DEVS Models of Palletized Ground Stacking in Storeyed Grain Warehouse

Shu-Yi HOU[1,a], Yu-Kun LIU[1] and Xiao-Guang ZHOU[1]

[1]*Logistics engineering, School of Automation, Beijing University of Posts and Telecommunications, China*

Abstract. Processed grain stored in storeyed warehouse is generally stacked on the ground without pallets. However, in order to improve the storing way, we developed a new stacking method, palletized ground stacking. Simulation should be used to present this new storing way. DEVS provides a formalized way to describe the system model. In this paper, DEVS models of palletized ground stacking in storeyed grain warehouse are given and a simulation model is developed by AutoMod.

1 Introduction

Mostly grain is stored by stack on the ground without pallets in the warehouse. However, this way of storing has many disadvantages: (1) Mechanization degree is low. A lot of workers are needed to do loading, unloading and stacking operations when the grain is being moved into or out of the warehouse. If an emergency happens, high-intensive manual operation would decrease efficiency and lead to the delay of relief. (2) This way of storing would damage the grain because loading and unloading manual operations must be done. (3) Frequently walking into and out of the storing room would pollute the storing environment.

In order to improve the storing way of grain, we develop a new stacking method, palletized ground stacking. It not only can increase the mechanization level, but decrease the cost and the probability of damaging grain as well. The system of storeyed grain warehouse is always complicated and has many random factors, so analyzing it using traditional mathematical tools is difficult. Simulation is a useful tool than can be used to analyze this new storing way better. In most cases, we abstract the target system in the real world into system model before designing simulation model. DEVS is short for discrete event system specification. It provides a formalized method to describe system model based on system theory. It is a mechanism to modularize, hierarchize and formalize system modeling and simulation. DEVS has been proved that it is a general formalized mechanism to describe every subclass of discrete event system. DEVS can make developers share their achievement at modeling and simulation at most. So we use DEVS to build the system model of palletized ground stacking in storeyed grain warehouse.

2 The DEVS Formalism

DEVS model can describe the system by two levels, namely, atomic DEVS and coupled DEVS. Atomic DEVS models describe autonomous behaviors of a system including transitions of system state, responses to external inputs, system outputs and so on. Several atomic DEVS models can be connected into a coupled DEVS model, generating a hierarchical construction. A coupled DEVS model is comprised of many components which can be either atomic DEVS or coupled DEVS.

An atomic DEVS model can be written as follows.

$$\textit{Atomic DEVS}=<X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta> \qquad (1)$$

X is external input event set. Y is output set. S is system state set. ta is time advance function. ta(s) denotes the time that the system state remains at s without any external events. ta(s)=+∞ means that when there are no external events arrive, the system state will be s forever. ta(s)=0 indicates the simulation clock will not be advanced as long as the system state remains s.ext is external state transition function. int is internal state transition function. If there are no external events arrive, state s will transit to int(s) after the time ta(s). At the same time, e will be reset to 0. is output function.

A coupled DEVS model can be written as follows.

$$\textit{Coupled DEVS}=<X_{self}, Y_{self}, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, select> \qquad (2)$$

self is coupled DEVS self. Xself is external input event set. Yself is output event set. D is unique component names set (self∉ D). {Mi} is coupled DEVS component set. {Mi|i∈ D} and every Mi is an atomic DEVS. {Ii} is a

[a] houshuyi2928@163.com

set of influences of a component. {Zi,j} is the transition of component i's output into component j's input. The input and output transition set is {Zi,j|i∈ D∪{self}, j∈ Ii}. {Zi,j} denotes the coupled construction. select is select function. When several components transform their states, select function is used to choose one of these state transitions as coupled model state transition.

