Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research

UNIVERSITE IBN KHALDOUN - TIARET

Master Thesis

Presented in:

MATHEMATICS AND COMPUTER FACULTY
COMPUTER SCIENCE DEPARTMENT

In view of obtaining:

MASTER'S DEGREE

Speciality: Software engineering

By:

NAGGAR Ilyes

On the subject

Specification of the lifetime of states in the DEVS formalism by fuzzy controller: Application to the propagation of forest fires

---

Presented the 23 / 12 / 2020 in Tiaret in front of the jury:

Mr MEBAREK Bendaoud	MCA	University Ibn Khaldoun	President
Mr DAHMANI Youcef	PROF	University Ibn Khaldoun	Supervisor
Mr MOSTEFAOUI Kadda	MAA	University Ibn Khaldoun	Examiner

Academic year: 2019 /2020

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Réseaux et télécommunications

Par :

NAGGAR Ilyes

Sur le thème

Spécification de la durée de vie des états dans le formalisme DEVS par contrôleur flou : Application à la propagation des feux de forêts

---

Soutenu le 23 / 12 / 2020 à Tiaret devant le jury composé de :

Mr MEBAREK Bendaoud	MCA	University Ibn Khaldoun	Président
Mr DAHMANI Youcef	PROF	University Ibn Khaldoun	Encadreur
Mr MOSTEFAOUI Kadda	MAA	University Ibn Khaldoun	Examinateur

Année Universitaire: 2019 /2020

Acknowledgement

First of all, we want to thank the Almighty and Merciful God, who has given us the strength and patience to do this modest work.

Secondly, we would like to thank our supervisor Mr. **Dahmani Yucef**, for his precious advice and his help throughout the period of work.

Our sincere thanks also goes to the jury for the interest they took in our memory by agreeing to examine our work and enrich it with their insights.

Finally, we would also like to thank all the people who participated directly or indirectly in the realization of this work.

DEDICATION

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my loving parents whose words of encouragement and push for tenacity ring in my ears. My sisters and brothers and loved one I do appreciate everything you've done for me.

I also dedicate this dissertation to my many friends who have supported me throughout the process. I will always appreciate all they have done.

Table of Contents

List of figures

List of acronyms

C

CoG
Center Of Gravitiy · 23, 24

D

D.O.D.
DEPARTMENT OF DEFENSE · 15, 49
DEVS
Discrete Event Systems · 10, 11, 13, 16, 17, 18, 19, 25, 27, 28, 29, 33, 35, 37, 38, 39, 42, 45, 48, 49

E

EIC
external input couplings · 18, 19, 29
EOC
external output couplings · 18, 19, 29

F

FFP
Fire Front Propagation · 44
FIS
Fuzzy Inference System · 22, 24, 30, 33, 39, 42, 43, 44
FLC
Fuzzy Logic Control · 22, 27

I

IC
internal couplings · 18, 19, 29
IDE
integrated development environment · 40

M

M&S
Modeling and simulation · 15

U

UML
Unified Modeling Language · 41

I. General introduction

Forest fire or more commonly named Wildfire, is defined by many as an uncontrolled and unplanned fire that devastates certain areas with high combustible vegetation (Cambridge Advanced Learner's Dictionary, 2008).

Every year wild fires burn a substantial amount of the forests area all around the world causing environment changes that afflicts nature and mankind alike. In Algeria alone statistics show 42,555 fires happened from the year 1985 to year 2010, resulting in 910,640 hectares to be burned (MEDDOUR, DERRIDJ, 2012).

Countries that suffer every year from fires keep developing their counter measures. These efforts range from modernizing the large air tanker fleet to using satellites that oversee the fire in a more detailed manner and properly trained firefighters with a good understanding of fire behaviour and all the factors that can affect the spread of fires, yet every year fires causes a lot of losses (U.S. FOREST SERVICE, 2020).

Though several studies were made by experts of the field, they resulted mostly in models that addresses time or temperature at which a certain area would ignite at, such as Porterie et *al*., (2005 and 2007 *in* DRISSI, 2013), most these models deal with fire itself focusing on its aspects such as the chemical or energy that relates to combustion…, but few deal with fires in a more global view such as how long does a fire need to spread and which factors do influence its propagation.

Through simulation this work intends to create a tool that is more focused on how the fire will spread and the time needed to. To give fire fighters a better chance in their fight. Forest fires have many factors that contribute to its behaviour such as wind speed, landscape, type of vegetation, humidity etc...(JONATHAN, 1998), which renders it impossible to create a simulation that can encapsulate all of it.

To be able to simulate a phenomenon such as wild fires, it must be viewed as a system with attributes and components that represents its structure and how it behaves in a defined manner using system modelling. Creating a model that describe the targeted system is not the final answer due to the existence of numerous system modelling techniques and formalisms such as the one used in this work DEVS “Discrete Event Systems”.

DEVS is a formalism introduced to the world in the work of Bernard P. Zeigler in 1976. It focuses on the key components and factors simplifying the system and thus facilitating the simulation affording a better computational ground for implementing real world phenomenon behaviours (Zeigler, 1976).

Being DEVS a modelling formalism that sees the development of a system by timing every state of the system, the time that each state the wild fire will have will be influenced by the aforementioned factors (Zeigler et *al*., 2000).

Since wild fires have different behaviours and their propagation speed differs whenever factors change even slightly makes it hard to evaluate how long and how far a fire will spread, thus the need to design a system that follows certain conditions and decide upon them how long each state will last.

Having such a mechanism where results aren't characterized by just fast or slow speeds of propagation based on the traditional logic that sees everything either true or false. To describe different levels of speeds to be as similar to the real world wildfire is to extend the Boolean logic with more values, known as Fuzzy Logic.

Fuzzy Logic has been developed to emulate the human reasoning, giving the possibility to have values can be in between 0 or 1 i.e. neither true nor false. A way to implement such logic rules based on the factors that are deemed to be important are defined to have more versatile results, this implementation is known as a Fuzzy Controller (Jager, 1995).

This document will proceed by establishing the needed theory in the first chapter by familiarising the reader with wild fires and factors that dictate its behaviour and an explanation of DEVS and Fuzzy logic inner workings that are important to this work, then in the 2nd chapter the model of the targeted system will be explained and how all its components shall work to finish with its implementation in the 3rd chapter.

CHAPTER 1

II. Chapter 1: Concepts

II.1 Wildfire

This work will be dedicated to the creation of a simulation that has for a system wildfires and depict them using DEVS formalism as a behavioural basis adding the fuzzy controller to determine the life time of the wildfire.

II.1.1 Definition

Forest fire is an exothermic (release of heat) reaction that develops without control in time and space. It is the most devastating factor of degradation due to the losses caused by its intensity and brutality affecting large areas of forests in short periods of time, becoming more and more frequent and violent (JAPPIOT et *al*., 2002) commonly known as wild fires.

Three critical parameters are needed for fire which are Heat the more the temperature is high the more a chance to have a fire are, Fuel which is the is mainly a material that will feed the fire to keep burning and finally Oxygen which is a compound in phenomenon of combustion. These conditions are traditionally portrayed in the triangle of fire (Fig.1) (Kutz, 2012).

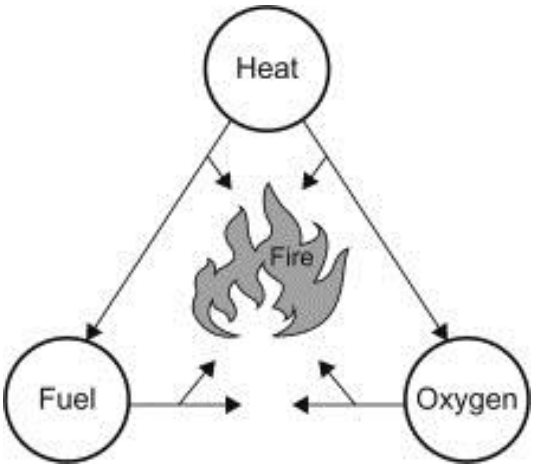

Fig.1 fire triangle (MYER Kutz, 2012)

II.1.2 Wildfire shapes

Wild fires can have different shapes (Fig.3) when spreading Circular, ellipse shaped or irregular. These shapes are caused by many factors related to the landscape the wind speed

humidity or the vegetation…etc. in many literatures they are called risk factors since they have hold over how the fire will behave (BENKRAOUDA. 2015).

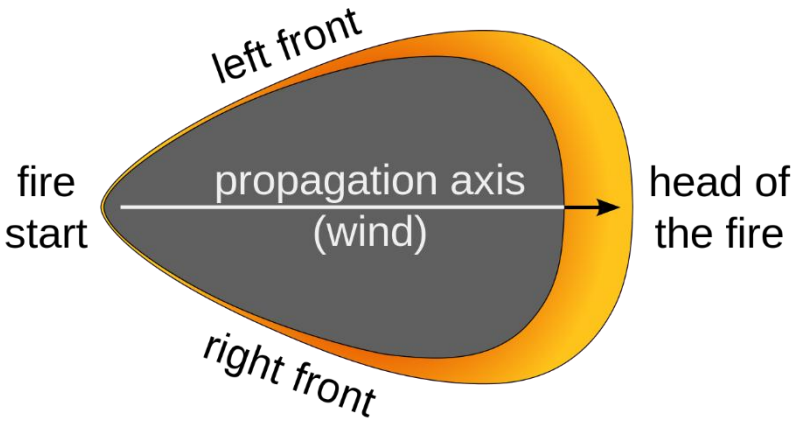

Fig.2 Fire propagation

Fires when propagating in an area different places can be distinguished (Fig.2) such as the starting point of the fire and which side is the most influenced by wind direction named head of the fire enveloped by the left and right fronts.

II.1.3 Risk Factors

Risk factors are all the natural factors that contribute to a wildfire's spreading speed and influence its behaviour and more the factors are combined more the fire's behaviour can become complex and impredictible (Ammari, 2011).

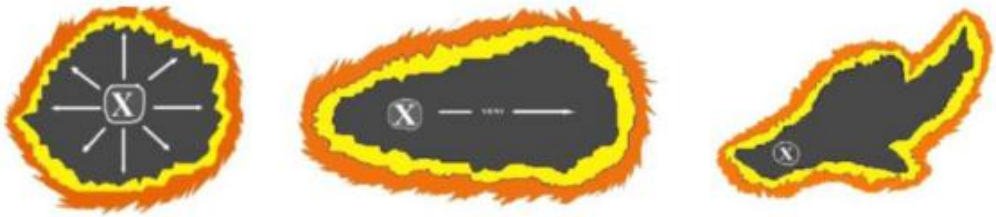

Fig.3 Wildfire's different shapes

The Circular shape is commonly found in flat terrains with calm winds and the vegetation has the same burning speed in all area which makes the starting point the center and all sides can be head of the fire making it the easiest to predict.

