Economy Transdisciplinarity Cognition www.ugb.ro/etc	Vol. 23, Issue 2/2020	100 - 108

Applications of DEVS Models and Finites Automaton in Economics

Neculai CRÎȘMARU
George Bacovia University, Bacau, ROMANIA
crismaru_nicolae@yahoo.com

Abstract: In this work we give an application of DEVS models, finites automaton and formal languages on economics.
Keywords: DEVS models, formal languages, finite automaton, deterministic finite automaton

Introduction

This chapter introduces the concept of a finite automaton, which is perhaps the simplest form of abstract computing device. Although finite automata theory is concerned with relatively simple machines, it is an important foundation of a large number of concrete and abstract applications. The finite-state control of a finite automaton is also at the heart of more complex computing devices such as finite-state transducers, pushdown automata, and Turing machines. [1]

Applications for finite automata can be found in the algorithms used for string matching in text editors and spelling checkers and in the lexical analysers used by assemblers and compilers. In fact, the best known string matching algorithms are based on finite automata. Although finite automata are generally thought of as abstract computing devices, other non-computer applications are possible. These applications include traffic signals and vending machines or any device in which there are a finite set of inputs and a finite set of things that must be "remembered" by the device. [1]

Briefly, a deterministic finite automaton, also called a recognizer or acceptor, is a mathematical model of a finite-state computing device that recognizes a set of words over some alphabet; this set of words is called the language accepted by the automaton. For each word over the alphabet of the automaton, there is a unique path through the automaton; if the path ends in what is called a final or accepting state, then the word traversing this path is in the language accepted by the automaton.

Finite automata represent one attempt at employing a finite description to rigorously define a (possibly) infinite set of words (that is, a language). Given such a description, the criterion for membership in the language is straightforward and well-defined; there are simple algorithms for ascertaining whether a given word belongs to the set. In this respect, such devices model one of the behaviours we require of a compiler: recognizing syntactically correct programs. Actually, finite automata have inherent limitations that make them un suitable for modelling the compilers of modern programming languages, but they serve as an instructive first approximation. Compilers must also be capable of producing object code from source code. [1]

1. Alphabets and Words

The devices we will consider are meant to react to and manipulate symbols. Different applications may employ different character sets and we will therefore take care to explicitly mention the alphabet under consideration.

Definition 1.1 Σ is an alphabet if Σ is a finite nonempty set of symbols.

An element of an alphabet is often called a letter, although there is no reason to restrict symbols in an alphabet to consist solely of single characters. Some familiar examples of alphabets are the 26-letter English alphabet and the ASCII character set, which represents a standard set of computer codes. In this text we will usually make use of shorter, simpler alphabets, like those given in Example 1.1. [1]

Example 1.1
i. $\Sigma=\{0, 1\}$
ii. $\Sigma = \{a,b,c\}$
iii. $\Sigma = \{\langle 0, 0\rangle,\langle 0, 1\rangle,\langle 1, 0\rangle,\langle 1, 1\rangle\}$.

It is important to emphasize that the elements (letters) of an alphabet are not restricted to single characters. In example (iii.) above, the alphabet is composed of the ordered pairs in $\{0,1\}\times\{0,1\}$. Based on the definition of an alphabet, we can define composite entities called words or strings, which are finite sequences of symbols from the alphabet. [1]

Definition 1.2 For a given alphabet Σ and a natural number n, a sequence of symbols a $a_1a_2 \dots a_n$ is a word (or string) over the alphabet Σ of length n if for each i=1,2,...,n,a, $a_i \in\Sigma$.

The order in which the symbols of the word occur will be deemed significant, and therefore a word of length 3 can be identified with an ordered triple belonging to $\Sigma\times\Sigma\times\Sigma$. Indeed, one may view the three-letter word *bca* as convenient shorthand for the ordered triple $\langle$b, c, a$\rangle$. A word over an alphabet is thus an ordered string of symbols, where each symbol in the string is an element of the given alphabet. An obvious example of words is what you are reading right now, which are words (or strings) over the standard English alphabet. In some contexts, these strings of symbols are occasionally called sentences. [1]

Example 1.2
Let $\Sigma=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$; some examples of words over this alphabet are
i. 42
ii. 242342

Even though only three different members of Σ occur in the second example, the length of 242342 is 6, as each symbol is counted each time it occurs. To easily and succinctly express these concepts, the absolute value notation will be employed to denote the length of a string. Thus, $|42| = 2$, $|242342| = 6$ and $|a1a2a3a4| = 4$. [1]

Definition 1.3 For a given alphabet Σ and a word $x = a_1a_2 \dots a_n$ over Σ, $|x|$ denotes the length of x. That is, $|a_1a_2 \dots a_n| = n$.

