Bakery Manufacture - supply chain multiagent system

31.01.2025

1. Details of the proposal

1.1 Supply chain

The proposed project focuses on **multi-agent coordination and task division in a collaborative supply chain environment**. It simulates a bakery production system where multiple agents interact dynamically to complete orders efficiently. The system operates within a **multi-agent framework**, ensuring smooth workflow execution through structured communication and decision-making processes.

Project is implemented in **JadeScript** and the environment consists of 4 distinct agent types, each with specialized roles:

- **Baker (4 instances):** Responsible for preparing baked goods. Bakers follow recipes, request ingredients, and manage the baking process.
- **Supplier (2 instances):** Supplies ingredients to bakers upon request.
- **Packer (1 instance):** Packs baked goods into appropriate packaging and prepares them for delivery.
- **Supervisor (2 instances):** Oversees the process, assigns orders, ensures quality control, and facilitates workflow coordination.

A workday is finished when all baked items are prepared, packed, and delivered to the supervisor. The project demonstrates the importance of agent-based cooperation in supply chain management, ensuring efficiency, adaptability, and resilience in handling real-world manufacturing constraints.

2. Introduction

2.1 Supply chain

A **supply chain** is a network of suppliers, factories, warehouses, distribution centres, and retailers through which raw materials are acquired, transformed, and delivered to customers. Its main task is to distribute the right products in the right quantity to the proper instances [1].

In the context of this project, the multi-agent system (MAS) models a supply chain environment where different roles—Baker, Supplier, Packer, and Supervisor—interact to complete the production and delivery process. Each role has distinct responsibilities, and agents communicate dynamically to coordinate tasks, ensuring the seamless flow of goods from production to delivery.

[1] Vivek Kumar, Dr. S. Srinivasan,2010. IJCSI International Journal of Computer Science Issues Vol.7,Issue 5, pp. 198.

The goal here was to create a Multi-agent systems which can dynamically adjust production and distribution plans to meet customer needs while minimizing waste and improving efficiency – which stands as one of specific use cases of AI Agents and multi-agent systems [2]

2.2 Short description of project

Initially, both bakers, packers and suppliers are required to report to their respective supervisors. Each baker is responsible for informing their designated supervisor which is assigned based on the type of orders Baker will be working on, while packers and suppliers report only to the supervisor who oversees "private" orders. Upon the completion of reporting by all workers, the supervisor commences the day's activities and provides the packer with a list of packages to be prepared for that day.

The baker is unable to prepare two goods concurrently; therefore, upon confirming that no goods are currently being prepared or that all goods in progress are complete (in state of baking), the baker submits a request for a new order. The supervisor then dispatches the orders, and the baker commences the work. Initially, the Baker verifies the sufficiency of ingredients, proceeding with the task only if they are available. In the event of insufficient ingredients, the Baker has two options: he can either request assistance from his colleagues, who are designated upon their creation, or he can request assistance from a supplier.

Once the Baker has obtained all the necessary ingredients, he begins the preparation phase, adhering to the specified time indicated in the recipe. During the baking process, the baker requests new orders to be made. The finished goods are then sent to the supervisor, who either accepts or rejects them. If an order is rejected, it is returned to the baker to begin the process over again. Additionally, if order is accepted and marked as 'private' it is sent to the packer for completion.

The day's operations conclude when all standard orders have been adequately fulfilled and all private orders have been prepared and packed.

3. Project Implementation

3.1 BakeryOntology

An **ontology** is a set of rules that defines how agents communicate and work together. It provides a shared framework that helps agents understand each other by defining key concepts, actions, predicates and propositions. This ensures that messages are interpreted the same way by all agents, allowing them to share information, make decisions, and coordinate their tasks effectively. These abstractions are used as building blocks for the structures that represent the knowledge of the agents. [3]

3.1.1 Concepts

- *IngredientQuantity* – The ingredient quantity field is a quantitative value that denotes the amount of a specific ingredient that is to be stored in Bakers stock or transferred during instances of ingredient shortage. The ingredient's name is expressed in text format, while the quantity is given as an integer.

