TOWARDS THE INTEGRATED SIMULATION AND PROGRAMMING OF PALLETIZING LINES

Antonello Calò[1], Davide Buratti[1], Dario Lodi Rizzini[2], and Stefano Caselli[2]
[1]*OCME S.r.l., Parma, Italy*
[2]*University of Parma, Italy*
{*a.calo, d.buratti*}*@ocme.it* {*dlr,caselli*}*@ce.unipr.it*

Keywords: Industrial Machine, Simulation, Visual Programming

Abstract: In this paper, we discuss the advantages and the issues of simulation and visual programming of palletizing machines and, in general, palletizing lines, and we illustrate an integrated software tool suite that meets such requirements. The increasing complexity of lines and the variability of product formats require a common machine model for all the tools, together with the independence from visualization to allow software reuse and extendibility. Furthermore, the model should also be able to include palletizing line components external to the machine that are critical for performance and whose model is only partially known.

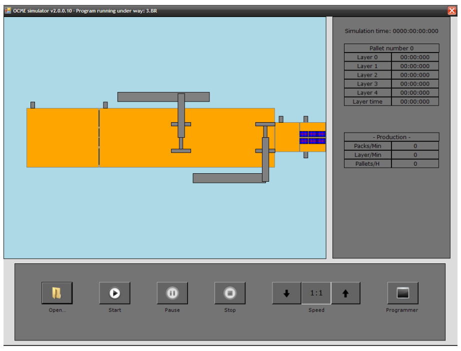
1 Introduction

Palletizing units have represented an important part of manufacturing systems for decades. However, the design of modern palletizing systems is facing new challenges dictated by market demands, which include increasing production rates, the larger variability in product packaging and formats, the thinner and more fragile primary containers and products to be palletized. The complexity of modern palletizing systems has increased: they have to ensure faster product manipulation and to adapt online to different pallet formats. This trend also challenges human operators, whose skills often do not match the new generation of machines.

Simulation and visual programming software tools can help to meet the requirements for modern palletizing units and assist both manufacturers and end users. Several software packages have been proposed for simulation of a machine component (Dong et al., 2002), for integrated simulation and parameters optimization of a manipulator (Kazi et al., 2002), and for high-level simulation of manufacturing systems (Inukai et al., 2007). Few tools, however, integrate simulation, programming and monitoring of a complete palletizing line.

In our previous work (Argenti et al., 2010), we proposed an integrated software tool suite for simulation, programming and monitoring of palletizing machines. The tool suite had been designed for a line of robot-based palletizing machines manufactured by OCME S.r.l., but the underlying concepts are suitable for a wide range of end-of-line machines. The proposed application was able to handle several machine configurations and some of these have been added long time after the end of software design. However, new demands and requirements have come out through the use and experience with the tool suite. In particular, upstream or downstream components of the palletizing line like transport conveyor belts or end-line manipulators, which are not part of the palletizing machine, are critical for the performance and, thus, should be included in the simulation. Unfortunately, the behavior of such components provided by third-part manufacturers is often unknown. Furthermore, a customized version of the application is required with respect to both the machine model and the user interface, when the tools are provided to a client. A more modular and independent organization of the software tools could simplify the customization.

In this paper, we discuss the advantages and the issues of the integrated simulation, visual programming and monitoring of palletizing machines with their extensions and illustrate an improved software tool suite to demonstrate our position. In particular, we claim that an integrated simulator and programmer that models machine logic and behavior avoiding unnecessarily detailed physical description, can improve and speed up the development and the programming of a machine. The simulation and modeling should also include components of the lines that can affect the performance of the palletizing machine. When

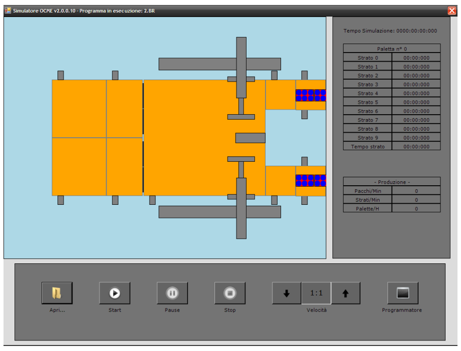

OCME simulator v2.0.0.10 - Program running under way: 3.BR

Simulation time: 0000:00:00:000

Pallet number 0	
Layer 0	00:00:000
Layer 1	00:00:000
Layer 2	00:00:000
Layer 3	00:00:000
Layer 4	00:00:000
Layer time	00:00:000

