formalism uses the notion of a description hierarchy, which permits the construction of coupled models, based on a collection of atomic models and/or coupled, and on three coupling relations.

Why DEVS? We mainly use this formalism to its openness and extensibility. For example, the physical model presented has already been used with conventional software tools, they give the same results! Our advantage is to open the application to other fields such as web applications or Web Services.

2.3 DEVS services

This section lists some of the work combining DEVS formalism and Web Services:

SOADEVS (Mittal, Risco, and Zeigler 2007) focuses on the interoperability at the application level, particularly, at the client level and hides the whole simulation engines. DEVS Modeling Language

(DEVSML) aims on standardizing DEVS models among different implementation, which an XML notation for representing models that can be shared between the different participant labs around the world. These models are subsequently realized on different implementations (based on Java, C++ or other languages), using a net-centric infrastructure.

DEVSML (Janoušek, Polášek, and Slavíček 2006; Mittal, Risco-Martín, and Zeigler 2007) proposed a standard representation of DEVS model based on XML. It provides model interoperability among DEVS models located at remote locations. The DEVSML environment is built on client-server paradigm and the simulation is executed at the server's end. The proposed DEVS atomic and coupled DTDs are open to standardization from the community for successful model sharing and collaboration. The DEVSML framework provides the needed feature of run-time composability of coupled systems using the SOA framework. DEVSML also provides the capability to translate model to and from XML and programming language leading to model composability and validation.

DEVS Namespace (Seo and Zeigler 2009) provides an interoperability between DEVS models designed as different programming languages. The difference with other works is that interoperability does not pass through models but by the messages. The proposed DEVS environment is based on SOA and DEVS simulation standards, and extended with the DEVS namespace. In this environment, SOA provides network interoperability, DEVS protocol implementation provides simulator interoperability, and DEVS namespace provides a step toward semantic level interoperability.

CD++ (G. Wainer 2002) is a modeling and simulation toolkit capable of executing DEVS and Cell-DEVS models that has proven to be useful for executing complex models.

D-CD++ (G. A. Wainer, Madhoun, and Al-Zoubi 2008) is an implementation of a CD++ with a distributed simulation engine. In addition to standard versions, there is an implementation of the Web-Service components which enable D-CD++ to expose the simulation functionalities to remote users. Enabling CD++ with Web-Services technology provides a solid framework for interoperating different DEVS implementations in order to achieve a standard DEVS Modeling Language and simulation protocols.

Many works have been proposed with the aim of mapping DEVS models in Web Services. Generally the aim of this works is twofold: (1) provide a service based on a DEVS model, and (2) extend the interoperability of DEVS formalism. We do not propose evolution at these levels, but simply the use of these concepts applied to another application field: the ASD calculation. Our approach is general because we want to use DEVS as a calculation tool. We also propose a comparison with a calculation program scientist and to equivalent results our environment is much more efficient. In addition, we can use DEVS to transform our calculation tool in online tool, and propose complete software architecture; efficient and adapted to the problem we were asked. That is to say: developing a field tool, portable, for fuelbreak dimensioning.

3 OVERALL APPROACH

Our approach is fairly standard, it resembles in this work (Harzallah et al. 2008). We constructed a computational service as Web Service. Our Web Service is based on the DEVS formalism. It allows sending a safety distance, used to planning aid (fuelbreak) and prevention against forest fires. For our calculations, we need a certain number of data, and these data are either acquired locally, such as geolocation, or retrieved through other Web Services, such as ground slope. The Figure 2 describes this architecture. A main Web Service hosting a DEVS simulator to calculate the ASD. It can connect to another Web Services. The client sends data and displays the results.

