FORMALIZING GEOGRAPHICAL MODELS USING SPECIFICATION AND DESCRIPTION LANGUAGE, THE WILDFIRE EXAMPLE

Pau Fonseca i Casas
Josep Casanovas
Jaume Figueras
Antoni Guasch

Universitat Politècnica de Catalunya
Jordi Girona 31
Campus Nord, B5 building, InLab FIB
08034 Barcelona, Spain

ABSTRACT

In this paper we explore how we can use Specification and Description Language, to represent simulation models that make an intensive use of geographical information, like environmental simulation models. The purpose is to perform a complete unambiguous, graphical and formal representation of a wildfire simulation model. Specification and Description Language is a modern object oriented language that allows the definition of distributed systems. It has focused on the modeling of reactive, state/event driven systems, and has been standardized by the International Telecommunications Union (ITU) in the Z.100. Thanks to the graphical representation of the simulation model, the interaction between the experts that usually come from different areas is simplified. Also, due to the unambiguous and modular nature of the language, all the details of the model can be validated by personnel that do not necessarily are used with programming languages or simulation infrastructures.

1 INTRODUCTION

From the different phases of a simulation model construction, the formalization phase sometimes is missed. The reasons can be diverse but often the time needed to define the simulation model using a formal language is not viewed by the members of the team as a key element that accelerates the process. Several authors that argue the formalization of a simulation model is a key phase, considering that it could be understood as a product by itself (Brade, 2000).

It is clear that the conceptual model helps in the implementation process and in the communication between the different personnel involved in the model construction, and if the models, like those present in this paper, need of the knowledge of personnel with diverse formation, this phase must be considered as a requirement. Regarding simulation models that use geographical information is often needed to use cellular automaton structures in order to represent this data (Benenson, et al., 2004). Since we want to formalize cellular automaton structures we also need to fully define this structure. We first need to define what we understand as a cellular automaton, and then to use a language -and in our case extend the language- to achieve a complete and unambiguous representation of this structure.

Regarding cellular automaton structure, we follow the approach presented on (Fonseca i Casas, et al., 2005), and in order to extend the cellular automaton we are based on the preliminary idea presented on (Fonseca i Casas, et al., 2010) to represent formally a cellular automaton.

Wildfire simulation model requires a behavior definition that clearly describes how the fire evolves over time. To do this we follow the approximation proposed by (Andrews, et al., 1989).

In this paper we explore the use of Specification and Description Language (SDL) to define the wildfire model, see (Doldi, 2003), (ITU-T, 1999). We want to remark that, of course, it is not our intention to state that SDL is the only language that can be used for this purpose: other alternatives exists like (Wainer, et al., 2003). We just want to analyze the feasibility of using SDL in this scope and finally analyze his weakness and benefits.

2 SPECIFICATION AND DESCRIPTION LANGUAGE

Specification and Description Language (SDL) (Doldi, 2003) is an object-oriented formal language defined by the International Telecommunications Union–Telecommunications Standardization Sector (ITU–T) (the Comité Consultatif International Telegraphique et Telephonique [CCITT]) on the Z. 100 recommendation (ITU-T, 1999). The language was designed for the specification of event-oriented, real-time and interactive complex systems. These systems might involve different concurrent activities that use signals to perform communication. In our current scope SDL SIGNALS represents the events of the simulation model, hence in the paper SDL SIGNAL or event can be considered equivalent, since the SIGNAL is the representation of the event in the language. SDL is based on the definition of four levels to describe the structure and the behavior of the models: system, blocks, processes and procedures. In SDL BLOCKS and PROCESSES are named AGENTS. The outermost block, the *system* BLOCK, is an agent itself. Figure 1 shows this hierarchy of levels.

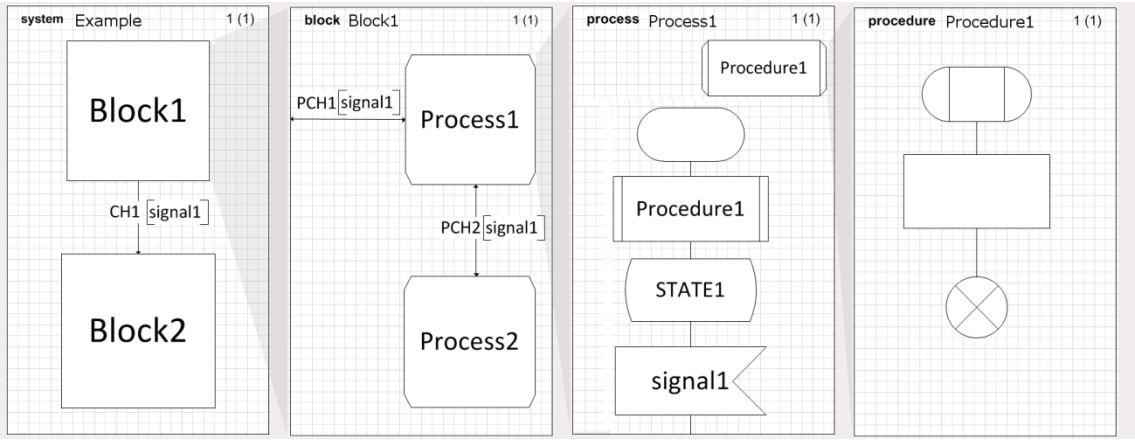

Figure 1. A structural vision of an SDL model. 4 main different levels exist.