It is possible to join together two strings to form a composite word; this process is called concatenation. The concatenation of two strings of symbols produces one longer string of symbols, which is made up of the characters in the first string, followed immediately by the symbols of the second string. [1]

Definition 1.4 Given an alphabet Σ, let $x = a_1, \dots a_n$ and $y = b_1, \dots b_m$, be strings where each $a_i \in\Sigma$ and each $b_j \in\Sigma$. The concatenation of the strings x and y, denoted by $x{\cdot}y$, is the juxtaposition of x and y; that is, $x{\cdot}y = a_1 \dots a_n b_1 \dots b_m$. [1]
Note in Definition 1.4 *that* $|x{\cdot}y| = n+m = |x|+|y|$.

Some examples of string concatenation are:
i. aaa·bbb=aaabbb
ii. home·run=homerun
iii. $a^2{\cdot}b^3$=aabbb

Example (iii) illustrates a shorthand for denoting strings. Placing a superscript after a symbol means that this entity is a string made by concatenating it to itself the specified number of times. [1]

In a similar fashion, $(ac)^3$ is meant to express *acacac*. Note that an equal sign was used in the above examples. Formally, two strings are equal if they have the same number of symbols and these symbols match, character for character. [1]

Definition 1.5 Given an alphabet Σ, let $x=a_1\dots a_n$ and $y= b_1\dots b_m$ be strings over Σ. *x* and *y* are equal if $n = m$ and for each $i=1, 2,\dots,n$, $a_i = b_i$.

The operation of concatenation has certain algebraic properties: it is associative, and it is not commutative. That is,
i. $(\forall x\in\Sigma^*)(\forall y\in\Sigma^*)(\forall z\in\Sigma^*)$, $x\cdot(y\cdot z)=(x\cdot y)\cdot z$.
ii. For most strings *x* and *y*, $x\cdot y\neq y\cdot x$.

When the operation of concatenation is clear from the context, we will adopt the convention of omitting the symbol for the operator (as is done in arithmetic with the multiplication operator). The s*xyz* refers to $x\cdot y\cdot z$. [1]

Definition 1.6 Given an alphabet Σ, and some *b*, $b\in\Sigma$, the length of a word w with respect to *b*, denoted $|w|_b$, is the number of occurrences of the letter b within that word.

Example 1.3
$|abb|_b = 2$
$|abb|_c = 0$
$|1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,8\,8\,8\,1\,8\,8\,8\,8\,8\,8|_1 = 5$

Definition 1.7 Given an alphabet Σ, the empty word, denoted by λ, is defined to be the (unique) word consisting of zero letters.

The empty word is often denoted by λ in many formal language texts. The empty string serves as the identity element for concatenation. That is, for all strings *x*,
$x\cdot\lambda=\lambda\cdot x=x$

Even though the empty word is represented by a single character, λ is a string but is not a member of any alphabet: $\lambda\notin\Sigma$. (Symbols in an alphabet all have length one; λ has length zero.)

A particular string *x* can be divided into substrings in several ways. If we choose to break *x* up into three substrings *u,v,* and *w*, there are many ways to accomplish this. For example, if *x=abccdbc*, it could be written *as* $ab\cdot ccd\cdot bc$; that is, *x=uvw*, where *u=ab*, *v=ccd*, and *w=bc*. This *x* could also be written as $abc\cdot\lambda\cdot cdbc$, where *u=abc*, $v=\lambda$, and *w=cdbc*. In this second case, $|x|=7=3+0+4=|u|+|v|+|w|$.