[2] Colin Masson, 2025. Unlocking Supply Chain Potential with AI Agents and Multi-Agent Workflows.

[3] Giuseppe Petrosino 2022. Jadescript Programmer’s Guide Version 1.0.

- *Good* – Represents the good that can be produced in the Bakery by Baker. There are five distinct types of good → bread, bun, cookies, cake and cupcakes. Good is built of name as text form, list of *IngredientQuantity* that are needed for production, baking time and preparation time both expressed as duration, batch size denoting the quantity of goods to be produced in a single preparation.
- *Order* – Represents the specific request for a particular good that has been directed to the Baker by the Supervisor. It consist of id as a text, the name of the good to be prepared, the status of the order (which can be one of the following → pending, toPrepare, preparing, baking, finished) and type of order. Two distinct types are recognised: "normal" and "private" orders. (in private order we put only cake and cupcakes)
- *OrderQuantity* – This field signifies the aggregate quantity of orders for a particular good. Information is stored by the supervisor at the beginning of the day. Initially, no distinction is made between individual orders; however, the supervisor, who is in possession of a list of quantities for each order, divides them into separate orders. *OrderQuantity* is built of good as text presenting the name of good to be made, the type of good, and the quantity. The quantity is expressed as an integer, denoting the number of orders for a specific good that should be prepared.
- *PackageOfGoods* – Represents the single package of goods. It is important to note that only orders with a 'private' classification are packed. Knowledge of packages is stored in Supervisor (of type “private”) but at the beginning Packer is also informed of the orders that they should pack. It consist of packageId as text, list of *OrderQuantity* which shows how many orders for specific good should be stored inside the package and status as text → pending or finished.
- *PackagePreparation* – Concept represents a tuple consisting of two variables: the packageId expressed as text, and the prep time measured in duration. This concept is essential for the storage of information regarding the time spent on package preparation by the packer.

3.1.2 Actions

- RequestOrder – Represents the action of requesting a new order. It is a message sent by Baker to Supervisor when Baker has nothing else to do at the moment. It consists of Baker's aid.
- AssignOrder – Represents an action of sending a new order after requesting an order by Baker. It is part of the message sent by the Supervisor to the Baker - of course if there are still orders to be assigned. It consists of asking Baker aid and order of type *Order*.
- AskForHelpColleague – Represents a task of asking for missing ingredients co-workers of a baker. It is a message sent to all the Baker's co-workers to find a Baker who can provide ingredients from their stock. It consists of the aid of the Baker in need and a list of the *IngredientQuantity* that are missing.
- RequestIngredientsColleague – After sending a request to several co-workers, if at least one of them replies that he has needed ingredients, the Baker sends a message containing this action to one of those co-workers to start the process of transferring the ingredients. It will again contain the aid of the Baker in need and the list of *IngredientQuantities* that he is missing.

- NeedRestock – Represents a task to ask the Supplier for restocking by the Baker. Performed in two situations: 1. when the Baker is slowly running out of ingredients and after using some of them sees that there is not much left in stock or 2. when none of the co-workers can or have the time to help. It contains the aid of the Baker in need, the list of *IngredientQuantity* that he is missing on and the boolean that informs if the question was asked during the preparation of the good and after getting the ingredients the Baker should go back to the preparation stage or not.
- ProvideIngredients – Represents an action to transfer the needed ingredient from a co-worker or Supplier to the Baker in need. Depending on the helper, Baker gets the exact amount of missing ingredient (from co-worker) or a fixed, but certainly higher than needed amount of missing ingredient (from Supplier). It contains a list of *IngredientQuantity* that Baker is missing on and the boolean that informs if the question was asked during preparation of good and after getting ingredients Baker should go back to preparation stage or not.
- ProvidePackingList – It is a task performed at the beginning where Supervisor, after reporting Packer's readiness to work, provides him with a list of packages to be packed that day. It consists of a list of *PackageOfGoods*.
- RedoOrder – Represents a task where the Supervisor informs the Baker whether there is a need to redo the order or not. It consists of the orderId of the order in question and the decision as a text, which says "No need" if there is no need to redo the package or another message, here I used "Order not done correctly", which states that the order needs to be redone.
- GoodToPack – Represents a task of sending the order to be packed from the baker to the Packer. The order has to be accepted by the Supervisor first, but if it is and is of type "private" it will be sent to the packer. Is contains the order of type *Order*.