The ellipse shaped one will have its head as the wider side which is following the wind direction when winds are strong enough in the direction of an axis known as the propagation axis.

The irregular one is influenced by terrains with slopes and studies show that fire is faster at climbing them than when descending, which results in different speeds in different areas of the fire causing some to be fast some to be slow creating a shape that isn't uniform and is much harder to predict compared to the other shapes.

II.2 Modelling and Simulation

A model is the establishment of a mathematical or logical representation of a phenomenon, or process in the real world. Implementing a model and observing its behaviour over time is called a simulation, a process that brings a model to life and demonstrates how a particular object or phenomenon will behave. Simulating a model is a very advantageous process that scientists can use due to the cost efficiency it provides (Fig.4), making tests, analysis and training more approachable (D.O.D., 2001).

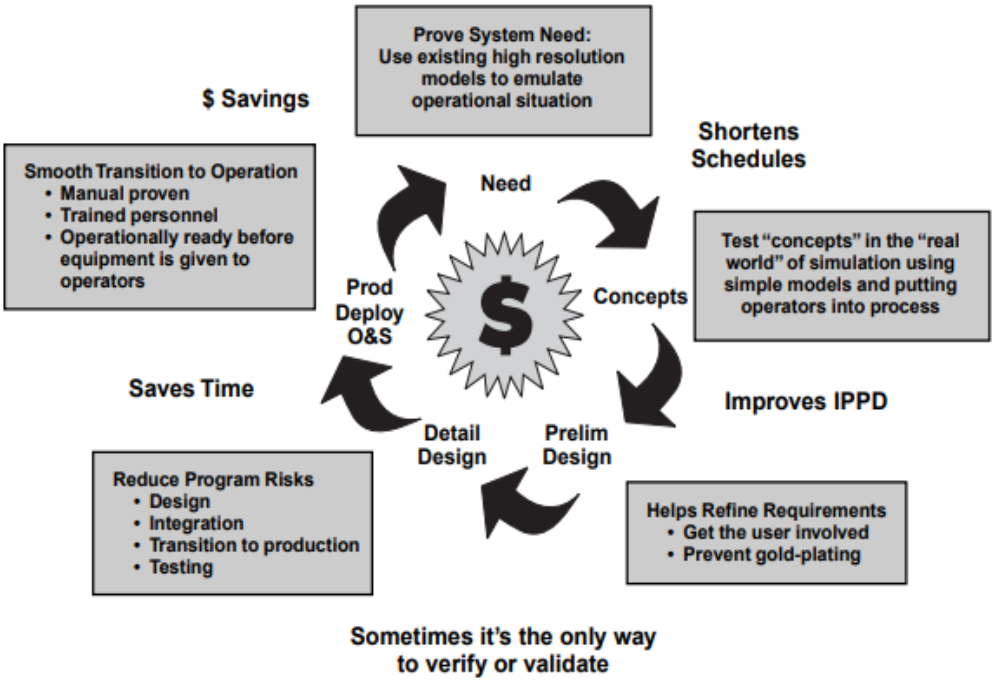

Fig.4 Representation of Advantages in Modelling and Simulation (D.O.D., 2001)

Modelling and simulation (M&S) provides a tool that can copycat products and processes, and represents them in a controllable environment and operationally valid compared to the real world (D.O.D., 2001).

As stated earlier DEVS is one of many techniques for creating models that describe complex systems and make it possible to simulate them.

II.2.1 DEVS

Discrete Event Simulation is used to model the complex behaviour of a system, using events occurring at different points in time changing the internal state of the said system. DEVS systems are specified using a structure that defines its components, events, states etc… (Zeigler, 1976).

The structure of a model is expressed in a mathematical language known as a formalism. Discrete event focuses on the changes that occur to variable values and calculates time segments that are constant i.e. an event is a change in a variable value that happens instantaneously. In essence the model defines how to obtain new values for variables and the when to apply them. One of DEVS important aspect is that the time intervals between event occurrences are not fixed (Zeigler et *al.*, 2005).

Any Model can have states that represent the current condition of the process or system being simulated. States are defined mainly by using two values "PHASE" that represent the state is being maintained, and "SIGMA" the time value that tells how long the state will be lasting i.e. it shows when the state might end or being changed (Wainer, 2009).

II.2.2 Atomic Model

The most basic model in the DEVS formalism is the atomic mode, it describes all the important parts and components a model needs to operate in good conditions. The atomic model is used in all DEVS simulations since he has the ability to exactly define all different components in complex systems through the mathematical representation (Cassandras et *al.*,2007):

$$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$$

- **X**: set of input values.
- **S**: set of states.
- **Y**: set of output values.
- $\boldsymbol{\delta_{int}}$: $S \rightarrow S$ is the internal transition function

- δ_{ext}: $Q \times X \rightarrow S$ is the external transition function, where
  $Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$ is the total state set

  e is the time elapsed since last transition

- λ: $S \rightarrow Y$ is the output function.
- t_a: $S \rightarrow R^{+}_{0,\infty}$ is the time advance function.

These components describe the behaviour a DEVS model will follow, at any time the system has a known state S, if no external events happens the system will keep it for time ta(s) (Time Advancement function). Ta(s) could be either a real number, 0 or ∞.

II.2.3 Atomic Model Behaviour

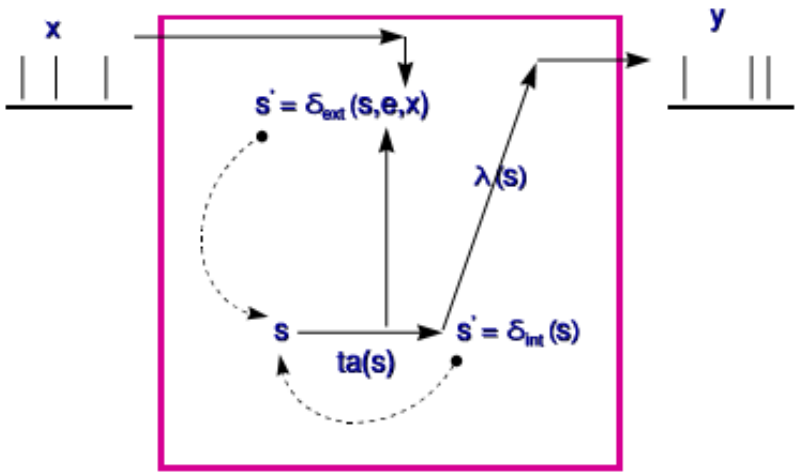

Fig.5 DEVS model behaviour

The behaviour in a DEVS model can have several scenarios that can happen (Fig.5). One Case is the duration of the state **S** is too short that no external event can intervene, the state is called transitory. Another case the system will keep the state **S** forever unless an external event occurs to change it, this state is passive.

As defined **e** is the amount of time since last state transition, if **e** exceeds the duration of the state **S** (calculated by **ta(s)**) the system will output a value **λ(s)** then change the state of the

system using **δint(s)**. Or an external event happens while **e ≤ ta(s)** the system will execute the **δext(s, e, x)**. Both transitions will dictate the new state **S′** and its duration **ta (S′)**.

Note that output is only possible just before internal transitions, which enables the model to output a value before any transition can occur if the model is designed to have an internal transition then the external in the confluent function.

II.2.4 Coupled Model

The DEVS formalism is designed to be built in a modular fashion using models called basic models then combined to create more complex models. Different to traditional simulation languages, DEVS models have input and output ports. Values appearing on each port are determined by events or functions. When external events occur outside the model, values are supplied to the corresponding input port. The model description must describe how it responds to each possible event, same goes for events that happen within the model generating values that are communicated through the corresponding output port to other models (B.P. Zeigler et al, 2005).

Following the mathematical description of a DEVS model a Basic Model is defined, starting with a set of inputs which deal with external events, another set of outputs to send external events, a set for all possible states. A time advance function to control internal transitions occurrences, internal and external transition functions that specifies the next state by changing the phase and sigma values computed based on the current state and/or the values from an input port. The confluent transition function which defines which transition function would be called first, and lastly an output function that generates an external output before the internal transition (B.P. Zeigler et al, 2005).

DEVS formalism introduced the coupled model as a solution to working in a parallel fashion since atomic models were limited to sequential processes. By adding the coupling aspect it became possible to build models from already made components using external and internal coupling (Fig.6) resulting in this specification:

$$CM = \langle D, X, Y, EIC, EOC, IC \rangle$$

- **D:** a set of components.
- **X:** a set inputs and possible values.
- **Y:** a set of outputs and possible values.

- **EIC:** external input couplings connect external inputs to component inputs**.**
- **EOC:** external output couplings connect component outputs to external.
- **IC:** internal couplings connect component outputs to component inputs.

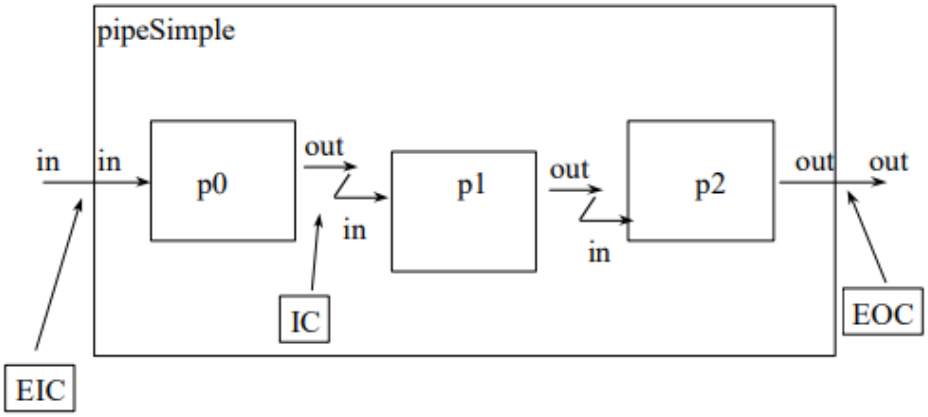

Fig.6 Coupling in a DEVS coupled model

II.3 Fuzzy Logic

Fuzzy Logic is a logic that sheds light on principles of human reasoning, Contrary to classical logic that focuses only on propositions that are either true or false. This classical logic works with combinations of variables in the form of propositions. Each variable stands for a hypothetical proposition, any given combination is evaluated to be either true or false, but never the combination of both (i.e., is not true and false at the same time) (Chen et *al*., 2000).

The principle of having one Truth value (true or false) has been questioned by many researchers even on a philosophical level. The more complex is the study the more unsatisfactory the results were, to be just true or false completely disregarding the factors that might render it opposite to what it might really be (Yager et *al.*,1992).