- Production -	
Packs/Min	0
Layer/Min	0
Pallets/H	0

(a)


Simulatore OCME v2.0.0.10 - Programma in esecuzione: 2.BR

Tempo Simulazione: 0000:00:00:000

Paletta n° 0	
Strato 0	00:00:000
Strato 1	00:00:000
Strato 2	00:00:000
Strato 3	00:00:000
Strato 4	00:00:000
Strato 5	00:00:000
Strato 6	00:00:000
Strato 7	00:00:000
Strato 8	00:00:000
Strato 9	00:00:000
Tempo strato	00:00:000

- Produzione -	
Pacchi/Min	0
Strati/Min	0
Palette/H	0

(b)

Figure 1: Configuration of palletizers: single (a) or double (b) channel with arbitrary number and position of manipulators

the behavior of such components cannot be fully described, an approximate model can be used possibly using a statistical model for some unknown variables (e.g. the time required to perform a specific operation).

We also claim that the design of the tool suite, while using a unique machine model for all tools, should be inspired by the *Model-View-Control* (MVC) software design paradigm, which allows code reuse and ensures consistency among the behaviors achieved by the applications. The new version of the tool suite has been implemented using *Microsoft Expression Blend Framework* (MEBF) to better meet such requirements. Such framework allows a greater independence of the machine model and control from the visualization, since it manages interface as vector graphics. Furthermore, the framework allows a more modular organization of the suite: the application consists of separate components, each described by a *XAML* (eXtensible Application Markup Language) file that specifies the graphic interface and the model and control program.

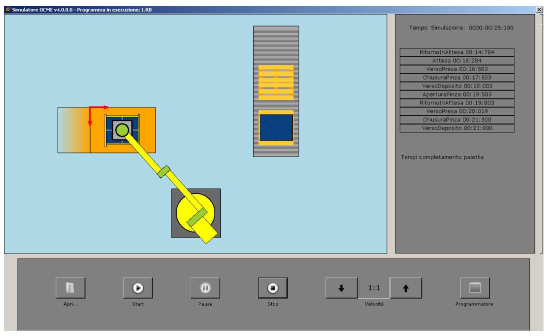
The paper is organized as follows. Section 2 illustrates the common model of the palletizing machine used by all tools. Section 3 describes the simulator. Section 4 presents the visual programming tool for the generation of control parameters. Finally, section 5 gives conclusion remarks.

2 Model of the Palletizing Machine

The integrated tool suite for simulating, programming and monitoring the palletizing lines is built on the model of the system. Since the structure of a palletizing line as well as its core palletizing machine is subject to variability, the model consists of the composition of the elementary components of the system. The accuracy level of each model depends on available information on the machine. A model should describe the elementary components, the dimensional parameters, and the state of the machine. Since the aim of the tool suite is to provide an user friendly interface for palletizer management, the graphical representation of each component is as important as the list of the attributes. However, the state of the machine is the most critical part for machines that are provided by third party manufacturers. Each tool uses the machine model according to its own purposes. The simulator modifies the state of the machine to emulate its behavior through state transitions. The monitoring tool represents the current state of the palletizer according to the messages received from the machine controller, in the case there is a communication channel with the controller. The programmer reads and writes the parameters that allow formation of pallet layers.

The model, the graphical representation and the behavior of the machines palletizer should be independent. The software design of the tool suite has adopted the MVC paradigm that proposes such separation between the three aspects as a general design criterion. Such approach has been applied to define the palletizer components in (Argenti et al., 2010). Figure 1 shows how the same components can be reused to represent different machine configurations.