The different concepts that the SDL language covers are:

- **System structure**: from the blocks to the processes and their related hierarchy.
- **Communication**: signals, communication paths or channels, parameters that can be carried out by the signals, etc.
- **Behavior**: defined by different processes.
- **Data**: based in Abstract Data Types (ADT).
- **Inheritance**: useful to describe relations between objects and their properties.

Although a textual SDL representation is possible (SDL/PR), this paper uses the graphical representation of the language (named SDL/GR). More details about the Specification and Description Language can be found in the recommendation Z.100 (Telecommunication standardization sector of ITU, 1999) or at the web site (IEC International Enginyeriing Consortium, 2000).

To briefly illustrate SDL and in order to explain some of the key elements of the proposed methodology we show a specification of a G|G|2 model, following the Kendall notation (Kendall, 1953), general distribution for the arrivals and for the services times, and two servers. The first diagram following the SDL language can be represented by a single box in the system diagram that contains the GG2 model. Going inside this BLOCK, Figure 2 shows the inner structure of the model, two servers and a queue.

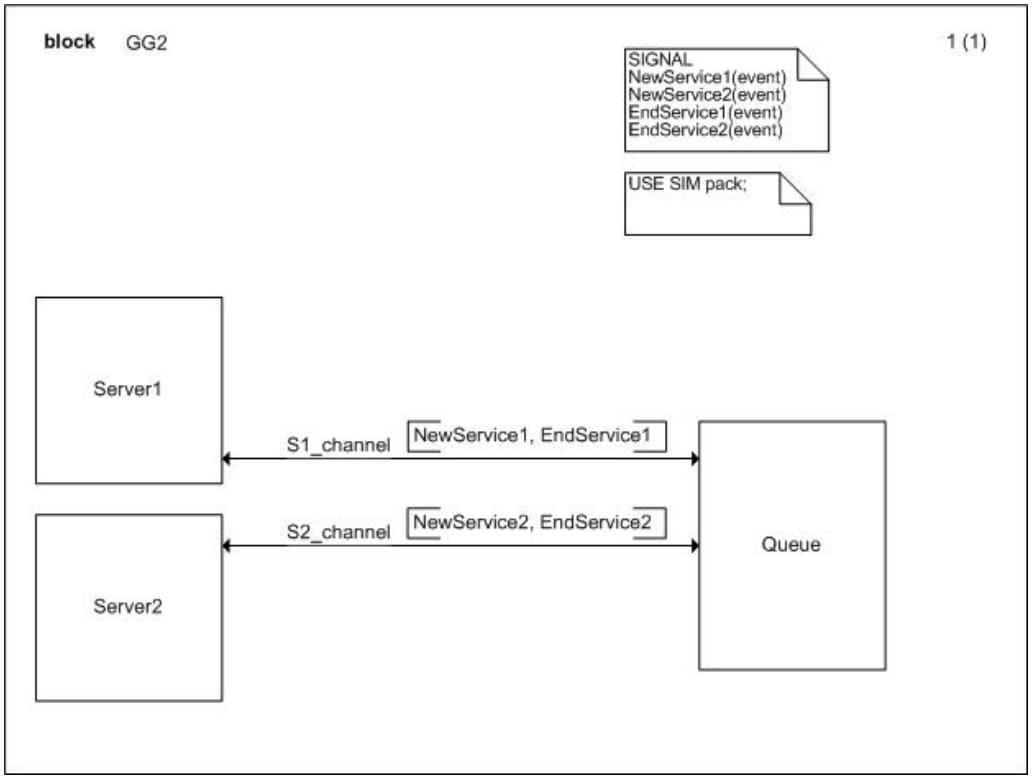

Figure 2: GG2 model blocks diagram.

Figure 2 details the structure of the model. The communication mechanism between the different model elements is asynchronous (we are defining all these elements in a BLOCK agent not inside a PROCESS agent). Also, we are here representing the signals that travel from one agent to others allowing this communication *EndService1, EndService2, NewService1* and *NewService2*. We can go further and analyze what is inside these blocks. We continue the decomposition with other BLOCK's agents. Finally we must define PROCESS that can be decomposed in other set of PROCESS. Finally the PROCESS must be defined with specific behaviors, as the one represented on Figure 4 for the PServer1.