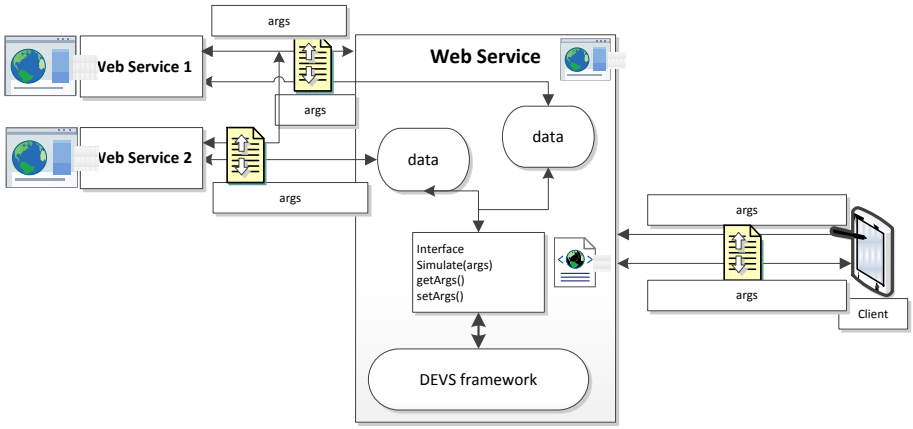


Figure 2: An overview of the overall approach

3.1 Mapping a physical model to a DEVS model

We have seen the complexity of the physical model, and the large number of parameters to be taken into account. Figure 3 shows a simplified diagram of the interconnectedness of all these parameters. The ASD calculation was divided into five blocks, to solve some equations. Each of these blocks has been mapped into DEVS model.

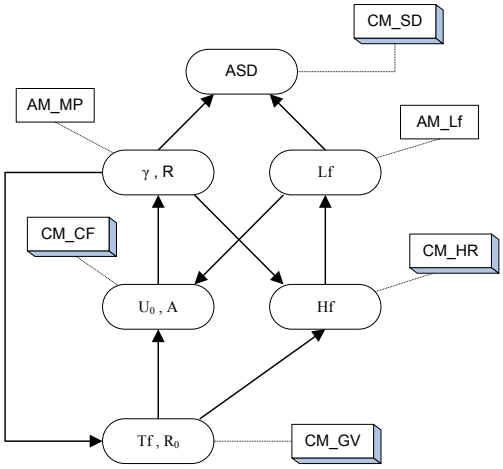


Figure 3: link between the ASD model parameters

Modeling the ASD model using the DEVS formalism gives a coupled model composed of several sub-models. The graphical representation of model is given in Figure 4; we can see the coupling relations: CM_SD gives the ASD; CM_GV gives the rate of climb of gas, and the energy ratio; CM_HR gives the flame height; CM_CF gives the flame temperature and the flat velocity; generator initializes all components by reading the input data from a file or from a Web Service; parameter r0 calculates a ratio surface / volume (depending on ground); AM_LF gives the length of the flame; and, AM_S calculates the flame inclination and the flame front velocity. Once the dependency relationships were obtained, we built the ASD-DEVS models.

These models have been implemented on our framework. A DEVS framework implement in C# language and, using simulation architecture called flat or direct coupling. Flat algorithms are presented in (Zacharewicz et al. 2010). The hierarchy of the simulation objects is flattened to reduce the communication overheads, using a flat simulation approach that eliminated the intermediate coordinator, in order to

reduce the time of simulation and to speed up the production of results. A flattened representation of the model is show in figure 5.

The models were executed with multiple test cases, these same examples were solved in (Rossi et al. 2011) using the software *Mathematica* and gave exactly the same values for the ASD. To validate our model, we made comparisons with the original physical model ASD. The physical model under *Mathematica* was validated by comparison with experimental results. The results obtained with our ASD-DEVS models are identical, therefore we can consider as valid.