3.1.3 Predicates

- WorkerReady – Represents the information sent to the Supervisor stating that the worker is ready to work. It consists of workerId, worker variable which is simply text indicating if this is a message from "baker", "supplier" or "packer" and type which is important for Bakers where its either "normal" or "private". Suppliers and Packers only report their availability to one of the Supervisors.
- IngredientAvaliable – Represents the information about the availability of the requested ingredients. The Baker's co-workers sends the information to the Baker to indicate whether he can help with the ingredients the Baker needs. It consists of aid of co-worker, available variable which is boolean indicating availability and the list of *IngredientQuantity* that Baker is missing.
- SupplierRunOutSupply – Represents an information about the low stock of the supplier. It is sent to the Baker to inform him that the Supplier cannot supply any Ingredients at the moment. It consists of bakerWhileProparing variable which is Boolean. It represents the information whether the baker was preparing the good at the moment and without ingredients will not do the work.
- OrderStatus – It represents and holds the information about the current status of the order. When the order reaches the status "finished", the message with the updated *OrderStatus* is sent to the Supervisor. It consists of orderId and status as text.

- PackageStatus – Contains information about the status of the package. When the status of the package reaches "finished" a message with PackageStatus is sent to the Supervisor. It consists of pcg variable as *PackageOfGoods*.
- BakerRaportInfo – Represents the finish report created by the Baker at the end of the working day. It consists of the Baker's aid, the integer representing the number of successful ingredient-related assistance requests made to co-workers, the integer representing the number of successful necessary ingredient-related assistance requests made to Suppliers. NumOrdersToRedo as integer, which shows number of orders that needed to be redone by the Baker, ordersDone as integer, which shows number of all orders done during work day, and restStock as list of *IngredientQuantity*, which represents the stock that was left at the end of the day.
- PackerRaport - Document is generated by the Packer at the end of the working day and informs about the final report. It consists of pcgPrep as list of *PackagePreparation*, which represents information about all packages and their preparation time.

3.1.4 Propositions

- AgentsReported – Expression to represent and inform that all agents reported before the work and that the working day can commence. This is sent from the Supervisor to all Baker to inform them that they can start asking for orders.
- EndOfPrivateOrders – Expression indicating that the list of private orders has been completed. The message containing this expression is then sent to the Bakers of the "private" type by their Supervisor, prompting them to transition to "normal" mode and begin processing a new set of "normal" orders.
- EndOfOrders – Expression marks the end of orders for a specific group of Bakers. This message is sent from the Supervisor to the relevant Bakers.
- GroupEndedDay – Expression to represent the end of day for the other group of Bakers. Each Baker group waits for the other to fully end the day and start sending reports. A message is sent from one Supervisor to another.
- EndOfDay – Expression to represent the end of the working day when all orders are done by both groups. The Supervisor then announces this to all the workers under their responsibility.
- AllPackagesReady – This expression states that all packages that needed to be packed are of state "finished". This is an information message from the Supervisor to the Packer.

3.2 Agents

As outlined previously, there are four distinct categories of agents: Baker, Supervisor, Supplier and Packer.