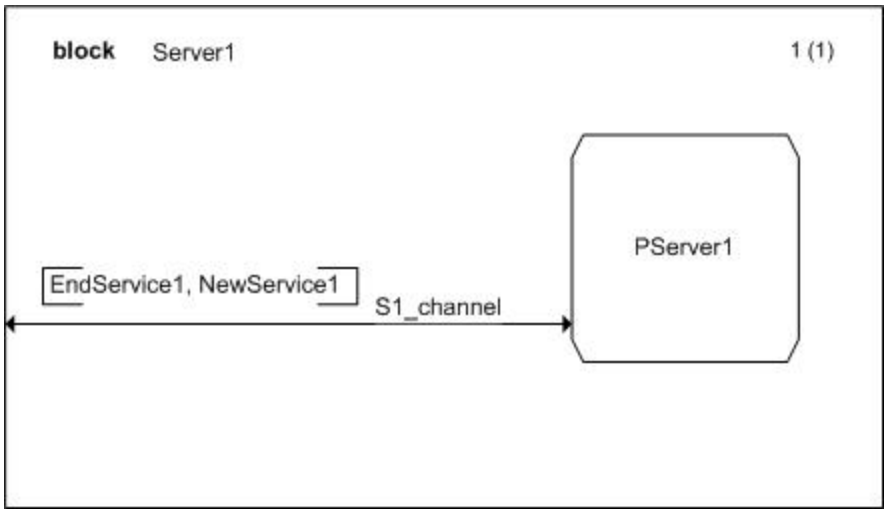

Figure 3:Server1 block processes diagram.

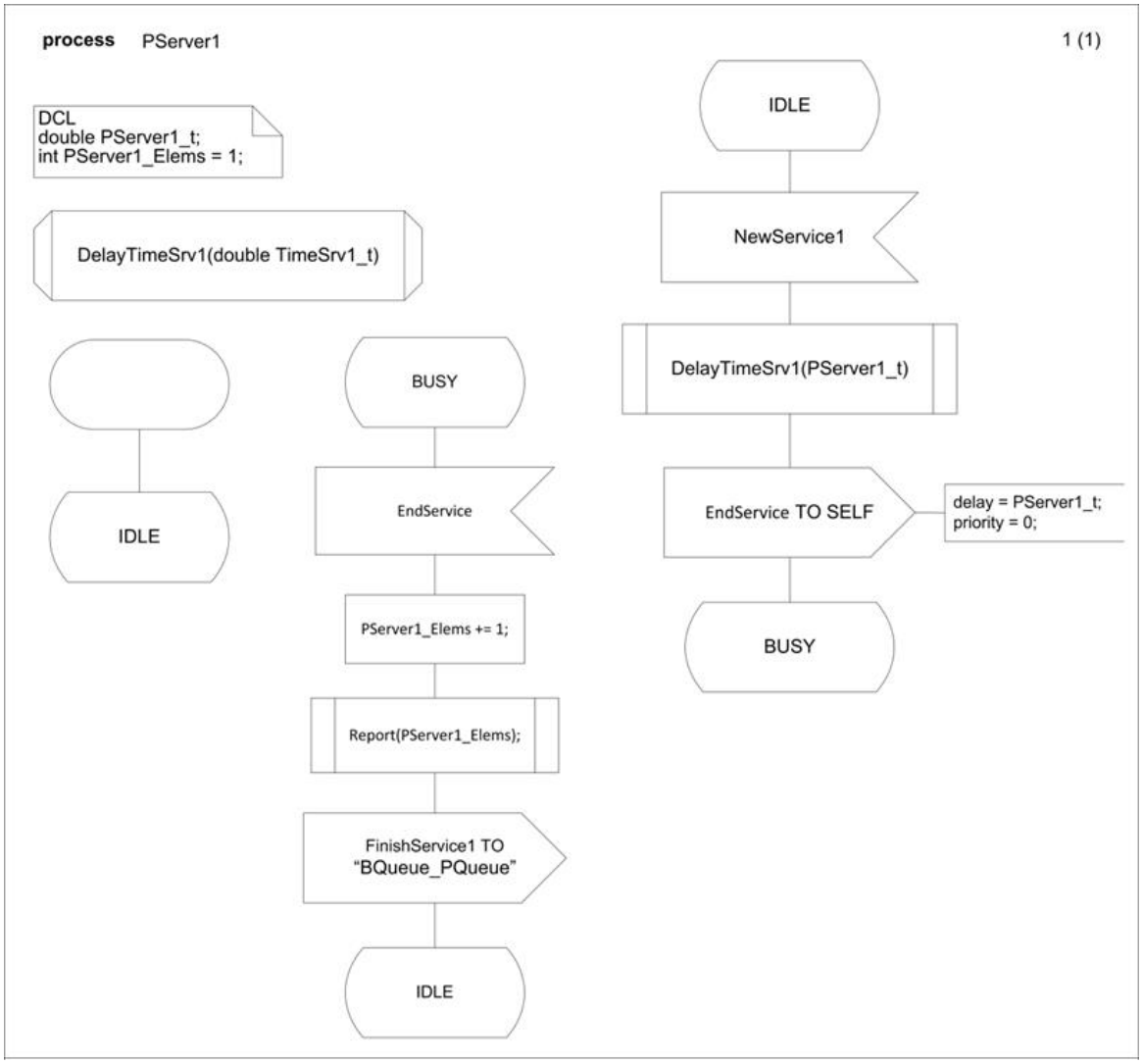

Figure 4: PServer1 process

Note that only one process is represented for each one of the blocks. If two or more processes are defined they are executed sequentially. Since each process defines the states for the object, the definition of two or more blocks implies that the element states definition are the combination of the states of each one of the processes. In the former example this is not needed, and a single process is enough.
Finally is necessary to define the process diagram for each process. In Figure 4 the start operation initializes the clock of the process (to 0) and finish in the state (IDLE). Two states are defined (IDLE and BUSY). The events that modify the state of the server are *NewService1* (from IDLE to BUSY) and *EndService* (from BUSY to IDLE). Note that the events have a parameter defining the time where the event takes place (delay). This time is used to update the clock of the element.

