A LINGUISTIC MODEL OF AGGREGATION IN VIRTUAL MANUFACTURING PROCESSES

Vasile CRĂCIUNEAN[1]

ABSTRACT: The current objectives to increase the standards of quality and efficiency in manufacturing processes can be achieved only through the best combination of inputs, independent of spatial distance between them. Electronic communication solves the problem of the distance between resources. As a natural consequence, achieving these objectives is solved by virtual production processes. Due to the complexity of these production processes they cannot be mastered without the help of accurate and complex models that reflect real-time evolution and breakdown of aggregate physical and logical actions focused on goals. This paper proposes a linguistic model for building virtual production systems by assembling intelligent elementary actions with a high degree of independence. Thus from a lot of independent actions which we call actions of aggregation, we define an aggregation system of actions capable of generating aggregate production processes. In our model the aggregated production process is a word in a language over the alphabet consisting of actions of aggregation.

KEY WORDS: action of aggregation, aggregation system of actions, aggregated production process, application of aggregation, application of disaggregation.

1 INTRODUCTION

Through the production process we understand the transformation of a given set of resources (raw materials, semi-finished goods, energy, labor, equipment) into finished products by aggregating specific actions according to their manufacturing recipes.

The technical and technological progress has contributed to increased communication capacity and optimal use of the resources and activities regardless of their distribution in space leading to the possibility of managing them virtual. It is clear that technology is essential to manage virtual actions but on the other hand the real need of collaboration in virtual organizations creates the impetus for the creation of appropriate technologies. Software and hardware technologies open opportunities for developing new better ways of coordination and synchronization of the actions involved in production processes.

It is a fact that virtual organizations based on the architecture of a distributed manufacturing system, well organized, can bring a significant increase in efficiency and product quality.

This paper proposes a model for building virtual production systems by assembling actions with a high degree of independence.

[1] "Lucian Blaga" University of Sibiu, Engineering Faculty, Romania

E-mail: craciunean@sln.ro

Construction of an aggregate production process from actions involves two distinct activities: create new actions and creating production processes by aggregating existing actions. Therefore when we want to develop a production process will have to first try to build on existing actions and just after that to build new actions needed for our production process requirement. This actions will enlarge our base of actions.

Our base of actions, on which we build production process, together with facilities for action construction and management is the actions infrastructure.

The actions infrastructure consists of three distinct models: a standard action model, an actions connection model, and an action deployment model.

A standard action model defines what a valid action is, and how to create a new action in the actions infrastructure. All reusable actions will be built, thus, according to the model of the standard action. Each actions infrastructure has a library of reusable actions conforming to the standard action model.

The actions connection model defines a collection of connectors and support facilities for actions aggregation. Therefore, the connection model determines how to build a production process or a larger action from existing actions.

The action deployment model describes how the actions will be implemented in a work environment.

Based on this reality and the belief that this is the evolution of production processes, this paper proposes a linguistic model for building complex production processes starting from actions. Simplifying things, in this paper a production process is a word in a language. All the actions of a production process are designed to operate in a parallel and distributed environment.

In 3 we introduce the notion of an aggregation system of actions (S) and the language L(S,f) associated with it. The aggregation system of actions is a general framework that defines a set of production processes. In 4. we introduce the notion of aggregated production process and functional aggregated production process.

We believe that the present paper provides the necessary mechanisms to build static or dynamic optimal production processes by assembling independent actions.

2 PRELIMINARY NOTIONS AND NOTATIONS

Let X be a set. The family of subsets of X is denoted by $\mathcal{P}(X)$. The cardinality of X is denoted by |X|. The set of natural numbers, {0, 1, 2, ...} is denoted by N. The empty set is denoted by Ø. An *alphabet* is a finite nonempty set of abstract symbols. For an alphabet V we denote by V^* the set of all strings of symbols in V. The empty string is denoted by λ. The set of nonempty strings over V, that is $V^*-\{\lambda\}$, is denoted by V^+. Let Sub(w) denote the set of subwords of w. Each subset of V^* is called a language over V. The *length* of a string $x \in V^*$ (the number of symbol occurrences in X) is denoted by |x|. The number of occurrences of a given symbol $a \in V$ in $x \in V^*$ is denoted by $|x|_a$. Let Symb(w) denote the set of symbol occurrences in w.