3.2.1 Baker

The role of the Baker agent is to represent the workers who prepare goods in the Bakery. The main goal of the agent is to successfully complete all orders. They possess exclusive knowledge on the production process of bakery goods and each baker maintains their own inventory of ingredients for use in the production of goods. As previously mentioned, there

are two types of baker: normal and private. The main difference between them is the knowledge of recipes stored in the Recipe Book.

The day of the Baker begins with the activation of the **DelayedBakerReady** function, which is exclusively used to notify the Supervisor of the Baker's readiness by transmitting a WorkerReady message. Behaviour is activated with a delay to ensure that the Supervisor is not overwhelmed with messages during the creation process. Additionally, we activate **ManageOrders** and **ListenCoworkers**, both of which are cyclic behaviours.

ManageOrders:

In **ManageOrders**, agents oversee the primary workflow and its conclusion. At the beginning of the day, the agent is informed that all workers are present and ready to go. The agent then activates **SendRequest** and **WaitForNewOrder**, allowing the day's work to commence.

One later stage of program this is also the place where the Baker is notified of the completion of baking and communicates the status to the Supervisor via the OrderStatus message. When the Supervisor reviews the order, they will send a message of RedoOrder when two options are available:

- If the order has been accepted, the Baker can proceed and if it has been marked as 'private', the Baker will send it to the Packer to pack
- If the order has been rejected, the Baker must redo the entire order and activate the **PrepareOrder** behaviour.

When the Baker primary is labelled as 'private' during the day, it changes its nature to 'normal'. This is because, even though 'private' Bakers have their own list of orders to make, they also have to help normal workers by doing an additional one-third of the work of ‘normal’ Bakers. Consequently, despite initially being designated as "private", they end the day by doing normal orders under type “normal”. This transition is triggered by the Supervisor's message, EndOfPrivateOrders, which notifies the Bakers that the private order list is now empty.

The Baker's work is concluded with the message EndOfOrders, which notifies them that the list of orders to make is now empty and the day is coming to a close. The day concludes with the message EndOfDay, after which Baker submits the daily report and removes the behaviours.

During the day, the behaviour checks one main property: is Baker able to ask the Supervisor for a new order to make? It makes this decision based on a few properties: are all orders currently made by Baker in status "baking"? Is Baker not doing anything at the moment? Is there no information about the end of orders? Is Baker not currently waiting for another order from the Supervisor?

SendRequest is responsible for submitting a request for new order to make to the Supervisor when the Baker is not occupied.

WaitForNewOrder: The Baker awaits an order, which is subsequently transferred from the Supervisor by AssignOrder. Prior to this, the Baker verifies whether they are familiar with the recipe for the order, and if so, they initiate the preparation process- **PrepareOrder**

PrepareOrder:

This is a one-time action during which the entire good is to be prepared. First, the Baker verifies that they have all the necessary ingredients to make the product.

- If all the ingredients are in stock, the status of the order is updated and the used ingredients are subtracted. Then, the Baker checks if the amount of each ingredient in stock is above the lower limit (set at 3). If not, a message is sent to the suppliers to request restocking. Finally, we activate **BakingOrderBehaviour**, which starts after the preparation time.
- If any ingredients are missing or insufficient for the preparation of the product, a message is sent to co-workers requesting the required supplies, and the Baker activates **HandleIngredientsShortage**.

During the execution of **BakingOrderBehavior**, we are constantly checking whether the baking time has passed. If so, the status of the order is changed to "finished" and the Baker is notified that one of its orders has been baked.

ListenCoworkers:

When one of the Bakers needs help the system will notify all of his co-workers via AskForHelpColleague.

