AdTrans 2023-1-PL01-KA220-HED-000158917

POLITECHNIKA GDAŃSKA

INTELLIGENT LOGISTICS SYSTEMS

ACTIVITY #5

Using lists in models

dr hab. Daniel Kaszubowski, prof. PG

Department of Transport 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 master the skills of creating and using lists with data in models. The material is divided into two parts. The first part contains an introduction to building and creating lists and adding items to them during simulation. The second part is an example model in which the lists will be used to store information about defective products, initiating the operation of other elements of the model.

New skills
Creating lists, adding elements to them that illustrate flows in the model
Retrieving information from lists
Saving items to lists according to selected criteria
Using a gantry crane to manipulate flow elements

2. Introduction to the use of lists

Lists in the model are dynamically updated tables that store various types of information, e.g. about flow elements. They allow you to add another flow control element in models. Lists are added in *Toolbox → Global List → Item List.* The default list created is shown in Fig. 1. Information is stored in subsequent columns of the table. Fields visible in the *Fields* section indicate subsequent headers of these columns. Values can be updated dynamically after selecting the *Dynamic* option next to the field name.

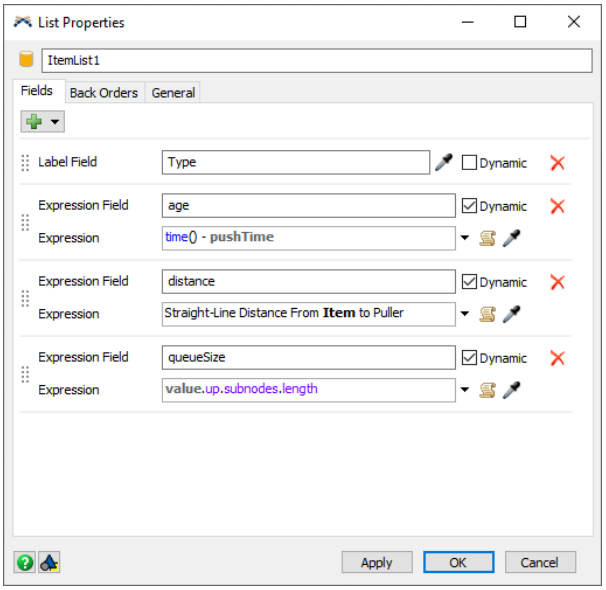


Fig. 1 List Properties

There are two types of fields in the table shown:

1. ***Label field*** – store the value of labels with the given names.
2. ***Expression field*** – store the results of given operations.

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

Products list will be created in the model with the settings as in Figure 2.

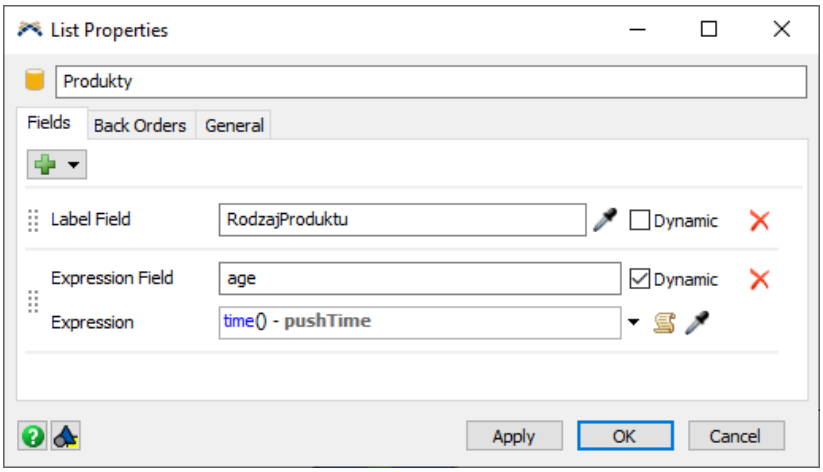


Fig. 2. *Products* list settings

To test the operation of the lists in practice, the following activities will be performed:

1. Adding a numeric label ProductType to the *Box* element in the *FlowItemBin* section .
2. Adding a source and buffer to the model, connected directionally.
3. Adding a trigger to assign a value to the ProductType label each time a flow element is generated (Fig. 3).

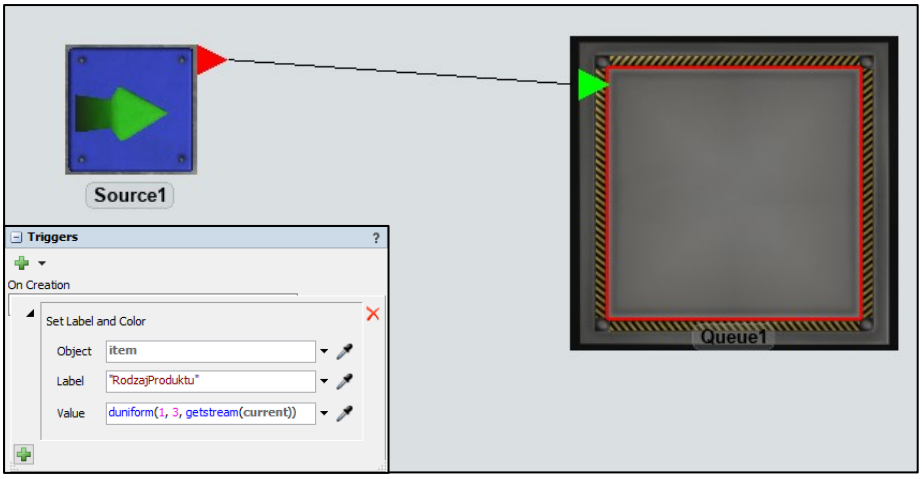
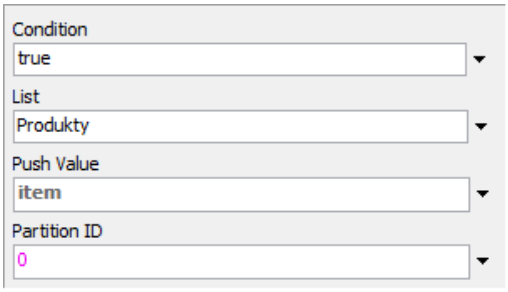