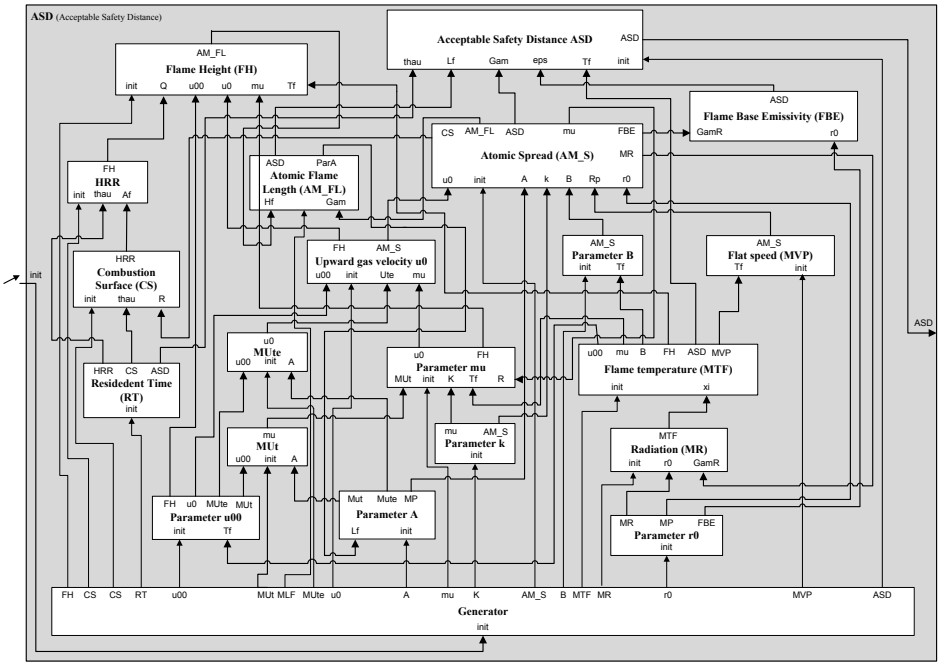


Figure 4: ASD-DEVS flattened representation

The ASD model is composed of twenty-one different atomic models (Figure 4). Except the Generator each atomic model AM_i (Figure 5) compute a specific equation for estimating the value x_i associated with the model. The models are initially loaded by assigning arbitrary values to all variables except the constants and the fixed values like the Stefan-Boltzmann constant (B), the threshold heat flux level: (Φ_{th}), the local terrain angle (α) or the ambient temperature (T_a). A new value is computed for each x_i variable by appropriated atomic model. During each iteration k, variables values from previous iteration are used to calculate the new value x_i. The variables are immediately updated with their new values. The process stops when the Atomic Speed Model (AM_S) reaches the fixed-point $\left| R^{(k)} - R^{(k-1)} \right| < 10^{-3}$. In almost ever case, the solution converges to the correct answer after 10 steps.

For each equation we have split the parameters into two groups: the interim and the fixed parameters. For each calculation of the ASD, fixed parameters are initialized through the "init" port by the "Generator" model. Interim parameters are cleared during the initialization phase. All models operate in the same basic way show in the Figure 5. Where p[i] is the model parameter.

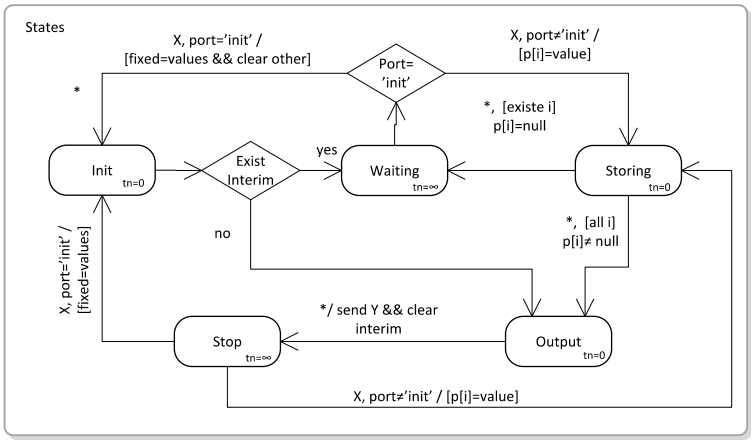
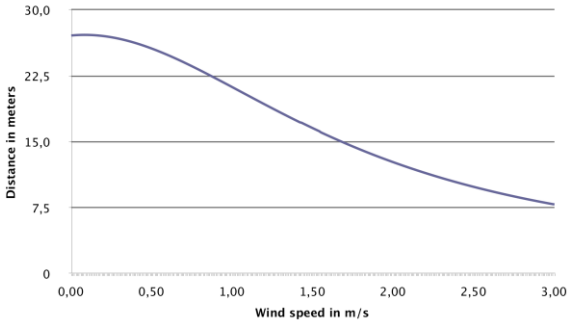