- If such co-worker is not busy preparing their own goods, they will begin checking for available ingredients in their stock.
  - If the quantity of even one of the required ingredients exceeds the available stock of co-worker, the co-worker will provide a notification of their inability to assist.
  - If the co-worker has the required ingredients, he will send information on the possibility of providing assistance.
- If the co-worker was occupied with their own work at the time, he will send information on the impossibility of providing assistance as they are already engaged in their own tasks

If a co-worker has provided information indicating their ability to assist with ingredients, there is a possibility that they will be the one that the Baker will actually ask by RequestIngredientsColleague for those supplies. In such a case, the co-worker will check their inventory, deduct the required items, and then provide them to the Baker by ProvideIngredients.

HandleIngredientsShortage:

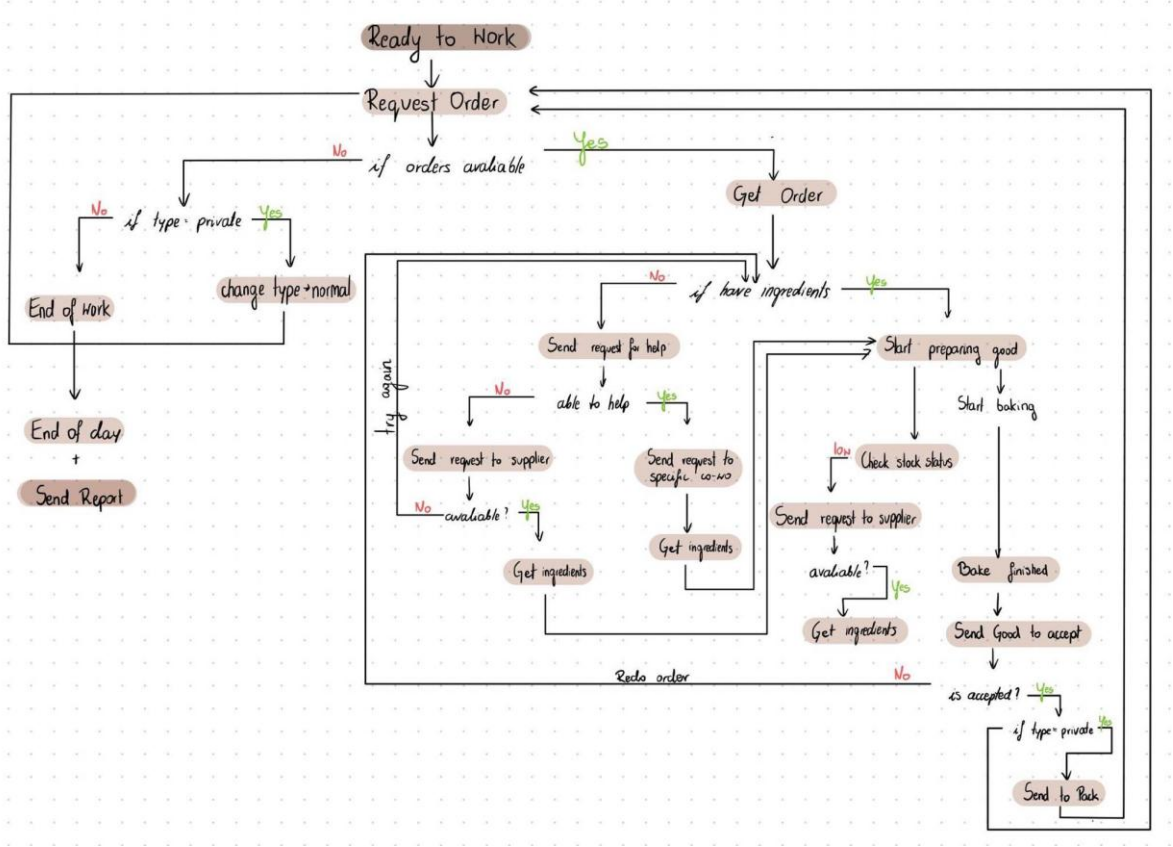
Is activated by Baker's request for assistance, and it awaits a response. Initially, we assume that no assistance will be provided by colleagues.