It is well established in recent decades that evaluating propositions should be regarded partially true and false at the same time. Multi valued logics were introduced to describe this new reasoning, the most basic is adding a third value to the already known True and False. By this addition it became possible to have more accurate and diverse results, especially propositions that were hard to determine their truthfulness (Chen et *al*., 2000).

II.3.1 Fuzzy Sets

In this work it is assumed that the reader is familiar with fundamentals of the theory of crisp sets which fuzzy sets are based on and introduced in the work of Lotfi Zadeh.

Humans and Computers, two different entities each with their advantages. While the computation power of computers far exceeds the human's, it still cannot cope with decision making of the human brain, partially due to the computer's logic that is derived from the Boolean logic rendering a lot of results unsatisfactory. In 1960s the idea of fuzzy Sets was introduced as a way to implement the human decision making in software engineering where the values of a given set are valued based on the degree of membership of each individual (Dimitrov et *al.*, 2002).

From this, we can understand the difference between a classical set and a fuzzy set. Classical sets contain elements that satisfy all properties of membership while fuzzy sets contain elements that satisfy partially the properties of membership.

II.3.1.1 Membership function

The characteristic function of a crisp set assigns a value of either 1 or 0 to each individual in the universal set, thereby discriminating between members and non-members of the set under consideration. This function can be used in a way that the values assigned to the elements of the set to fall within a specified range and indicate the membership grade of these elements in the set in question. Larger values denote higher degrees of set membership, such a function is called a membership function, and the set defined by it becomes a fuzzy set (Klir et *al.*, 1995).

The most commonly used range of values of membership functions is the unit interval [0, 1]. In this case, each membership function maps elements of a given universal set X, which is always a crisp set, into real numbers in [0, 1].

Each fuzzy set is completely and uniquely defined by one particular membership function. A fuzzy set A will have a membership function denoted μ_A.

$$\boldsymbol{\mu_A : X \rightarrow [0, 1]}$$

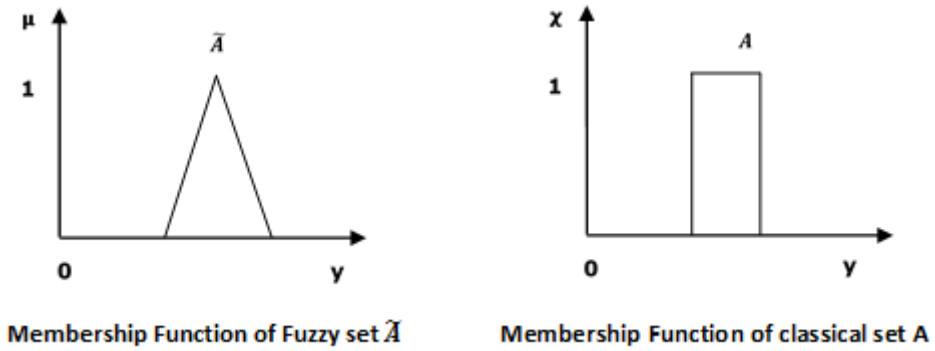

Fig.7 Membership in fuzzy and crisp sets

In (Fig.7) the membership of individuals in the set. On the right represents a crisp set, the individual must satisfy all the properties of the Set to get a result 1 (true), but in the fuzzy set in the left the individual can satisfy partially the properties of the set and have different degrees of membership ranging from non-membership with a value 0 to full membership with value 1.

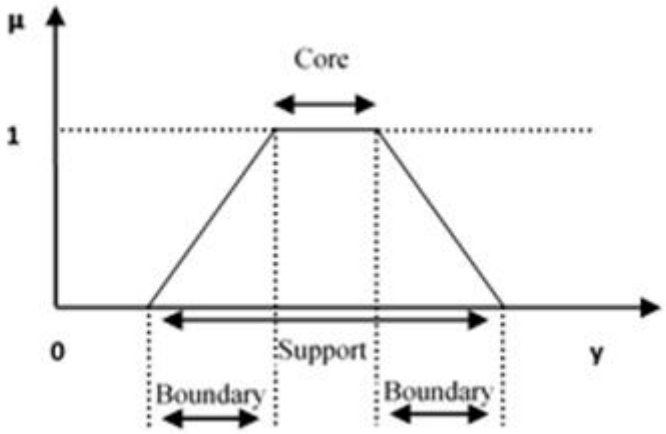

Fig.8 Membership Function Regions

Membership functions consist of 3 possible regions core, support or boundary as shown in (Fig.8).

- **Core**: represents values that are fully members of the fuzzy set i.e. $\mu_{\tilde{A}}(y) = 1$.
- **Support**: represents values that have a non-zero membership of the fuzzy set, $\mu_{\tilde{A}}(y) > 0$.
- **Boundary**: consists of the values that are non-zero but does not satisfy a full membership i.e. incomplete membership, $1 > \mu_{\tilde{A}}(y) > 0$.

II.3.2 Fuzzy Control system

Fuzzy Set Theory, has been the subject of many researchers. Used in many applications in different fields. Amongst the most successful applications of Zadeh's theory was in the area of Fuzzy Logic Control (FLC) pioneered by the work of Mamdani and Assilian. FLC has had considerable success in many commercial products using this technology such as braking systems in vehicles (McNeill et al., 1994).

FLCs are based on rules in the decision making process to have results similar to what a human expert would obtain in an automated frame. FIS or Fuzzy Inference System is the core component of the FLC, it defines the needed rules as IF-THEN sequences with AND or OR that are applied to the input values that are represented by linguistic terms that subdivide the fuzzy set to sub-sets. Linguistic terms help make decisions by determining the membership of input values to their respective sub-sets using the membership function (Hampel et al, 2000).

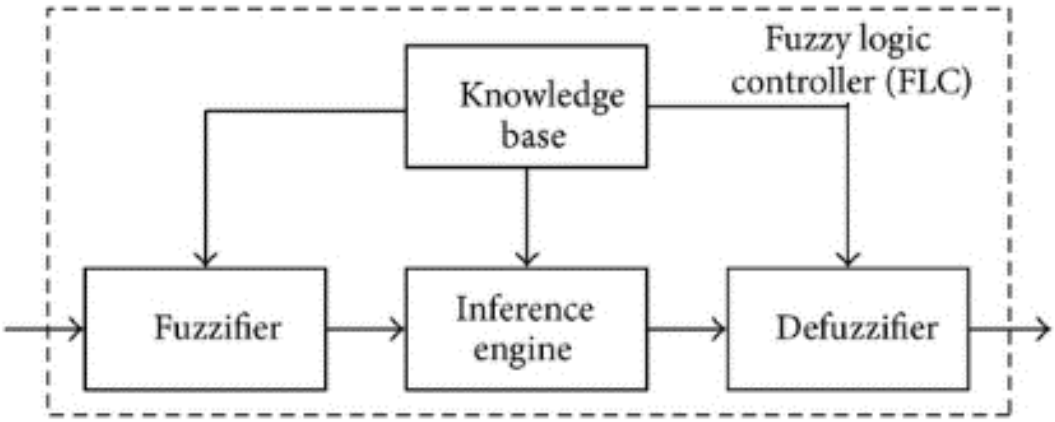

Fig.9 Simple architecture of a FLC

The inner working of Fuzzy Controllers is based several important components shown in (Fig.9) (Cirstea et *al.*, 2002):

- **Knowledge Base**: It contains the two most crucial information in making decisions, the rule base that has the IF THEN rules and a DATA BASE defining membership functions of the used fuzzy set.
- **Inference engine**: it acts as a brain by taking the right decision using the rules on the fuzzy values it receives from the fuzzifier then sends the results to the deffizifier.
- **Fuzzifier:** It converts the crisp values from input to fuzzy values the passes them to the inference system.
- **Defuzzifier**: It converts the fuzzy values resulting coming from the inference engine into crisp values.

II.3.2.1 Fuzzy Inference System

The inference process of a fuzzy controller performs two steps, rule matching and rule firing. The matching step compares each rule to the input variables. The firing rule step is to generating membership function for the output for the matched rule using modus ponens compositional rule of inference (Baudoin et al., 2016).

Each input variable is fuzzified by mapping it using the defined membership function to one of its regions. Using mathematical methods such as Support Fuzzification or Grade Fuzzification (Azeem, 2012).

Deffuzification renders the fuzzy results to crisp values by the use of methods like the Max-Membership Method that is limited to peak output functions or the Centroid Method that focuses on the center of the area, for discrete triangular linear functions the CoG method is obtained by moments of area as defined by (Baudoin et al., 2016):

$$\hat{c} = \frac{\sum_{c_i} c_i \mu_c(c_i)}{\sum_{c_i} \mu_c(c_i)}$$

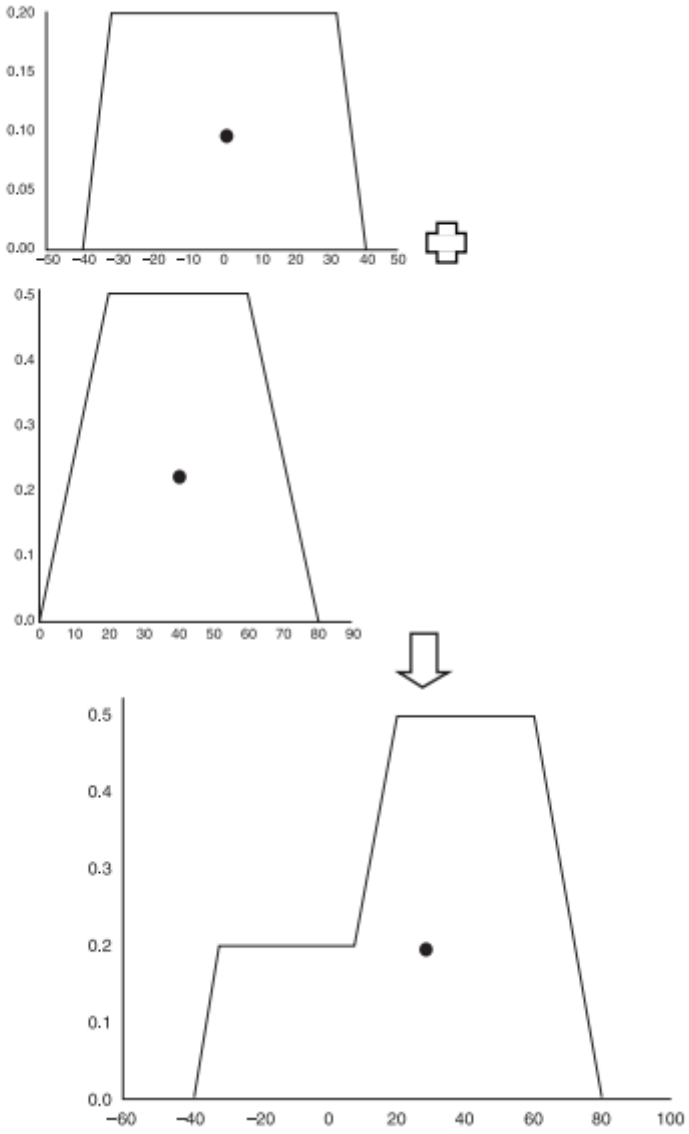

Fig.10 Example of CoG

Deffuzification is the last step in the FIS, using the center of gravity method the input variables membership functions are combined in what is called the fuzzification step then the center gravity of the membership functions of the chosen rule in the matching process then the center of the new membership function is calculated to be defuzzfied resulting in a crisp value as an output (Fig.10).