Fig. 3. Adding label at *Queue1*

Information about generated products will be saved in the created list. The model assumes that this action will be performed when a flow element enters the buffer after the source. To do this, select *Triggers* → *On Entry* → *Lists* → *Push to List.*



Rys. 4 Push to List

A window will appear as in Fig. 4, with the following fields:

1. ***Condition*** – condition that must be met to enter a given item into the list. The default value *true* means that all items will be entered.
2. ***List*** – the name of the list where the selected items will be entered – must be selected from the drop-down list (here – *Products*).
3. ***Push value*** – the item that will be entered into the list. By default, it is *item* , i.e. a flow element.
4. ***Partition ID*** – ability to define subgroups on the created list. The default value of "0" means no subgroups.

The list content can be viewed in *Toolbox* → *Global Lists* → *Products* → *View Entries* (right click). The *value* column contains a reference to a specific product (flow element), *ProductType* saves the value of the label with this name, while *age* is the time the element stays on the list. The empty and filled lists are shown in Fig. 5.

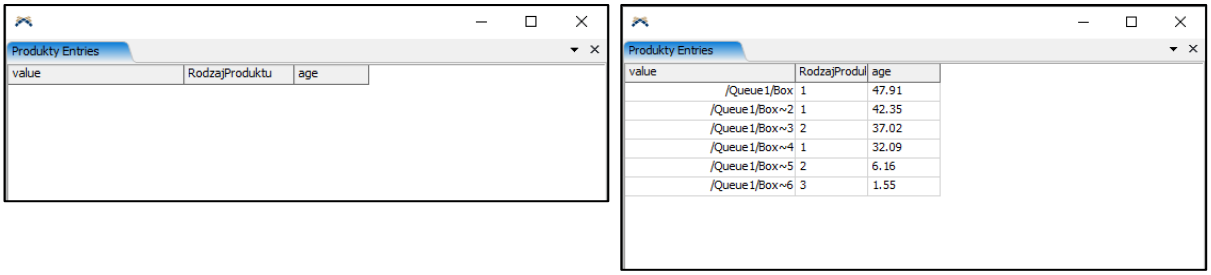


Fig. 5. Contents of the *Products list*

For greater readability of the list, you can modify the setting of the trigger responsible for entering flow elements on the buffer by defining subgroups in it. In *Partition ID,* select **item** ProductType. The result will be grouping the entries on the list according to the type of product stored on this label (Fig. 6).

Produkty Entries

value	RodzajProdul	age
Partition ID: 1		
/Queue1/Box	1	88.25
/Queue1/Box~2	1	82.69
/Queue1/Box~4	1	72.42
/Queue1/Box~9	1	1.55
Partition ID: 2		
/Queue1/Box~3	2	77.35
/Queue1/Box~5	2	46.50
/Queue1/Box~7	2	39.25
0x0	2	0.78
Partition ID: 3		
/Queue1/Box~6	3	41.88
/Queue1/Box~8	3	28.99

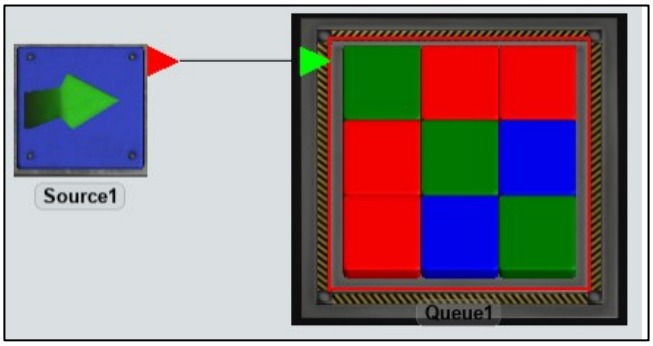


Fig. 6. Grouping entries in the list by ProductType

3. List model

3.1. Assumptions and input data

The model reproduces the transport of products using roller conveyors and the removal of defective products using an overhead crane:

1. The source generates products in 3 seconds. The products are sent to a conveyor, which transports them to the system output.
2. The system consists of two 10 m long straight conveyors and a connecting curved conveyor. The minimum distance between products on the conveyors is 25 cm.
3. Conveyors are subject to failure over time, causing product damage. The probability of failure increases linearly by 20% per 5 minutes of operation.
4. Before leaving the system, each product is checked - if it appears damaged, the entire conveyor system is stopped and the crane removes the defective products from it. After this operation, the system is restarted.

The target layout of the 3D model and the data table is shown in Fig. 7, while Fig. 8 shows two *Process Flow* blocks– *Generator1* (generating failures) and *Generator2* (removing faulty elements from the system).