3 AGGREGATION SYSTEM OF ACTIONS

In this paper we consider that the atomic production unit of the process is the action.

The actions and the resources on which the actions are produced, are distributed in a virtual network of companies working together. Actions are very inhomogeneous beginning with actions of design, procurement, logistics, manufacturing, quality control etc. There will be software actions that will store and dynamically process all data on the evolution of the production process. Therefor we begin by defining the notion of action of aggregation on witch our model is based.

Definition 3.1. An action of aggregation *a* is a construct of the form

a = (I,O,M)

where

$I=\{(r_i,c_i,e_i) \mid i=1,2,...m$; r_i is a resource, c_i is the quantity of that resource, e_i is an event$\}$;

The resource r_i is called an input resource and represents the necessary resource in the c_i quantity for the action a, and e_i is called an input event.

$O=\{(r_o,c_o,e_o) \mid o=1,2,...n$; r_o is a resource or a final product, c_o is the quantity of the resource, e_o is an event$\}$;

If r_o is the resource then it's called the output resource and c_o is the quantity of that resource, else r_o is the final output product. The event e_o is the output event.

M is a set of data and associated metadata such as duration of action, the minimum allowed stock, costs, technical data about the action, software components, etc.

Although virtual production processes are words in a language that are strings of actions, the execution order is not strictly that from the word of aggregate production process.

Each component embeds in it a certain degree of intelligence. It will know how to check if it has provided the necessary context for starting, for example it will know to check if it has the necessary resources for the smooth conduct. If this conditions are not satisfied it will wait until these conditions are met. The wait will last until it gets a message about updating the input resources and then it will restart the process with context verification. The action will know how to decrement stock, for example, with consumed resources and increment the stock with the resource created by it and also sending the corresponding event. An action of aggregation also does a very important thing that is real-time update of developments related to various data such as start time, the actual length, waiting time, costs, etc.

We consider the set A of actions of aggregation involved in a production process, then we define the precedence actions by two functions, namely: a bottom-up function of precedence $f: A \to \mathcal{P}(A)$, defined as follows:

$f(a) = \{b \in A \mid O_a \in I_b\}$, $(\forall)\ a \in A$

and a top-down function of precedence $g: A \to \mathcal{P}(A)$, defined as follows:

$g(b) = \{a \in A \mid O_a \in I_b\}$, $(\forall)\ b \in A$.

In the following we define some useful languages over the set A. The symbols from the A set, represent real elementary actions and therefore the semantic load of the elements of the vocabulary is well defined.

Definition 3.2. An aggregation system of actions is a construct of the form

$$S = (A, f, g, \{L_X \mid X \in \mathcal{P}(A)\})$$

where

A is a finite set of actions called action of aggregation;

f is a function f: A → $\mathcal{P}$(A) named bottom-up function of precedence;

g is a function g: A → $\mathcal{P}$(A) named top-down function of precedence;

L_X is a language over A, (∀) X∈$\mathcal{P}$(A). It is obvious that L_X can be an empty language for some X∈$\mathcal{P}$(A).

Based on the function f we define the application of aggregation φ as follows:

Definition 3.3. An application of aggregation is an application φ: $\mathcal{P}$(A) → $\mathcal{P}$(A) defined as follows:

$$\varphi(X) = \bigcup_{x \in X} f(x) \quad (\forall) X \in \mathcal{P}(A).$$

In this conditions $\varphi^n(X)$ can be defined as:

$\varphi^0(X) = X$ (∀) X∈$\mathcal{P}$(A);

$\varphi^n(X) = \varphi(\varphi^{n-1}(X))$.