II.4 Conclusion

Learning the different aspects pertaining to fires shows that a wildfire a very complex system that thought it seem predictable from an experts point of view, from the view of an Algorithm or model a lot of factors and conditions are needed to express it in a good manner making it very hard to simulate.

DEVS shows its power by its simplicity to focus on the main aspects nullifying the hardships that a simulation like a wildfire might entail scrapping it to its most basic behavioural driven components and key points.

Fuzzy Logic has the potential to mimic uncertainty and quantifying it so a computer can emulate what real world phenomenon can do and expanding the benefit to wider horizon allowing us the ability to create expert systems that are reliable not to replace the humans but to aid them in their work.

CHAPTER 2

III. Chapter 2: Modelling

The theory that was previously discussed will be used to create a model that represents a wild fire, with states that show the different steps a wild fire will follow such as unburned and burning… through DEVS formalism.

The land to be simulated will be divided to cells where every cell has an atomic model that controls its state's transitions, these cells will be inter connected through the use of coupled models so that each cell can communicate with its neighbours to have a fire spread.

To capacitate the model with the ability to compute how long each state will last it will have access to a FLC that uses rules based off of researchers in the field or firefighter's experiences.

This FLC will have the capacity to determine how long a fire to burn a piece of land based on the natural risk factors that human experts will select, mostly to be the most pertinent factors that have great influence on the fire's behaviour such as wind speed, humidity.

III.1 Fire behaviour modelling

Establishing an Atomic model for a cell as the most basic component, the cell must have the capacity to change its own state following an input that notifies it of a fire or one that is already burning, thus the need to specify all the possible phases it can transition to. These states will represent the different stages a piece of land will go through while burning:

- Unburned: this state that shows the piece of land that is represented by the model is not burning with a sigma of ∞, since it can keep this state forever if no fire is ignited through an external event.
- Burning: this state happens with an external transition that will occur when an input of a fire is received with a sigma that will be computed with a fuzzy control system.
- Ember: the burning vegetation will become ember with a sigma using the same time advance function as the burning state.
- Burned: the last state the cell will transition to with a sigma of ∞.

This atomic model will be used to create a grid, to enable each cell to communicate to its neighbours that a fire is spreading, the connections will be established with the use of inputs and outputs.

Each cell needs will have up to eight neighbours, the cell can either notify or be notified by one of them. Using coupled models we can link all cells to each other creating a grid that can share the information of a fire spreading. The possible values to be sent between cells will be the word “Fire” which will ignite the first cell only to be replaced by the coordinates of the sending cell.

The internal transition function after the time elapses of the Burning state will transition the model to Ember with a Sigma that can be also be computed with the fuzzy controller, through a mathematical equation or a simple time value decided upon by the field’s expert, then after the life time of the Ember state finishes the internal transition function will set the state to Burned signalling the end of the life cycle with a Sigma = ∞.

A grid representing the area to be simulated will be created using the DEVS coupled model by interlacing a defined number of cells together through their input and output ports. This coupled model enables the cell to be connected through the use of internal coupling since the grid’s model doesn’t need any external input or output.

III.1.1 Cell Model

The previous texts described the behaviour of the model, through it the main aspects of the model can be extracted and refined to construct a working model that has the basic behaviour an area under fire would have in certain conditions such as wind’s speed, direction and humidity resulting in:

$$Cell = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$$

- **X**: “Fire”, position of Cell.
- **S**: Unburned, Burning, Ember, Burned.
- **Y**: position of Cell.
- δ_{int} (Burning) = Ember
  δ_{int}(Ember) = Burned.
- δ_{ext} (Unburned, “Fire”) = Burning.
  δ_{ext} (Unburned, sending Cell’s position) = Burning / Unburned.
- λ (Burning) = Cell Position.
- t_a (Burning) = T.

T will be calculated using the Fuzzy controller to determine the propagation rate of fire then used in a mathematical equation.

Each cell will determine its chances of catching fire following the wind direction by calculating the angle between the receiving cell and sending cell, the smaller the angle means the cell is in the way of the fire and has a high chance of burning and the bigger the angle the less chance the cell will burn, the angle will calculated trigonometry equation:

$$\arccos((P122 + P132 - P232) / (2 * P12 * P13))$$

Where P12 is the length of the segment from P1 to P2, calculated by:

$$sqrt((P1x - P2x)2 + (P1y - P2y)2)$$

III.1.2 Grid Model

The coupled model that will represent the grid will also follow the DEVS representation where the set of components will have a defined number of the Cell model with no external inputs or outputs and only Internal Coupling will be used since the grid doesn't receive input or sends output and only inner components will communicate with each other.

$$Grid = \langle D, X, Y, EIC, EOC, IC \rangle$$

- **D: (**row * Cell) * (col * Cell).
  Where row: number of rows in grid, col: number of columns in the grid.
- **X:** None.
- **Y:** None.
- **EIC:** None**.**
- **EOC:** None.
- **IC:** all eight neighbour's outputs are connected to the input of center cell of the eight. And output of center cell is connected to all inputs of the eight neighbours.

III.2 Fuzzy Controller

As mentioned in the DEVS model description the time advance function will use a fuzzy inference system that imitates a firefighter's experience to determine the speed the fire will advance by computing the lifetime of the Burning state.

The inference system will use a set of membership functions that represent the risk factors wind speed and humidity. These membership functions will be used to fuzzify the crisp values of wind and speed to obtain a result that is called the flaming front propagation which according to an expert is 3% to 8% of the wind's speed by Deffuzifiying the result (Bisgambiglia, 2008).

The membership functions will show the FIS the different intervals at which the wind and humidity can be characterized through the use of linguistic terms that represent different speeds or percentage of humidity. Wind can be described in literature as calm, slight or powerful …etc. as for humidity dry, little or too much...etc. There can be more detailed intervals to have more accurate results, as for the propagation range will have slow, medium and fast.

III.2.1 FIS Rules and linguistic terms

The linguistic terms that defines both wind speed and humidity membership functions will be used in creating a block of rules that the FIS will follow, different combinations of wind speed and humidity will have different influence on fire spread or more exactly the fire front propagation.

The rule block will use the IF-Then notations to infer what the result should be, from the previous texts the following rules were created:

- **RULE 1: IF** humidity **IS** dryHumidity **AND** windSpeed **IS** calmWind **THEN** propagationRate **IS** mediumTime.

- **RULE 2: IF** humidity **IS** dryHumidity **AND** windSpeed **IS** slightWind **THEN** propagationRate **IS** fastTime.

- **RULE 3: IF** humidity **IS** dryHumidity **AND** windSpeed **IS** powerWind **THEN** propagationRate **IS** fastTime.

- **RULE 4: IF** humidity **IS** littleHumidity **AND** windSpeed **IS** calmWind **THEN** propagationRate **IS** slowTime
- **RULE 5: IF** humidity **IS** littleHumidity **AND** windSpeed **IS** slightWind **THEN** propagationRate **IS** mediumTime.

- **RULE 6: IF** humidity **IS** littleHumidity **AND** windSpeed **IS** powerWind **THEN** propagationRate **IS** fastTime.

- **RULE 7: IF** humidity **IS** tooMuchHumidity **AND** windSpeed **IS** calmWind **THEN** propagationRate **IS** slowTime.

- **RULE 8: IF** humidity **IS** tooMuchHumidity **AND** windSpeed **IS** slightWind **THEN** propagationRate **IS** mediumTime.

- **RULE 9: IF** humidity **IS** tooMuchHumidity **AND** windSpeed **IS** powerWind **THEN** propagationRate **IS** mediumTime.

Wind speed membership functions (Fig.11):

- **calmWind:** [0, 50]
- **slightWind:** [20, 100]
- **powerWind:** [75, 118]

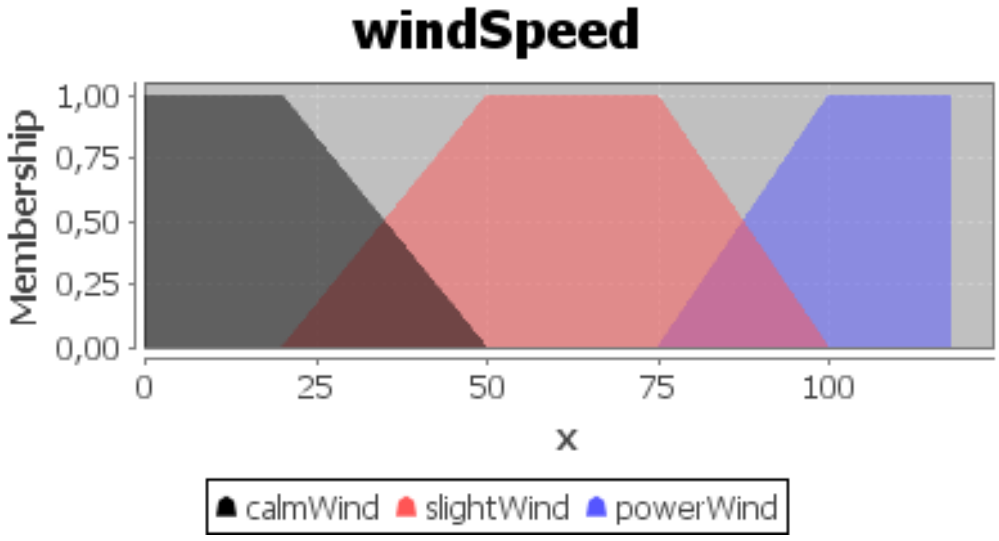

Fig.11 Graph of Wind speed membership functions

Humidity membership functions (Fig.12):

- **dryHumidity:** [0, 30]
- **littleHumidity:** [20, 80]
- **tooMuchHumidity:** [70, 100]