If we were at the stage of asking co-workers for help, we would be waiting for their messages IngredientAvailable, which states whether the co-worker has the required ingredients. When all co-workers respond, we check if at least one of them answered with variable available=true and Baker sends the message RequestIngredientsColleague to the first co-worker who claimed availability. If none of the co-workers claimed availability, the message is sent to Suppliers.

The Baker is now only waiting for the supplies from the co-worker or Supplier. When ingredients are sent by ProvideIngredients, we also have to check if the Baker was asking for the ingredients during preparation, because if so, he needs to go back to **PrepareOrder** to prepare this specific order.

It is also possible for the message to be sent to the supplier and to receive the information SupplierRunOutSupply, which means that the supplier is unavailable at that moment due to personal restocking. If the Baker was in the middle of good preparation, the **PrepareOrder** will be returned to and will attempt to request the supplies again shortly.

Basic workflow sketch*:



* The purpose of the sketch is to outline the order in which a single agent operates and to show roughly the order in which it performs its tasks and generates results. However, it is important to remember that in a running program, several tasks may be happening at the same time.

3.2.2 Supervisor

The Supervisor's key responsibilities include overseeing the Bakery's operations. Both Supervisors distribute orders according to the group they are responsible for, accept work and announce the end of the work and day. They possess full knowledge of the orders that should be made that day. These orders cover both “normal” and “private” orders, (“private” orders are categorised into packages)

At the start of the day, the supervisor prepares a list of individual orders to be made, based on the initial estimates of product requirements. The supervisor then activates **ReportingWorkers** and awaits the notification of all workers' readiness.

In **ReportingWorkers**, once all workers have reported, the message AgentsReported is sent to all Bakers, prompting them to start requesting new orders. Additionally, the list of packages to be prepared that day is sent to the packer by the ProvidePackingList message. Following this, the Supervisor activates the **SendOrder** function.

SendOrder:

Its primary function is to dispatch orders to the Bakers. Initially, we also initiate **WaitingForFinishedOrders** to allow the Supervisor to await completed orders.

Subsequently, we proceed to receive RequestOrder, at which point we ascertain whether the requesting Baker has previously reported their readiness. If so, we conduct a series of checks before dispatching the order:

1. Firstly, we verify whether the Baker is of type 'private' and if the length of the list of orders to be done (now of type 'private') is equal to 0. If so, the Supervisor's type is changed to normal, a new list of orders with 'normal' orders is created with the amount of 1/3 of orders that the other group had to do. The eandOfPrivateOrders variable is then changed to true.
2. When the current Baker is of type "private" (in the previous check, the type of Supervisor had only been changed) and the eandOfPrivateOrders variable is set to true, the Supervisor sends the EndOfPrivateOrders message to the Baker to prompt him to change his type to "normal". We also set the momentOfChangeType variable to true.
3. When a request for orders is made and the momentOfChangeType variable is not current (so set to false), it indicates that the Baker is entitled to demand the order at that time. We then check if the list of orders is not empty and, if not, we proceed with sending the order. However, if the list is empty, we send the EndOfOrders message.

WaitingForFinishedOrders:

The Supervisor is waiting for finished orders provided by Baker and also for packages provided by Packer.