We observe that $\varphi^n(X) \in \mathcal{P}(A)$ (∀) n∈N and (∀) X∈$\mathcal{P}$(A), and therefore over the set $\varphi^n(X)$ we have the language $L_{\varphi^n(X)}$ as defined in *Definition 3.2*.

The set $\{\varphi^n(X);\ n \geq 0,\ X \in \mathcal{P}(A)\}$ is finite, because A is a finite set.

Similarly, we define the application of disaggregation ψ:

Definition 3.4. An application of disaggregation is an application ψ: $\mathcal{P}$(A) → $\mathcal{P}$(A) defined as follows:

$$\psi(X) = \bigcup_{x \in X} g(x) \quad (\forall) X \in \mathcal{P}(A).$$

Similarly, $\psi^n(X)$ can be defined as:

$\psi^0(X) = X$ (∀) X∈$\mathcal{P}$(A);

$\psi^n(X) = \psi(\psi^{n-1}(X))$.

We observe that $\psi^n(X) \in \mathcal{P}(A)$ (∀) n∈N and (∀) X∈$\mathcal{P}$(A), and therefore over the set $\psi^n(X)$ we have the language $L_{\psi^n(X)}$ as defined in *Definition 3.2*.

The set $\{\psi^n(X);\ n \geq 0,\ X \in \mathcal{P}(A)\}$ is finite, because A is a finite set.

In the following we define the language specified by the aggregation system S. We must point out that one can define two important languages specified by the aggregation system S, namely one starting from the bottom-up precedence function f witch we denote by L(S,f) and the other starting from the top-down precedence function g witch we denote by L(S,g). In this paper we will deal only with the L(S,f) language, which is why we use the aggregation application φ:$\mathcal{P}$(A)→$\mathcal{P}$(A) as defined above, by the bottom-up precedence function f.

Definition 3.5. Let

$$S = (A, f, g, \{L_X \mid X \in \mathcal{P}(A)\})$$

be an aggregation system of actions. Then the language L(S,f) specified by the aggregation system of actions S is:

$$L(S,f) = \bigcup_{X \in \mathcal{P}(A)} \bigcup_{i=0}^{\infty} \prod_{k=0}^{i} L_{\varphi^k(X)}$$

Lemma 3.1. Let

$$S = (A, f, g, \{L_X \mid X \in \mathcal{P}(A)\})$$

be an aggregation system of actions and the language

$$R(X) = \bigcup_{i=0}^{\infty} \prod_{k=0}^{i} L_{\varphi^k(X)}$$

then

$$R(X) = R_1(X) \cup R_2(X)(R_3(X))^* R_4(X)$$

where

$$R_1(X) = \bigcup_{i=0}^{i_1-1} \prod_{k=0}^{i} L_{\varphi^k(X)}$$

$$R_2(X) = \prod_{k=0}^{i_1-1} L_{\varphi^k(X)}$$

$$R_3(X) = \prod_{k=i_1}^{i_2-1} L_{\varphi^k(X)}$$

$$R_4(X) = \bigcup_{i=i_1}^{i_2-2} \prod_{k=i_1}^{i} L_{\varphi^k(X)}$$

Proof. For X∈$\mathcal{P}$(A) we consider the following sequence:

X, φ(X), $φ^2$(X), …, $φ^k$(X), …

Because the set X⊂ A is finite, it's obvious that (∃) i and j such that $φ^i(X) = φ^j(X)$ and therefore $L_{\varphi^i(X)} = L_{\varphi^j(X)}$.

Let i_1 and i_2 ($i_1<i_2$) be the smallest such values for i and j. If we denote

$$R_1(X) = \bigcup_{i=0}^{i_1-1} \prod_{k=0}^{i} L_{\varphi^k(X)}$$

$$R_2(X) = \prod_{k=0}^{i_1-1} L_{\varphi^k(X)}$$

$$R_3(X) = \prod_{k=i_1}^{i_2-1} L_{\varphi^k(X)}$$

$$R_4(X) = \bigcup_{i=i_1}^{i_2-2} \prod_{k=i_1}^{i} L_{\varphi^k(X)}$$

then the Lemma 3.1 is proved.