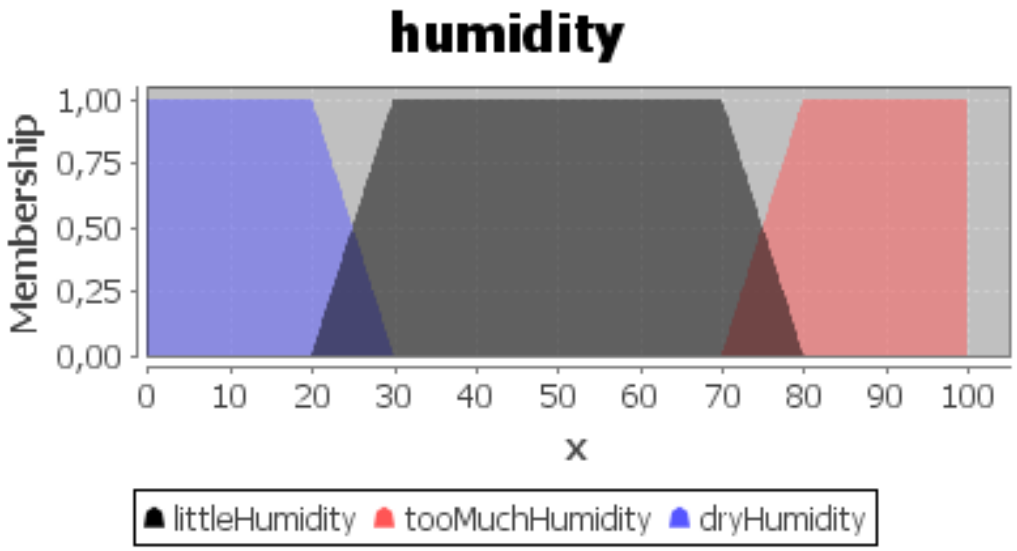

Fig.12 Graph of Humidity membership functions

Flaming front propagation membership functions (Fig.13):

- **slowTime:** [3, 5]
- **mediumTime:** [4, 7]
- **fastTime:** [6, 8]

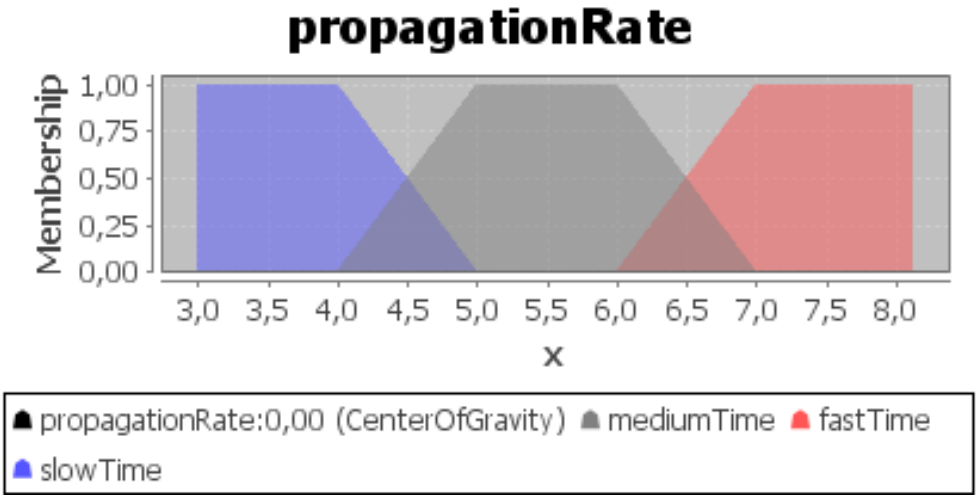

Fig.13 Graph of Fire front propagation membership functions

III.3 Class Diagram

Combining the DEVS models and the fuzzy logic controller will result in an application, which simulates a wildfire spreading in space through time controlled events. To be able engineer this piece of software some diagrams can be used to facilitate the programming aspect.

Most of the time several diagrams are made for the purpose of creating a functional reliable software, but due to the fact that this work will only focus on creating models and fuzzy controllers since the simulator will be the DEVS-Suite and that models use the java syntax to be defined only the Class Diagram shall be presented in this work to showcase how the models were defined and brought together to have FireSim.

As shown in the class diagram all components needed for the simulation are present in (Fig.14):

- **Cell:** represents the Cell model with, it inherits from the atomic model class.
- **Grid:** represents the Grid model, it inherits from the DEVS model class and implements the coupled model interface.
- **Atomic and DEVS:** are classes of the DEVS java that are used in the simulator.
- **Direction:** is a simple class used to represent (x, y) coordinates of each Cell, used to calculate if a cell is following the wind direction, perpendicular or completely opposite.
- **GuiGrid:** is used as a graphical interface to facilitate the uses to observe the simulation in a global manner and allow him to change wind speed, humidity and wind direction in real time.
- **FIS:** is the fuzzy logic library that will deal with fuzzifying and defuzzifying the input and output variables to calculate the fire front propagation.

The fuzzy controller was used in the CalculatedStateTime method of the Grid class since the Fuzzy Inference System will be created using fuzzy libraries instead of manually programming to ensure the results be accurate since the fuzzy controllers use mathematical functions such as the center of gravity in determining the results.

As for ChanceOfBurning method it calculates the chances a cell might burn by calculating the angle between the cell and wind direction if the angle is 0° then the chance is 100%, 45° then 75%, 90° then 50%, 135° then 25% and 180° is 10%.

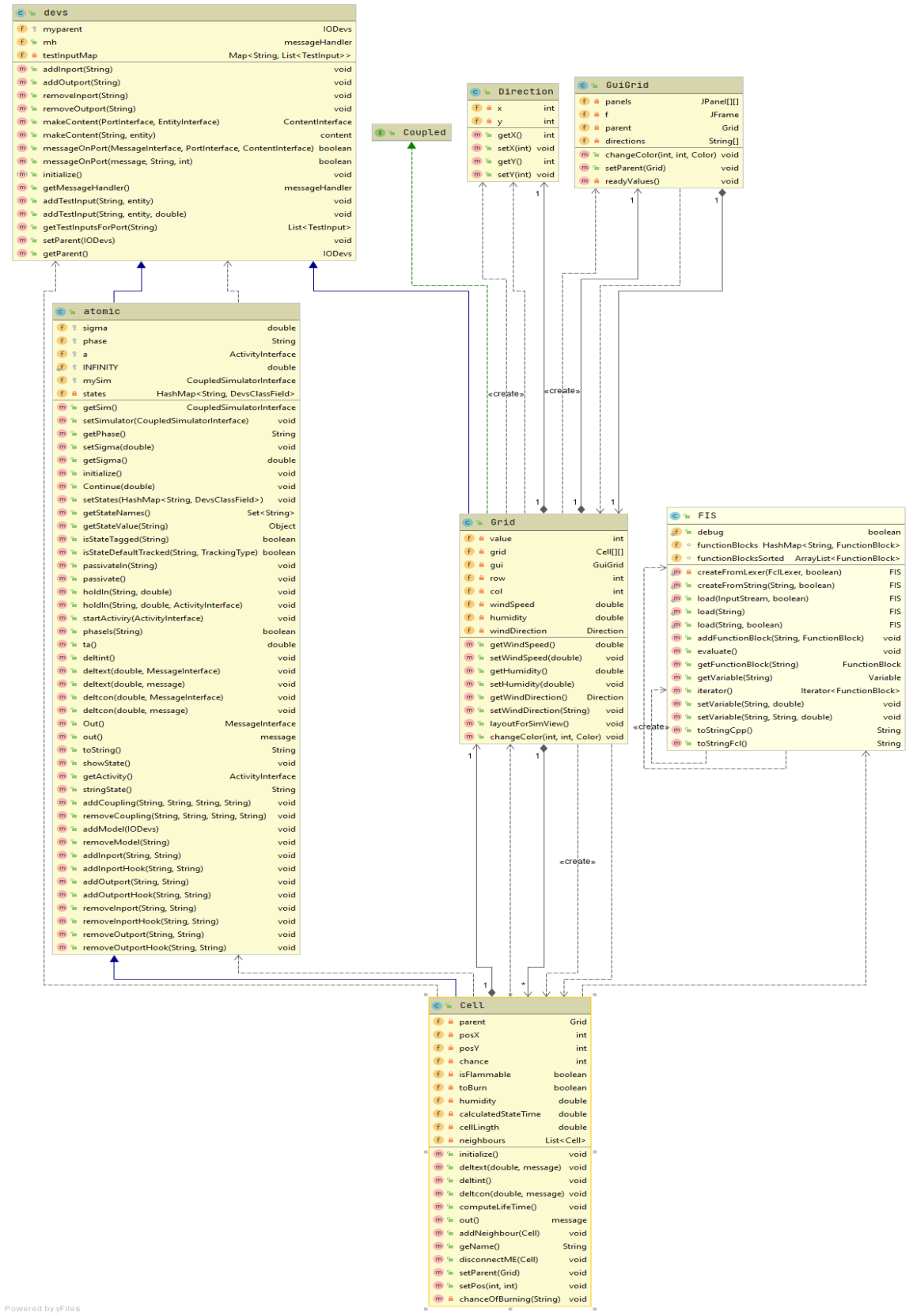

Fig.14 Class diagram

III.4 Conclusion

DEVS formalism enabled us to create a model that can perfectly represent a small area's behaviour under fire and dictate how it will transition its states after a calculated time make it simple yet powerful and modular rendering it updateable and upgradeable.

Coupled with a fuzzy controller that follows precise rules that are extracted from hundreds of hours of experience of experts and firefighters alike makes it possible to transfer all of it to a system that can enhance the results using computational power.

CHAPTER 3

IV. Chapter 3: Implementation

Culminating this work is the implementation of the conceptualized models of the 2nd chapter with the theory of the 1st chapter, by using different computer tools and ranging from simulator to libraries a Fire simulator was created.

The FireSim was created to allow simulating the fire spread behaviour based on different factors using a fuzzy controller system in making decision i.e. the FireSim uses the possibilities that the fuzzy logic offers to determine the fire front propagation to have a more natural behaviour.

The DEVS models were designed in a way that allows the addition of different factors and functions that can help advancing the inner workings of the fire simulator to obtain results that can be as close as to a real world fire.

DEVS-Suite was used a simulator since it is a well-established software in DEVS formalism created by the Arizona university, strengthened with capabilities and simplicity that JFuzzyLogic library ensures even if the user isn't familiar with fuzzy logic or programing for that matter.

These tools were chosen due to the fact that both are programed and still beign maintained based on the Java programing language, making it a combination that allows great results.

To finish this last chapter a presentation of the finished product will be conducted using screenshots with an actual simulation.