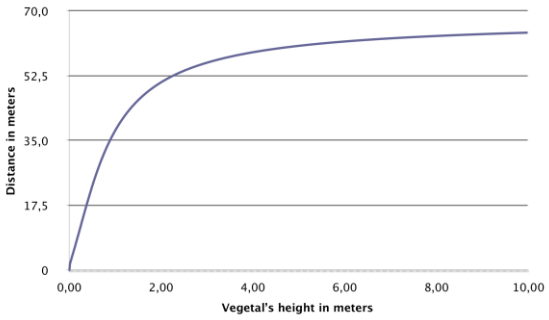
Figure 5: evolution of model state

3.2 Identification of the parameters influence

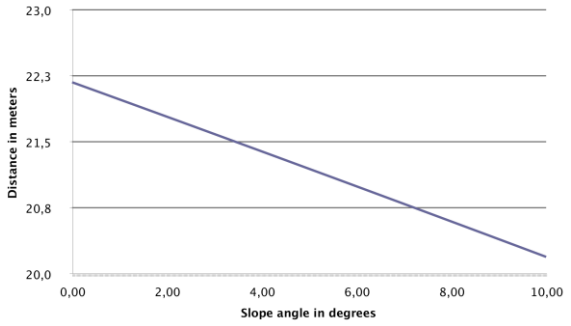
A sensitivity analysis is conducted to identify the model parameters that must be chosen with care because of their large impact on model predictions, and the other parameters that may have only a small impact on the model. Three kinds of environmental parameters are particularly sensitive to the model; exactly twelve parameters are used in the formulas. We can point to the physical parameters: the local terrain angle (α) and the fuel depth (e); to the chemical parameters: the heat yield of the fuel (ΔH_c), the fuel absorption coefficient (C_p) and the moisture content of the dead fuel ($\dot{m}$); and to the meteorological data like the normal wind speed or ambient temperature (T_a).



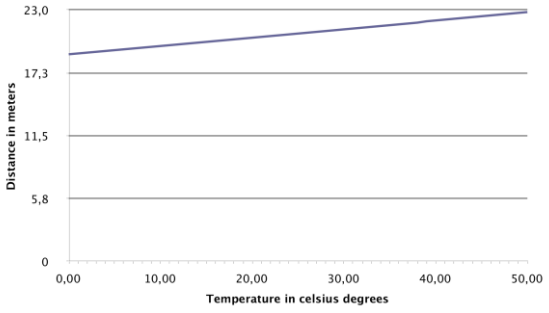
(a) Normal wind speed (m/s)



(b) Fuel depth (m)



(c) Local terrain slope (°)



(d) Ambient temperature (K)

Figure 6: The influence of wind speed, temperature, vegetal height and slope parameters on ASD results

Currently, there is no chemical data available on the net, and all other parameters are not sufficient in temporal and spatial resolution to be uses directly in a surface fire spread model. In this study, a simple univariate sensitivity analysis was performed to assess how the ASD values are affected by a parameter change. The chosen parameters correspond to the local terrain angle, the fuel depth, the normal wind speed and the ambient temperature. All other values are estimated depending on experimental measures.

This analysis indicates that parameters with significant effects on ASD evaluation are the normal wind speed and the fuel depth. However, local terrain slope but above ambient temperature have less influence on ASD and do not produce significant changes on the final results. The results in Figure 6.a and 6.c may be surprising, but it is easily explained, since the radiation rate directly depends on the fire front length. Wind and slope incline the fire front, thus decreasing the radiation rate perceived by firefighters. So, it is not necessary, but interesting, to have a highly accurate estimate on these parameters.