3 DEVS Models of Palletized Ground Stacking in Storeyed Grain Warehouse

The model of delivering grain into warehouse is similar to the model of delivering grain out of warehouse. Hence, there are only given the DEVS models of delivering grain out of warehouse. The grain is palletized stacked on the ground. The entities in this system include forklifts, control center (core), trucks, elevators, slideways, reciprocating lifts and every single room in the warehouse. Forklifts move the grain out of every single storage room. The grain on the first floor is delivered out of warehouse to trucks by forklift. The grain on the higher floor is delivered to the first floor by elevator, slideway or reciprocating lift and then be taken into trucks by forklift. In this paper, coupled DEVS is built first to illustrate the relationship of every entity and then each entity is described as atomic DEVS.

3.1. Coupled DEVS Models

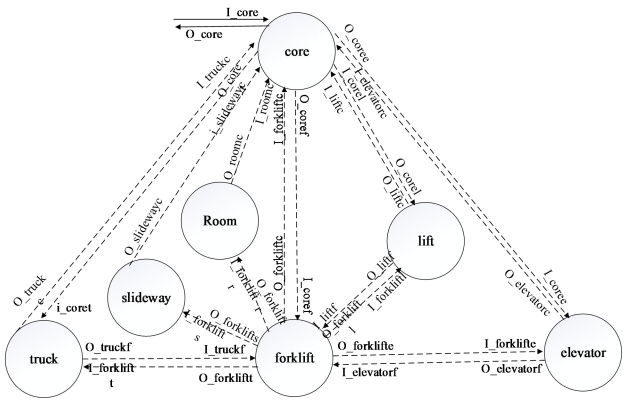

Figure 1. Coupled DEVS of Palletized Ground Stacking In Storeyed Warehouse.

The circles in Fig.1 stand for components in coupled DEVS including every atomic DEVS model, namely, truck, forklift, room, slideway, reciprocating lift, elevator, core and room. Arrows stand for the relationship between external world and coupled DEVS and the relationship between input and output of every atomic DEVS. The words on the arrow stand for the input and output port of atomic DEVS. For example, i_truckf stands for the input port of forklift which can receive information from truck. i stands for input port and f is short for forklift. O_elevatorc stands for the output port of elevator which can send information to core. o stands for output port and c is short for core. It is the same expressing way as other input and output ports. Coupled DEVS inputs external events by i_core (the information about the grain in the warehouse) and outputs statistical information such as grain quantity, departure time and machine utility through o_core.

3.2 Atomic DEVS Models

The atomic DEVS model of reciprocating lift is described in detail by Fig. 2 and equation 3-10. The atomic DEVS model of elevator is described in detail by Fig. 3 and equation 11-18. The atomic DEVS model of truck is described in detail by Fig. 4 and equation 19-26. The atomic DEVS model of forklift is described in detail by Fig. 5 and equation 27-34.

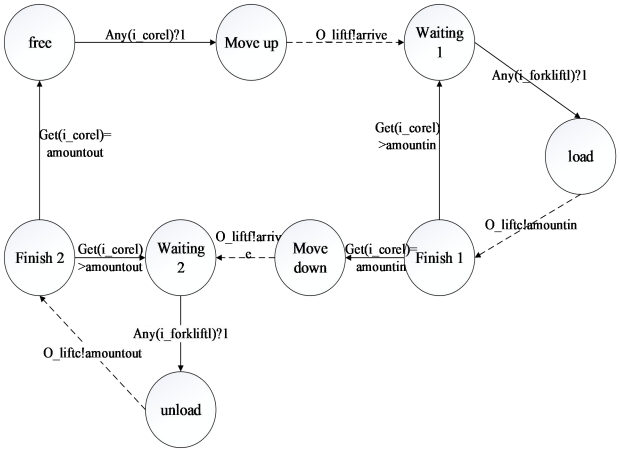

Figure 2. Reciprocating Lift Atomic DEVS

Any(i_corel)?1: receive outbound task from core
O_liftf!arrive: inform forklift that it has already been arrived
Any(i_forkliftl)?1: loading and unloading request from forklift
O_liftc!amountin: the accumulated quantity of grain that is loading into the lift
Get(i_forkliftl)>amountin: the outbound task has not been finished
Get(i_corel)=amountin: finish outbound task
O_liftc!amountout: the accumulated quantity of grain that is unloading out of the lift
Get(i_corel)=amountout: finish loading and unloading task

