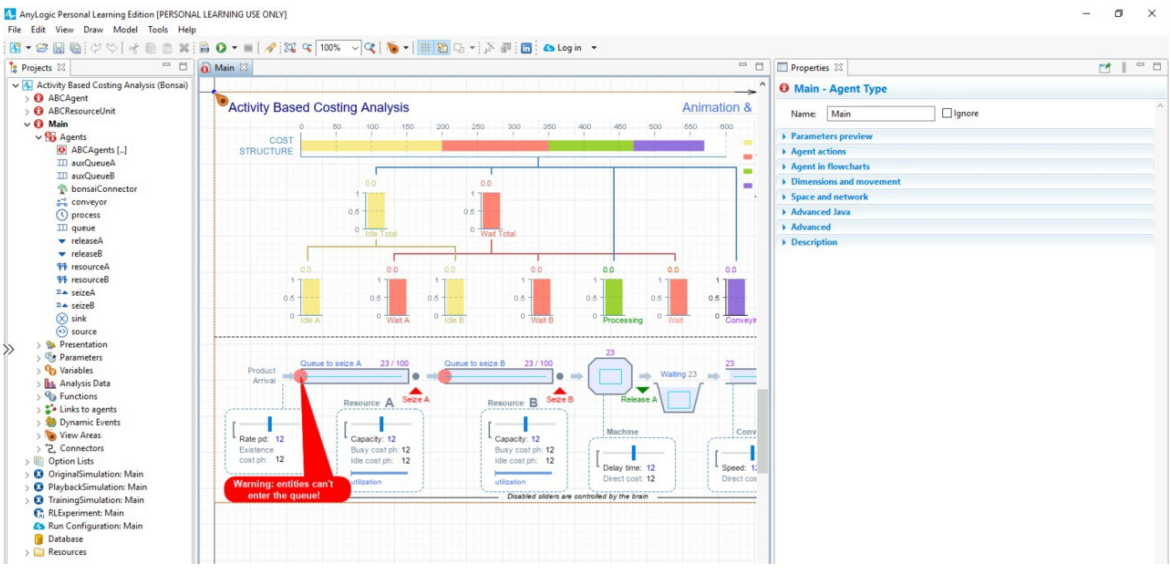
Model overview: activity-based costing analysis for factory logistics in AnyLogic

6/15/2022 • 14 minutes to read

> **NOTE**
>
> The ABCA model is used by the Factory logistics sample.

Activity-based costing analysis (ABCA) is a factory logistics control mechanic that optimizes output while reducing operational costs. Activity-based costing analysis depends on accurate monitoring of factory activities and tracing of resource consumption in addition to the typical costing analysis of final output.

The AnyLogic simulator included in the Factory Logistics Accelerator implements a discrete event simulation that models a simplified factory floor. The model focuses on high-level processes and abstracts away some real-world details to support flexibility and customization so that it can be applied to similar manufacturing processes across varied industries.



agent.

How the ABCA model works

The model traces product lot movements as the lots arrive, pass through machine and conveyer operations, then exit the system:

- **product lot**: a discrete unit of goods created in the factory being modeled.
- **system**: a finite portion of the factory being modeled.
- **resource pool**: a finite pool of inputs required to move a product lot through the system, which are seized (claimed) when a product lot enters the system and potentially released when the product lot leaves the system. Resource pools have explicit, quantified capacities.
- **machine**: a generic representation of factory equipment used to process an incoming product lot. Machines have an associated queue length (how many lots are waiting for the machine) and throughput (how lots can be processed simultaneously). Machines are constrained by the number of available resources.

- **conveyer**: a generic representation of factory infrastructure used to move a product lot through the system.
- **cost object**: a representation of the costs associated with using or consuming a resource unit.

