POLITECHNIKA GDAŃSKA

INTELLIGENT LOGISTICS SYSTEMS

CLASSES #2 and #3

Process Flow – model expansion

dr hab. Daniel Kaszubowski, prof. PG

Department of Transportation Engineering

ADTRANS
ADVANCED DATA ANALYSIS FOR TRANSPORTATION AND MOBILITY ACADEMIC TEACHERS

Co-funded by the European Union

Co-funded by the European Union. Views and opinions expressed are however those of the author or authors only and do not necessarily reflect those of the European Union or the Foundation for the Development of the Education System. Neither the European Union nor the entity providing the grant can be held responsible for them.

1. Objective and new skills

The aim of the task is to introduce new features to *Process Flow* and expansion of the current model. It will be modified in two steps (Part I and II of the material). Using the example of new activities, issues related to the use of decision points, references to the object hierarchy and grouping of elements, allowing for complex flow control, will be introduced.

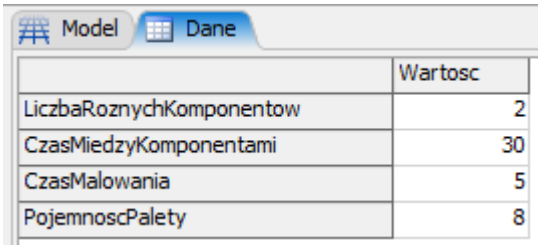
New skills
Decide – a block enabling conditional decisions about the direction of token flow
Batch – a block that creates batches of tokens, allowing grouping of elements
Move Object – displacement of elements
Using object hierarchy references to create element flow logic
Using feedback between blocks of a diagram

2. Part I – model assumptions and input data

The model being created will be a modification of the model from Lesson #1. The following new elements will be introduced to it:

1. The *Palette* source *Zrzut* buffer receives a single palette on which the painted elements are placed.
2. The pallet capacity is 8 elements, once full it is moved by the operator to the *strefaWysylki* buffer.

pojemnoscPalety has been added to the global *Data* table (Fig.1), while the modified model layout is shown in Fig. 2.



	Wartosc
LiczbaRoznychKomponentow	2
CzasMiedzyKomponentami	30
CzasMalowania	5
PojemnoscPalety	8

Fig. 1. Model parameters in the *Data table*

The material was based on: K. Jurczyk, FlexSim , user manual.
InterMarium Sp. z o . o. , Krakow 2022

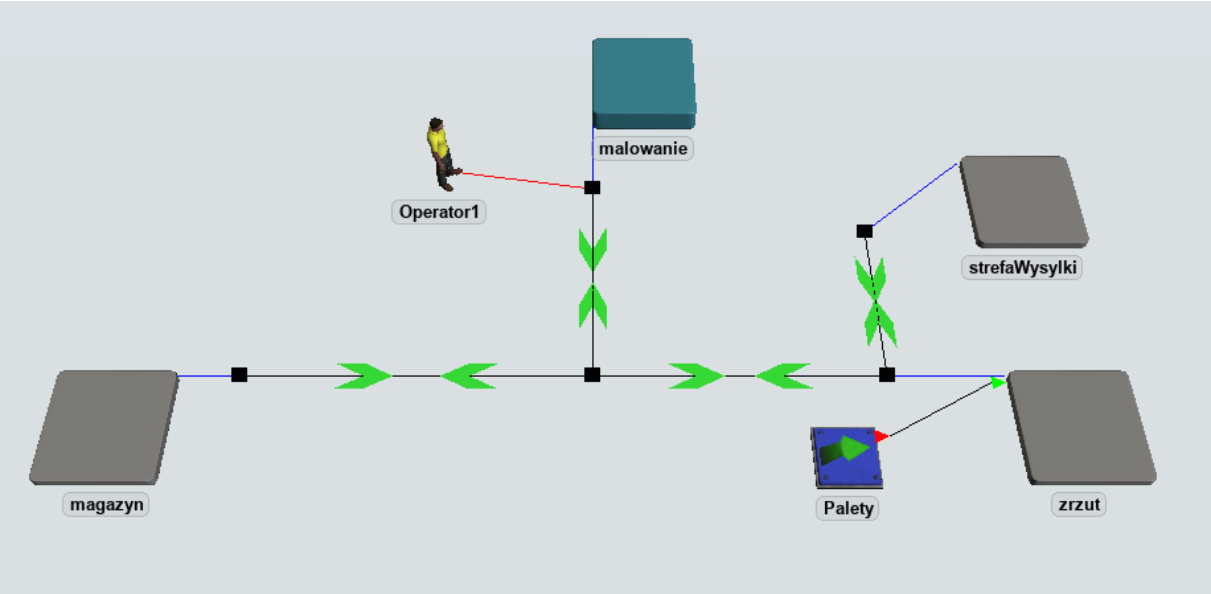


Fig. 2. Modified 3D model layout

2.1. Model control logic

2.1.1. Model structure

2.1.1.1. Modified Process structure Flow

The initial (left fragment) and modified structure (right) of the model block diagram are shown in Fig. 3.