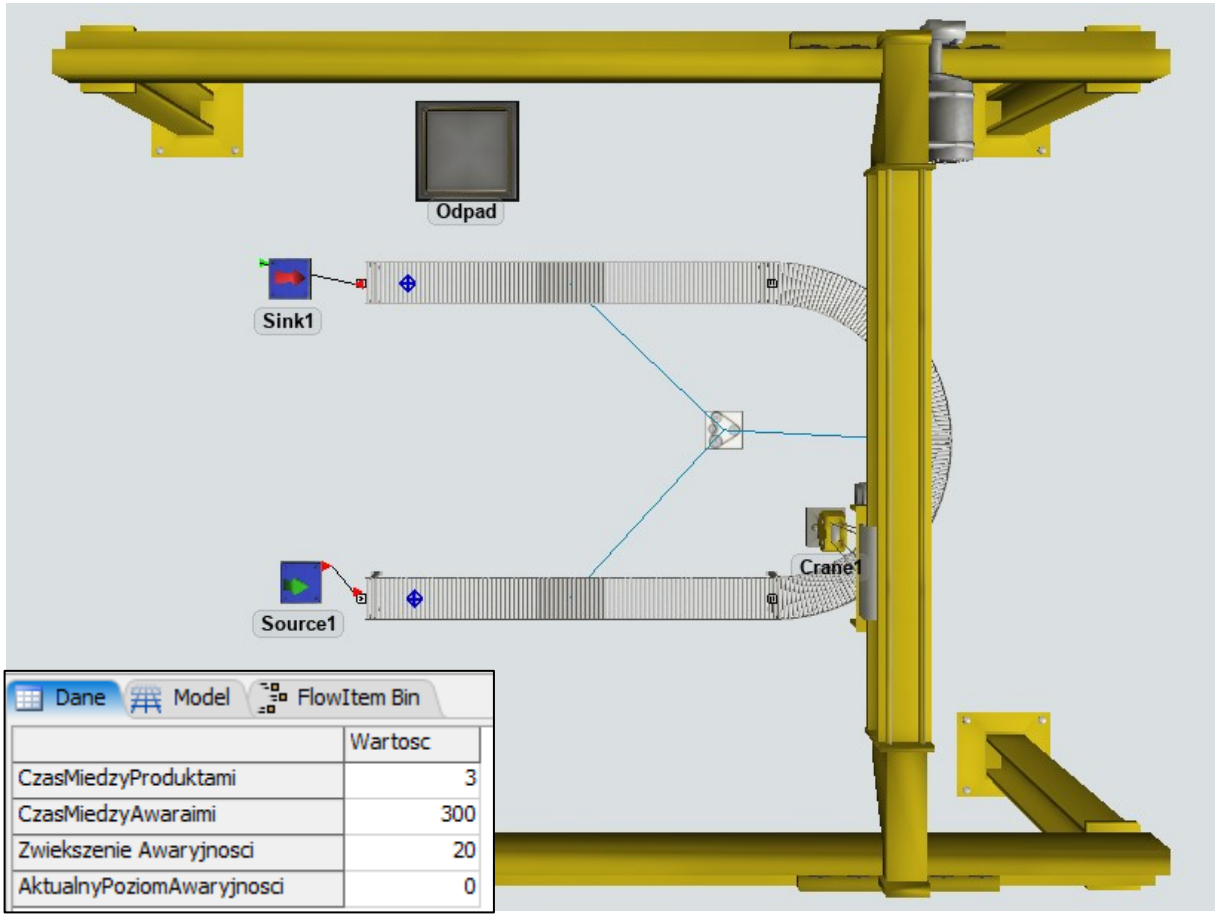


Fig. 7. 3D model elements

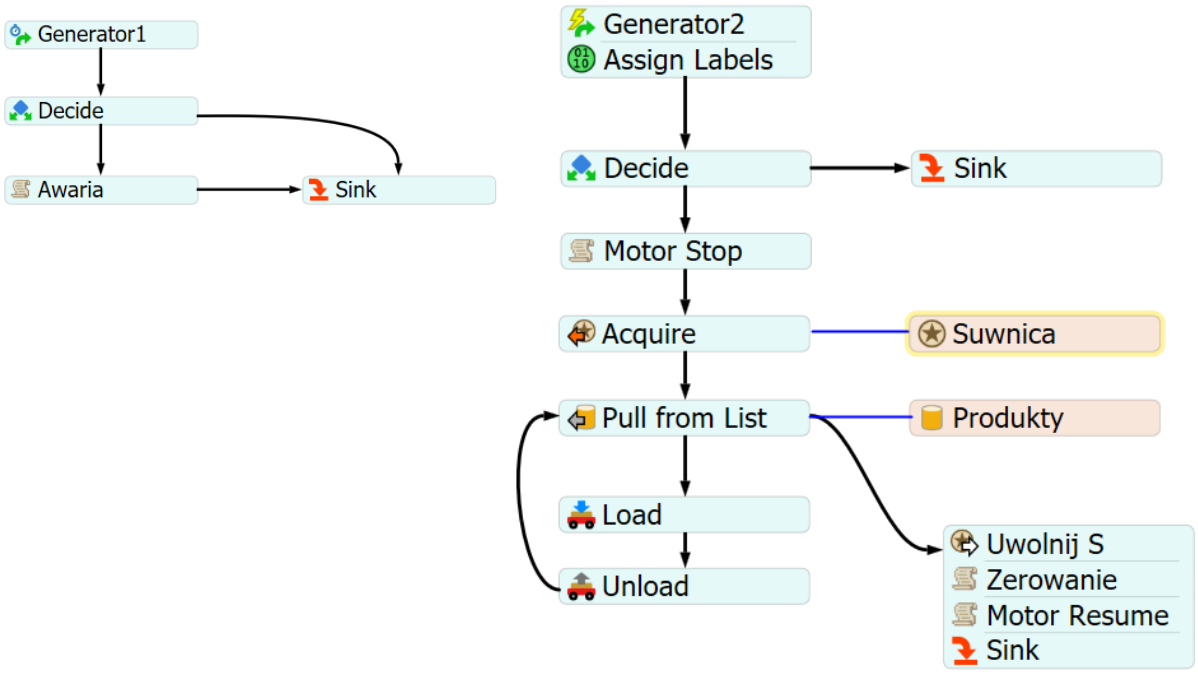
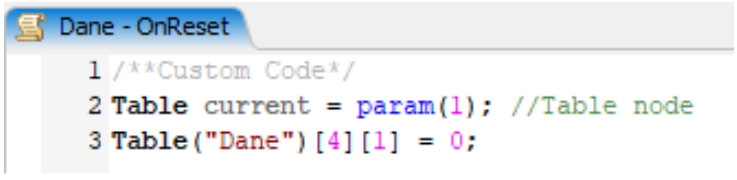


Fig. 8. Structure of two *Process Flow* blocks– *Generator1* and *Generator2*

In the *Data* table, a modification should be made to the fourth row, the failure rate. The current level of this parameter will be updated during the simulation, and its initial value is 0.