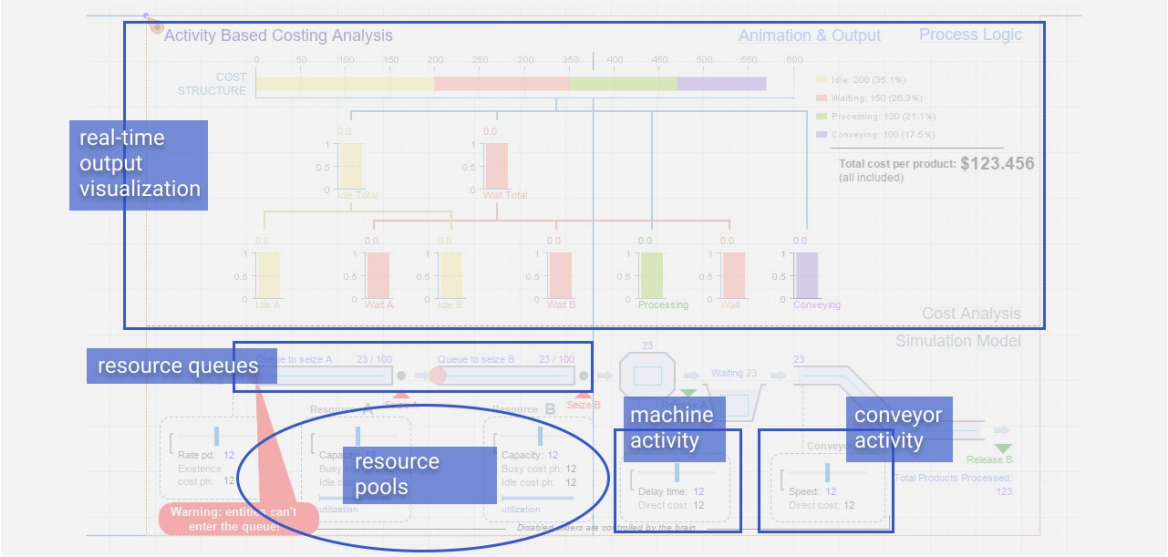
The optimization goal of the ABCA model is to reduce the operational cost per product lot while maintaining a high production throughput according to the configured manufacturing schedule. Incoming lots claim resources from the two available resource pools, complete processing, then release any unconsumed resources.

The model also tracks resources allocation and resource consumption management based on dynamic demand to calculate the operational costs associated with resource utilization. The operational costs accumulated at each production step are broken-down into several categories for analysis and optimization.

ABCA model components

The ABCA model has many moving parts, but the key components are:

- simulation visualizer
- source generation
- resource queues
- resource pools
- machine activity modeler
- conveyor activity modeler



activity".

The settings for source generation, resource pool capacity, machine processing time, and conveyor speed all have default settings, but can be changed while the simulation is running by dragging the associated slider.

Simulation visualizer

The simulation visualizer lets you monitor in real time as things change while the simulation runs. The visualization is useful when running the local (unmanaged) version of the simulator during training on Bonsai.

ABCA model output includes responsive meters, the current cost per product lot, and summary information on:

- the number and percentage of product lots currently processing.
- the number and percentage of product lots being conveyed.
- the total number and percentage of resources going unused (idling).
- the total number and percentage of product lots blocked and waiting for resources.

Source generation

The source generation component creates new product lots and injects them into the system. ABCA model implements source generation as an AnyLogic Source block.

Product lot generation follows a Poisson distribution, which mimics a situation where product lots arrive with exponentially distributed inter-arrival times and an average arrival time of $[\ ^{1}/_{\text{configured\_arrival\_rate}}\ ]$. The parameterized arrival rate is the expected number of product lots that will enter the system over a specific period of time, for example, 2 per day.

> **IMPORTANT**
>
> Arrival rate is an **expected** value, not a **guaranteed** value. Using a Poisson distribution means that the arrival time calculated during simulation is not deterministic. For example, if the configured arrival rate is 2 per day, when the simulation runs, the effective product lot arrival rate value could fluctuate between 0.5 to 2 per day. The intent is to model a realistic arrival rate that is subject to change due to external factors.

Resource queues

Resource queues give the simulation a place to put product lots when something stops them from moving forward in the process. The ABCA model implements resource queues as AnyLogic Queue blocks.

Queues work as transitional blocks within the simulation:

- If there **is available space** on the conveyor after the queue, the product lots passes through in zero simulated time.
- If there **is no available space** on the conveyor after the queue, the product lot idles in the system until space becomes available.

The most important property of a queue is the capacity setting, which specifies the maximum number of entities that can be accumulated inside that queue at any point in time. Product lots cannot enter a queue while it is at capacity.

Resource pools

Resource pools (**Resource A** and **Resource B**) represent the set of available system resources for processing product lots. The ABCA model implements a system resource pool as a set of AnyLogic ResourcePool, Seize, and Release blocks with a parameterized resource capacity.

The ABCA model includes two resource pools that can be configured independently. When the simulator starts, it creates a set of resource units based on the associated parameter settings. Once created, resource units remain idle until they are seized by a product lot.

Machine activity modeler

The machine activity modeler captures the time and factory equipment cost for processing product lots. The ABCA model implements machine activity as an AnyLogic Delay block with a parameterized mean delay value. Delay blocks stop simulation progress by holding the product lot in the block for a given amount of time and block capacity is dependent on the number of resources in the system.