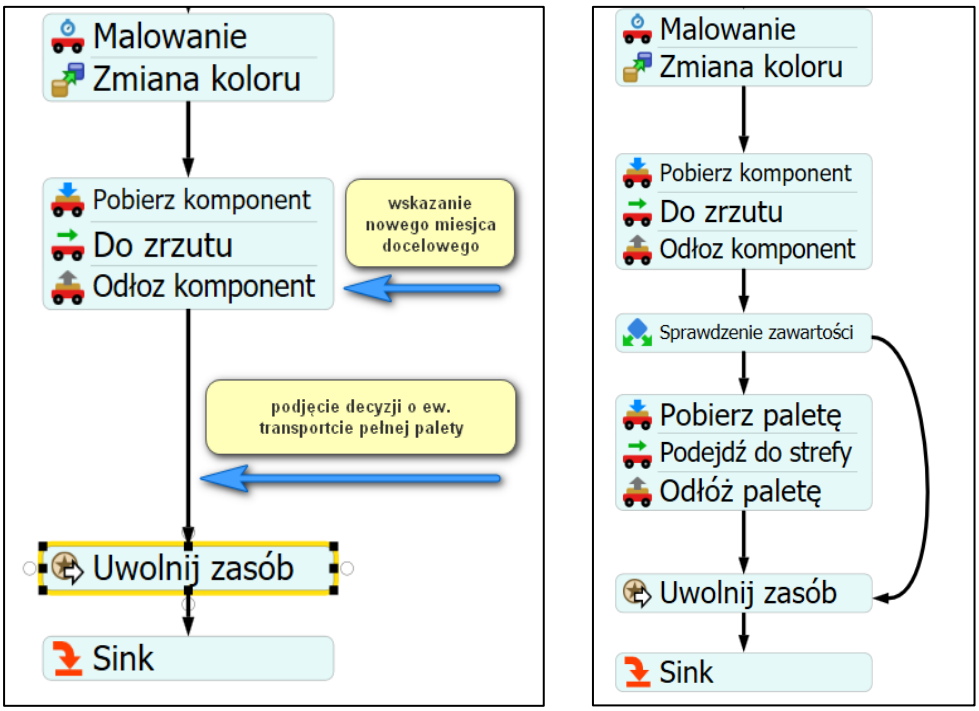


Fig . 3. Initial and modified block diagram

2.1.1.2. 3D model update

Palettes source has been added to the model, the settings of which are shown in Fig. 4. Palettes are generated continuously (time 0).

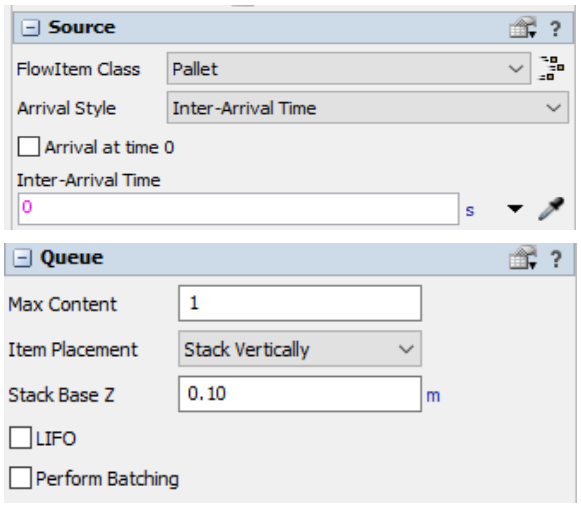


Fig. 4. *Palette Source*

zrzut buffer is used to store pallets and has a capacity of 1 piece (Fig. 5)). The buffer *strefaWysylki* was connected to the existing network of nodes (Fig. 6).

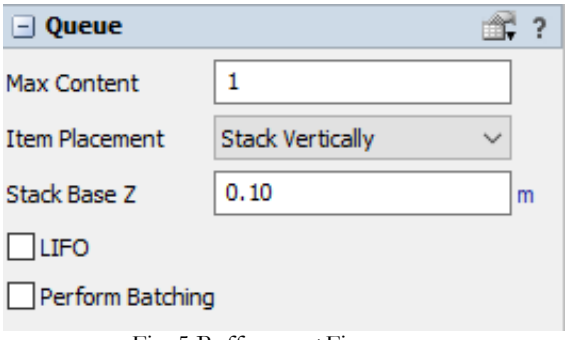
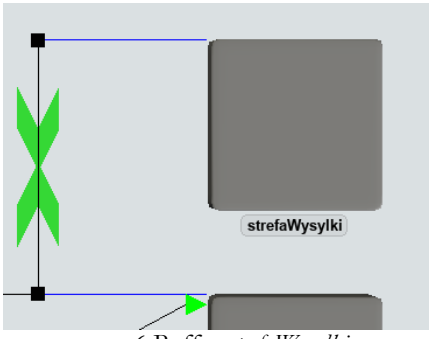


Fig. 5 Buffer *zrzut* Fig.



6 Buffer *strefaWysylki*

2.1.2. Modification Process Flow

2.1.2.1. Assign Labels – new labels

Two new labels should be added to *the Assign labels* block *(Fig. 6):*

1. capacity – its value to refer to row [4] of the *Dane* table,
2. strefaWysylki – reference to the *strefaWysylki* buffer.

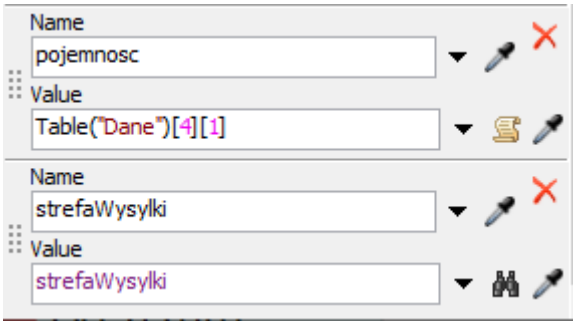


Fig. 6. Label update

2.1.2.2. Decide – put the component aside and decide on further actions

In the modified model, the place where the operator puts down the components will change, from the *zrzut* buffer to the pallet located on this buffer. In addition, after putting down the component, the operator checks whether it is maximally full (8 pcs.). If so, he moves the full pallet to the *strefaWysylki* buffer .

The change in the pallet's place of deposit was introduced in the *Unload* block (Fig. 7), in accordance with the assumptions in Fig. 3. It concerns the entry in the *Station* field.