It should be possible to reset the failure rate after each simulation run. For this purpose, a *Custom* trigger will be added to the *On Reset* field in the table properties *Code* (Fig. 9): → *Directly edit code for this trigger*. The code in the third line means to enter 0 into cell [4][1]



Rys. 9 Zawartość Custom Code

Additionally, a Correctness (Poprawnosc) label must be added to the flow element. (fig. 10).

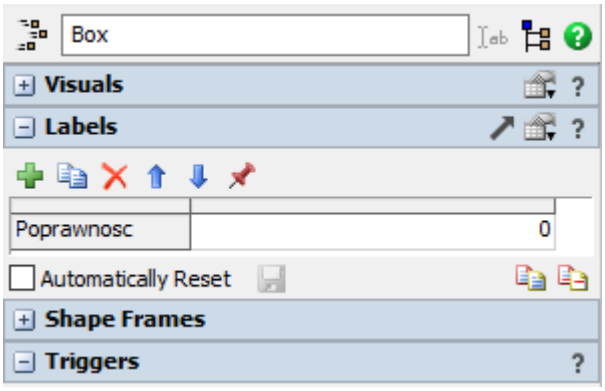
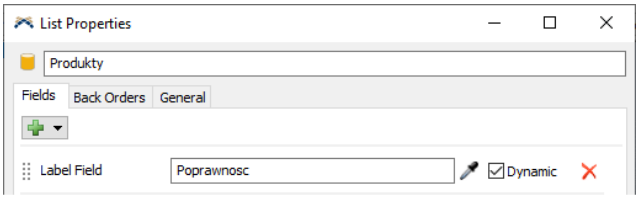


Fig. 10. Label on the flow element

3.2. Flow Element List

Products will be added in *Toolbox* → *Global List* → *General List* (Fig. 11). It will contain information about the elements located on the conveyors. One *Label Field* type field should be added to it, referring to the Correctness label. Selecting the *Dynamic* option will allow the table to be updated during the simulation.



Rys. 11 Tabela Produkty

3.3. Parameterization of model objects

Products are generated by the *Source* at fixed intervals given in the *Data* table and are delivered to the conveyor system. It is controlled by a *Motor* element that is connected directionally to each conveyor (blue lines).

At the beginning of the first conveyor there is a decision point *DP1*. It is responsible for entering products into the *Products* list and assigning a value to the Correctness label and giving the elements the appropriate color. *On Arrival* should be selected → *Data* → *Set Label by Percentage* and enter the settings as in Fig. 12.

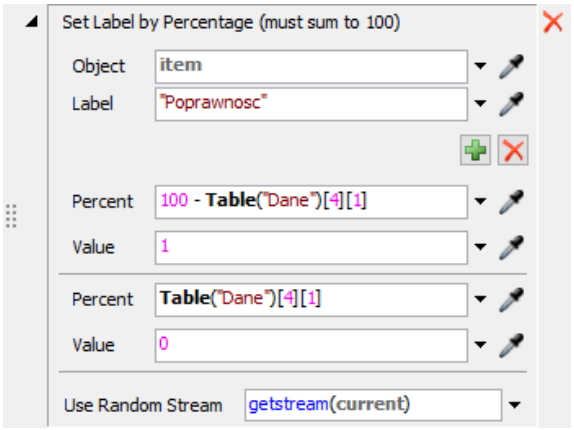


Fig. 13 *Set Label by Percentage* settings

Next, select *Visual* → *Set Color by Case* and enter the settings as in Fig. 13. The flow elements will receive colors according to the value of the Correctness label. Undamaged products with a value of 1 will be red, damaged products with a value of 0 will be black.

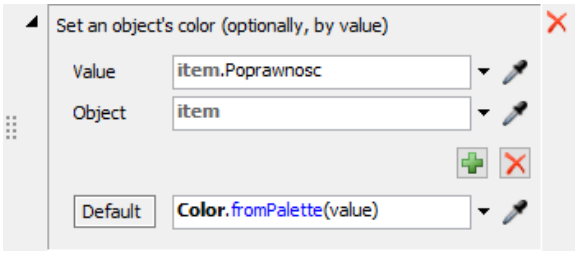


Fig. 13 *Set Color by Case*

The last step is to enter the flow items into the *Products* list by selecting *Lists* → *Push to List* and entering the settings from Fig. 14.

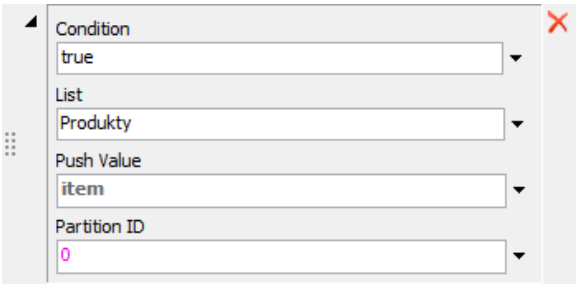


Fig. 14 *Push to List*

According to the assumptions, the conveyors also require some modification. This involves introducing settings that ensure a distance of 25 cm between the moved objects (Fig. 15).