$$Lift=<X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta> \quad (3)$$

$$X=\{i\_corel, i\_forkliftl\} \quad (4)$$

$$Y=\{O\_liftc, O\_liftf\} \quad (5)$$

$$s = \begin{Bmatrix} free \\ move\ up \\ waiting1 \\ load \\ finish1 \\ move\ down \\ waiting2 \\ unload \\ finish2 \end{Bmatrix} \quad (6)$$

$$\delta_{int} = \begin{Bmatrix} \delta_{int}(move\ up) = waiting1 \\ \delta_{int}(load) = finish1 \\ \delta_{int}(move\ down) = waiting2 \\ \delta_{int}(unload) = finish2 \end{Bmatrix} \quad (7)$$

$$\delta_{out} = \begin{Bmatrix} \delta_{out}(waiting1, i\_forkliftl) = load \\ \delta_{out}(free, i\_corel) = move\ up \\ \delta_{out}(finish1, i\_corel) = move\ down \\ \delta_{out}(finish1, i\_corel) = waiting1 \\ \delta_{out}(waiting2, i\_corel) = finish2 \\ \delta_{out}(finish2, i\_corel) = free \\ \delta_{out}(waiting2, i\_forkliftl) = unload \end{Bmatrix} \quad (8)$$

$$\lambda = \begin{Bmatrix} \lambda(move\ up) = O\_liftf \\ \lambda(load) = O\_liftc \\ \lambda(move\ down) = O\_liftf \\ \lambda(unload) = O\_liftc \end{Bmatrix} \quad (9)$$

$$ta = \begin{cases} ta(free) = ta(waiting1) = ta(waiting2) = ta(finish1) = ta(finish2) = +\infty \\ ta(move\ up) = move\ up - time \\ ta(move\ down) = move\ down - time \\ ta(load) = load - time \\ ta(unload) = unload - time \end{cases} \quad (10)$$

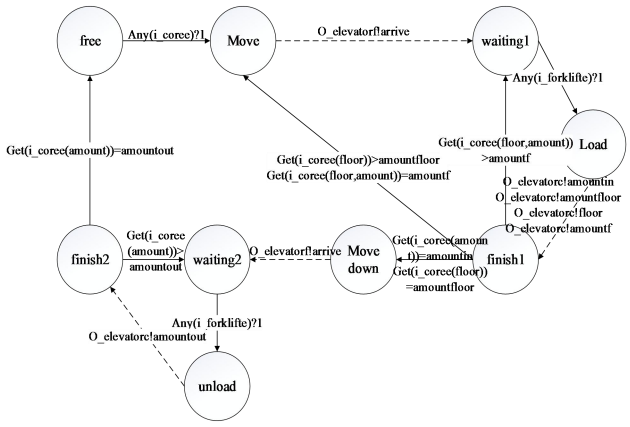

Figure 3. Elevator Atomic DEVS

Any(i_coree)?1: receive outbound task from core
O_elevatorf!arrive: inform forklift that it has already been arrived
Any(i_forklifte)?1: loading and unloading request from forklift
O_elevatorc!amountin: the accumulated quantity of grain that is loading into the elevator
O_elevatorc!amountfloor: the accumulated floor quantity that elevator stops
O_elevatorc!floor: the floor number that elevator is stopping at
O_elevatorc!amountf: the loading quantity from the floor that elevator is stopping at
Get(i_coree(floor,amount))>amountf: the task of loading into the elevator at the current floor has not been finished
Get(i_coree(floor,amount))=amountf: finish the task of loading into the elevator at the current floor
Get(i_coree(floor))>amountfloor: the amount of total floor that elevator should stop has not been reached
Get(i_coree(floor))=amountfloor: reach the amount of total floor that elevator should stop
Get(i_coree(amount))=amountin: finish loading task
O_elevatorc!amountout: the accumulated quantity of grain that is unloading out of the elevator
Get(i_corec(amount))>amountout: the task of unloading from elevator has not been finished
Get(i_corec(amount))=amountout: finish loading and unloading task