The same approach has been applied to model the end-line manipulator that moves the already formed pallet layers from the palletizing machine to the pallet layer stack. The end-line manipulator is not strictly part of the machine, but it is manufactured by a third-part producer. A complete model of the physical arm and of its kinematics is either difficult to obtain because the information is unavailable or too complex and detailed for our purposes. Thus, an approximate planar model of the manipulator has been exploited by partially reusing components designed for the palletizer. The position of the manipulator components has been expressed with respect to a relative frame. Figure 2 illustrates the graphical representation of the end-line manipulator. The design of the machine model has proven sufficiently flexible and robust to the introduction of this new palletizing line element.



```
Tempo  Simulazione:  0000:00:25:190

RitornoInAttesa 00:14:784
Attesa 00:16:284
VersoPresa 00:16:503
ChiusuraPinza 00:17:503
VersoDeposito 00:18:003
AperturaPinza 00:19:503
RitornoInAttesa 00:19:803
VersoPresa 00:20:019
ChiusuraPinza 00:21:300
VersoDeposito 00:21:800

Tempi completamento paletta
```

Figure 2: End-line manipulator to move and the formed pallet layer from the palletizing machine (middle-left) to the pallet layer stack (top-right).

Although product units are not strictly part of the machine, the concept of component has been conveniently extended also to units and group of units. A component is defined as the basic part of the system that contributes to the description of the system state. Components also have a graphical representation and their visualization is handled by proper routines. A machine configuration is defined through a configuration file in text format. During initialization, this file is parsed and used to create the palletizer model with the aggregation of the specified components. In the future, we expect to convert the specifications contained in the custom configuration file to the XAML format to fully exploit the features of Microsoft Expression Blend Framework.

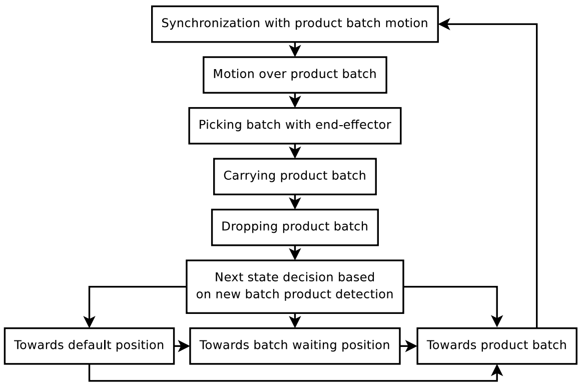

Synchronization with product batch motion
Motion over product batch
Picking batch with end-effector
Carrying product batch
Dropping product batch
Next state decision based on new batch product detection
Towards default position
Towards batch waiting position
Towards product batch

Figure 3: Finite state machine representing the manipulator behavior.

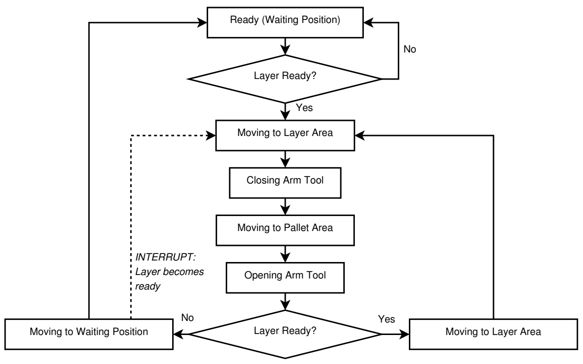


Figure 4: Finite state machine approximating the behavior of the end-line manipulator.

3 Simulator

The proposed tool suite allows an accurate simulation of the palletizing line to estimate its throughput and the time required to complete specific production steps. The simulating tool can also be used to display the final system to designers and customers. The accuracy of simulation closely depends on the model described in the previous session, since simulation is essentially defined by the system state variables and the transition function that modifies the value of state.

The simulation of the palletizing machine is focused on the product units. Indeed, the palletizing machine operates on batches of product units in order to arrange them into pallet layers. In this case, the activity of each machine component is naturally modeled as a queue or FIFO representing the sequential processing operated on product units as illustrated in (Argenti et al., 2010).

The simulation of an external element like the end-line manipulator illustrated in the previous section has required a different approach. Indeed, the end-line manipulator does not operate on single product units or batches, but on pallet layers. Thus, the FIFO interface, which is suitable to model the operations inside the palletizing machine, cannot be used in this case. For this purpose, a hierarchy of simulators and and a high level interface `Simulator` have been defined. `Simulator` is a composite object that allows the composition and joint management of multiple simulators. In this perspective, the palletizing machine and the manipulator can share the same event queue.