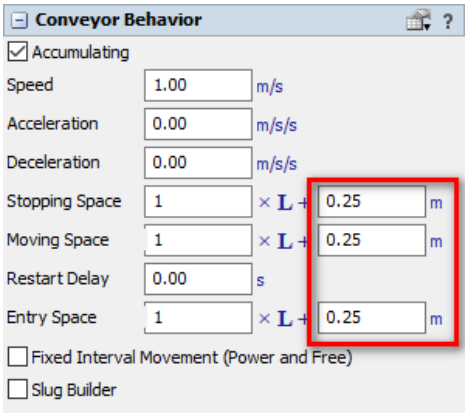


Fig. 15 Conveyor settings

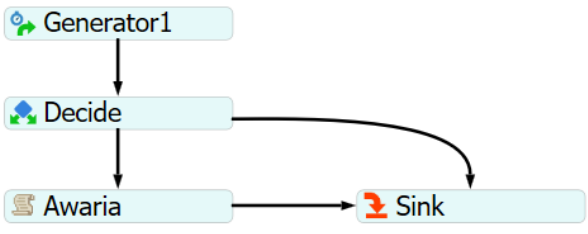
4. Model Control – *Process Flow*

4.1. Failure Generator

The model control logic is composed of two schemes:

1. Diagram responsible for generating failures according to the given parameters *(Geenerator1)*.
2. Diagram responsible for removing faulty elements from the system *(Generator2)*.

The first diagram is shown in Fig. 16. It starts with the *Inter- Arrival Source* block generating tokens that will be responsible for increasing the failure rate (Fig. 17).



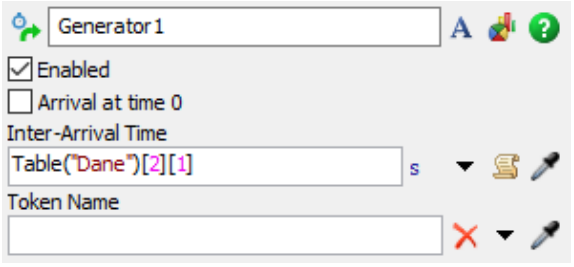
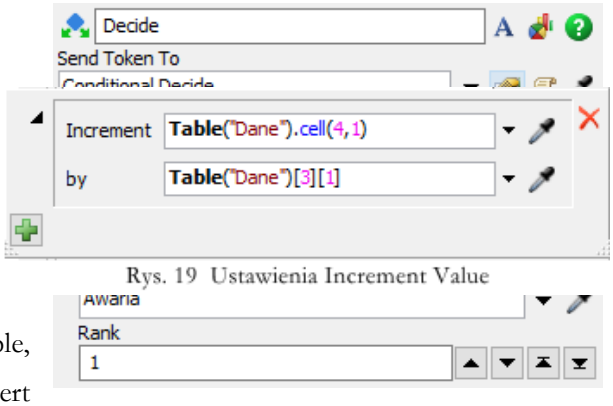


Fig. 16. Fault generation diagram Fig. 17. *Generator1* block

Decide block protects against a situation in which the failure rate would be increased above 100%. The *Send To → Conditional* option is responsible for this *Decide*. After entering the settings according to Fig. 18, if the parameter value is less than 100% it will be redirected to the *Custom Code* block *(Failure),* otherwise it will be deleted from the system.

Fig. 18 *Decide* block



Rys. 19 Ustawienia Increment Value

The Failure block is responsible for increasing the failure rate. From the *Data* group, select *Increment Value* and enter the settings from Fig. 19. The different format of cell references in both fields requires attention. The reason is the need to correctly recognize the type of the entered variable, because in the *Increment* field you can also insert a reference, e.g. to the label value. Therefore, the notation cell (i,j) clearly points to a specific table field.

4.2. Manipulation of defective products

4.2.1. Creating defective products

Process Flow scheme will be responsible for removing defective products from the system, stopping it and restoring it to its initial state. *(* fig. 20 *)* .

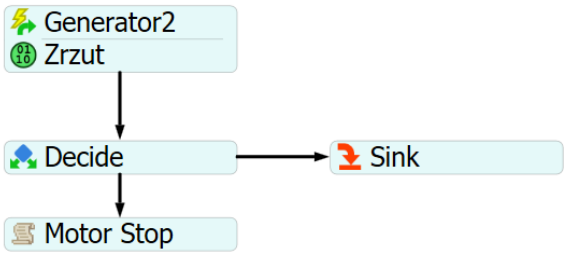


Fig. 20 Process Flow – removing defective products

The initial block will be the *Event- Triggered Source* (Fig. 21), generating a token each time a flow element appears at decision point *DP2*. The reference to the flow element generating the token is contained in the product label .

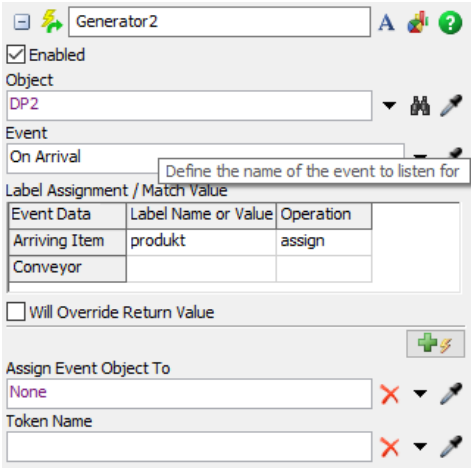


Fig. 21 Event - Triggered Source – *Generator2*

The token goes to the *Assign Labels* block)(Fig. 22). The label dump is added there, referring to the queue named *Waste (Odpad).*

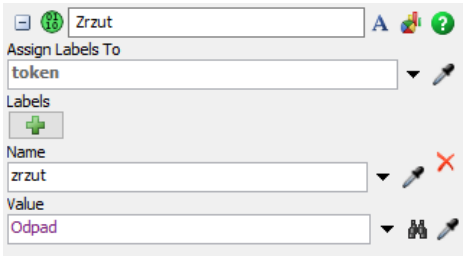


Fig. 22 *Dump* block

The next task is to check if the element on *DP2* is faulty. This is done using the decision block and the *Conditional Decide* option. If the value of the Correctness label is equal to 0 (faulty product), the token goes with output 1 to the next block *(Custom Code – Motor Stop)* stopping the conveyors. Otherwise it is removed to the *Sink* block (fig. 23).