A fundamental structure in formal languages involves sets of words. A simple example of such a set is Σ^k, the collection of all words of exactly length *k* (for some $k\in N$) that can be constructed from the letters of Σ. [1]

Definition 1.8 Given an alphabet Σ and a nonnegative integer $k\in N$, we define:
$\Sigma^k=\{x|\ x \text{ is a word over } \Sigma, \text{ and } |x| =k\}$

Example 1.4
If $\Sigma=\{0, 1\}$, then $\Sigma^0=\{\lambda\}$, $\Sigma^1=\{0, 1\}$, $\Sigma^2=\{00, 01, 10, 11\}$,
$\Sigma^3=\{000, 001, 010, 011, 100, 101, 110, 111\}$.

λ is the only element of Σ^0, the set of all words containing zero letters from Σ. There is no difficulty in letting λ be an element (and the only element) of Σ^0, since each Σ k is not necessarily an alphabet, but is instead a set of words; λ, according to the definition, is indeed a word consisting of zero letters.

Definition 1.9 Given an alphabet Σ, define $\Sigma^* = \sum_{k=0}^{+\infty} \Sigma^k = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$, and

$$\Sigma^+ = \sum_{k=1}^{+\infty} \Sigma^k = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$

Σ^* is the set of all words that may be constructed from the letters of an alphabet Σ. Σ^+ is the set of all nonempty words that may be constructed from Σ. [1]

2. Definition of a Finite Automaton

A deterministic finite automaton is a mathematical model of a machine that accepts a particular set of words over some alphabet Σ. [1]

A useful visualization of this concept might be referred to as the black box model. This conceptualization is built around a black box that houses the finite-state control. This control reacts to the information provided by the read head, which extracts data from the input tape. [1]

There is no limit to the number of symbols that can be on the tape (although each individual word must be of finite length). As the input tape is read by the machine, state transitions, which alter the current state of the automaton, take place within the black box. [1]

Applications of finite automata

There are several reasons why the study of automata and complexity is an important and is part of computer science. Let us list some important aspects of the automata theory.

Automata theory plays an important role when we are making software for designing and checking the behaviour of digital circuit.

The “Lexical analyses” of the typical compiler that is the compiler component that breaks the input text into logical units, such as identifiers Keywords and Punctuation.

Software for scanning large bodies of text, such as collections of web pages to find occurrences of words, phrases or other patterns.

Automata theory is key to software for verifying systems of all types that have a finite number of distinct states, such as communication protocols or protocol for secure exchange of information.
Automata theory is most useful concept of software for natural language processing.

Direct application to formulate traffic light, making play toys, It plays a major role in making electronic machine (e.g. Vending machine, Toll Machine,…) and Android games (Pac man, Treasure Hunt, monkey and Banana, ...). [2]

Any problem with interacted participants and actions can be treated as a game. When car drivers put a plan to drive in heavy traffic, they are actually playing a driving game. When users bid on bidding-based Websites, they are actually playing an auctioning game. In election, choosing the platform is a political game.([3]) The owner of a factory deciding the price of his product is an economic game. Obviously, game theory can be presented in wide range of applications. [3]

Game theory is a mathematical tool that can analyse the interactions between individuals strategically. The interactions between agents, who may be individuals, groups, firms are interdependent. These interdependent interactions are controlled by the available strategies and their corresponding payoffs to participants. However, game theory studies the rational behaviour in situations involving interdependency. Therefore, game theory will only work when people play games rationally and it will not work on games with co-operational behaviour. [3]

Generally, the game consists of the following entities:

Players: Where one side of the game tries to maximize the gain (payoff), while the other side tries to minimize the opponent's score. However, these players can be humans, computer applications or any other entities.

Environment: This includes board position and the possible moves for the players.
Successor Function: The successor function includes actions and returns a list of (move, state) pairs, where each pair indicates a legal move and the resulting state.

Terminal Test: The terminal test specifies when the game is over and the terminal state is reached.

Utility Function: The utility function is the numeric value for the terminal states.