IV.1 Software

This section will introduce the softwares and tools that were used in this work, to familiarise the reader with it such as DEVS-Suite and JFuzzyLogic library:

IV.1.1 DEVS-Suite

Fig.15 DEVS-Suite Logo

DEVS-Suite (Fig.15) is a Parallel DEVS simulator that helps with automating designs of experiments in combination with animated models, generating timed data trajectories at run-time, synchronized viewing these trajectories alongside hierarchical components and I/O messaging animation. it offers rich visual modelling for specifying families of models and storing them in a Database, the software follows a lifecycle of modelling, simulation, and evaluation (University of Arizona website, 2020) .

- Action-Level Real-Time DEVS
- Black-Box Testing
- OSATE-DEVS-Suite is a tool built based on an integrated framework for the AADL (Architecture Analysis and Design Language) and DEVS (Discrete Event System Specification) modelling approaches.

The interface of the DEVS-Suite is simple (Fig.16), it comprises of a models viewer in the top left, and it enlists all the model present in simulation with a graphical representation on the right.

Each Model's state is also shown on the middle left portion with a "inject input" button that allows to send a predefined value to any input.

The left bottom panel has all the controls of the simulation such as run and request pause.

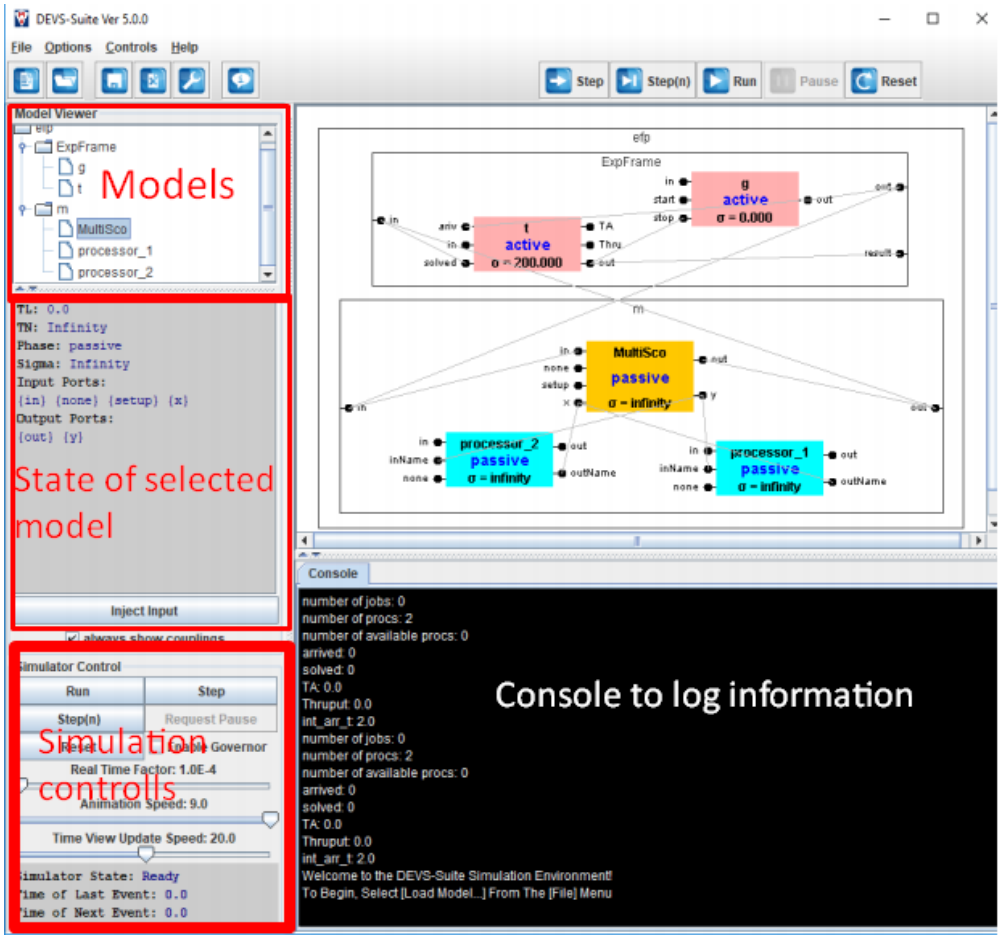

Fig.16 DEVS-Suite main menu

IV.1.2 JFuzzyLogic

JFuzzyLogic is an open source library created for java based applications. It was designed to facilitate the creation of fuzzy controllers and fuzzy inference systems according to the standard IEC 61131-7. This library was created relieve researchers of the substantial amount of technical aspects or those with lack of knowledge in fuzzy logic control when applying FLCs in projects, it provides graphical graphs of the FIS and results it obtains and an implementation for C/C++ compilers, allowing easy implementation of embedded control systems (P.CINGOLANI, J. Alcalá-Fdez, 2012) (P.CINGOLANI, J. Alcalá-Fdez, 2013).

JFuzzyLogic allows the user to visualise the graphs of the membership functions and the results of the fuzzy inference system with simple code by adding dialogs to the code when inputting the values to be calculated. And the rules and all input and output variables are defined in a file with the extension ".fcl" in simple terms to facilitate the work even further.

IV.1.3 Java

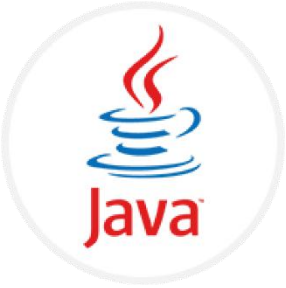

Fig.17 Java Logo

Java (Fig.17) is a programming language created by James Gosling and Patrick Naughton in 1995. Java is an Object oriented language of SunWorld then bought by Oracle. Created as a general-purpose programming language for developers to write once, run anywhere, in other words the compiled Java code can run on all platforms with Java support without recompilation.

Java applications are compiled to bytecode which can run on any Java virtual machine regardless of the computer architecture. The syntax is similar to C and C++, but has fewer low-level facilities than either of them (Oracle, 2020).

IV.1.4 Intellij idea

Fig.18 intelij IDEA Logo

IntelliJ IDEA (Fig.18) is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains, and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development (JetBrains, 2020).

The IDE offers many out of the box tools and functionalities that help the developers in their work making it easier and efficient, such as smart code completion, framework-specific assistance, and deep intelligence that indexes the programmer's source code allowing a fast and reliable suggestions while working (JetBrains, 2020).

IV.1.5 Unified Modelling Language

Fig.19 UML Logo

The Unified Modelling Language (Fig.19) is the modern go to developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system (Visual Paradigm, 2020).

UML 2 has many types of diagrams, which are divided into two categories. Some types represent structural information, and the rest represent general types of behaviour, including those that represent different aspects of interactions.

IV.2 Hardware

Fig.20 Lenovo Yoga Logo

All works were done including conception and implementation were performed using a Lenovo YOGA (Fig.20) pc with the following hardware configuration:

- Model: YOGA 520-14IKB.
- Processor: Intel(R) Core(TM) i3-7100U CPU 2.40GHz.
- RAM: 4Go DDR4 2133MHz.
- Storage Capacity: 128Go SSD.

IV.3 FireSim

FireSim is the result of this work, it is a java application that uses the DEVS formalism as a foundation and built on top of DEVS-Suite. The models that were created for the simulation are Cell the atomic model (Fig.21) and Grid the coupled model that uses hundreds of examples of the Cell model (Fig.22).

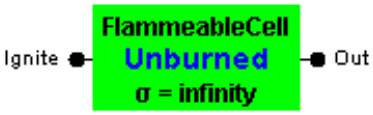

Fig.21 graphical reprsentation of one cell in DEVS-SUite

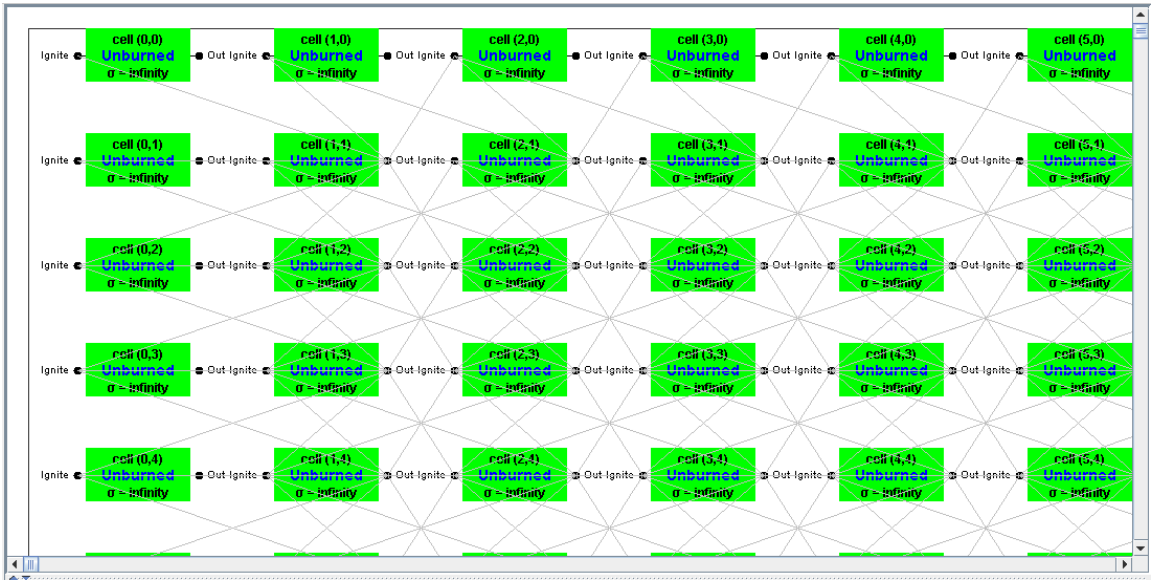

Fig. 22 Graphical reprsentation of the coupled model of the grid

The coupled model as shown comprises of a substantial amount of atom models which are Cells interconnected with every cell having a coupling with all eight neighbours.

As for the fuzzy controller using the JFuzzyLogic library a FCL file was created following a set of blocks. These blocks define the membership functions, which will be fuzzified and which to be deffuzzified and how the values shall be determined in this instance:

A block that defines the input and output variables of the FIS (Fig.23):

```
2
3     VAR_INPUT
4         windSpeed : real;
5         humidity: real;
6     END_VAR
7
8     VAR_OUTPUT
9         propagationRate : real;
10    END_VAR
11
```

Fig.23 FIS input and output variables

Then the membership function of all variables (Fig.24):

```
11
12    FUZZIFY windSpeed
13        TERM calmWind := (0,1) (20,1) (50,0);
14        TERM slightWind := (20,0) (50,1) (75,1) (100,0);
15        TERM powerWind := (75,0) (100,1) (118,1);
16    END_FUZZIFY
17
18    FUZZIFY humidity
19        TERM dryHumidity := (0,1) (20,1) (30,0);
20        TERM littleHumidity := (20,0) (30,1) (70,1) (80,0);
21        TERM tooMuchHumidity := (70,0) (80,1) (100,1);
22    END_FUZZIFY
23
24    DEFUZZIFY propagationRate
25        TERM slowTime := (3,1) (4,1) (5,0);
26        TERM mediumTime := (4, 0) (5,1) (6,1) (7,0);
27        TERM fastTime := (6,0) (7,1) (8,1);
28        METHOD : COG;
29        DEFAULT := 0;
30    END_DEFUZZIFY
31
```

Fig.24 FIS membership functions

The numbers are coordinates used to draw the graph of each membership function and the Deffuziying is done using the COG function.