The ABCA models processing delay with a parameterized triangular distribution based on the target mean processing time. As a result, product lots can spend between 1 to 12 days in processing (plus or minus 30%). For example, if simulation is configured with a mean processing delay of 5, a given product lot will spend between 3.5 and 6.5 days in the Delay block. As processing time is based on a random distribution, each individual product lot will be delayed for a slightly different amount of time.

Conveyor activity modeler

The conveyor activity modeler captures the time and cost related to moving product lots through the system. The ABCA model implements conveyor activity as an AnyLogic Conveyor block with a parameterized speed.

Conveyor blocks have a defined length and move product lots along a path with a constant speed while

preserving a minimum space between the lots.

ABCA Anylogic objects

NAME	DESCRIPTION	RELATED PARAMETER SET	STATIC SETTINGS
ABCAgent	A discrete product lot	General	None
ABCAResourceUnit	A discrete resource unit	None	None
Main.ABCAgents	Models the total set of product lots currently in the system	None	None
Main.auxQueueA	Models the resource queue for resource pool A	Resource pool	None
Main.auxQueueB	Models the resource queue for resource pool B	Resource pool	None
Main.bonsaiConnector	Container for Bonsai workspace configuration details	Bonsai	Simulator name Timeout
Main.conveyor	Models current conveyor activity in the system	Conveyor	Length
Main.process	Models current machine activity in the system	Machine	Capacity
Main.queue	Models the product lot capacity of the conveyor	None	Capacity
Main.releaseA	Models the process of releasing resources back to resource pool A	None	None
Main.releaseB	Models the process of releasing resources back to resource pool B	None	None
Main.resourceA	Models the current set of resource units in resource pool A	Resource pool	Speed
Main.resourceB	Models the current set of resource units in resource pool B	Resource pool	Speed
Main.seizeA	Models the process of claiming resources from resource pool A	None	Queue capacity
Main.seizeB	Models the process of claiming resources from resource pool B	None	Queue capacity

NAME	DESCRIPTION	RELATED PARAMETER SET	STATIC SETTINGS
Main.sink	Models the process of product lots leaving the system	None	None
Main.source	Models the process of product lots entering the system	General	None

Simulation parameters

Bonsai parameters

Bonsai parameters let you connect the unmanaged version of the ABCA simulator to your Bonsai workspace.

- **accessKey** [Unset] An authorization key used by the simulator to connect to your Bonsai workspace. Follow the instructions in Get your workspace access key to set the parameter string.
- **bonsaiMode** [`-1`] Indicates whether the simulation is connected to a Bonsai brain. A value of 1 indicates the simulation should expect input from a brain. A value of -1 indicates that the simulation is running in isolation.
- **workspaceID** [Unset] The Azure object ID associated with your Bonsai workspace. Follow the instructions in Get your Bonsai workspace details to set the parameter string.

Conveyor parameters

Conveyor parameters define the capabilities of the conveyors associated with the system.

- **ConveyorSpeed** [`0.001`] The speed in meters per second that at which products move through the system. Speed is constant.
- **RelativeMoveCost** [`0.3`] The per-hour cost of moving a product lot along the conveyor.

General parameters

General simulation parameters set holistic state details about the system.

- **ArrivalRate** [`0.65`] Average number of product lots that arrive at the system per day. Inter-arrival times are exponentially distributed.
- **ExistenceCostPerHour** [`1`] Captures the cost units per hour incurred by a product lot by existing in the system. For example, the cost of electricity required to run refrigeration for lots that have not shipped yet.

Machine parameters

Machine parameters define the state of activity for the factory equipment associated with the system.

- **MeanProcessDelay** [`2`] The average time in days (N) that an individual product lot spends processing in the machine. Simulation calculations are based on a triangular distribution with a minimum value of 0.7N and maximum of 1.3N.
- **RelativeProcessingCost** [`0.3`] The per-hour cost of processing a product lot in the machine.