Each `Simulator` is responsible of the subsystem state update. The state consists of both continuous and logic variables. The transition events of logic variables are handled at the beginning of a simulation time interval as illustrated in (Argenti et al., 2010). The coordination among the simulators is achieved using the same simulation time step. In particular, the global time step is adapted to the minimum update time of the simulators (Zeigler et al., 2000).

3.1 The Queues of Palletizing Machine

The control and simulation of palletizing machine has been decomposed in four queues or FIFOs: the input conveyor belt queue, the manipulation queue, the movable barriers queue and the conveyor belt queue. As explained before a FIFO represent a specific operation or group of operations performed on the product units that are stored in the FIFO.

The *input conveyor belt queue* controls the state of the first group of two conveyor belts in each channel, which separates product units in batches.

The *manipulation queue* controls the manipulators that move product batches in the required position. Since these manipulators are part of the machine, their model and control is completely known (the finite state machine is shown in Figure 3). The aim of these manipulators is to place the batches that are not correctly aligned with respect to their final position in the pallet layer. The manipulator can pick its target only after a given position has been reached (third state diagram in Figure 3) and releases the product batch at a given final position. Then, the arm moves toward the new incoming batch, if there is one, or to a waiting position (branching of state diagram).

The *movable barriers queue* controls the product batches piling on the movable barriers located at the end of the manipulation conveyor belt. Barriers are raised in order to form a section of the pallet and are then lowered when the cumulated items are transferred to the merging conveyor belt. To avoid compenetration of product units, the simulator must handle object collision. Management has be simplified by assuming that all the incoming product units are orthogonally oriented with respect to the barriers and that the torques generated by contacts are negligible independently from the contact surface. The accumulation and contact has been handled using a *segment tree* (De Berg et al., 2008).

The *final conveyor belt queue* controls the last conveyor belt that transfers the formed pallet. The number of configurations of the conveyor belt handled by the simulator has been increased from the model in (Argenti et al., 2010). In particular, pallet transfer may be performed by a manipulator (discussed in the next section), by a mechanical pusher, or by a lowering platform.

3.2 End-line Manipulator

The manipulator is a component of the palletizing line that is provided by a third party producer. In some cases, the performance of a palletizing line depends on the performance of the manipulator, e.g. when the time required by the palletizer to form a pallet layer is comparable with time required by the robot arm to transport the layer. Thus, a simulation that aims at assessing the throughput of the production line should also consider the behavior of the robotic arm. Unfortunately, the knowledge of its physical and logical model is usually incomplete and its exact behavior is difficult to reproduce (i.e. the time required to execute a motion).

Instead, the approximate finite state machine represented in Figure 4 has been used. The diagram is similar to the finite state machine of the palletizer manipulators. Since the transition time between two states is only approximately known, the simulator returns a time estimation proportional to the path length or provided as an empirical constant. The user may also choose to randomly generate the time according to common statistical distributions (normal, negative exponential, etc.).

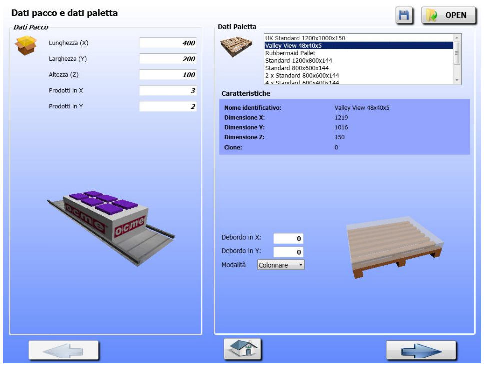
The end-line manipulator does not operate on product units or batches like the various parts of the palletizing machine, but on already formed pallet layers. Hence, the simulation of its behavior cannot be handled by a FIFO of product units. The control of the *end-line manipulator* has been implemented as a specific instance of `Simulator` interface. Such interface may be integrated with other simulators like the palletizer simulator to emulate the behavior of the palletizing line.

4 Programmer

The aim of palletizing line programming is the generation of machine parameters consistent with the pallet layer layouts specified by users. The human operator must only provide high level specification, choose among the solutions proposed by the programming tools according to convenient policies, and possibly edit the solution. Finally, the tool generates the required parameters. These four steps are achieved by the proposed tool that improves the previous version illustrated in (Argenti et al., 2010).