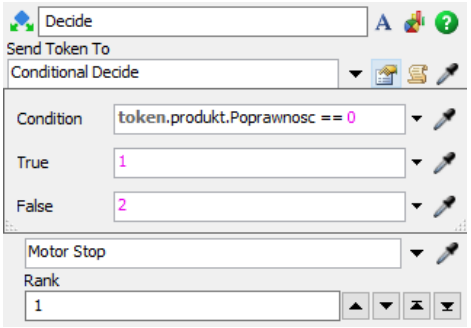


Fig. 23 *Decide Block*

The Motor Stop block should be configured by entering the settings in *the Control → Stop Object* option as shown in Fig. 24, inserting a reference to the *Motor1* object from the model using the picker.

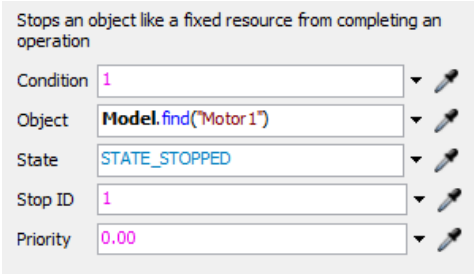


Fig. 24 *Stop Object* Settings

4.2.2. Removal of defective products

After entering all settings and starting the simulation, defective products (black) that appear after some time cause the conveyors to stop. To remove them using a crane, use the previously created product list.

The first task is to set the resource (crane) seizure mechanism according to Fig. 25.

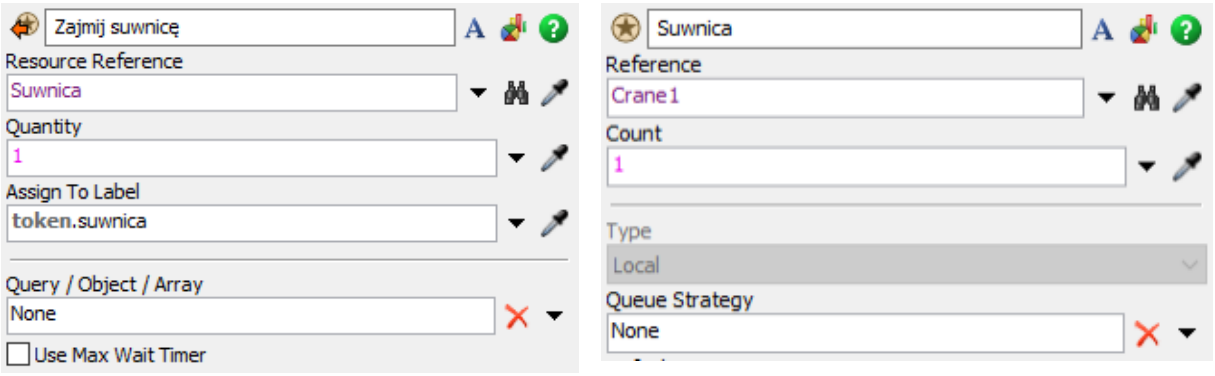
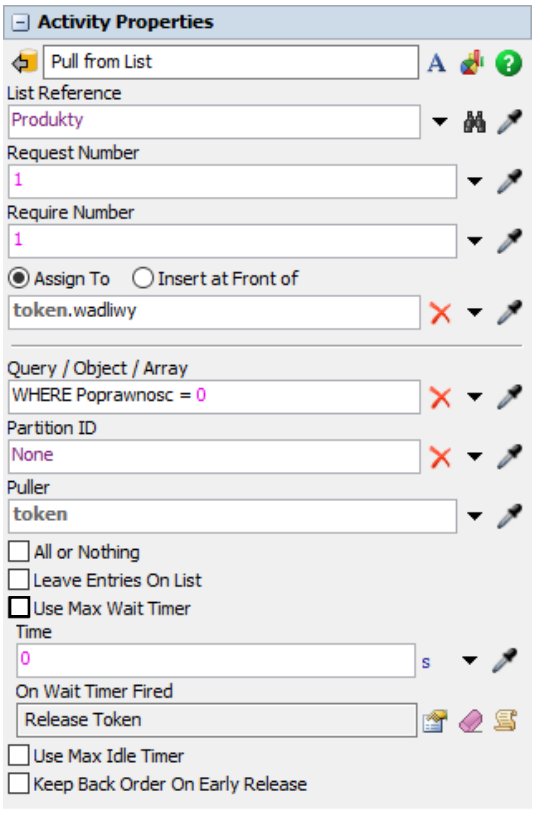


Fig. 25 Resource utilization – overhead crane

Then, the blocks responsible for searching for defective products are added in the existing *Products* list (Fig. 26). In the *Products* block, a reference to the *Products* list should be selected from the *Global List* .

Fig. 26 Scrolling through the *Products list*

Pull from List settings are shown in Figure 27. The *List Reference* automatically sets the reference to the linked list (here *Products*/*Produkty*). The remaining fields indicate:



Rys. 27 Opcje Pull from List

1. ***Request Number*** – the number of items to retrieve from the list.
2. ***Require Number*** – the number of items required to be retrieved from the list, cannot be greater than *Request Number* .
3. ***Assign To*** – assigning a label to a token with reference to an item from the list.
4. ***Insert at Front of*** – as above, but values are added as subsequent elements of the variable.
5. ***Query/Object/ Array*** – ability to specify restrictions on retrieving items from the list.
6. ***Partition ID*** – indicates the partition from which items are retrieved (requires the use of partitions in lists).
7. ***Puller*** – an element that retrieves items from a list, most often a token .
8. ***All for Nothing*** – items will be taken from the list only if there are enough of them, equal to *Requrie* Number.
9. ***Leave Entries on List*** – after downloading items from the list, they will be visible on the list at all times. Used e.g. for permanent resources used by many users.

10. ***Use Max Wait Timer*** – specifies the actions to be taken if the required item is not downloaded within the defined time.
11. ***Use Max Idle Timer*** - works like in *Max Wait Timer* , the difference is measuring the idle time of the token in this block.
12. ***Keep Back Order On Early Release*** – if the token does not fetch the required item, it will be released further (e.g. via *Max Wait Timer*) and the information about the uncollected item will be remembered and supplemented in the further elements of the process.