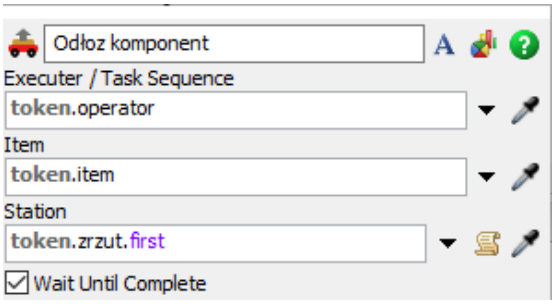


Fig. 7 Settings *Unload* block

The syntax used in the *Station* field indicating where to put the components results from the hierarchy of object references in the program. Objects in the model are related to each other on the basis of subordination – one object is the parent node, while the others are its child nodes (*Fig* . 8).

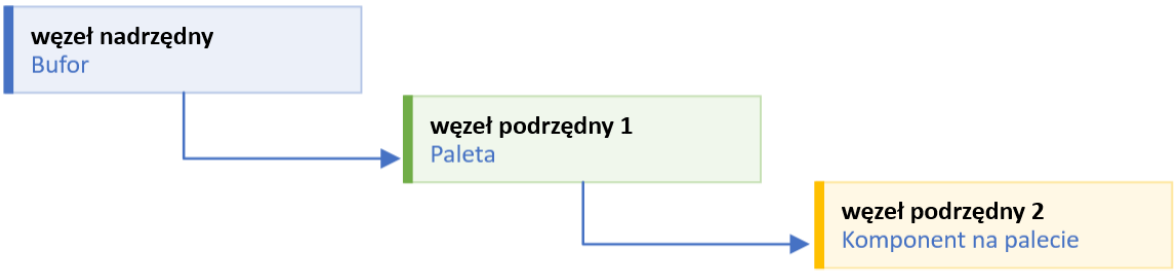


Fig. 8 Hierarchical structure of objects

To access child nodes, use the following notation:

1. **token.buffer.first** – access to the first child node of the buffer.
2. **token**. buffer. last – access to the last child node of the buffer.

In the example considered, there is only one palette in the buffer, so both writes will be equivalent. In the same way, you can access the elements in the palette:

1. **token**. pallet. first – access to the first child element of the palette.
2. **token**. pallet. last - access to the last child element of the palette

To access elements other than the first and last, the subnodes notation is used, specifying the subnode number:

1. **token.palette.subnodes** [2] – access to the second child node of the pallet; this will be the second element loaded onto the pallet.
2. **token**.palette.subnodes.length – the reference specifies the number of child nodes (relative to the pallet); in the analyzed case it will be 8, which is the maximum number of components on the pallet.

In a similar way, from the level of a child object, you can obtain information about the parent object, e.g. from the level of a component element, you can obtain information about the palette it is on and the buffer on which the palette is:

1. **token**. component.up – access to the parent object of the component, e.g. the palette,
2. **token**. component.up.up – access to the parent object two levels above the component, i.e. the buffer.

The next step is to add a *Decide* block , where the operator decides whether to return the full pallet to the shipping zone or continue adding components to it. *Conditional* option *Decide* allows you to choose the port from which the token will leave this block depending on the fulfillment of a specific condition. In this case, it will be checked whether the number of components is equal to the pallet capacity. **Token**.dump.first. subnodes.length notation will be used for this purpose (checking the number of components on the pallet on the dump buffer) and a reference to the label describing the pallet capacity **token**.capacity (fig. 9).

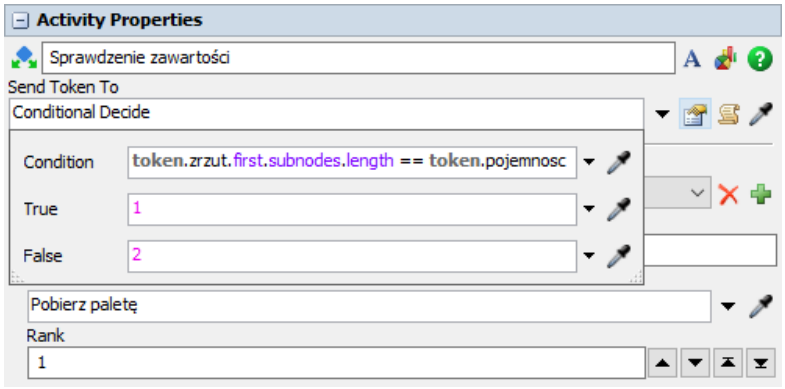


Fig. 9 *Decide* block settings

If the values are equal (the pallet is full, 8 pcs.), the token will leave the block through connector 1. Otherwise, the token will leave the block through connector 2, i.e. the operator is released and waits for the next tasks (Fig.10).

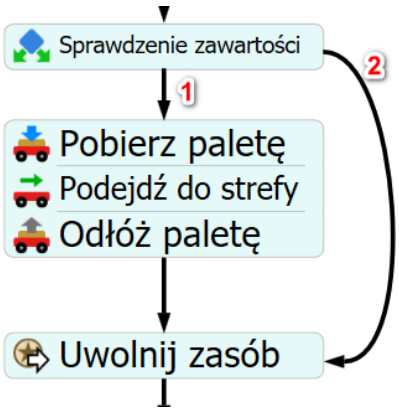


Fig. 10. *Decide* block operation

2.1.2.3. Task sequence - transporting a pallet to the shipping zone

Moving the pallet from the *zrzut* buffer to the *strefaWysylki* will be done using the *Load, Travel, Unload* block sequence (fig. 11).