We have a prefix from 0 to i_1 which is not repeated, followed by R_3 which can repeat from 0 to infinite and then we have a remainder (a part from a period).

Theorem 3.1. Let

S = (A, f, g, {L_X | X∈$\mathcal{P}$(A)})

be an aggregation system of actions. If all languages L_X, X∈$\mathcal{P}$(A) are of type i∈{0,1,2,3} in the Chomsky hierarchy, then the language L(S,f) is of type i in the Chomsky hierarchy.

Proof. It follows immediately from closing the languages from the Chomsky hierarchy at union, catenation, catenation closure and from Lemma 3.1.

4 AGGREGATED PRODUCTION PROCESS

We will further look at an action of aggregation as a basic production process which allows as input a set of resources in specified quantities and a specified type of event and produces as output a resource or a final product and a specific event.

If we have a string of actions of aggregation $α=a_1a_2...a_m$ where a∈Symb(α) is of the form

$$a = (I_a, O_a, M_a)$$

then we denote

$$I_\alpha = \bigcup_{a\in Symb\,(\alpha)} I_a$$

and with

$$O_\alpha = \bigcup_{a\in Symb\,(\alpha)} O_a$$

Definition 4.1. An action of aggregation block of level k initiated by X is a construct of the form:

$$Ɓ_k(X) = (I_k, O_k, L_k(X), B_k)$$

where

$$L_k(X)=L_{\varphi^k(X)}$$

$$I_k = \bigcup_{\alpha\epsilon B_k(X)} I_a$$

$$O_k = \bigcup_{\alpha\epsilon B_k(X)} O_a$$

B_k is an object, named block-board, that can store information of the form (resource, quantity, event). B_k is used by the actions of aggregation for communication.

Definition 4.2. Let

S = (A, f, g, {L_X | X∈$\mathcal{P}$(A)})

be an aggregation system of actions and X∈$\mathcal{P}$(A). Then the language

$$R(X) = \bigcup_{i=0}^{\infty} \prod_{k=0}^{i} L_{\varphi^k(X)}$$

is called an aggregated production network generated by X.

Definition 4.3. Let S be an aggregation system of actions and

$$R(X) = \bigcup_{i=0}^{\infty} \prod_{k=0}^{i} L_{\varphi^k(X)}$$

be an aggregated production network generated by X∈$\mathcal{P}$(A). Then the aggregated production network R(X) is a functional aggregated production network if for all k=0,1,2 … we have the action of aggregation block $Ɓ_k(X) = (I_k, O_k, L_k(X), B_k)$ and $I_{k+1}=O_k$.

Lemma 4.1. Let

S = (A, f, g, {L_X | X∈$\mathcal{P}$(A)}) be an aggregation system of actions and X∈$\mathcal{P}$(A). If

$$R(X) = \bigcup_{i=0}^{\infty} \prod_{k=0}^{i} L_{\varphi^k(X)}$$

is a functional aggregated production network then:

i) $\varphi(\psi(X))=X\ (\forall)X\in\mathcal{P}(A)$,

ii) $\psi(\varphi(X))=X\ (\forall)X\in\mathcal{P}(A)$,

iii) $\varphi^n(\psi^n(X))=X\ (\forall)X\in\mathcal{P}(A)$.

iv) $\psi^n(\varphi^n(X))=X\ (\forall)X\in\mathcal{P}(A)$.

Proof. Points i) and ii) are obvious from the definitions of φ and ψ. Statements iii) and iv) are proved by induction. We prove only the assertion iii) because iv) has the same demonstration.

For n=1 we have the assertion i). We assume the statement is true for n, i.e. $\varphi^n(\psi^n(X))=X$.