Web Services for the local ground slope and meteorological service are more than enough to calculated safety distance. There is no service which provides access to fuel depth, the value will be given with great precision between zero and two meters, by the user.

3.3 Mapping a DEVS model to a DEVS service

To transform our ASD-DEVS models into Web Service, we relied on existing work. Our application is based on the following technologies: a simulation framework in C# and based on the flat DEVS algorithms. Our framework is hosted on a server; an interface in C# makes a link between the simulator and the data retrieved by the client, and/or other Web Services. This interface allows remote start the simulation, and sends the results. The underlying technologies are Internet standards: XML, KML and JSON for data transfer; REST architecture to drive communications; to display on the client, we use the Goolge Map SDK. The Client, tablet or browser, sends data to the server interface. The interface collects local data, that is to say the client data, and other external data by querying other Web Services, and then it runs the simulation and finally returns the results to the client.

3.3.1 Underlying Technologies

Since REST (REpresentational State Transfer) architecture has emerged as a predominant Web Service design model for its simplicity and clarity, we chose to expose our Web Service with this architecture, where we only rely on URIs and HTTP verbs to expose our methods.

Client / server communications are made through URIs to pass on parameters, and results from server are sent as KML. KML or Keyhole Markup Language is an XML notation for expressing geographic annotation and became an international standard of the Open Geospatial Consortium in 2008. The main interest of sending back data as KML format is that we can lighten up the clients since they doesn't have to interpret raw results before drawing them on a map. Also, we can use directly this format to store the results and visualize them later.

3.3.2 ASD-DEVS service

The Web Service relies on a custom DEVS framework in C#. Originally used as a desktop application for modeling purposes, we added web components on top of it. Since the original application were developed in a typical SOA fashion, no extraction of code was necessary because the core simulators were already isolated. Software Oriented Architectures aims to organize several software components as services providing properties including loose coupling, abstraction and reusability among others.

The solution is composed of several projects, each representing a service:

- DataContracts: provides APIs to access data
- Data: provides an implementation of DataContracts APIs, providing elevation data, temperature and wind speed through external Web Services and providing ASD from local simulation results

- DEVSModeling: the modeling abstraction of our original DEVS framework. It contains ASD models implementations among others
- DEVSSimulation: like above, this layer was part of the original DEVS framework and provides the DEVS simulators
- ServicesContracts: represent the API of our service
- Services: implements the Web Service with the API described in ServicesContracts

The API of the ASD Web Service is quite simple. It's composed of one single method available through an HTTP GET request with the following pattern:

```
/asd/{encoded_polyline}.{format}
```

The encoded_polyline variable is a string representing a set of latitudes and longitudes encoded following Google's encoded polyline format (« Google Encoded Polyline »), which allow to represent several locations in a concise manner. For each of those points, we launch a simulation and build by default a KML string that will contain the polyline passed in arguments and a polygon representing the ASD area. The format parameter is optional and can take either json, xml, or kml value. The first two values returns a list of distances in meters for each given location (the raw results). The last value is the default one, which sends back KML as described above. We chose to minimize the amount of parameters passed to our Web Service, but some missing parameters are essential to calculate results. That's why we have to retrieve them from external Web Services.

3.3.3 Web Services connection and data

Some parameters needed to compute the ASD depends on environment related data around a given location. Since not all of parameters can be easily retrieved, we selected those having a significant influence on the results. For each of them, we either propose to retrieve the information (if available) or to launch the simulation, graduating the value of the parameter from the simple case to the worst case. Then, the user will have to evaluate these different results. Since the goal is to evaluate the extent of a fuelbreak, we're interested in data representing a critical case.