The attractive point in studying games is that models used in games are applicable to be used in real-life situations. Because of this, game theory has been broadly used in economics, biology, politics, low, and also in computer sciences. Examples on the use of game theory in computer science include interface design, network routing, load sharing and allocate resources in distributed systems and information and service transactions on Internet [3]. One of the crucial factors which affect the performance of players in a given game is the behaviour (strategy) representation. We found that different techniques are used to represent players' behaviour in different games. One of the successful techniques is the use of automata-based model to represent and control participated agents. In the next sections, we will discuss different types of automata and their affect on games. [1]

Formally, a deterministic finite automaton is a 5-tuple ($Q, \Sigma, \delta, q_0, F$) such that:
Q is a finite set called the **states**
Σ is a finite set called the **alphabet**
$\delta: Q \times \Sigma \rightarrow Q$ is the **transition function**
$q_0 \in Q$ is the **start state**
$F \subseteq Q$ is the **set of accept states**

We also discuss Mealy machines, which add to the expressiveness of DFAs by producing output values at each transition, where the output depends on the current state and input. Rather than accepting or rejecting strings, Mealy machines map an input string to an output string. They can be thought of as functions that receive a string and produce a string in response.

Formally, a Mealy machine is a 6-tuple ($Q, \Sigma, \Gamma, \delta, \omega, q_0$) such that:
Q, Σ, δ, and q_0 are defined as in a DFA.
Γ is a finite set called the **output alphabet**
$\omega : Q \times \Sigma \rightarrow \Gamma$ is the **output function** [4]

3. The main Result

We will continue to use a more general mathematical notion than that of the finite automatic (determinist), namely the mathematical simulation model of a system with discrete events (S), in the formalization of DEVS [5].

It is called the *DEVS (mathematical) simulation model of a system with discrete events (S)*([5]), denoted by (S_{DEVS}), (S_{DEVS})=(T,X,U,Y, γ, η), where:
T named time of (DFA), and $T \subseteq R$,
X named set of input of (DFA) and $(X_1,X_2,\ldots,X_s,\ldots,X_n) \in X \subseteq R^n$,
U named set of segment of input, and U has the next general forms $U=\{u_i([t_0,t_1]) \mid i \in I \neq \Phi$,
$u_i : [t_0,t_1]) \rightarrow R\}$, the element of U are sub-set of R and has the form $u_i([t_0,t_1])$ for each $i \in I$, and I are know. Here $u_i([t_0,t_1])=\{ u_i (t) | t \in [t_0,t_1]\}$

V named set of internal states of (S), $V=\{v_1,v_2,\dots,v_k,\dots,v_m\}$=finits, or named V=non-finits $V\subseteq R$, and $v_k\in R^n$, $k=1,2,\dots,m$.
Y named set of outputs of (S), $Y=\{y_1,y_2,\dots,y_j,\dots,y_p\}$= finits, sau Y=non-finits ,$Y\subseteq R$,
γ named the map of output of (S_{DEVS}), $\gamma:XxV\to Y$,
$(t,u_i(t),v_i))\xrightarrow{\gamma}\gamma(t,u_i(t),v_i))=y_j\in R$, $(t, u_i(t))=X_s\in U$, so $\gamma:UxV\to Y$, where $(t,u_i(t))=X_i$, $i=1,2,\dots,n$.
η named the map of transitions of states v_i of (S_{DEVS}), $\eta:XxV\to V$ or
$\eta:UxV\to V$, $(t,u_i(t),v_k)\in UxV$, $(t,u_i(t),v_k)\xrightarrow{\eta}\eta(t,u_i(t),v_k)=v_s\in V$ si $(t,u_i(t))=X_i$.

Next we will apply this mathematical model to model the economic activity of a company (enterprise), denoted with (F). For this we will specify what all 7 elements above look like for the company case (F).

1.The time. The operating time of the company is considered in this case a discrete crowd $T\subseteq N$. If we assume that the company works endlessly then $T\subseteq N$, if the company (F) has a limited time of activity, then $T\neq N$.

Here we assume that T= N, so T ={0,1,2,...,i,i 1,...}. The initial moment when the company starts its activity (at its inception) is the 0 T moment, after which the moment 1 T follows,... etc.

2.The se of all inputs. We assume that the company (F) has at some point $i\in N$ certain inputs of: raw materials and materials, machinery and installations, specialized knowledge (now how), production orders,... etc.

These inputs constitute a Xi vector $X_i=(x_1^{(i)},x_2^{(i)},\dots,x_k^{(i)},\dots x_n^{(i)})$ called input vector for (F). So the crowd of entries is in this case $X=\bigcup_{t=0}^{+\infty} X_t$.