Lastly the rules are added so the FIS can work properly (Fig.25):

```
31
32  RULEBLOCK No1
33
34      AND : MIN;
35      ACT : MIN;
36      ACCU : MAX;
37
38      RULE 1 : IF humidity IS dryHumidity AND windSpeed IS calmWind THEN propagationRate IS mediumTime;
39      RULE 2 : IF humidity IS dryHumidity AND windSpeed IS slightWind THEN propagationRate IS fastTime;
40      RULE 3 : IF humidity IS dryHumidity AND windSpeed IS powerWind THEN propagationRate IS fastTime;
41
42      RULE 4 : IF humidity IS littleHumidity AND windSpeed IS calmWind THEN propagationRate IS slowTime;
43      RULE 5 : IF humidity IS littleHumidity AND windSpeed IS slightWind THEN propagationRate IS mediumTime;
44      RULE 6 : IF humidity IS littleHumidity AND windSpeed IS powerWind THEN propagationRate IS fastTime;
45
46      RULE 7 : IF humidity IS tooMuchHumidity AND windSpeed IS calmWind THEN propagationRate IS slowTime;
47      RULE 8 : IF humidity IS tooMuchHumidity AND windSpeed IS slightWind THEN propagationRate IS mediumTime;
48      RULE 9 : IF humidity IS tooMuchHumidity AND windSpeed IS powerWind THEN propagationRate IS mediumTime;
49
50  END_RULEBLOCK
51
```

Fig.25 FIS Rules

Using the membership functions and rules by fuzzifying the crisp inputs of wind speed and humidity we can obtain a fire front propagation by deffuzzifying the result of applying the center of gravity to the fuzzifiyed values of wind speed and humidity.

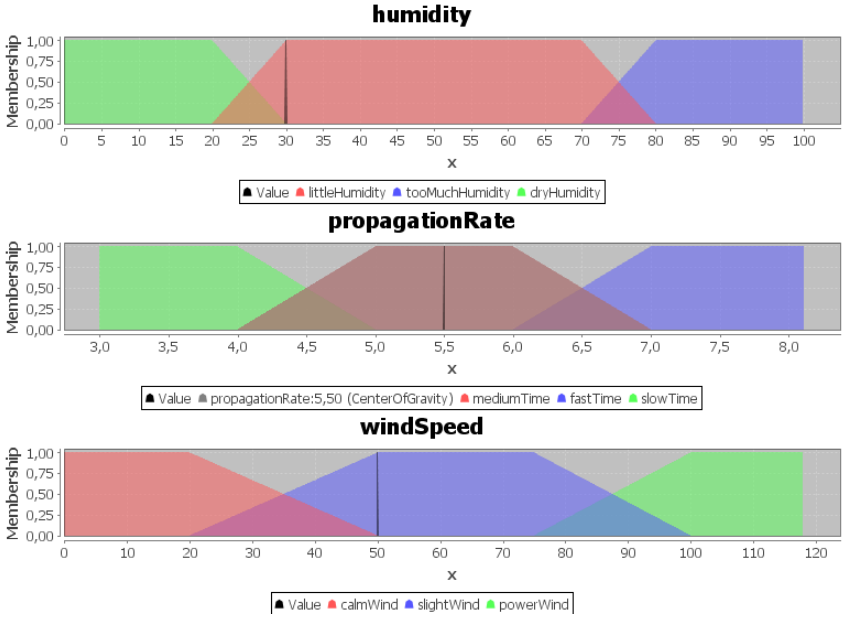

Fig.26 Example of FIS result

As seen by giving the crisp values Humidity = 30% and Wind Speed = 50Km/h the FIS determined using the membership functions that Humidity is “littleHumidity” and wind speed is “slightWind” then using “Rule 5” the FIS found that fire front propagation is memdiumTime using the COG function the FFP = 5.5% (Fig.26).

To have a better view of the simulation an interface that represents the whole coupled model for a better observation was designed for this purpose and allowing the user to have some in real time control over the simulation Fig.27.

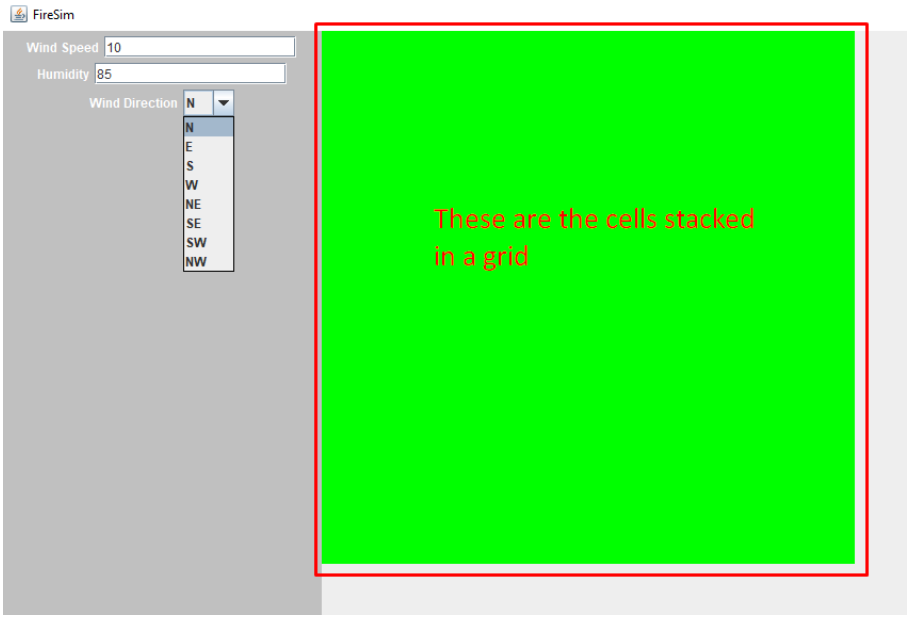

Fig.27 FireSim global view of the grid with options

The user selects a cell as the starting point of the fire in the DEVS-Suite's models viewer by injecting an input that represents the external event "Fire" the simulation can be started to be viewed in both DEVS-Suite for a detailed observation or the FireSim for a global observation (Fig.28) (Fig.29).

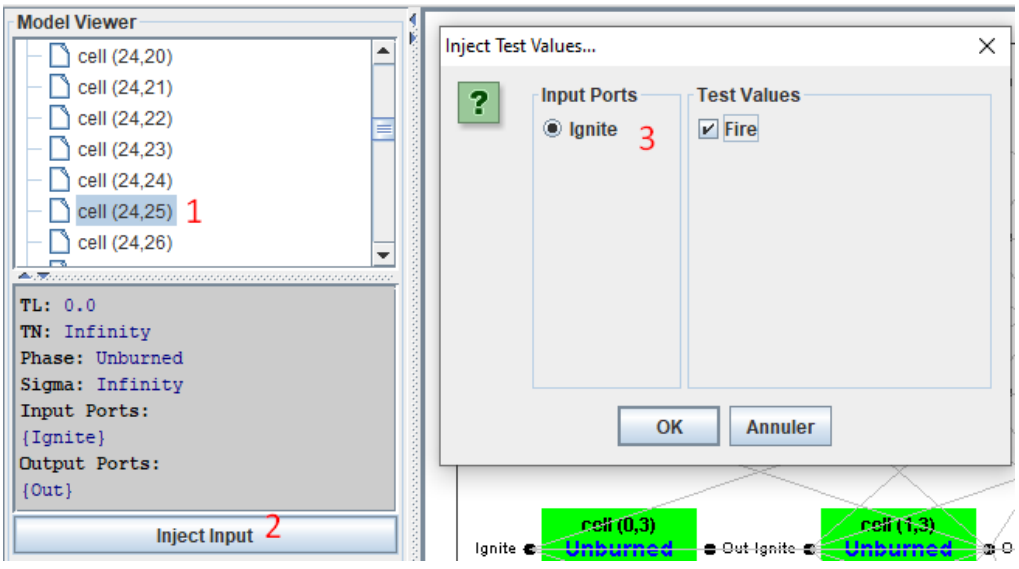

Fig.28 selection of a starting point

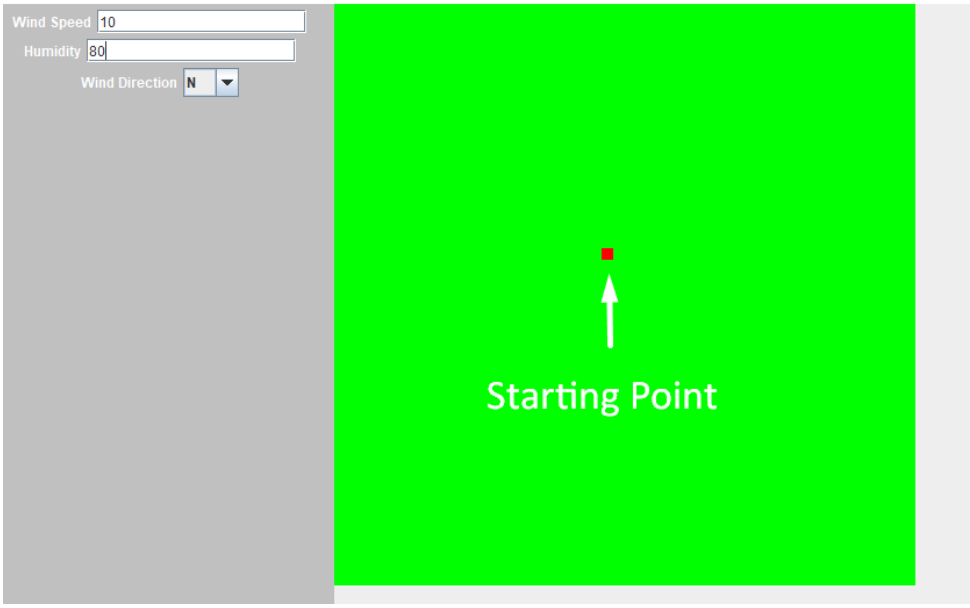

Fig.29 global view of the grid with the fire starting point

After the wind speed and Humidity are given and the wind direction is specified the simulation can be started. In This instance wind speed is 10 Km/h, the humidity is 80% for a grid of 10*10 and the wind direction is "North" we obtained the following results.