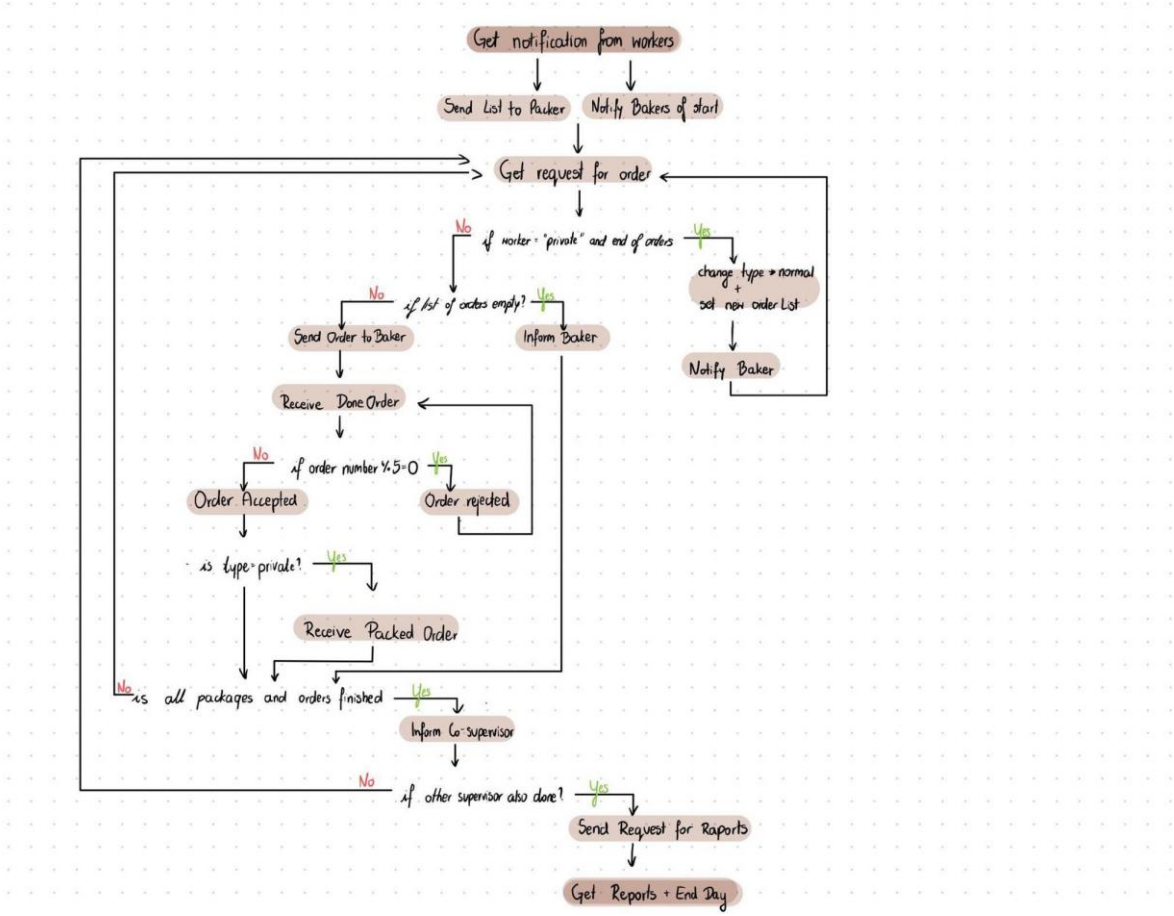
- Orders are dispatched by Baker via OrderStatus message. The Supervisor's standard procedure is to accept orders; however, each fifth order is rejected and returned to Baker for redoing. If an order is accepted, a message is also sent, but with the decision of 'No need' (for redoing the order).
- Packages are dispatched using the PackageStatus message. The status of each package is then updated in the relevant list, and the number of finished packages is checked. If all packages have status "finished", the message AllPackagesReady is sent to Packer.

Meanwhile, the Supervisor constantly verifies that all orders have been completed. Two checks are used for this purpose:

- The first checks if the work for a specific Supervisor and their team of Bakers has finished. If the work is finished, the Supervisor will send a message to the other Supervisor about the finished work.
- The second check is related to the end of the day for all workers. It is entered when all workers have completed their tasks successfully. If this is the case, the EndOfDay message is sent to all workers and the **FinishRaports** function is activated by the Supervisor**.**

At the end of the day in **FinishRaports** Supervisor awaits the reports from all Bakers and Packers. Once all reports have been submitted, they are printed and the day is concluded.