3 CELLULAR AUTOMATA

Cellular automata must be defined in SDL because they simplify the interaction of simulation models and Geographical Information Systems (GIS) data (Benenson, et al., 2004). The cellular automaton we are using is an extension of the common cellular automaton named m:n-CA^k. Its definition can be found on (Fonseca i Casas, et al., 2005). Cellular automata are discrete dynamical systems whose behavior is completely specified in terms of a local relation (Emmeche, 1998), hence is needed to represent all the data needed to perform its evolutions.
One-dimensional cellular automata are based in a row of "cells" and a set of "rules". A two-dimensional cellular automaton uses rectangular grids of cells. Each one of the different cells can be in one of different "states" (the number of possible states depends on the automata). Thinking states as numbers, in a two-state automaton, each cell can be only in 1 or 2 state. Cells represent automata space; time advances in discrete steps following "the rules", the laws of "automata universe", usually expressed in a small look-up table. At each step every cell computes its new state in function of its closer neighbors. Thus, system's laws are local and uniform.
Next figure shows one-dimensional cellular automaton initial state and successive two states after rules application.

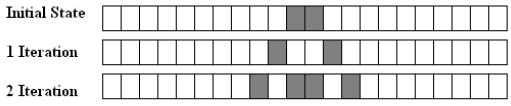

Figure 5:One-dimensional cellular automaton

4 MULTI:N-DIMENSIONAL CELLULAR AUTOMATON (M:N-CA)

A multi:n-dimensional cellular automaton (m:n-CA) is a generalization of a cellular automata defined as follows (Fonseca i Casas, et al., 2005):

Definition 1: m:n-CA^k
A multi n dimensional cellular automaton is a cellular automaton generalization composed by m layers with n dimensions each one.
The representation is:

$$m : n - CA^k \tag{1}$$

Where

- m: is the automaton number of layers.
- n: is the different layers dimension.
- k: is the number of main layers (1 by default). A layer in a m:n-CA^k is a main layer if a transition function Λ is defined in order to modify its state. A m:n-CA automaton only presents one main layer, while m:n-CA^k automaton presents k main layers.

A two dimensional cellular automaton is represented by a 1:2-CA. A transition in a m:2-CA cellular automata is defined as in a 2-dimensional cellular automata, but main layer cell state is a combination of data contained in the m-1 secondary layers at the same position.

All cellular automatons defines two functions: **vicinity** function that allows to represents the cells (or the space in the continuous case) that must be analyzed to perform the propagation, and the **nucleus** functions that defines the cells (or again, the space in the continuous case) that must be modified once the propagation finalizes. Also, the **propagation** function can work with several layers allowing to represent all the necessary model information, and a **combination** function permits to combine the information that comes from all the layers. All layers must be georeferenced. The GIS data classification is shown in the next table based in the table of (Fonseca i Casas, et al., 2004).

Table 1. GIS data classification in a simulation model.

Layer	GPS integration	Description
2DLayers	Geo referenced	Point, polylines, texts or lines.
3DLayers	Geo referenced	Fixed population of elements over a matrix, and DEM.
Routes	Track points	Represent *Objects* movement.
2DObjects	Waypoints	a 2D object in an specific position
3DObjects	Waypoints	a 3D object in an specific position.

Suitable data that can be represented in the m:n-CA^k layers are vectorial data (2DLayers) or raster data (3DLayers). Other elements can be represented using common simulator elements.
Since multiple layers belong to a single automaton its state is defined as follows.

5 IMPLEMENTATION

Usually in a simulation study we begin defining the model and then describing what is the platform selected to implement this model. However in the present case, and since it is needed to extend the language, this section must be previous.
To implement our models we can use different existing tools that understand SDL language, like Cinderella (CINDERELLA SOFTWARE, 2007), Telelogic (IBM, 2009) of IBM or PragmDev (PragmaDev SARL, 2012). However, we are using our tool Specification and Description Parallel Simulator (SDLPS) (Fonseca i Casas, 2008), (Fonseca i Casas, 2011) because it allows to add the needed capabilities to the language without the need to define complex SDL structures.

5.1 Extensions added to the language

The current version of SDL (SDL-2010 that appears at the end of 2012) allows defining delays and priorities in the SIGNALS. This allows defining the time needed to process an specific event (see Figure 6). However the recommendations do not define a structure that directly allows to work with cellular automaton structures.

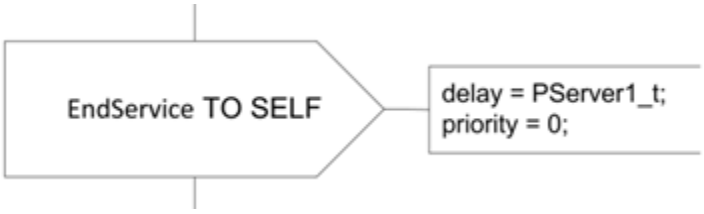

Figure 6: Defining the delay and priority on a SIGNAL on SDL-2010.