Resource pool parameters

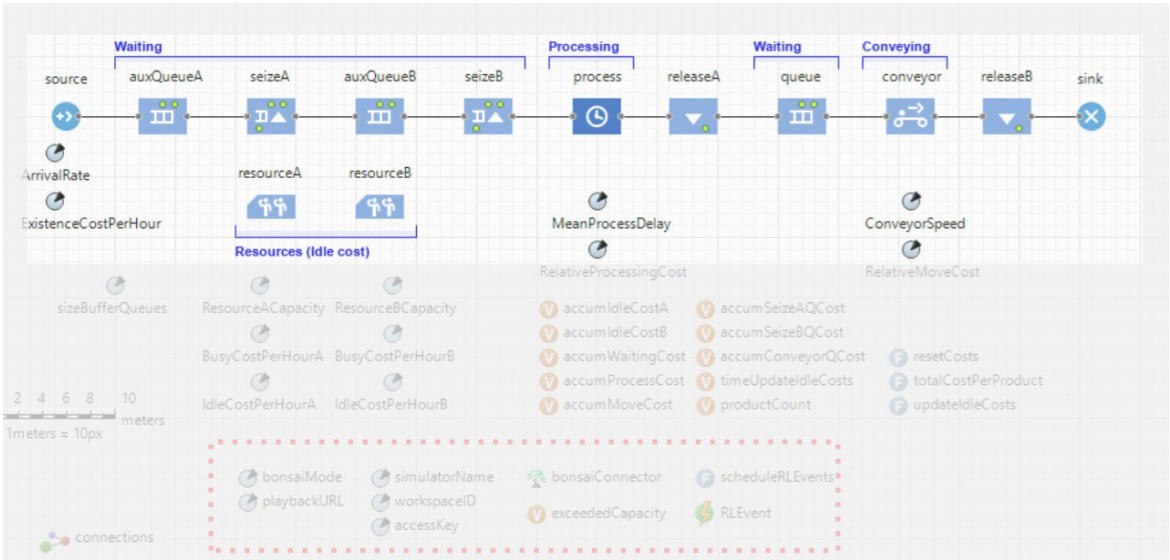
Resource pool parameters define the state of activity for the resource pools associated with the system.

- **ResourceACapacity** [12] The number of resource units available in resource pool A.
- **ResourceBCapacity** [12] The number of resource units available in resource pool B.
- **BusyCostPerHourA** [3] The per-hour cost of a resource in use from pool A.
- **BusyCostPerHourB** [5] The per-hour cost of a resource in use from pool B.
- **IdleCostPerHourA** [2] The per-hour cost of a resource sitting unused in pool A.
- **IdleCostPerHourB** [2] The per-hour cost of a resource sitting unused in pool B.
- **sizeBufferQueues** [45] The number of product lots that can be queued while waiting for resources from either of the pools. The value should be less than or equal to the total capacity of the conveyor.

Simulation process flow

The ABCA model is a continuous simulation with the following process flow:

1. The source block generates product lots with exponentially distributed inter-arrival times.
2. Each incoming product lot tries to seize one resource unit from resource pool A.
   - If units of resource A **are not** available, the lot waits in a queue (auxQueueA) until a resource unit is available.
   - If units of resource A **are** available, the product lot seizes the resource.
3. The product lot is processed (delayed) in the machine block.
4. When the product lot leaves the machine it releases resource A, which is now available for other product lots to seize.
5. The product lot is conveyed to an arbitrary point.
6. The product lot tries to seize one resource unit from resource pool B.
   - If units of resource B **are not** available, the lot waits in a queue (auxQueueB) until a resource unit is available.
   - If units of resource B **are** available, the product lot seizes the resource.
7. The product lot is processed (delayed) in the machine block.
8. The product lot is conveyed to the sink block, which represents the lot exiting the system and releases resource B, which is now available for other product lots to seize.



resourceB widgets are labeled "Resources (Idle cost).

The simulation is forcibly terminated if the system becomes overloaded. The system is considered overloaded whenever the number of lots in auxQueueA (the queue for seizing a resource unit from resource pool A)

exceeds the buffer capacity set for the queue. Whenever auxQueueA exceeds capacity, it indicates that the number of product lots entering the system greater than the number of product lots leaving the system, which typically happens for one (or more) of the following reasons:

- There are too few resources available in resource pool A
- There are too few resources available in resource pool B
- The machine processing time is too long
- The conveying speed is too slow

Operational cost calculations

Operation cost for an individual product lot is defined to be the combination of costs for conveying and processing the lot in addition to any inherent cost associated with having the lot in the system.

```
ProductLotCost = ExistenceCost +
                 BusyCost +
                 IdleCost +
                 ProcessingCost +
                 ConveyingCost
```