4.1 Layout Generation

The computational counterpart of the physical layer planning problem is the well-known bin packing



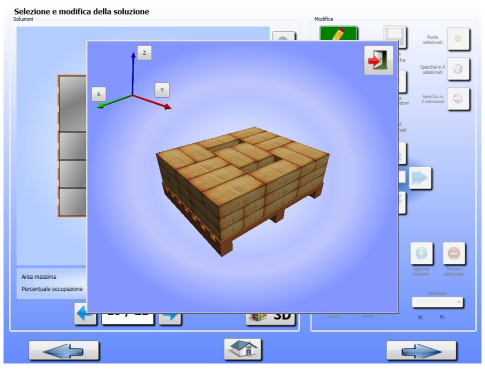
Lunghezza (X)	400
Larghezza (Y)	200
Altezza (Z)	100
Prodotti in X	3
Prodotti in Y	2

UK Standard 1200x1000x150
Valley View 48x40x5
Rubbermaid Pallet
Standard 1200x800x144
Standard 800x600x144
2 x Standard 800x600x144
4 x Standard 600x400x144

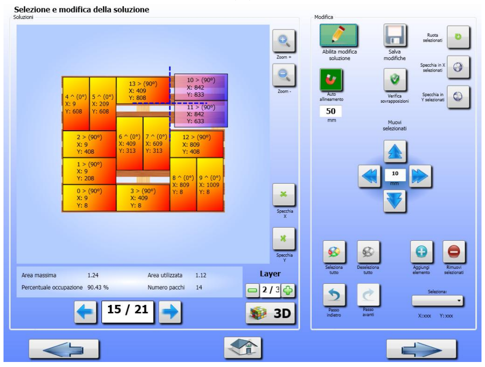
Nome identificativo:	Valley View 48x40x5
Dimensione X:	1219
Dimensione Y:	1016
Dimensione Z:	150
Clone:	0

Debordo in X:	0
Debordo in Y:	0
Modalità	Colonnare

(a)



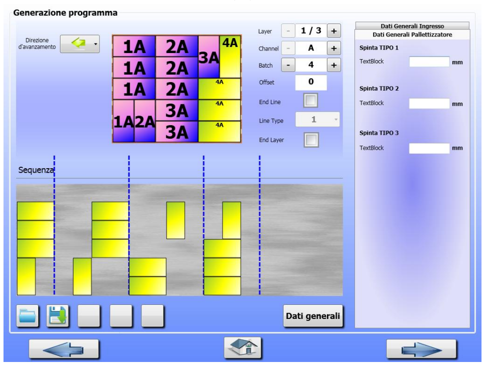
(b)



Area massima	1.24	Area utilizzata	1.12
Percentuale occupazione	90.43 %	Numero pacchi	14

Layer 2 / 3
15 / 21

(c)



Layer	1 / 3
Channel	A
Batch	4
Offset	0
End Line	☐
Line Type	1
End Layer	☐

(d)

Figure 5: The four steps of the programming tool: the selection mask of pallet format and other requirements (a), layout generation (b) and editing (c), and the batch decomposition and computation of machine parameters (d).

problem (Lodi et al., 2002), which is usually transformed into an unidimensional knapsack problem. Unfortunately, bin packing algorithms assist in maximizing the number of product units per layer or pallet but do not consider other practical requirements such as regularity in products arrangement, layer stability, and presence of symmetries. Alternative computational approaches such as rectangular tessellation and rectangle tiling suffer from similar drawbacks and do not exploit special features of the problem such as the uniform product size.

For these reasons, the proposed layout generation system is based on a set of predefined patterns incorporating user expectations as well as typical industrial requirements. The input data for layout computation consist of product size, allowed product overflow or required margin from pallet borders, and pallet style (plain or hierarchical). Figure 5(a) illustrates the mask for inserting high level specifics like pallet format and product unit size. Then, the programmer generates possible pallet layer layouts according to patterns like Binary Cut, Single Spiral, Column and Double Spiral and maximizes the number of units per layer (Figure 5(b)). The user can graphically edit the found solutions through a drag-and-drop interface and select the layouts to be used in each layer (Figure 5(c)). More details on layout patterns are available in (Argenti et al., 2010).