To work with cellular automata we will extend the language. To do this, we first need to understand that all the cells have the same behavior. This implies that is not required to represent all the cells, but only one

cell. Also is necessary to represent the relation with the neighborhood (that of course is specific of the cellular automata), and the relation with the other layers. That implies to define the **combination**, **vicinity** and **nucleus** functions, as previously are defined (Fonseca i Casas, et al., 2005), the **propagation** function, that defines the evolution of the cellular automaton representing his behavior can be represented graphically as a SDL PROCESS we can see next.

In SDL language we can use **types** to define blocks that have the same behavior (as is usual in any OO language). This leads to a simplification in the representation of SDL cellular automata models using two types VT_P_mnCACell (to build a PROCESS that represents the propagation function) and VT_P_mnCA (to represent a BLOCK containing the default dimension and main layers that all cellular automata must have). This block also implements some structures that permits to send the SDL SIGNALS to all the cells of the cellular automaton layer (we can use ALL_CELLS, to send the signal to all the cells of the VT_P_mnCA agent) or just to a selection (represented by an array). The definition of these two virtual SDL blocks, due to space reasons are not detailed in this paper.

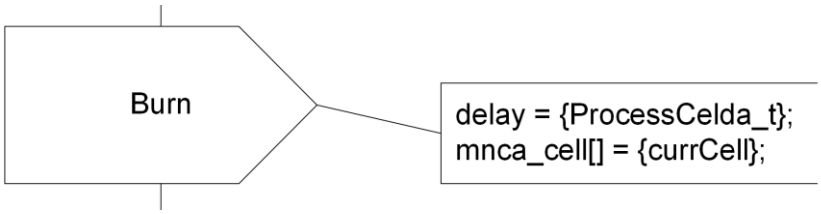

Figure 7: Extension to the language that allows to send a signal to all the cells of the cellular automaton represented on the layer. Here we are sending the SIGNAL to the cell that originates the SIGNAL.

6 WILDFIRE MODEL

As we said previously, we are following the model proposed by (Andrews, et al., 1989) to represent the fire spread. In Figure 8 we are showing the block implementing the cellular automaton that represents the wildfire propagation model. Since here we are only representing a single mode (fire spread) no other PROCESSes are included. If we want to combine the fire spread model with other models (containment models, etc.) can be here specified as new SDL agents.

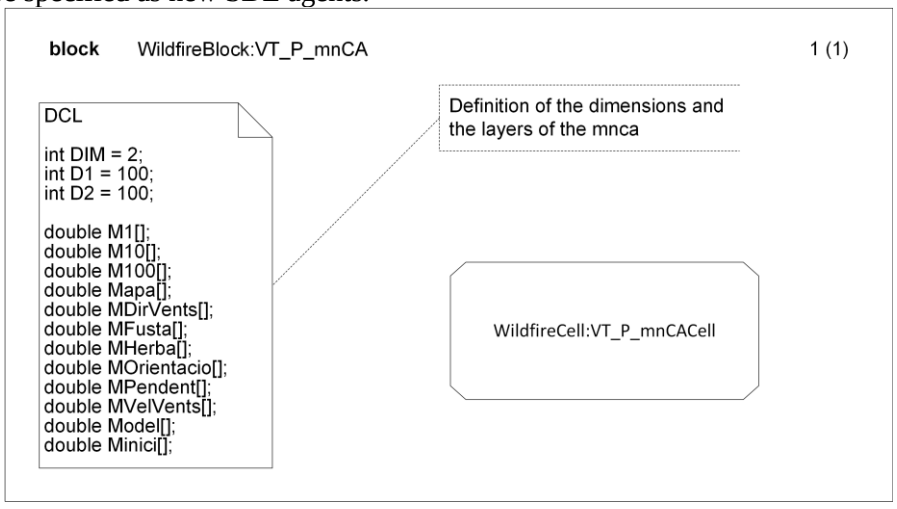

Figure 8: m:n-CAk cell representation for wildfire model.

Inside this we can find the definition of the behavior of the model. In Figure 9 we describe the *Unburn* and *Burned* state for the cellular automaton cells, on Figure 10 we represent the *Burning* state.