$$Elevator = <X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta> \quad (11)$$

$$X = \{i\_coree, i\_forklifte, i\_corec\} \quad (12)$$

$$Y = \{O\_elevatorf, O\_elevatorc\} \quad (13)$$

$$s = \begin{Bmatrix} free \\ move \\ waiting1 \\ load \\ finish1 \\ move\ down \\ waiting2 \\ unload \\ finish2 \end{Bmatrix} \quad (14)$$

$$\delta_{int} = \begin{Bmatrix} \delta_{int}(move) = waiting1 \\ \delta_{int}(load) = finish1 \\ \delta_{int}(move\ down) = waiting2 \\ \delta_{int}(unload) = finish2 \end{Bmatrix} \quad (15)$$

$$\delta_{out}=\begin{Bmatrix}\delta_{out}(waiting1,i\_forklifte)=load\\ \delta_{out}(free,i\_coree)=move\\ \delta_{out}(finish1,i\_coree)=move\ down\\ \delta_{out}(finish1,i\_coree)=waiting1\\ \delta_{out}(waiting2,i\_coree)=finish2\\ \delta_{out}(finish2,i\_coree)=free\\ \delta_{out}(finish1,i\_coree)=waiting1\\ \delta_{out}(finish1,i\_coree)=move\\ \delta_{out}(waiting2,i\_forklifte)=unload\end{Bmatrix} \quad (16)$$

$$\lambda=\begin{Bmatrix}\lambda(move)=O\_elevatorf\\ \lambda(load)=O\_elevatorc\\ \lambda(move\ down)=O\_elevatorf\\ \lambda(unload)=O\_elevatorc\end{Bmatrix} \quad (17)$$

$$ta=\begin{cases}ta(free)=ta(waiting1)=ta(waiting2)=ta(finish1)=ta(finish2)=+\infty\\ ta(move)=move-time\\ ta(move\ down)=move\ down-time\\ ta(load)=load-time\\ ta(unload)=unload-time\end{cases} \quad (18)$$

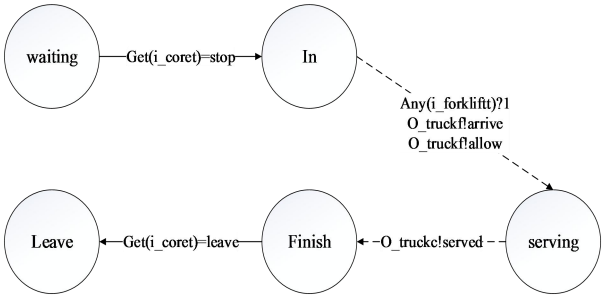

Figure 4. Truck Atomic DEVS

Get(i_coret)=stop: receive stopping information from core

Any(i_forklifttt)?1: loading and unloading request from forklift

O_truckf!arrive: truck has already been stopped

O_truckf!allow: truck can be loaded and unloaded

O_truckc!served: finish loading and unloading task

Get(i_coret)=leave: receive leaving information from core

$$Truck=<X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta> \quad (19)$$

$$X=\{i\_coret, i\_forklifttt\} \quad (20)$$

$$Y=\{O\_truckf, O\_truckc\} \quad (21)$$

$$s=\begin{Bmatrix}waiting\\ in\\ serving\\ finish\\ leave\end{Bmatrix} \quad (22)$$

$$\delta_{int}=\begin{Bmatrix}\delta_{int}(in)=serving\\ \delta_{int}(serving)=finish\end{Bmatrix} \quad (23)$$

$$\delta_{out}=\begin{Bmatrix}\delta_{out}(in,i\_forklifttt)=serving\\ \delta_{out}(waiting,i\_coret)=in\\ \delta_{out}(finish,i\_coret)=leave\end{Bmatrix} \quad (24)$$

$$\lambda=\begin{Bmatrix}\lambda(in)=O\_truckf\\ \lambda(serving)=O\_truckc\end{Bmatrix} \quad (25)$$

$$ta=\begin{cases}ta(waiting)=ta(leave)=ta(finish)+\infty\\ ta(in)=in-time\\ ta(serving)=serving-time\end{cases} \quad (26)$$