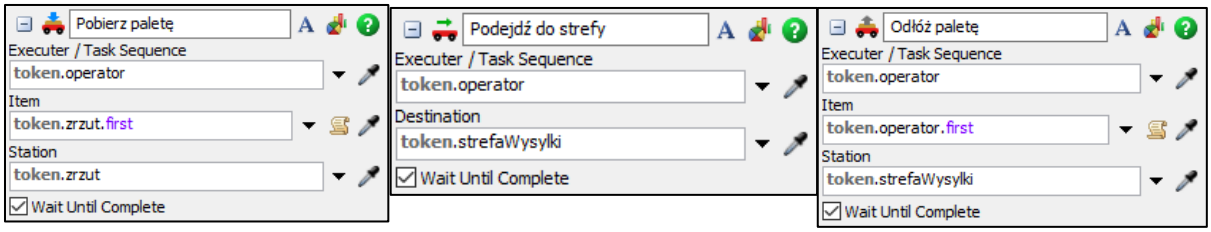


Fig. 11 Pallet transport sequence to *strefaWysylki* buffer

In the *Load* block , the element to be picked up is indicated by the formula **token**.zrzut.first , i.e. the first pallet on this buffer. The operator then goes to **token**.strefaWysyłki . In the *Unload block*, the element to be put away is indicated by the formula **token**.operator.first . This results from the fact that the pallet picked up by the operator changed its parent object from the *zrzut* buffer to the operator (Fig. 12).

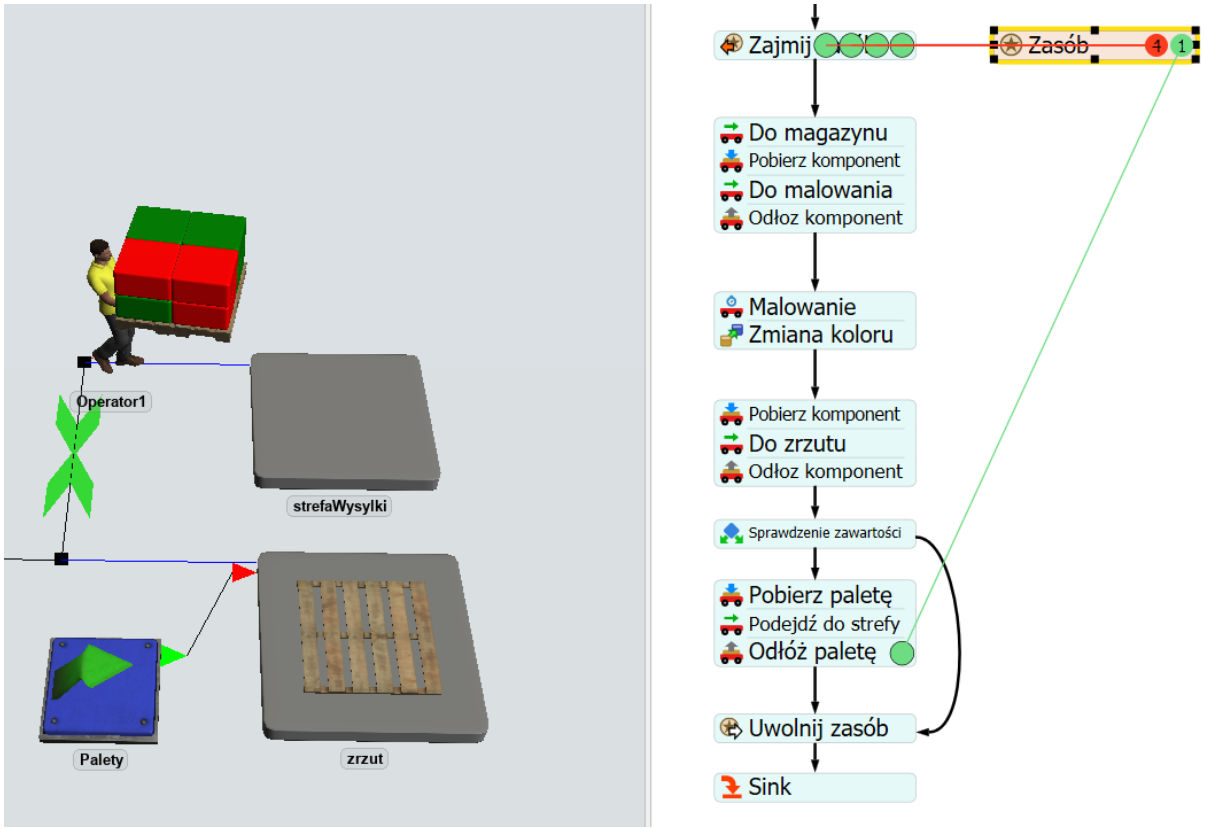


Fig. 12 3D model and *Process Flow*

3. Part 2 – Model Modification and Input Data

The current model will be expanded by adding more *ProcessFlow* blocks corresponding to the following assumptions:

1. Once the pallet arrives at the shipping zone, a second operator unloads it and the components are placed in a buffer before the conveyor belt.
2. The components are placed on the conveyor in batches of three.
3. Empty pallets are placed in a buffer area.

Due to the limitations of the test version of FlexSim regarding the maximum number of elements in both 3D and *ProcessFlow* modes , it will be necessary to modify the previously developed model to introduce new elements. It will be necessary to:

1. Removing the existing *ProcessFlow* and replacing its logic with new 3D model elements (*Palette* element sources and *Combiner*).
2. *Magazynier* operator .
3. Adding a buffer in the 3D model of the *strefaKompletacji* zone and a connected roller conveyor.

To build a new usage model, the existing *Dane* table (Fig. 13) will be used, where the 5th row *WielkoscPartii* has been added. The target model layout is shown in Fig. 14.