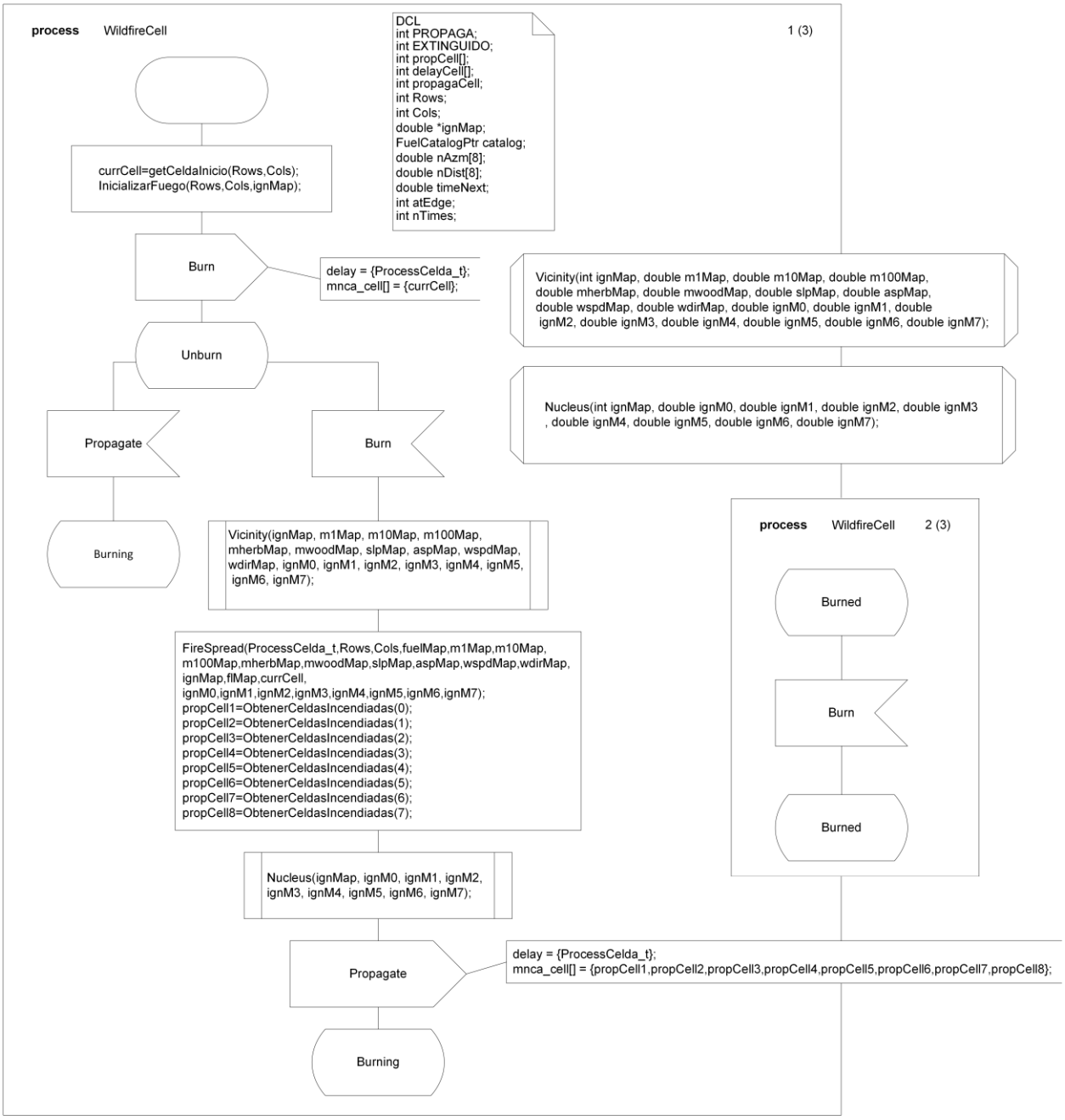

Figure 9: states diagram for a cell.