We managed to get some parameters by using external Web Services. The local slope of the terrain is calculated by sampling elevation data around the given location. Services such as (« Google Elevation API ») or (« MapQuest Open Elevation ») can give such data. Once we sampled data around the location, we use a gradient descent to find the local slope of the terrain.


```
{   "results" : [
      {
         "elevation" : 428.9814758300781,
         "location" : {"lat" : 42.30640,"lng" : 9.151400000000001},
         "resolution" : 152.7032318115234
      }
   ],   "status" : "OK"
}
```


Figure 7: Result of a Google Map Elevation API query

For wind speed and temperature data, we used the (« Wunderground ») Weather API, which is one of the Web Services offering access to worldwide weather data, not only for live or forecast but also for historic data, which is particularly what we're looking for. Their RESTful API exposes the history feature, which accept a date along with latitude and longitude coordinates and which returns a summary of the observed weather at the specified location and date in JSON format.

```
{"history":{
      "dailysummary":[
         {
            "date":{"pretty":"12:00 AM CET on January 01, 2011",},
            "meantempm":"11",     // Mean temp in C
            "meanwindspdm":"10",  // Mean wind speed in kph
            "meanwdire":"NE",     // Mean wind direction description
            "meanwdird":"46",     // Mean wind direction in degrees
            "maxtempm":"15",      // Max temp in C
            "mintempm":"7",       // Min temp in C
            "maxwspdm":"15",      // Max wind speed in kph
            "minwspdm":"2",       // Min wind speed in kph
         }
       ]
}}
```

Figure 8: Result extract of a weather history query

As shown in Figure 8, the query returns a JSON string that contains weather information for a given location including: maximum, average and minimal values for wind speed, direction and temperature along with various parameters that we omitted. With this data, we can establish mean values for a given period, e.g the mean of max temperatures and max wind speed for the last summer.

Once all parameters cited above are in our hands, we're able to launch the simulation and send back the results to the client which will display them.

4 DISPLAYING

ASD results provided by our Web Service are useful for firefighters on field when evaluating distances for fuelbreak, the client side must then provide a clear GUI to visualize the terrain and our results, an unobtrusive way to feed input data and a way to retry computations in case of a network failure. In order to fit such requirements, the client has been implemented as a mobile application on both iOS and Android platforms and designed to run primarily on tablets. The user interface will be essentially composed of:

- a sliding panel on the left side providing a section containing a list of previous cached computations and the ability to create a new one. Another section is there for configuration purposes.
- a map fitting the whole screen and representing the current selected computation. The map contains the user location if available; the path for which we want the results and when these are available, the ASD zone is drawn as a polygon. The user can touch the zone to display detailed information about results.
- The map has three distinct modes: the normal mode, the user-tracking mode and the drawing mode. Each of these are detailed respectively below :
- The normal mode displays all available data for the current selected computation and allows user interaction to show detailed information.
- Since our Web Service input is essentially a set of coordinates, these GPS enabled devices allow us to provide a handy user-tracking mode. Thereby, as the user location is updated we can spawn requests to our Web Service and draw a polygon that will represent the ASD.
- Finally, the drawing mode allows the user to place a set of locations directly on the terrain and adjust each of them.

The final result is shown in Figure 9. We can see the contours of the safety zone.

Figure 9: Display example

5 CONCLUSIONS

We have introduced a Web Service combining three tools: a physical model describing the forest fires mechanisms and to estimate the acceptable safety distances (ASD); a DEVS simulator as a calculation tool, to compute safety distances and, a set of Web Services to get or send data. Our service is based on web standards technologies: XML, REST, KML, etc. We used these standards to enhance performance, interoperability and usability. Our Web Service was designed to transform information (position and elevation) and to bring a visualization support to a DEVS based ASD model. A visualization support is a mobile application for tablet. It allows, from a geographical positions, draw a shape describing the safety distance to be respected (ASD).