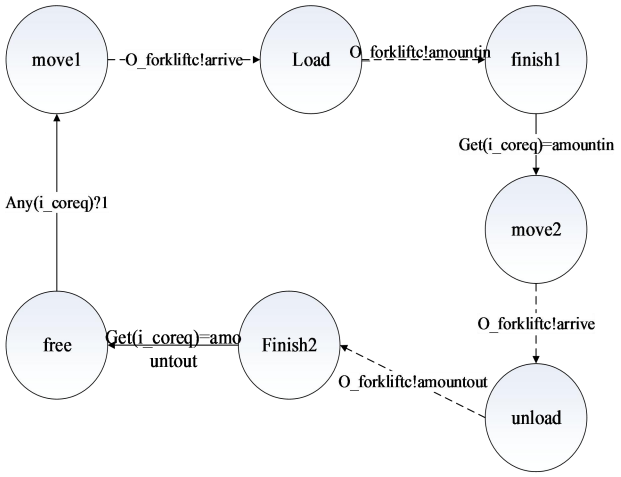

Figure 5. Forklift Atomic DEVS

Any(i_coref)?1: receive outbound task from core

O_forkliftc!arrive: arrive at the destination

O_forkliftc!amountin: the quantity of grain that the forklift picks up

Get(i_coref)=amountin: finish the task of picking up a pallet of grain

O_forkliftc!amountout: the accumulated unloading quantity

Get(i_coref)=amountout: finish outbound task

$$Forklift=<X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta> \quad (27)$$

$$X=\{i\_coref\} \quad (28)$$

$$Y=\{O\_forkliftc\} \quad (29)$$

$$s = \begin{Bmatrix} move1 \\ load \\ finish1 \\ move2 \\ unload \\ finish2 \\ free \end{Bmatrix} \quad (30)$$

$$\delta_{int} = \begin{Bmatrix} \delta_{int}(move1) = load \\ \delta_{int}(load) = finish1 \\ \delta_{int}(move2) = unload \\ \delta_{int}(unload) = finish2 \end{Bmatrix} (31)$$

$$\delta_{out} = \begin{Bmatrix} \delta_{out}(finish1, i\_coref) = move2 \\ \delta_{out}(finish2, i\_coref) = free \\ \delta_{out}(free, i\_coref) = move1 \end{Bmatrix} (32)$$

$$\lambda = \begin{Bmatrix} \lambda(move1) = O\_forkliftc \\ \lambda(load) = O\_forkliftc \\ \lambda(move2) = O\_forkliftc \\ \lambda(unload) = O\_forkliftc \end{Bmatrix} (33)$$

$$ta = \begin{Bmatrix} ta(waiting1) = ta(waiting2) = ta(finish1) = ta(finish2) = +\infty \\ ta(move1) = move1 - time \\ ta(move2) = move2 - time \\ ta(load) = load - time \\ ta(unload) = unload - time \end{Bmatrix} \quad (34)$$

4 Simulation Experiment

We use the DEVS models above as the system model and then develop a simulation model to present this new storing way, palletized ground stacking, by AutoMod.

4.1. Transformation Process

The following steps indicate the process that transforming DEVS models into simulation model.

4.1.1 Model initialization

Information about the grain in the storeyed warehouse is input into control center through i_core. Trucks, forklifts, elevators and reciprocating lifts inform their current states to control center through o_truckc, o_forkliftc, o_elevatorc, o_liftc. We use variables and load attributes in AutoMod to replace input and output ports in DEVS models. Every atomic DEVS model is transformed into sub models in AutoMod.