4.2 Batch Decomposition and Parameters Computation

The second step of palletizer programming is the computation of control parameters required to generate the given layout. The layout generated in the previous step consists of product units arranged inside the planar shape of the pallet. Since the manipulation of product units is oriented to batches, the decomposition of the layouts in batches is crucial for parameters computation. Batch formation must follow some rules about maximum number and proximity of product items. A batch corresponds to a group of units that are manipulated together. Thus, its weight and size are limited by the payload and the end-effector size of the manipulators. Furthermore, the batch should have single or double row rectangular shape depending on the machine configuration, since it is formed through commands to the input conveyor belts. Batch selection follows different rules. The sequence number of each batch must be consistent with the order of accumulation. Batch feasibility is checked using a graph-like data structure that represents proximity relationships between units. The selection of batch items can be performed manually, semi-automatically or completely automatically.

Given the batch decomposition of the pallet layer, machine parameters are completely determined. The parameters related to a batch include the number of product items, the number of rows, the spacing from the previous batch, the offset positions for manipulation picking and release, the rotation flag, and the offset positions for barrier raise and release. Barrier offsets depend on the relative displacement of batches in the pallet layer. Thus, the programmer divides the batches in groups according to their alignment to a common front line. Each group corresponds to product batches simultaneously accumulated on the same barrier. Figure 5(d) illustrates the final step of programming tool with the batch numbering and the final product unit sequence. The new version of the programming tool allows parameter generation for a larger number of machine configurations and supports the addition of other machines.

5 Conclusion

In this paper, we have discussed the simulation and programming of palletizing lines and have illustrated the related issues through a tool suite developed for a specific palletizer. A palletizing line consists of components, whose behavior can be fully described or only partially known. In the first case, the simulation and the programming of the machine is better performed on a common model. The simulation should include the control logic and a kinematic model of its parts sufficiently accurate to estimate the time, but should also avoid unnecessary modelling of physical interaction and dynamics of bodies. When the behavior of some machine of the model is unknown, an approximate finite state diagram with empirical estimation of transition times could be used. Such solution is effective when such machine is not a bottle-neck of the whole system.

The MVC design paradigm can effectively support software reuse and enforce consistency between the different tools of the tool suite by separating the management of data, graphical interface and logic control. All the tools should be based on a common model representing the structure of the machine, but each tool should operate differently on the machine data. Furthermore, a software organized in modular components allows customizable tools for the end user.

All these design solutions have been derived from a tool suite for the simulation, programming and monitoring of a specific palletizing machine supporting a high number of configurations. The simulator has been extended to simulate an end-line manipulator using an approximate model. The programming tool assists non-expert human operators in visual programming of the palletizing task, and supports a variety of configuration options automatically planning feasible layers. The organization of the suite according to Microsoft Expression Blend Framework has improved the decoupling between model and interface. The monitoring tool supports diagnostic activities and allows efficient recovery when failure occurs. Furthermore, it has enforced the modular subdivision of the tools, e.g. the decomposition of the programming steps, making possible a better customization of the tool for a specific machine. The proposed tool suite has been already successfully used by several operators on different palletizing lines in working plants.

Acknowledgment

This research has been supported by Regione Emilia-Romagna, Italy, as part of the Integrapack project.

REFERENCES

Argenti, M., Buratti, D., Lodi Rizzini, D., and Caselli, S. (2010). An integrated tool suite for simulation and programming of palletizing units. In *Proc. of the ISR 2010*.

De Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2008). *Computational Geometry, Algorithms and Applications*. Springer-Verlag, 3rd ed.

Dong, W., Palmquist, F., and Lidholm, S. (2002). A simple and effective emulation tool interface development for tricept application. In *Proc. Int. Symp. on Robotics (ISR)*.

Inukai, T., Hibino, H., and Fukuda, Y. (2007). Simulation environment synchronizing real equipment for manufacturing cell. *J. Advanced Mechanical Design, Systems, Manufacturing*, 1(2):238–249.

Kazi, A., Merck, G., Otter, M., and Fan, H. (2002). Design optimization of industrial robots using the modelica multi-physics modelling language. In *Proc. Int. Symp. on Robotics (ISR)*.

Lodi, A., Martello, S., and Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. *Discrete Appl. Math.*, 123:379–396.

Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of Modeling and Simulation*. Academic Press, 2nd ed.