Model | Dane

	Wartosc
LiczbaRoznychKomponentow	2
CzasMiedzyKomponentami	30
CzasMalowania	5
PojemnoscPalety	8
WielkoscPartii	3

Fig. 13 Model parameters in the *Data* table

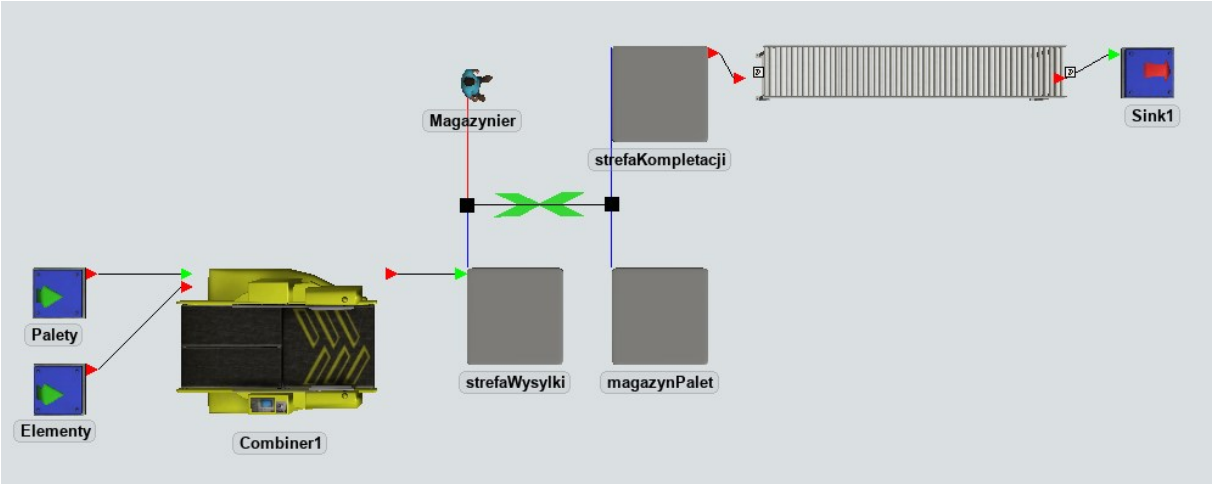


Fig. 14. Target model layout after all changes have been made.

The structure of the new *Process Flow* is shown in Fig. 15. It should be noted that it contains two separate groups of activities responsible for controlling the *operator* work on the buffers *strefaWysyłki* and *magazynPalet* (group 1) and creating a batch of components *(Batch* block) in the *strefaKomplatacji* buffer, which are then sent to the roller conveyor (group 2).

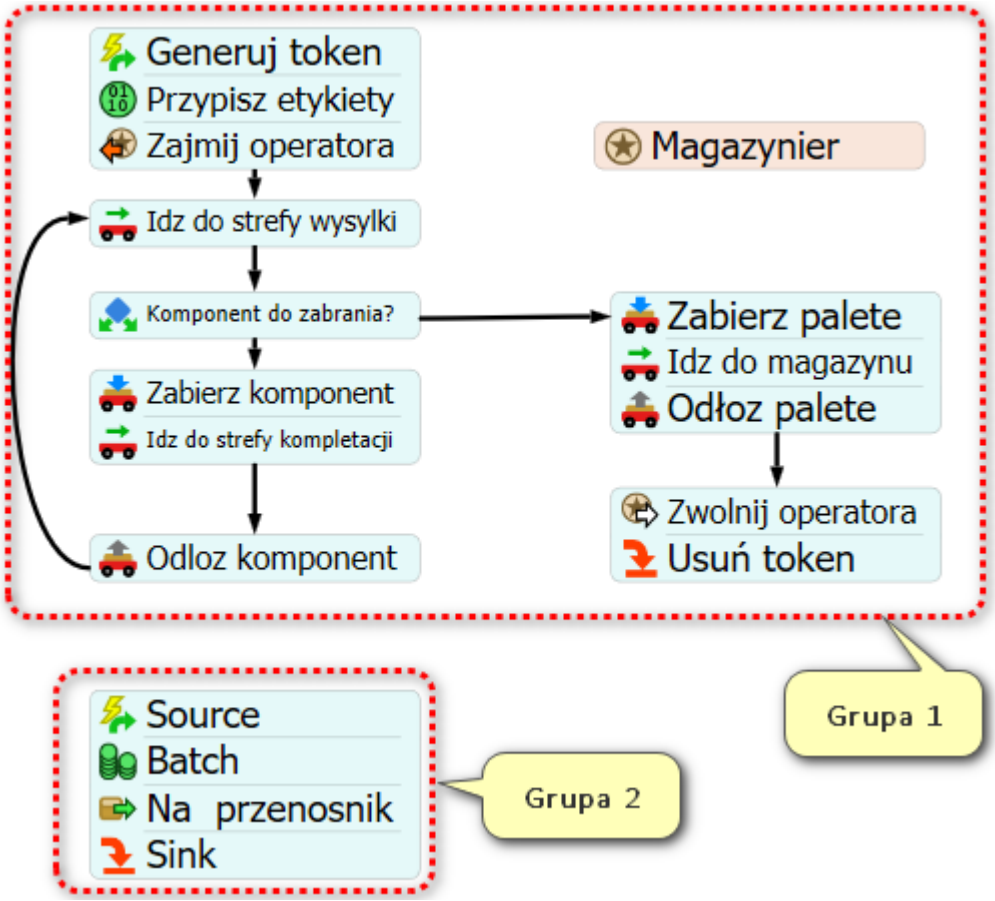


Fig. 15. The final model layout after all changes have been made.