4.1.2 Model implementation

(1) According to the outbound task information sent by core, forklifts send information of arriving and the amount of loading and unloading to input port i_truckc through output port o_truckc.

(2) After trucks arrive, trucks send information of arriving and loading amount to i_truckf through o_truckf. If the loading task has already been finished, trucks send information of finishing to input port i_truckc through o_truckc. Then core orders truck to leave through input port i_coret.

(3) According to the outbound task information sent by core, reciprocating lifts send information of arriving to input port i_liftf through output port o_liftf. Then reciprocating lifts send information of loading and unloading amount to input port i_liftc through output port o_liftc.

(4) According to the outbound task information sent by core, elevators send information of arriving to input port i_elevatorf through output port o_elevatorf. Then elevators send information of loading and unloading amount to input port i_elevatorc through output port o_elevatorc.

4.2. A Simulation Example

A storeyed grain warehouse has five floors. The first floor is 7.2 meters high and there are five single rooms to store processed grain. The second floor is 7.2 meters high and there are six rooms. Other floors are all six meters high and there are four rooms on every floor. The storing way is palletized ground stacking. 40 bags of grain can be stacked on one pallet and one bag of grain weighs 25 kilograms. One reciprocating lift, two elevators and two slideways can be used to deliver grain from higher floors to the first floor. The reciprocating lift can be only used by the grain on the second floor. The capacity of elevators is 4 pallets of grain. The grain can be delivered by either elevator or slideway. When the grain on two different floors need to use one and the same elevator, this elevator can be used to deliver grain on both floors but the grain on higher floor has the priority to be delivered first. The storeyed grain warehouse has 9 platforms for trucks. We developed a simulation model of this storeyed grain warehouse in AutoMod and the simulation panorama is shown in Fig. 6.

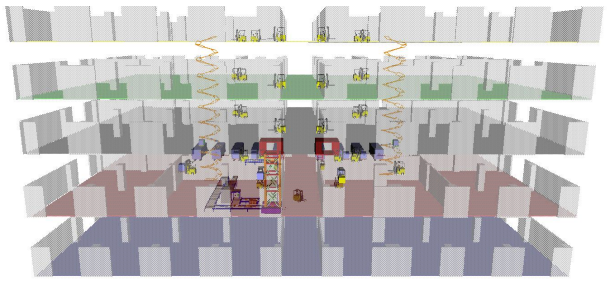

Figure 6. Simulation Panorama

5 Summary

To improve the way of storing grain in storeyed warehouse, a new storing way, palletized ground stacking, is developed. Furthermore, to better present this new storing way, we build DEVS models as the system model before developing simulation model. Finally, we develop a simulation model according to the DEVS models.

Acknowledgements

This work is supported by Natural Public Industry (Food) Special Funds for Scientific Research (No. 201313009-06).

References

1. W. Bin, Communication and network 12, 128-130, (2008)
2. K.H. Kim, Y.R. Seong, T.G. Kim, and K.H. Park, Transactions of the Society for Computer Simulation International, 13, 135-154,(1996)
3. S. Robinson, R.E. Nance, R.J. Paul, M. Pidd, and S.J.E. Taylor, Simulation modeling practice and theory, 12, 479-494, (2004)
4. H. Rui, Z. Guo-chun, and L. Shao-jun, Journal of system simulation, 23, 30-33, (2001)
5. M. Spiegel, P.F. Reynolds, and D.C. Brogan, Proceedings of the 2005 winter simulation conference, 437-444, (2005)
6. H. Vangheluwe, CS522 Fall Term 2001, (2001)
7. Q. Xiao-gang, and D. Wei, Journal of system simulation, 21, 6697-6709, (2009)
8. C. Yun, Q. Yue, and Z. Yu, Technology and Method, 33, 253-256, (2013)