In terms of DEVS formalism, we offer nothing really new. There have been a lot of works to transform a DEVS model in Web Service. We based our architecture on this works. By cons, we propose a new application of the formalism to model physical systems. This new application allows us to demonstrate the possibility to consider DEVS formalism as a simulation tool and also a calculation tool. Based on these initial results, we can draw a first conclusion. For nearly forty years, the DEVS formalism has evolved. The community extended to the new application domain, it has been associated with many other formalisms: cellular automata (Muzy et al. 2002), fuzzy logic (Bisgambiglia, Filippi, and Gentili 2006), multi-agent system (Quesnel, Duboz, and Ramat 2009), etc. It is now regularly described as multi-formalism. Beyond these advances and, from our results, we think that DEVS can be used as scientific computing tool, just like proprietary solutions. Our results are identical, obtained in equivalent time, if not better. It remains to improve the intuitive software interface, this ease of use, but much work is already well advanced in the field. Although are still gaps compared to proprietary tools, expansion capacity is definitively an advantage. In our case, it will allow us to go further, by deploying our application as a Web Service.

This preliminary work is to be included in a more general approach to provide a set of tools for decision support for the fight against forest fires. Each of these tools meets a specific problem and is based on a Web Service architecture or web application, like *Fore Fire* for the simulation of forest fire spread (Nader, Filippi, and Bisgambiglia 2011). For example, we believe precompute the ASD to provide faster service. Modify the model, taking into account the type of material and its resistance to radiation, to re-

turn the ASD for a structure such as a building, a fire truck, etc. We also can improve the model results by refining the input data.

ACKNOWLEDGMENTS

The present work was supported in part by the French Ministry of Research, the Corsican Region and the CNRS.

A APPENDICES

List of symbols used in equations, text and figures:

- ASD is the Acceptable Safety Distance facing the fire (m);
- A is the ratio energy radiated / necessary;
- B Stefan-Boltzmann constant (W/m^2/K^4);
- L_f The length of the flame to the ground (m);
- R The speed of the flame front (m.s^{-1});
- R_0 The flat velocity without wind (m.s^{-1});
- T_f Temperature of ignition (K);
- U_n Normal wind speed (m.s^{-1});
- u_0 The rate of climb of gas (m.s^{-1});
- α Local terrain slope (°);
- ε Dimensionless equivalent flame emissivity;
- γ Flame tilt angle, the inclination angle between the flame and the ground normal (°);
- τ Dimensionless atmospheric transmissivity;
- Φ_{th} The threshold heat flux level (W.m^{-2});

REFERENCES

Balbi, Jacques-Henri, Jean-Louis Rossi, Thierry Marcelli, and François-Joseph Chatelon. 2010. « Physical Modeling of Surface Fire Under Nonparallel Wind and Slope Conditions ». *Combustion Science and Technology* 182 (7): 922‑939. doi:10.1080/00102200903485178.

Bisgambiglia, P.-A., J.B. Filippi, and E. de Gentili. 2006. « A fuzzy approach of modeling evolutionary interfaces systems ». In *Proceedings of the ISEIM 2006, Corte (France)*, édité par IEEE, 98‑103. Corte (France).

« Google Elevation API ». https://developers.google.com/maps/documentation/elevation/.

« Google Encoded Polyline ». https://developers.google.com/maps/documentation/utilities/polylinealgorithm.

Harzallah, Yosri, Vincent Michel, Qi Liu, and Gabriel Wainer. 2008. « Distributed Simulation and Web Map Mash-Up for Forest Fire Spread ». In , 176‑183. IEEE. doi:10.1109/SERVICES-1.2008.74. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4578321.

Janoušek, Vladimír, Petr Polášek, and Pavel Slavíček. 2006. « Towards DEVS Meta Language ». In *ISC 2006 Proceedings*, 69–73. http://www.fit.vutbr.cz/research/view_pub.php?id=8109.

« MapQuest Open Elevation ». http://developer.mapquest.com/web/products/open/elevation-service.

Mittal, Saurabh, José L. Risco, and Bernard P. Zeigler. 2007. « DEVS-based simulation web services for net-centric T&E ». In *Proceedings of the 2007 Summer Computer Simulation Conference*, 357–366. SCSC '07. San Diego, CA, USA: Society for Computer Simulation International. http://dl.acm.org/citation.cfm?id=1357910.1357967.