The screenshots of the figures below were taken at different intervals to show how the fire spread in the entire grid, the first was taken at t= 33s (Fig.30), 2nd was taken at t= 132s (Fig.31), 3rd at t= 198s (Fig.32) and the last one at t= 311s (Fig.33).

The time that the whole grid took to burn is not that long due to the fact that the simulation was not of a large scale due to the lack of power in the machine used for this simulation, a strong computational power is needed to be able to simulate a large area submitted to a fire.

Other factors for the fast times is that this model focuses mainly only on wind speed and humidity whereas in real life there are numerous factors that influence its propagation, but the goal of this work is more the creation of a tool that knows how a fire should behave as for the formulas or factors that are linked to its behaviour are left more to the researcher's choice.

The model was designed in a way that makes it completely possible to change the fuzzy controller or add more factors, the FireSim has the potential of becoming a strong tool in simulating fire spread with reliable results.

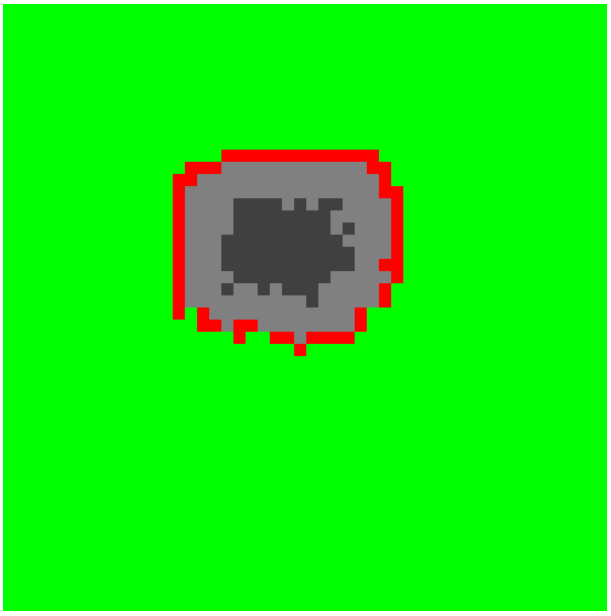

Fig.30 grid state at t = 33s

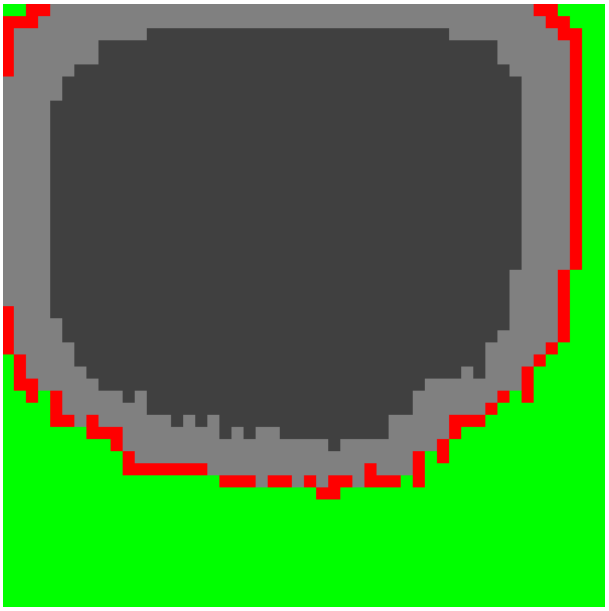

Fig.31 grid state at t = 132s

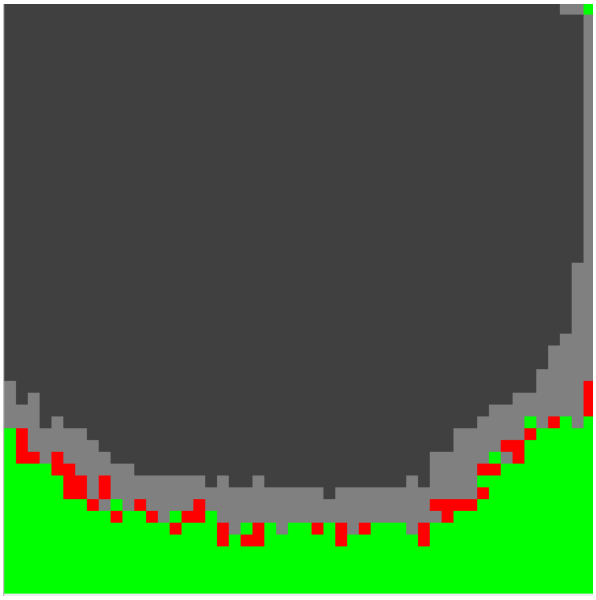

Fig.32 grid state at t = 198s

Fig.33 grid state at t = 311s

IV.4 Conclusion

The obtained results of the FireSim while simulating the time and behaviour a wildfire can have shown that using the DEVS formalism allows researchers to have a solid base to creating a software that knows exactly how a fire will behave in different circumstances. Coupled with the fuzzy controller accurate life times for the states can be determined furthering and widening the possibilities.

Further improvements can be added to this work by using the geographic information that provide more detailed data on different lands such as slopes, types of vegetation and its distribution…etc. to be used in fuzzy controllers more adapted to have the possibility to simulate a fire in its entirety accounted all factors.

V. BIBLIOGRAPHY

- B.P. Zeigler, HESSAM S. Sarjoughian. Introduction to DEVS Modeling and Simulation with JAVA: Developing Component-Based Simulation Models. Arizona USA: 2005, 147p.
- B.P. Zeigler. Theory of Modelling and Simulation. USA: John Wiley & Sons Inc, 1976, 435p.
- B.P. Zeigler, HERBERT Praehofer, TAG Gon Kim. Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems 2nd edition.USA: Academic Press, 2000, 544p.
- C.G. Cassandras, S.Lafortune. Introduction to discrete event systems. New York USA: Springer, 2007, 781p.
- Cambridge Advanced Learner's Dictionary (Third ed.). Cambridge University Press. 2008. ISBN 978-0-521-85804-5. Archived from the original on 13 August 2009.
- D.O.D. Department Of Defense. Systems Engineering Fundamentals. Virginia Usa: Defense Acquisition University Press, 2001, 222p.
- Dimitrov, V.Korolkich. Studies in fuzziness and soft computing: Fuzzy Logic: A Framework for the New Millennium. Physica, 2002, 395p.
- DRISSI M. Un modèle de propagation de feux de végétation à grande échelle. Thermique [physics.class-ph]. Université de Provence - Aix-Marseille I, 2013. Français.
- F. MARTIN McNeill, E.Thro. Fuzzy logic a practical approach. Morgan Kaufmann Pub, 1994, 309p.
- G.A. Wainer. Discrete-event modeling and simulation: a practitioner's approach. New York USA: CRC Press, 2009, 503p.
- GEORGE J. Klir, BO Yuan. FUZZY SETS AND FUZZY LOGIC Theory and Applications. New Jersey USA: Prentice Hall Press, 1995, 574p.
- GUANRONG C., TRUNG T.P. Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems. USA: CRC Press 2000, 328p.
- JAPPIOT, M., BLANCHI, R et ALEXANDRIAN, D. 2002. Cartographie du risque : recherche méthodologique pour la mise en adéquation des besoins, des données et des méthodes. CEMAGREF. ENSMP-ARMINES. Agence MTDA., Colloque de

REFERENCES BIBLIOGRAPHIQUES restitution des travaux de recherche du SIG Incendies de forêt. 4 Décembre 2002. Marseille (France).14p.

- JetBrains. IntelliJ IDEA **[online]**. Available at: < https://www.jetbrains.com/fr-fr/idea/ > (viewed the: 17/12/2020).
- JONATHAN M. 1998. Modélisation et simulation numériques de la propagation de feux de forêts, Institut National Polytechnique de lorraine.
- M. AMMARI. Etude de la dimension fractale du front dans un système désordonné binaire : Application aux feux de forêt, Mémoire Magister, Université Mohamed Mokhtar Oran. 2011, 90p.
- M. Azeem. Fuzzy Inference System - Theory and Applications. [engineering]. Intech,2012, 518p.
- M. Kutz. Handbook of Environmental Degradation of Materials. USA: William Andrew Publishing, 2012, 936p.
- M.Cirstea, A.Dinu, M.McCormick, J.G.Khor. Neural and Fuzzy Logic Control of Drives and Power Systems. Oxford UK: Newnes, 2002, 408p.
- MEDDOUR S O, DERRIDJ A, 2012. Bilan des feux de forets en Algérie: analyse spatio-temporelle et cartographie du risque (periode 1985-2010). Sécheresse 23: 133-41. doi: 10.1684/sec.2012.0342.
- Oracle. The Java Language Environment **[online]**. Available at: www.oracle.com/java/technologies/introduction-to-java.html (08/12/2020).
- P. CINGOLANI, A.F JESÚS. jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming. (2013).
- P. CINGOLANI, A.F. JESÚS. jFuzzyLogic: A Robust and Flexible Fuzzy-Logic Inference System Language Implementation. (2012).
- P.A. Bisgambiglia. Approche de modélisation approximative pour des systèmes à événements discrets : Application à l'étude de propagation de feux de forêt. Modeling and Simulation. Université Pascal Paoli, 2008. French.
- R. Hampel, MICHAEL Wagenknecht, N asredin Chaker. Advances in Soft Computing Fuzzy Control Theory and Practice. Physica, 2000, 400p.
- R. Jager. Fuzzy Logic in Control. Europe: 1995, 322p.
- R.R. Yager, L.A. Zadeh. An Introduction to Fuzzy Logic Applications in Intelligent Systems. USA: Springer, 1992, 363p.

- S. BENKRAOUDA. Détection des feux de forêts à partir d'images satellitaires infrarouges thermiques IRT en utilisant l'image de l'inverse de la probabilité d'appartenance. Doc LMD. Spécialité : Génie Electrique, 2015.
- U.S. Forest Service. Equipment and Tools **[online]**. Available at: https://www.fs.usda.gov/science-technology/fire/equipment-tools (viewed the: 16/10/2020)
- University of Arizona. General Purpose Tools **[online]**. Available at: https://acims.asu.edu/software/ (viewed the: 16/01/2020).
- Visual Paradigm. What is Unified Modeling Language (UML)? **[online]**. Available at: < https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/> (viewed the: 17/12/2020).
- Y. Baudoin, MAKI K. Habib. Using Robots in Hazardous Environments Landmine Detection, De-Mining and Other Applications. Cambridge UK: Woodhead Publishing, 2016, 692p.