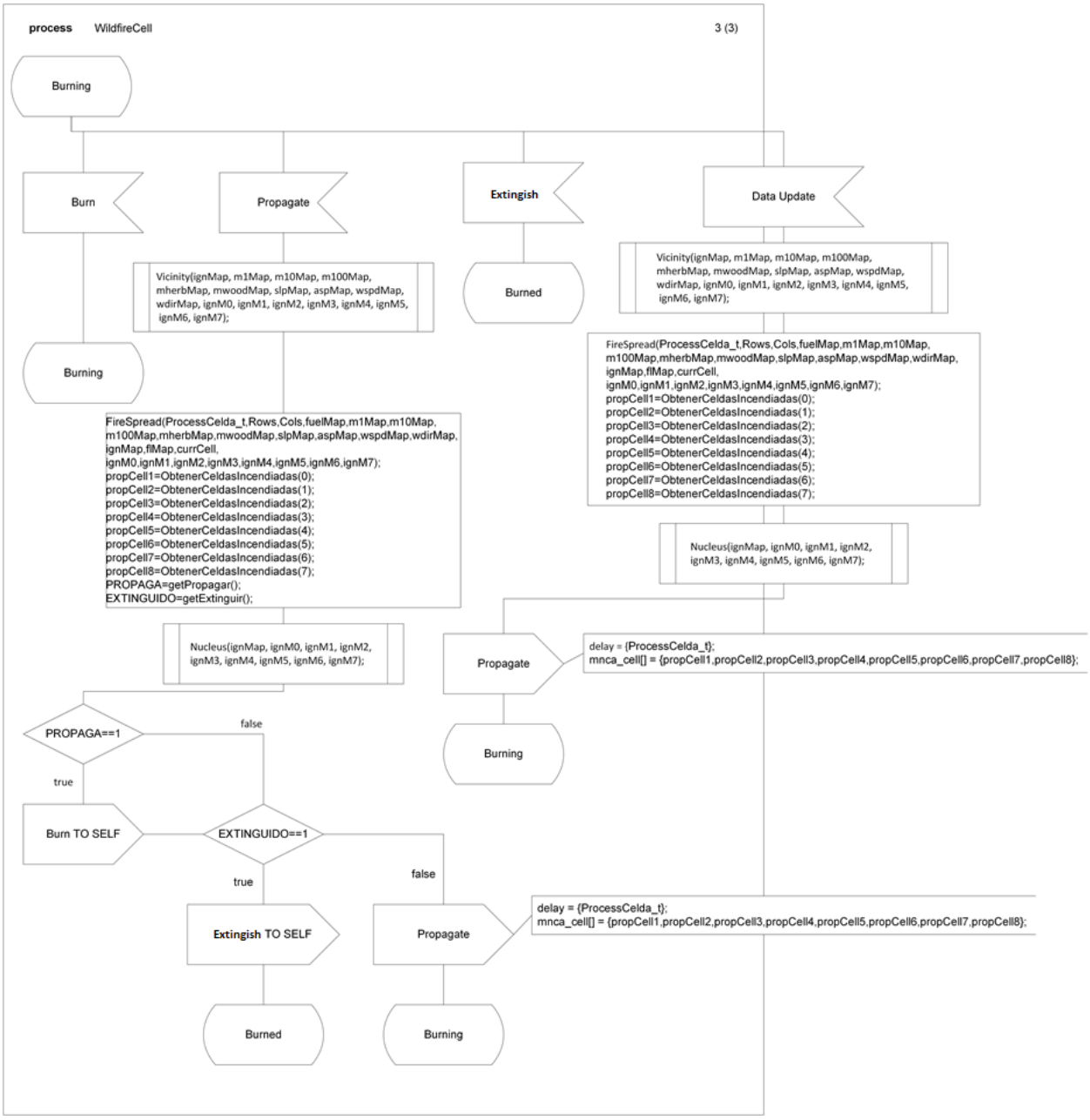

Figure 10: propagation of the wildfire.

Each cell of the model can be in three states, *Unburn* (that means that is no fire in the cell), *Burning* (that means that the cell is on fire) and *Burned* (that means that the cell is completely burned).

The signals that can travel from one state to others are *Burn*, *Propagate*, *Extinguish* and *Data Update*. *Data Update* is needed in order to recalculate the state of the cells in the case that the values of the cells changes due to an external model. Regarding the procedures that are used in the model, just to note that **Vicinity** and **Nucleus** can also be represented graphically in the last level of the SDL diagrams (PROCEDURE diagrams). For the **FireSpread** function, that represents the internal calculus of the BEHAVE (Benenson, et al., 2004) model this is not allowed, since the method is called in a TASK. Conceptually this is done in that way because BEHAVE model in this implementation only is representing the mathematical calculus that defines the modification of the physical state of a cell.

7 CONCLUDING REMARKS

This paper shows how we can model environmental systems using Specification and Description Language. Specifically we present a formalization of a wildfire model based on BEHAVE model (Benenson, et al., 2004).To do this, the main concern is how to model the behavior of cellular automata graphically using SDL, and how to manage time. Time management is well solved in the new recommendation of SDL (SDL-2010) however it does not propose a convenient solution in order to represent cellular automata structures. In order to do this we propose to add to our specification two virtual blocks that allow to define the main structures that all cellular automaton requires. This implies that the proposed extensions to the recommendation can be implemented over the existing language using its current structures. This leads to the definition of a library in the language that simplifies the definition of environmental simulation models.

From the point of view of the model, the unambiguous and graphical representation of the cellular automaton increases the understanding of its behavior by the experts of the system to be modeled, and allows to perform a Conceptual Validation as is defined in (Sargent, 2007). Since all the data that the automaton needs is also represented in the SDL diagrams, it remains clear what the dependencies of the model are in order to be executed. SDL is not a symmetric language, is more complex to write a model than read a model (in less than one hour everyone can be capable to read any SDL diagram). This is a clear advantage, since the modelers must be capable to write the model correctly, at least to beneficiate from the capabilities of the automatic execution, but this is not mandatory for the system experts, facilitating a faster and deeper use of its knowledge in the model. This also avoids a possible refusal to use the language by people not used with this type of specification strategies since the learning curve to understand it is relatively fast.

Other clear advantage is related to the fact that SDL is a standard ISO language and several tools understand our models. This leads to automatic implementations an easier use of different platforms, and the simplification of the verification process.

REFERENCES

Andrews, P.L., and C.H. Chase. 1989. BEHAVE: Fire behavior prediction and fuel modeling system-BURN subsystem, part 2. Gen. Tech. Rep. INT-260., Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Research Station, 93.

Benenson, Itzhak, and Paul M. Torrens. 2004. Geosimulation, Automata-based Modeling of Urban Phenomena. West Susses PO19 8SQ: John Wiley & Sons Ltd.

Brade, D. 2000. "Enhancing modeling and simulation accreditation by structuring verification and validation results." Edited by J. A. Joines, R. R. Barton, K. Kang and P. A. Fishwick. Winter Simulation Conference.

CINDERELLA SOFTWARE. 2007. Cinderella SDL. Accessed 03 31, 2009. http://www.cinderella.dk.

Doldi, Lauren. 2003. Validation of Communications Systems with SDL: The Art of SDL Simulation and Reachability Analysis. John Wiley & Sons, Inc.

Emmeche, Claus. 1998. Vida Simulada en el ordenador. Barcelona, Catalunya: Gedisa.