Mittal, Saurabh, José Luis Risco-Martín, and Bernard P. Zeigler. 2007. « DEVSML: automating DEVS execution over SOA towards transparent simulators ». In *Proceedings of the 2007 spring simula-*

tion multiconference - Volume 2, 287–295. SpringSim '07. San Diego, CA, USA: Society for Computer Simulation International. http://dl.acm.org/citation.cfm?id=1404680.1404725.

Morandini, F., P.A. Santoni, and J.H. Balbi. 2001. « The contribution of radiant heat transfert to laboratoryscale fire spread under the influences of wind and slope ». *Fire Safety Journal* (36): 519‑543.

Muzy, A., E. Innocenti, A. Aiello, J.-F. Santucci, and G. Wainer. 2002. « Cell-DEVS quantization techniques in a fire spreading application ». In *Simulation Conference, 2002. Proceedings of the Winter*, 1:542 ‑ 549 vol.1. doi:10.1109/WSC.2002.1172929.

Nader, B., J.-B. Filippi, and P. -A Bisgambiglia. 2011. « An experimental frame for the simulation of forest fire spread ». In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, 1010‑1022. doi:10.1109/WSC.2011.6147825.

Quesnel, Gauthier, Raphaël Duboz, and Éric Ramat. 2009. « The Virtual Laboratory Environment – An operational framework for multi-modelling, simulation and analysis of complex dynamical systems ». *Simulation Modelling Practice and Theory* 17 (4) (avril): 641 ‑ 653. doi:10.1016/j.simpat.2008.11.003.

Rossi, J.L., A. Simeoni, B. Moretti, and V. Leroy-Cancellieri. 2011. « An analytical model based on radiative heating for the determination of safety distances for wildland fires ». *Fire Safety Journal* 46 (8) (novembre): 520‑527. doi:10.1016/j.firesaf.2011.07.007.

Seo, Chungman, ad B.P. Zeigler. 2009. « DEVS namespace for interoperable DEVS/SOA ». In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, 1311 ‑ 1322. doi:10.1109/WSC.2009.5429701.

Vangheluwe, H. 2001. « Multi-Formalism Modelling and Simulation ».

Wainer, Gabriel. 2002. « CD++: a toolkit to develop DEVS models ». *Software - Practice and Experience* 32: 1261–1306.

Wainer, Gabriel A., Rami Madhoun, and Khaldoon Al-Zoubi. 2008. « Distributed simulation of DEVS and Cell-DEVS models in CD++ using Web-Services ». *Simulation Modelling Practice and Theory* 16 (9) (octobre): 1266‑1292. doi:10.1016/j.simpat.2008.06.012.

William, R.J. 1982. « Urban and wildland fire phenomenology ». *Progress in Energy and Combustion Science*, sect. 8.

« Wunderground ». http://www.wunderground.com/weather/api.

Zacharewicz, Gregory, Maâmar El-Amine Hamri, Claudia S. Frydman, and Norbert Giambiasi. 2010. « A Generalized Discrete Event System (G-DEVS) Flattened Simulation Structure: Application to High-Level Architecture (HLA) Compliant Simulation of Workflow ». *Simulation* 86 (3): 181‑197.

Zárate, Luis, Josep Arnaldos, and Joaquim Casal. 2008. « Establishing safety distances for wildland fires ». *Fire Safety Journal* 43 (8) (novembre): 565‑575. doi:10.1016/j.firesaf.2008.01.001.

Zeigler, Bernard P. 2003. « DEVS Today - Recent Advances in Discrete Event-Based Information Technology ». In *proc. of the 11th IEEE/ACM International Symposium on*.

Zeigler, Bernard P., Herbert Praehofer, and Tag Gon Kim. 2000. *Theory of Modeling and Simulation, Second Edition*. Édité par Academic Press.

AUTHOR BIOGRAPHIES

PAUL-ANTOINE BISGAMBIGLIA is an Associate Professor at the University of Corsica. His research interests include fuzzy systems, multi agent systems, and DEVS. Her e-mail and web addresses are bisgambiglia@univ-corse.fr and http://paul-antoine-bisgambiglia.univ-corse.fr/, respectively