3.1. The structure of the new model

3.1.1. 3D model update

Objects for generating palettes and flow elements in 3D mode should be introduced to the model, replacing the previous *ProcessFlow* blocks. Two sources are used for this purpose: *Palety* and *Elementy*, and *a Combiner*,

which will be used to assemble the elements on the palette. The configuration of the sources is shown in Fig. 165

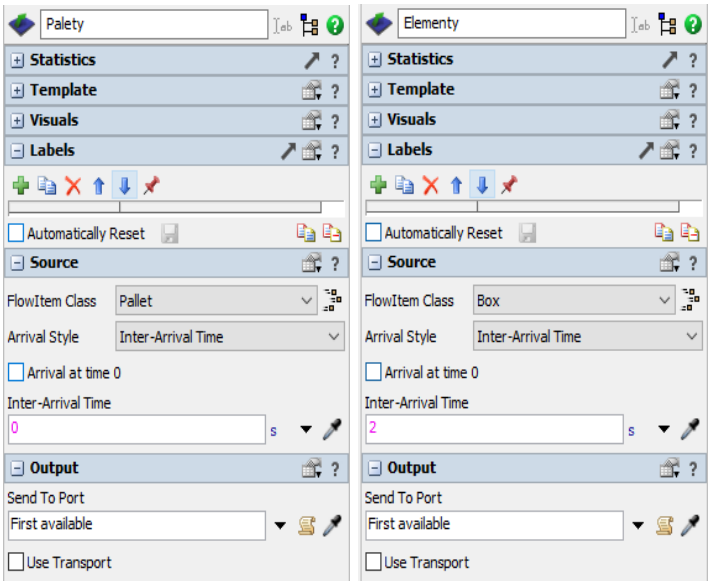


Fig. 16. Configuring new sources in the 3D model

Combiner is responsible for creating pallets with elements (Fig.17). It connects one pallet from input port 1 and 8 elements from input port 2. It is connected by a directional connection to the buffer *strefaWysylki* . The presented actions replace the deleted previous *ProcessFlow* blocks responsible for controlling the flow of flow elements and pallets.

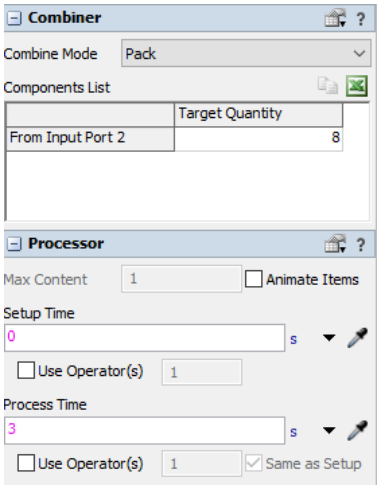


Fig. 17 *Combiner* object

New elements of the model include adding the *magazynPalety* and *strefaKompletacji* buffers, as well as the *Magazynier* operator and two network nodes. It is also necessary to block the possibility of automatic transfer of flow elements to the conveyor by inserting "-1" in the *Output* field of the *strefaKompetacji* zone buffer (fig. 18).

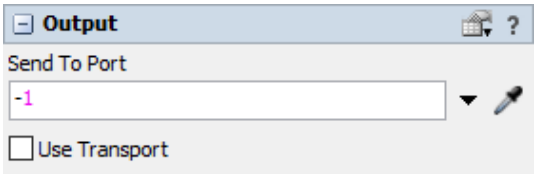


Fig.18. *Output* filed of the *strefaKompletacji* buffer

3.1.2. Model Control – *Process Flow*

3.1.2.1. Generating tokens

Controlling the flow between new model elements requires generating tokens responsible for the execution of individual tasks. *The Event- Triggered Source* block will be responsible for this, creating tokens after a specific event occurs on a specified object (Fig. 19).

After inserting the block, you need to indicate the object *(strefaWysylki)* with a red exclamation mark and indicate the event that initiates the appearance of tokens *(Event – On Entry)*.

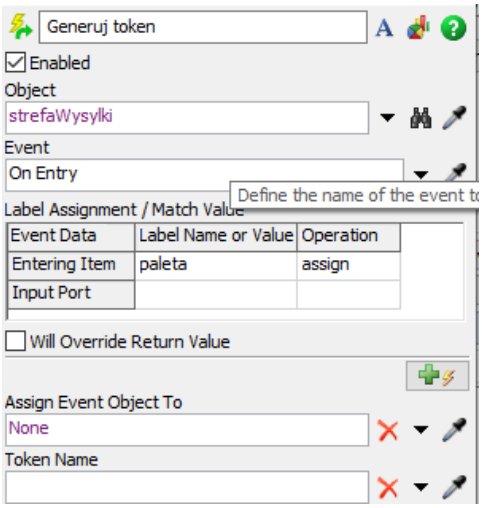
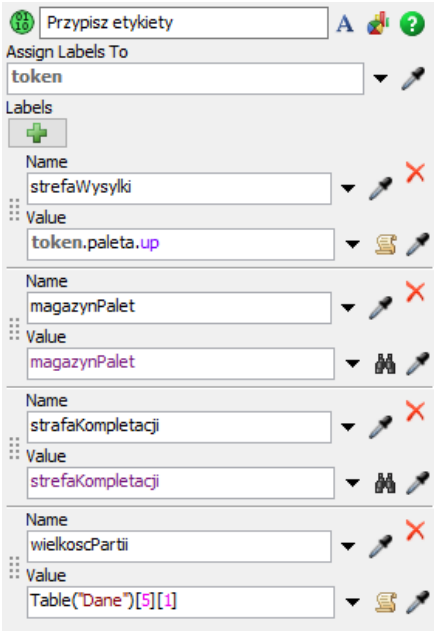


Fig. 19. *Generuj token* block

An additional reference to the flow element causing the token to be generated can be saved under the label in *the Label section Assignemt / Match Values*. The label name *paleta* is added with the *assign operation* .

3.1.2.2. Adding labels

New labels will be added to the generated token *(Przypisz etykiety)* (Fig. 20). It should be noted that the shipping zone will be defined as a parent object to the pallet, hence the notation **token**.pallet.up. The locations to which the flow elements will be moved are defined as references to specific objects from model 3 D. *WielkoscPartii* is referenced to the *Dane* table.