Fonseca i Casas, Pau. 2008. "SDL distributed simulator." Winter Simulation Conference 2008. Miami: INFORMS.

Fonseca i Casas, Pau, and Josep Casanovas. 2005. "Simplifying GIS data use inside discrete event simulation model through m:n-AC cellular automaton." Proceedings ESS 2005.

Fonseca i Casas, Pau, Josep Casanovas, and Jordi Montero. 2004. "A cellular automata and intelligent agents use to model natural disasters with discrete simulation." Proceedings EMS 2004.

Fonseca i Casas, Pau, Màxim Colls, and Josep Casanovas. 2010. "Representing Fibonacci function through cellular automata using Specification and Description Language." Procediings of the 2010 Summer Simulation Multiconference. Ottawa, ON, Canada.

IBM. 2009. TELELOGIC. Accessed 03 31, 2009. http://www.telelogic.com/.

IEC International Enginyeriing Consortium. 2000. SDL Tutorial. Accessed January 2009. http://www.iec.org/online/tutorials/sdl/.

IEC. n.d. SDL Tutorial. Accessed May 2010. http://www.iec.org/online/tutorials/sdl/topic04.html.

ITU-T. 1999. "Specification and Description Language (SDL)." Series Z: Languages and general software aspects for telecommunication systems. International Telecommunication Union. Accessed April 2008. http://www.itu.int/ITU-T/studygroups/com17/languages/index.html.

Kendall, David G. 1953. "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain." Annals of Mathematical Statistics 24 (3): 338-354. doi:10.1214/aoms/1177728975.

PragmaDev SARL. 2012. http://www.pragmadev.com/product/codeGeneration.html.

Sargent, Robert G. 2007. "Verification and Validation of simulation models." Edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. Proceedings of the 2007 Winter Simulation Conference. IEEE.

Telecommunication standardization sector of ITU. 1999. "Specification and Description Language (SDL)." Series Z: Languages and general software aspects for telecommunication systems. International Telecommunication Union. Accessed April 2008. http://www.itu.int/ITU-T/studygroups/com17/languages/index.html.

Wainer, G., and W. Chen. 2003. "A framework for remote execution and visualization of Cell-DEVS models." Simulation: Transactions of the Society for Modeling and Simulation International 626-647.

AUTHOR BIOGRAPHIES

PAU FONSECA I CASAS is a professor of the department of Statistics and Operational research of the Technical University of Catalonia, teaching in Statistics and Simulation areas. He obtained his master degree in computer engineering on 1999 and his Ph.D. on 2007 from Technical University of Catalonia. He also works in the InLab FIB (http://inlab.fib.upc.edu/) as a head of the Environmental Simulation area, developing Simulation projects since 1998. His website is http://www-eio.upc.es/~pau/. His research interests are discrete simulation applied to industrial, environmental and social models, and the formal representation of such models.

JOSEP CASANOVAS Professor Josep Casanovas (Ph.D. in Computer Science, Industrial Engineer, MSc in Economics) is the head of inLab FIB, formerly called LCFIB, at Barcelona School of Informatics. He is a full professor of the Statistics and Operations Research Department at UPC. His main research ar-eas are Modeling and Simulation, Internet and Information Systems. He is the author of numerous re-search articles and other kinds of publications and has collaborated in the development of many projects for the European Union and other companies and institutions. Between 1998 and 2004 he was dean of the Barcelona School of Informatics. In addition, Prof. Casanovas has been vice-rector of university policies of the Technical University of Catalonia (2006-2011) with responsibilities in strategic projects like the definition of new university governance models, reformulation of university departmental structure or de-sign and promotion of the new Diagonal-Besos Campus in Barcelona. He was also responsible for ICT policies at UPC. Currently, Josep Casanovas is co-director of LogiSim (Centre of Simulation and Optimi-zation of Logistic Systems) and coordinator of the Severo Ochoa Research Excellence Program in the Barcelona Supercomputing Center (BSC-CNS).

JAUME FIGUERAS born in 1974, had his degree in Computer Science in 1998. His research is in Automatic Control and Computer Simulation and Optimization. He has designed and developed CORAL, an optimal control system for sewer networks, applied at Barcelona (Spain); PLIO, an optimal control sys-tem and planner for drinking water production and distribution, applied at Santiago de Chile (Chile) and Murcia (Spain). Nowadays He participates in different industrial projects, like the power consumption optimization of tramway lines in Barcelona with TRAM and SIEMENS and the development of tooPath (http://www.toopath.com) a free web tracking system of mobile devices. He is also the local representative of OSM (http://www.openstreetmap.org) in Catalonia and participates in different FOSS projects.

ANTONI GUASCH is a research engineer focusing on modelling, simulation and optimization of dynamic systems. He received his Ph.D. from the UPC in 1987. He is an Associate Professor in the department of "Ingeniería de Sistemas, Automática e Informática Industrial" in the UPC and head of Simulation and Industrial Optimization at inLab FIB (http://inlab.fib.upc.edu/). Since 1990, Prof Guasch has lead more than 40 industrial and research projects related with modelling, simulation and optimization of nuclear, textile, transportation, car manufacturing, water, pharmaceutical and steel industrial processes