In the case under consideration, items for which the label value Correctness = 0 should be retrieved from the list. This condition is entered in the *Query/Object/Array field.* A reference to the retrieved item will be saved under the label defective.

The loading and unloading of products by the crane from the conveyor is carried out using standard *Load* and *Unload* blocks, the settings of which are shown in Table 1.

Table 1. *Load* and *Unload* block settings

Block	Block name	Settings box	Value
Load	*Download*	*Execute / Task Sequence*	token. gantry
		Item	token. defective
		Station	token. defective. up
Unload	*Put it aside*	*Execute / Task Sequence*	token. gantry
		Item	token. defective
		Station	token.drop

After unloading, it is necessary to check whether there are other defective products on the list. For this purpose, a feedback from *Put away* to *Pull from List* is added (Fig. 28).

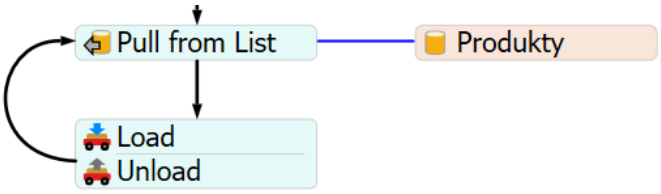


Fig. 28 Loop for checking the availability of defective products

After running the simulation, you can see that at some point all defective products are removed from the list, which results in blocking the token on the *Pull from List block* (Fig. 29). Additionally, the conveyor system

has been stopped and no new products are being delivered – so there is no possibility of entering them on the list.

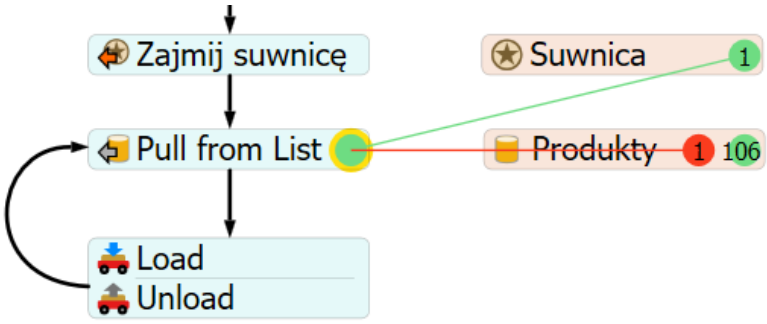


Fig. 29 Blocking the token when downloading from the *Products* list

In this situation, you should:

1. Release the token from the *Pull from List* block .
2. Start the engine responsible for the operation of the conveyors and resetting their failure rate.

In the *Pull from List settings,* check *Use Max Wait Timer* , which allows you to define an action in a situation when the appropriate item has not been picked from the list. After selecting it, you should:

1. Remove the *Set Label* function .
2. Modify *Release Token* by entering value 2 in the *Destination* field, indicating a new connector to which the token will go that failed to pick up an item from the list. It directs to a new block that releases the previously occupied resource (crane), resets the failure parameter value to zero, and resumes the operation of the conveyors.

Creating a new block on connector 2 involves adding the following blocks:

1. *Free crane/Uwolnij S(Release Resource):* *Resource(s) Assigned To* → **token**. crane .
2. *Reset/Zerowanie (Custom Code):* *Data* → *Write to Global Table* → settings Fig. 30.
3. *Motor Resume (Custom Code):* *Control* → *Resume Object* → settings fig . 30.
4. *Sink* .

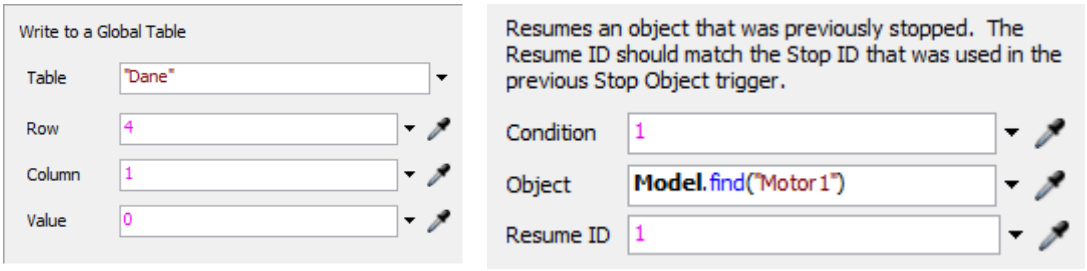


Fig. 30 Settings in *the Zeroing* and *Motor Resume blocks*

After adding all the elements, the entire *Generator2* section looks like Figure 31.