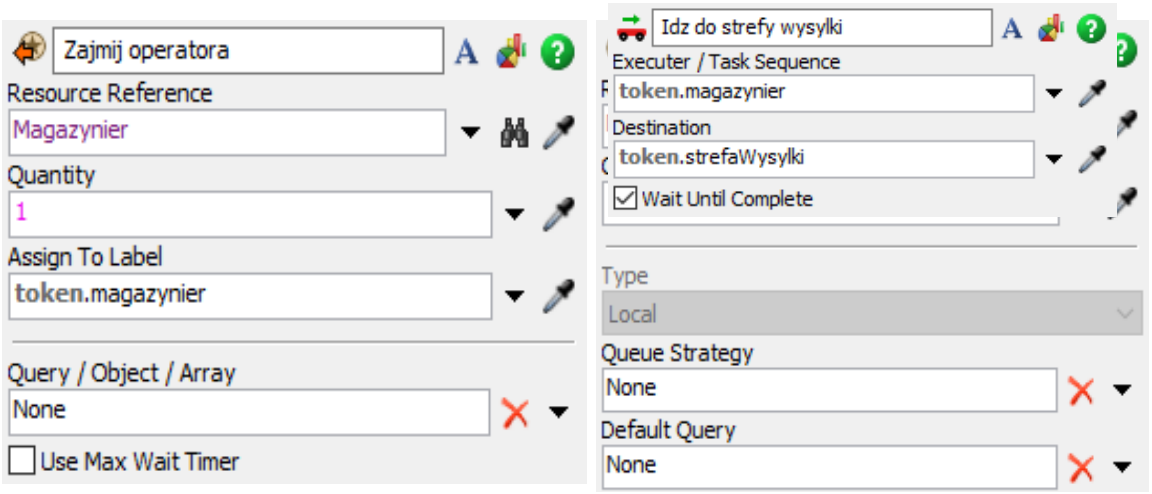
Assign labels block

3.1.3. Warehouseman – *Resource*

The flow is handled by an operator named *Magazynier* . It is added in the *Resource* block, which is referenced by the *Acquire Resource* block (Fig. 21). In the *Resource* block , a reference to the 3D object is created, while

in the *Acquire Resource,* the reference is under the value of the magazynier label at the time of the seizure of this resource by the token.

Fig. 21 *Acquire resource* and *magazynier (Resource)* blocks



Next, the warehouse worker approaches the shipping zone (Fig. 22). At this point, it is necessary to make a decision on whether or not to pick up the components.

Fig. 22 *Move To Block*

3.1.4. Decision to download a component

The *Decide* block is added with the *Conditional Decide* option (Fig. 23). The condition being checked concerns the number of components on the pallet - token.pallet.subnodes.length > 0. If the condition is met (there are components on the pallet) the token will be sent via connector #1, otherwise via connector #2.

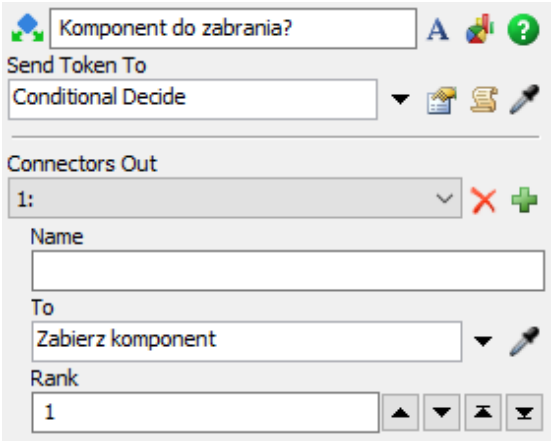


Fig. 23 Decide Block – *Komponent do zabrania?*

3.1.5. Task sequence – from shipping zone to picking zone

The structure of the model up to the moment the element is placed in the picking zone is shown in Fig. 24. The system will be expanded to include the final tasks related to transferring the elements to the roller conveyor.

Fig. 24 *Process Flow* – tasks until the components are placed in the picking zone

The activities concern two tasks: moving an item from the shipping zone to the picking zone and moving an empty pallet from the shipping zone to the pallet warehouse. They are accomplished using the same sequence of *Load, Travel*, and *Unload* blocks .

If the condition specified in the *Decide* block is met (there are components on the pallet), the operator takes the last one that was there - reference **token.paleta**.last (fig. 25).

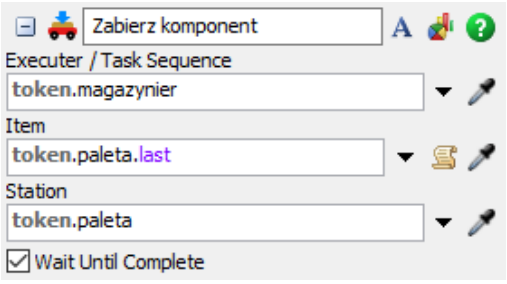


Fig. 25 *Load Block*

The warehouseman goes to the picking zone (Fig. 26), and then puts the component on the *strefaKompletacji* buffer *(*Fig. 27). It should be noted that the parent of the transferred component changes – the reference to it has the form **token.magazynier**.last.

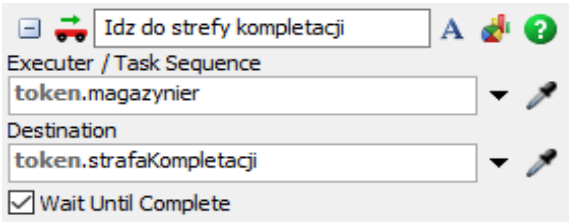


Fig. 26 *Travel b*lock

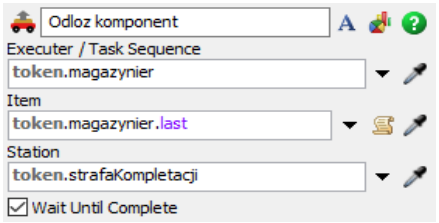


Fig. 27 Unload block

After performing these actions, it is necessary to check whether there are any components left on the pallet in the *strafaWysylki* buffer. This is done by calling back from the *Odloz komponent* block to *Idz to strefy wysyłki*, from which the token will go to the *Decide* decision block again. If the pallet is empty, the token will go to the *Zabierz palete* block via connector #2 .