For example, $x_1^{(i)}$=20 tons of raw materials and production materials, so the values of $x_1^{(i)}$=real numbers, $x_2^{(i)}=\{I_1,I_2,\dots,I_r\}$ the multitude of installations and machines that enter the company for production at the next moment (i+1), with $x_2^{(i)}=\{I_1,I_2,\dots,I_r\}\subseteq I(i)$=the multitude of all installations and machines that the company can procure at the given time i,
$I(i)\subseteq I$ take the crowd of all installations to which the company can access at different times,
$I=\bigcup_{t=0}^{+\infty} I(t)$.

$x_3^{(i)}=\{C_1^{(i)},C_2^{(i)},\dots,C_t^{(i)}\}$ represents a lot of knowledge (technical documentation) that enters the company at the time (F), knowledge necessary for the production process at the time I, so $x_3^{(i)}=\{C_1^{(i)},C_2^{(i)},\dots,C_t^{(i)}\}\subseteq C(i)$ = the multitude of all the technical and scientific knowledge to which the company (F) can access at the time i, $C(i)\subseteq C$= the multitude of all the technical and scientific knowledge to which the company can access (F) at different times, $C=\bigcup_{t=0}^{+\infty} C(t)$.

$x_4^{(i)}=\{P_1^{(i)},P_2^{(i)},\dots,P_d^{(i)}\}$ is the multitude of production orders to be manufactured at the time i by the company (F), a crowd that has a number of $d\in N$ orders, $x_4^{(i)}=\{P_1^{(i)},P_2^{(i)},\dots,P_d^{(i)}\}\subseteq P(i\ )$ = the multitude of orders that the company (F) can receive at the given time i, $P(i)\subseteq P$ = the multitude of orders that the company can receive (F) at different times, $P=\bigcup_{t=0}^{+\infty} P(t)$.

The number d of the orders at the time i may depend on the time i and in this case will be noted so $d_i \in N$. So for this example the input vector at the time i is: $X_i=(x_1^{(i)},x_2^{(i)},x_3^{(i)},x_4^{(i)})$ and the set of entrances is $X=\bigcup_{t=0}^{+\infty} \{X_t\}=\{X_t \mid t\in N\}$.

3. The set of inputs segments, U. We assume it is in this case $U=\{u_1,u_2,u_3,u_4\}$ where $(u_1,u_2,u_3,u_4):T\rightarrow \underline{R}\mathrm{x}\underline{P(I)}\mathrm{x}\underline{P(C)}\mathrm{x}\underline{P(P)}$, cu $t\in T$. Here P(A) = the set of all parts of the A, so P(I(t)) = the set of all parts of the set I(t) at the time $t\in T$.

So for the t moment = i we have:
$(u_1,u_2,u_3,u_4)(t) = (u_1,u_2,u_3,u_4)(i) = (u_1(i),u_2(i),u_3(i),u_4(i)) = (20 \text{ tons},\{I_1^{(i)},I_2^{(i)},\ldots,I_r^{(i)}\},\{C_1^{(i)},C_2^{(i)},\ldots,C_t^{(i)}\}, \{P_1^{(i)},P_2^{(i)},\ldots,P_d^{(i)}\})$ cu $u_1(i) = x_1^{(i)} = 20$ tons, $u_2(i) = x_2^{(i)} = \{I_1,I_2,\ldots,I_r\}$, $u_3(i) = \{C_1^{(i)},C_2^{(i)},\ldots,C_t^{(i)}\}$, $u_4(i) = \{P_1^{(i)},P_2^{(i)},\ldots,P_d^{(i)}\}$. All these sets depend on the $t\in T$ moment, so there are functions defined on T and with multiple values.

So <u>for u_1</u> we have, $u_1:T\rightarrow R$ with $u_1(t)= x_1^{(t)}$ and for t=i we have $u_1(i)= x_1^{(i)}=20\in R$, so $u_1(t)$ it's not set, it's real number.

<u>For u_2</u> we have, $u_2:T\rightarrow P(I(t))$= set of all sub-set of $I(t)\subseteq I$, $u_2(t)= x_2^{(t)}=\{I_1^{(t)},I_2^{(t)},\ldots,I_r^{(t)}\}\subseteq I(t)\subseteq I$ and for t = i we have
$u_2(i) = x_2^{(i)}=\{I_1^{(i)},I_2^{(i)},\ldots,I_r^{(i)}\}\subseteq I(i)\subseteq I$.