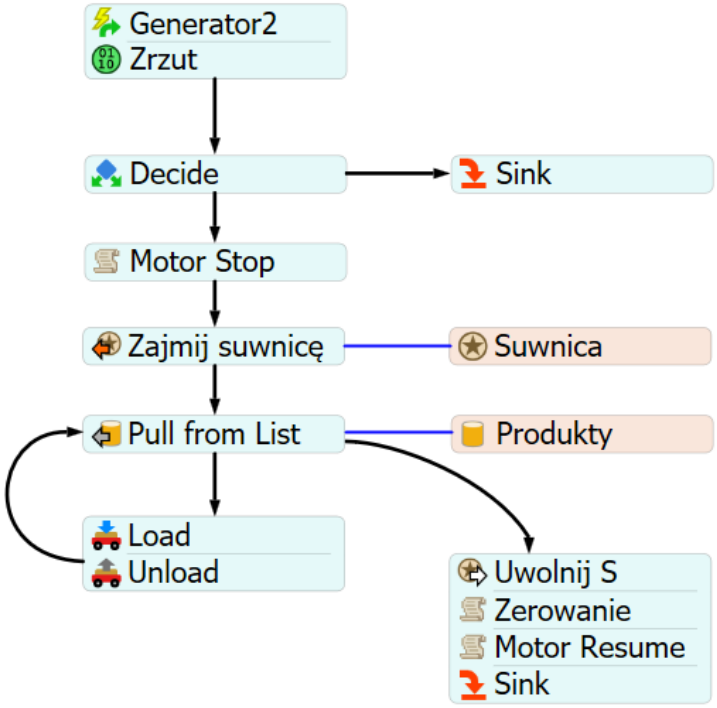


Fig. 31 Complete *Generator2 section*

4.2.3. Options for entering and downloading products from the list

After starting the simulation, the products on the conveyors are recorded in the table (Fig. 32, left view). If some of the correct products leave the system, new items will appear in the table (right view). This means that the deleted products have also been removed from the list. Their place on the list has been taken by entries in the form 0x0. This can make it difficult to analyze the data in the model, so there are two possible solutions:

1. Entering only the items that are to be downloaded into the created lists.
2. Defnitrly remove an item from the list before removing the related flow element from the model.

Produkty Entries

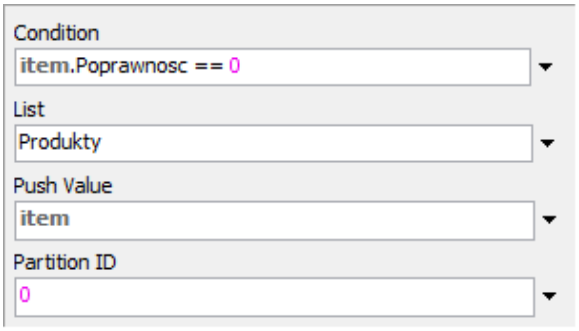
value	Poprawnosc
/StraightConveyor2/Box	1
/StraightConveyor2/Box~2	1
/StraightConveyor2/Box~3	1
/CurvedConveyor1/Box	1
/CurvedConveyor1/Box~2	1
/CurvedConveyor1/Box~3	1
/CurvedConveyor1/Box~4	1
/StraightConveyor1/Box	1
/StraightConveyor1/Box~2	1
/StraightConveyor1/Box~3	1

Produkty Entries

value	Poprawnosc
0x0	
0x0	
0x0	
0x0	
0x0	
0x0	
0x0	
/StraightConveyor2/Box	1
/StraightConveyor2/Box~2	1
/StraightConveyor2/Box~3	1
/CurvedConveyor1/Box	1
/CurvedConveyor1/Box~2	1
/CurvedConveyor1/Box~3	1
/CurvedConveyor1/Box~4	1
/StraightConveyor1/Box	1
/StraightConveyor1/Box~2	1
/StraightConveyor1/Box~3	1

Fig. 32. Changes in the *Products* table during simulation

The first option requires modifying the *On Arrival* trigger in *DP1*. *The Push to List* action should be changed by specifying in the *Condition* field condition as in Fig. 33. Only items for which the value of the Correctness label equal to 0 will be entered into the list will be.

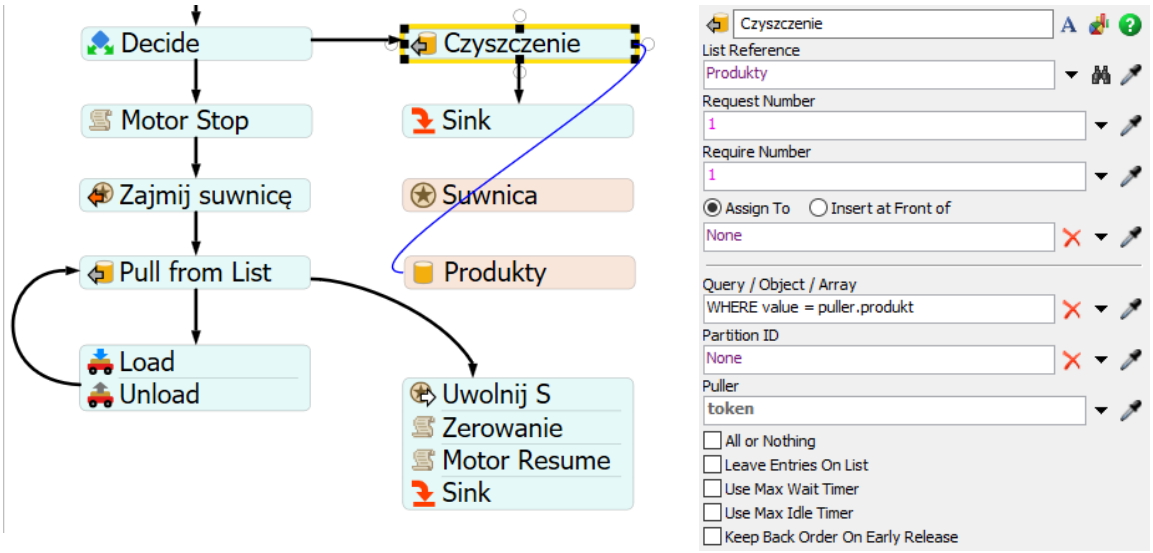


Rys. 33 Modyfikacja Push to List

At the same time, since only information about defective products is saved, you can skip specifying the download condition *(None)* in the *Query* field in the *Pull from List* block.

The second solution, which involves removing from the list the item associated with the removed flow element, requires adding another *Pull from List (Removing/Czyszczenie)* block to the *Process Flow* as shown in Figure 34. In the *Query* field, enter the condition WHERE value = puller.product. The reference to the flow element in the list is always stored in the column with the header value. In the model, this reference must match the reference to the product, saved as **token.product**.

The *Query* field uses SQL syntax that does not recognize the **token** keyword used in the FlexSim programming language. Therefore, the option available in the *Puller* field was used, where **token** was indicated as the element taking from the list. Hence, instead of WHERE value = token.product WHERE value = puller.product is used .



Cleaning block and its settings