In this situation, the warehouse worker takes the pallet from the shipping zone, goes to the pallet warehouse and puts the pallet down (Fig. 28).

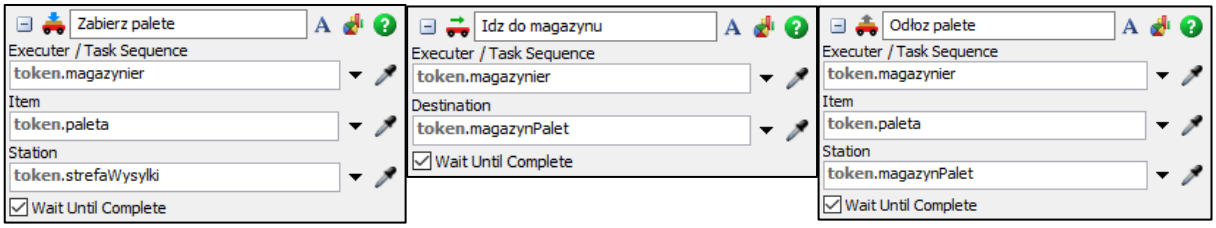


Fig. 28. Sequence of actions when putting down the pallet

After completing these activities, the operator can be released and the token removed from the model (Fig. 29).

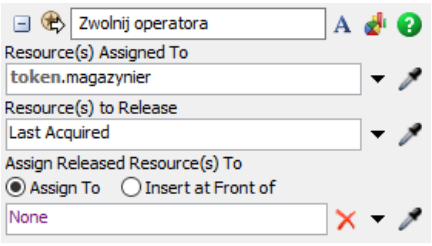


Fig. 29 *Release resource* block

3.1.6. Task sequence – from picking zone to conveyor

New *ProcessFlow* layout must be created to cover these operation. It can be added in the same tab as the existing layout or a separate tab can be created for it.

The *Event- Triggered Source* block (Fig. 30) will be responsible for **generating tokens**. They will be created each time a component hits the *strefaKompletacji* buffer. The reference to the flow element that triggers the token generation is located under the label named component. You can also save a reference to the object in which this event occurred - under the label buffer there will be an indication of *the strefaKompletacji* buffer.

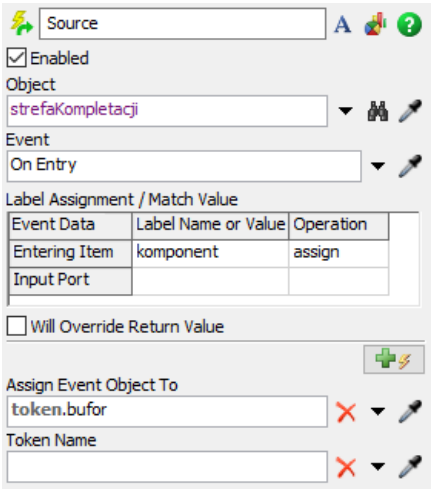
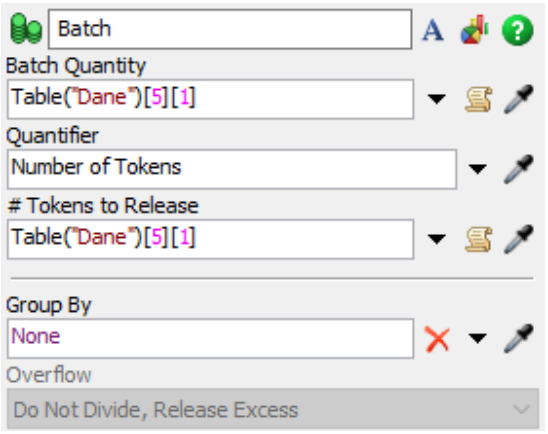


Fig. 30 Event-triggered source block

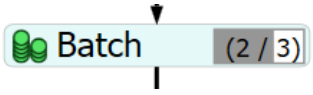
In the next step, you need to create **a batch of 3 components** that will be sent to the conveyor belt. The *Batch* block is used for this purpose (Fig. 31), generating batches of tokens - this is not the same as generating batches of products (components). In the *Batch Quantity* field the number of tokens that should



go into this block is specified so that a given batch of them can be released (*#Tokens to release field)* . In this case, both values will match.

Fig. 31 *Batch* block settings

After running the simulation, you can see



that *Batch* collects tokens according to their given target number (Fig. 32).

Fig. 32. *Batch* block operation

The last step in the model is to move the components onto the conveyor. The *Move Object* block is used for this (Fig.33). The component is moved onto the object to which the buffer is connected by a direction port (**token**.bufor.outobjects [1]). Finally, *ProcessFlow* responsible for moving the elements from the picking zone to the conveyor will have a layout as in Fig. 34.

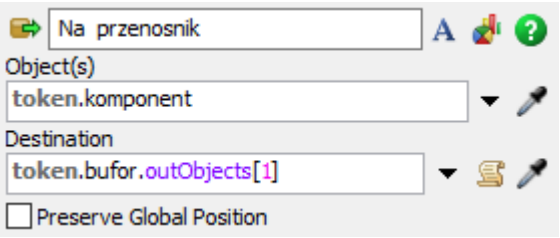


Fig. 33 *Move Object* block

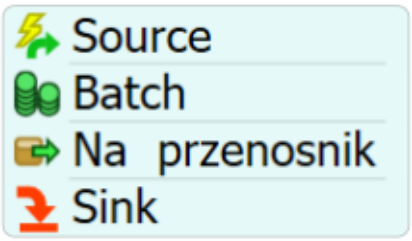


Fig. 34 *ProcessFlow* to the conveyor