<u>For u_3</u> we have, $u_3:T\rightarrow P(C(t))$= set of all sub-set of $C(t)\subseteq C$, $u_3(t)= x_3^{(t)}=\{C_1^{(t)},C_2^{(t)},\ldots,C_t^{(t)}\}\subseteq C(t)\subseteq$
$\subseteq C$ and for t=i we have $u_3(i)= x_3^{(i)} =\{C_1^{(i)},C_2^{(i)},\ldots,C_t^{(i)}\}\subseteq C(i)\subseteq C$.

<u>For u_4</u> we have, $u_4:T\rightarrow P(P(t))$= set of all sub-set of P(t),
$u_4(t)= x_4^{(t)}=\{P_1^{(t)},P_2^{(t)},\ldots,P_t^{(t)}\}\subseteq P(t) \subseteq P$ and for t=i we have $u_4(i)= x_4^{(i)} =\{P_1^{(i)},P_2^{(i)},\ldots,P_t^{(i)}\}\subseteq P(i)\subseteq P$.

So input segments are the $t\in T$ distributions of X_t inputs, i.e. they are functions that provide input values at every moment $t\in T$. Instead the sets of entrances is the X set of all the entrances at every moment $t\in T$, so $X=\bigcup_{t=0}^{+\infty} \{X_t\}=\{X_t \mid t\in N\}$. They are different mathematical notions, $X \neq U$, the set of inputs segments offer more information than the set of X, X= meeting all input segments at all times $t\in T$. U represents distributions during $t\in T$, X_t inputs, distributions given by functions u_1,u_2,u_3,u_4 .

4. <u>Multiple V states of the model</u>. In general, in the case of modelling firms with these mathematical tools, the model's moods represent the facilities and condition of the company at that time $t\in T$.

We assume that the company's facilities are:
$V_1^{(t)}$= stocks of raw materials (total) at the time of $t\in T$, necessary for production at the time of t. The values of the $V_i^{(t)}$ variables represent real positive numbers, $V_1^{(t)} \in R$, $t\in T$ and if $V_1^{(t)}>0$, then there are raw materials in the warehouse (sufficient or not-in this case it is necessary to replenish). If $V_1^{(t)}=0$ then there are no raw materials and need to be replenished.

If $V_1^{(t)}<0$ then $V_1^{(t)}$ represents the amount of raw materials required for production at this time $t\in T$.

$V_2^{(t)}$= energy resources from the time $t\in T$, (coal stock, technological steam, electricity,... etc.) existing in the company.

$V_3^{(t)}$= human resources stock (number of workers-on specialties, no technicians, no specialist engineers,... etc.) existing in the company at the time $t\in T$.

$V_4^{(t)}$= the stock of knowledge (now how) existing in the company at the time $t \in T$. Thus the state at time $t \in T$ is represented by vector $V^{(t)}=(V_1^{(t)}, V_2^{(t)}, V_3^{(t)}, V_4^{(t)})=S_k \in R^4$,S_k is the vector with the number k from V and is a S_k point from R^4, so the crowd of the company's model states is $V=\bigcup_{t=0}^{+\infty} \{V^{(t)}\}=$ $\{V^{(t)}=(V_1^{(t)}, V_2^{(t)}, V_3^{(t)}, V_4^{(t)}) \mid t \in T\}=\{S_0, S_1, ... ,S_k, ..., S_q\}$.

Here S_k does not represent the state at the time $k \in T$, but represents the vector with the number k in $V=\{S_0, S_1, ... ,S_k, ..., S_q\} \subset R^4$.

On the figure no. 1 we give a graphical interpretation of the states at 2 consecutive moments of time, t and t +1, V_t state and V_{t+1} state that takes values $a_k^{(t)}$, k=1,2,3,4, and $a_k^{(t+1)}$ k=1,2,3,4.