We replace X with ψ(X) and obtain $\varphi^n(\psi^n(\psi(X)))=\psi(X)$. We apply φ on this relationship and get $\varphi^{n+1}(\psi^{n+1}(X)) = \varphi(\psi(X)) = X$.

Thus in each block, two control loops of the aggregated production process close, one downward and one upward. Each block receives through breakdown a particular path to follow by the ascending block and reports back its progress by aggregation.

Definition 4.4. Let

S = (A, f, g, {L_X | X∈$\mathcal{P}$(A)})

be an aggregation system of actions, R(X) an aggregated production network generated by X and $\alpha=\alpha_1\ \alpha_2\ \ldots\ \alpha_n$, $\alpha_i \in L_{\varphi^i(X)}$ a finite word from R_X. Therefore, $\mathcal{A}$= (I, O, α, φ, ψ, B) is called an aggregated production process where:

$$I = I_{\alpha_1} \qquad O = \bigcup_{\alpha_i} O_{\alpha_i}$$

The φ and ψ applications are the restrictions of the φ and ψ applications from 3.3 and 3.4 definitions.

B is an object, called process-board, that can store information of the form (resource, quantity, event). B is used by the actions of aggregation for communication.

Definition 4.5. Let S be an aggregation system of actions and $\mathcal{A}$=(I,O,α,φ,ψ,B) an aggregated production process where $\alpha=\alpha_1\ \alpha_2\ \ldots\ \alpha_n$, $\alpha_i \in$ *Lφi(X)*.Then the aggregated production process $\mathcal{A}$ is a functional aggregated production process if (∀) i=1, …, n-1 we have $I_{\alpha_{i+1}} = O_{\alpha_i}$.

Definition 4.6. Let

S = (A, f, g, {L_X | X∈$\mathcal{P}$(A)})

be an aggregation system of actions. Then

$$R(X,n) = \prod_{k=0}^{n} L_{\varphi^k(X)}\ , n \in N$$

is called an aggregated production network generated by X and length n.

We remark that R(X, n) contains all aggregated production processes generated by X and length n, namely all aggregated production processes of the form:

$\mathcal{A}$= (I, O, α, φ, ψ, B)

where:

$$I = \bigcup_{a\in X} I_a \qquad O = \bigcup_{k=0}^{n} \bigcup_{a\in\varphi^k(X)} O_a$$

α∈R(X,n) , B is like in the *Definition 4.2.*

In practice we are interested in an aggregated production processes that has a lot of inputs I and plenty of outputs O. Obviously, we need to search for our production process in a subset of R(X, n), which we denote R(X, Y, n) where:

X is a minimal subset Z with the property

$$\bigcup_{a\in Z} I_a = I$$

$$n = \min\left\{ i \mid O \subset \bigcup_{k=0}^{i} \bigcup_{a\in\varphi^k(X)} O_a \right\}$$

$$Y = \{a\in\varphi^n(X) \mid O_a \cap O \neq \emptyset\}$$

$$R(X,Y,n) = \prod_{k=0}^{n} L_{(\varphi^k(X)\cap\psi^{(n-k)}(Y))}\ , n \in N$$

Definition 4.7. The language R(X, Y, n) is called a network of acceptable aggregated production process.

As can be seen from the above there are a lot of equivalent acceptable production processes in the sense that they make the same products. You should therefore choose the optimal aggregated production process, problem which will be the subjects of future papers. Also, building generative grammars for languages associated with an aggregation system of actions will be treated in future papers.

5 CONCLUSION

The purpose of this model is to generate aggregated production processes by combining independent actions.

A process as defined, will run in a parallel and distributed environment. Starting the process implies parallel activation of all actions of aggregation. Each action will get its input data from the block-board and write all its output data also on the block-board. Of course, these operations will synchronize dynamically.

We note that we can get more functional aggregated production processes that do the same from the perspective of the process, namely aggregated processes that have the same inputs and outputs.