- **ExistenceCost** is a configured "work-in-progress" or holding cost in dollars-per-hour (`ExistenceCostPerHour`) that applies as long as the product lot exists in the system.
- **BusyCost** is the sum of all configured costs in dollars-per-hour that apply while a resource unit is claimed from a specific resource pool X for performing process operations on a product lot (`BusyCostPerHourX`).
- **IdleCost** is the sum of all configured costs in dollars-per-hour that applies while a resource unit is sitting unclaimed in a specific resource pool X (`IdleCostPerHourX`).
- **ProcessingCost** is a calculated cost in dollars-per-hour that applies while a product lot is delayed in the machine. While not explicitly modeling processing activities, the calculated processing cost can be imagined as attributable to the associated maintenance costs, where a faster machine speed (less delay) results in more internal wear and thus an increased upkeep cost. The processing cost of a product lot is calculated as the configured processing cost divided by the square of the mean processing time [`RelativeProcessingCost` / (`MeanProcessDelay`)^2].
- **ConveyingCost** is a calculated cost in dollars-per-hour that applies while a product lot is delayed in the machine. While not explicitly modeling a conveyor, the calculated conveying cost can be imagined as attributable to the associated maintenance costs, where a faster or longer conveyor results in more break points and increased wear and thus an increased upkeep cost. The conveying cost of a product lot is calculated as the configured conveying cost multiplied by the square of the conveyor speed [`RelativeMoveCost` X (`ConveyorSpeed`)^2].

In the ABCA model, the cost in dollars-per-hour for an individual product lot is the combination of the existence cost, the cost of the resources used, and the cost of actually processing the lot in the machine:

```
ProcessCost = ExistenceCostPerHour +
              [ BusyCostPerHourA + BusyCostPerHourB ] +
              [ RelativeProcessingCost / (MeanProcessDelay)^2 ]
```

To calculate the full operational cost over time, the ABCA model accumulates the individual per-lot costs for:

- processing the product lots (`accumProcessCost`)
- resource units sitting idle (`accumIdleCostA` , `accumIdleCostB`)
- product lots waiting in the system (`accumWaitingCost`), which includes:
  - idling in resource queues (`accumSeizeAQCost` , `accumSeizeBQCost`)

  - idling on the conveyor (`accumConveyorQCost`)
  - seizing resources (`accumSeizeAQCost` , `accumSeizeAQCost`)
- movement through the system (`accumMoveCost`)

The total operational cost per product is calculated by dividing the sum of these costs by the number of product lots that have left the system.

```
TotalOperationalCost = accumProcessCost +
                       [ accumIdleCostA + accumIdleCostB ] +
                       accumWaitingCost +
                       accumMoveCost

OperationCost = TotalOperationalCost / productCount
```

Bonsai brain integration

Once you set the parameters for the Bonsai connector, you can run the unmanaged (local) version of the ABCA simulator to train a new version of the Bonsai brain included with the solution accelerator. The goal of the brain is to adaptively adjust the number of available resources, processing time, and conveyor speed to maintain a minimum cost per product lot for a specific arrival rate.

> **NOTE**
> Any time you move one of the component sliders, the cost statistics will be reset, so the observed metrics will be specifically for the newly chosen arrival rate.

Observable and action spaces

Information is passed between the simulation and Bonsai according to the observable space and the action space defined in the Inkling file associated with the brain:

- The **observable space** defines the information provided by the simulation that the brain will use to make predictions.
- The **action space** defines the simulation parameters the brain can change.

Information about the observable space and action space is exchanged using the input parameters defined in the ABCA model and the Inkling fields defined in the ObservableState and Action types.

SPACE	MODEL PARAMETER	INKLING FIELD	ALLOWABLE RANGE
Observable space	ArrivalRate	`ObservableState.arrivalRate`	`[0.5, 2.0]`
Action space	ResourceACapacity	`Action.numResourceA`	`[1, 20]`
Action space	ResourceBCapacity	`Action.numResourceB`	`[1, 20]`
Action space	MeanProcessDelay	`Action.processTime`	`[1.0, 12.0]`
Action space	ConveyorSpeed	`Action.conveyorSpeed`	`[0.01, 1.0]`

Training

The Inkling file included with the solution accelerator will train a brain in episodes of six-months. At the start of each episode, the brain attempts an action based on the information in the observable state. The next observation occurs after six simulated months pass. At that time, the brain observes the outcome of its action

before beginning a new episode. As a result, the brain only has one chance to find an optimal configuration for the action space. The long duration between episodes is due to the relatively low rate of arrivals in the simulation (0.5 – 2 product lots per day) and the relatively long processing time per product lot (1 – 2 days).

The success of the configuration selected by the brain (the reward) is calculated as the negated cost per product lot, with a large penalty if the system becomes overloaded. With each episode, the brain tries to maximize the negation in order to minimize the overall cost per product lot.

Once trained, the brain will change the action space parameters of the simulation whenever you move one of the component sliders to minimize the operational cost per product unit as much as possible.