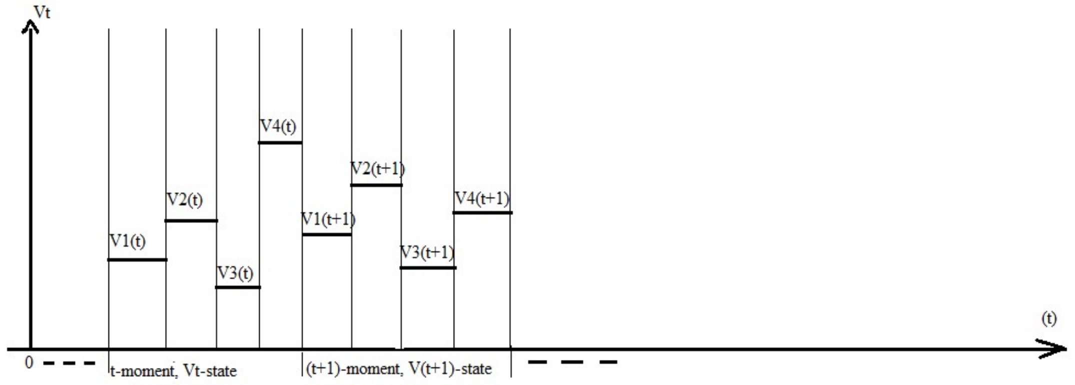

Figure no. 1 Legend: V_t are states at t moment, $V_i^{(t)}$ are the i-state at the t moment

5.<u>The set of outputs</u>, denoted with $Y=\{y_1,y_2,...,y_t ,...,y_m\} \subset R_+$, represents the results of the production activity at the time of $t \in T$. These results are expressed in monetary units and represent the (positive) value received from the sale of products obtained in the production process from time t, value denoted by y_t.

6. <u>Model response function (or model output function)</u>, γ:XxV$\rightarrow$Y, sau γ:UxV$\rightarrow$Y,

$(X_i ,V_i) \xrightarrow{\gamma} \gamma(X_i ,V_i) = y_i$ the output (system response) from time i = the monetary value obtained after the delivery of the products manufactured by the company at time i. If we consider γ:UxV$\rightarrow$Y, then we have the following diagram

$(X_t,S_k)= (X_t,V_t)= ((u_1(t), u_2(t), u_3(t), u_4(t)), (V_1^{(t)}, V_2^{(t)}, V_3^{(t)}, V_4^{(t)}))\xrightarrow{\gamma}$

$\xrightarrow{\gamma} \gamma((u_1(t), u_2(t), u_3(t), u_4(t)), (V_1^{(t)}, V_2^{(t)}, V_3^{(t)}, V_4^{(t)}))=\gamma(X_t,V_t)= \gamma(X_t,S_k)=y_t \in Y$, $t \in T$.

7. The <u>transition function</u> of the Sk states of the system (of the system model) η:XxV$\rightarrow$V, sau η:UxV$\rightarrow$V, $(X_i ,V_i) \xrightarrow{\eta} \eta(X_i ,V_i)=V_{i+1}$=state at the moment (i+1). If $V_i=S_p$ and $V_{i+1}=S_q$, then $S_q=\eta(X_i,S_p)$, noted by $(X_i,V_i)=(X_i,S_p)\xrightarrow{\eta} S_q=V_{i+1}$.

With this type DEVS model you can model the evolution of the company over a given period of time and you can also simulate its behaviour to various "environmental disturbances" that may occur during this time.

References
[1] Marinescu D., *Limbaje formale si teoria automatelor*,
https://horatiuvlad.com/unitbv/limbaje_formale/Limbaje%20Formale_Marinescu.pdf
[2] Anitha E., *Applications of Automata in Electronic Machines and Android Games (Finite Automata)*, International Journal of Computing Algorithm Volume: 03, Issue: 01 June 2014, p. 40-43, ISSN: 2278-2397,
[3] Arshad M. R. M., Khaled S., *The Applications of Automata in Game Theory,* https://www.researchgate.net/publication/252627321_The_Applications_of_Automata_in_Game_Theory
[4] Gribkoff E., *Applications of Deterministic Finite Automata*, 2013,
https://web.cs.ucdavis.edu/~rogaway/classes/120/spring13/eric-dfa.pdf
[5] Solcany V., *Simulation Algorithms for DEVS Models*, September 2008,
http://www2.fiit.stuba.sk/~solcany/files/devs_article.pdf .