It is natural to ask the question of choosing the best solution for our process, optimal in terms of execution time and cost. The problem of finding the optimal aggregate can be put in two ways: static determination of an optimum aggregated process or building an aggregated process that evolves over time by changing his actions according to their efficiency.

6 REFERENCES

►Mowshowitz A., *Virtual Organization: A vision of Management in the Information Age*, The Information, 1994

►Matthias Dehmer, Abbe Mowshowitz, and Frank Emmert-Streib - *Advances in Network Complexity* - 2013 Wiley-VCH Verlag GmbH & Co. KGaA, Boschstr. 12, 69469 Weinheim, Germany

►Păun Gh., *Mecanisme generative ale proceselor economice*, Ed Tehnică, Bucureşti, 1988

►Dassow J., Păun Gh., *On the power of membrane computing*, J. Universal Computer Sci., 5, 2 (1999), 33-49, (http://www.iicm.edu/jucs).

►Calude Ch. S., Păun Gh., *Computing with Cells and Atoms an introduction to quantum, DNA and membrane computing*, Taylor & Francis , 2001

►Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison Wesley, 2001.

►Salomaa, A. , *Formal Languages*, New York, Academic Press, 1973.

►Crăciunean, V., *Proiectarea Translatoarelor*, Sibiu, Editura Universității Lucian Blaga Sibiu, 2014, ISBN 978-606-12-0842-5

►Crăciunean, V., *Stochastic grammars for intelligent software systems modelling*, Proceedings of the 8th WSEAS International Conference, Mathematical Methods and Computational Techniques in Electrical Engineering (MMACTEE), Bucharest, Romania 2006, pag. 159-163, ISSN 1790-5117, ISBN 960-8457-54-8.

►Crăciunean, V., *Stochastic Grammars in Simulation Models* – WSEAS Transactions on Computers, Issue 2, Volume 6, February 2007, pag. 355-360, ISSN 1109-2750.

►Fabian R., Crăciunean, V., Popa E. M., *Intelligent system modeling with total fuzzy grammars*, – Proceedings of the 8th WSEAS International Conference, Mathematical Methods and Computational Techniques in Electrical Engineering (MMACTEE), Bucharest, Romania 2006, pag. 82-87, ISSN 1790-5117, ISBN 960-8457-54-8.

►Crăciunean, V., Fabian R., Popa E. M., *Package architecture optimization in software application design*– Proceedings of the 8th WSEAS International Conference, Mathematical Methods and Computational Techniques in Electrical Engineering (MMACTEE), Bucharest, Romania 2006, pag. 99-102, ISSN 1790-5117, ISBN 960-8457-54-8.

►Crăciunean, V., Fabian R., Popa E. M.,*Total Fuzzy Grammar Driven Architecture for Intelligent Software Sytems*– WSEAS Transactions on Computers, Issue 2, Volume 6, February 2007, pag. 248-253, ISSN 1109-2750.

►Crăciunean, V., Fabian R., Popa E. M. ,*Algorithm for Optimal Decomposition of Software Applications*– WSEAS Transactions on Computers, Issue 2, Volume 6, February 2007, pag. 242-247, ISSN 1109-2750.

►Crăciunean, V., Aron C.E. , Fabian R., Hunyadi I.D., *Multi Level Recursive Specifications for Context Free Grammars* - Proceedings of the 11th WSEAS International Conference on COMPUTERS, Agios Nikolaos, Crete Island, Greece, 2007, pag. 275-280, ISSN 1790-5117, ISBN 978-960-8457-95-9.

►Crăciunean, V., Fabian R., Hunyadi I.D, Popa E. M.,*Fix point internal hierarchy specification for context free grammars* - Proceedings of the 11th WSEAS International Conference on COMPUTERS, Agios Nikolaos, Crete Island, Greece, 2007, pag. 247-251, ISSN 1790-5117, ISBN 978-960-8457-95-9.