<u>Basic</u> work flow sketch*:



* The purpose of the sketch is to outline the order in which a single agent operates and to show roughly the order in which it performs its tasks and generates results. However, it is important to remember that in a running program, several tasks may be happening at the same time.

3.2.3 Supplier

The Supplier agent is responsible for providing supplies to the Baker when required. We distinguish two types of suppliers, each of which provides different goods. Each supplier starts the day with a specific quantity of goods in their inventory.

The first action of the Supplier is to report their readiness to the Supervisor, after which the **ProvidingRestock** and **WaitForEandOfDay** is activated.

While the **ProvidingRestock** supplier awaits a request from the Baker. If the NeedRestock is received, the procedure of restocking is initiated. First, it checks the inventory list to see if it has the necessary ingredients. If it does and the quantity one ingredient in Bakers' stock is lower than 4 (the maximum amount for ingredient required for the recipe) and the inventory is higher than 6, then the ingredient is provided. The amount provided is fixed at 6, ensuring that the supply of ingredients is slightly higher than the potential demand, with products supplied in excess/reserve.

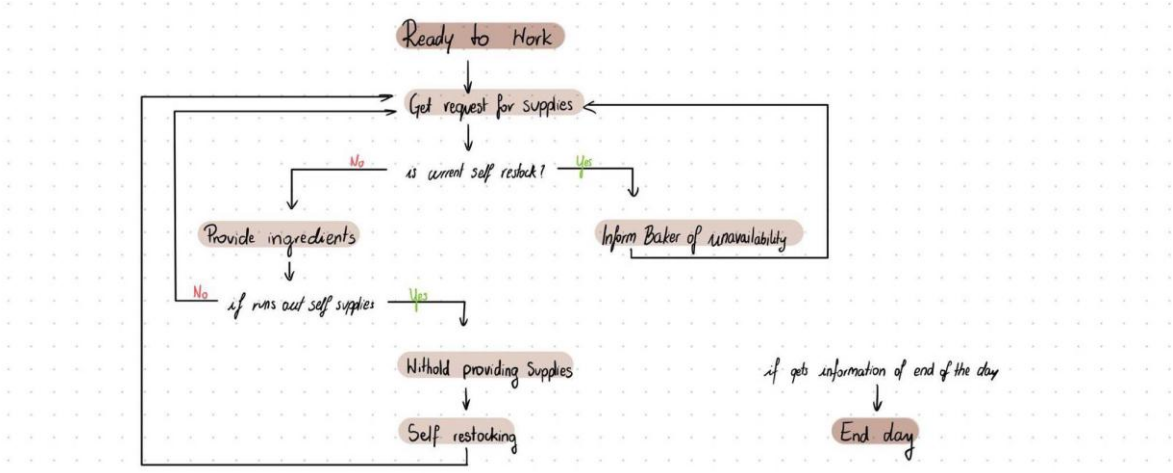
If supplier runs out of supplies in stock (quantity falls below 6), Supplier activates **GoForSupplies**.

When the Supplier activates **GoForSupplies**, it simulates the moment of restocking supplies. The supplier also activates **CurrentlyRestocking**, which provides the Baker with

information during restocking time if the Baker requests it. After a set time has elapsed, the system reverts to its previous state.

WaitForEandOfDay is designed to help manage information about the end of the work day.

Basic work flow sketch*:



* The purpose of the sketch is to outline the order in which a single agent operates and to show roughly the order in which it performs its tasks and generates results. However, it is important to remember that in a running program, several tasks may be happening at the same time.

3.2.4 Packer

The Packer agent is responsible for the packing of provided orders, based on the package list. The packer does not possess any specific knowledge. A list of packages is provided at the beginning of program execution, upon which the packer bases all subsequent actions. Prior to this, Packer notifies the Supervisor of its readiness and activates **WaitForListWithPackages**. The Supervisor then provides the Packer agent with a list of packages, enabling the activation of **WaitAndPackOrders**.

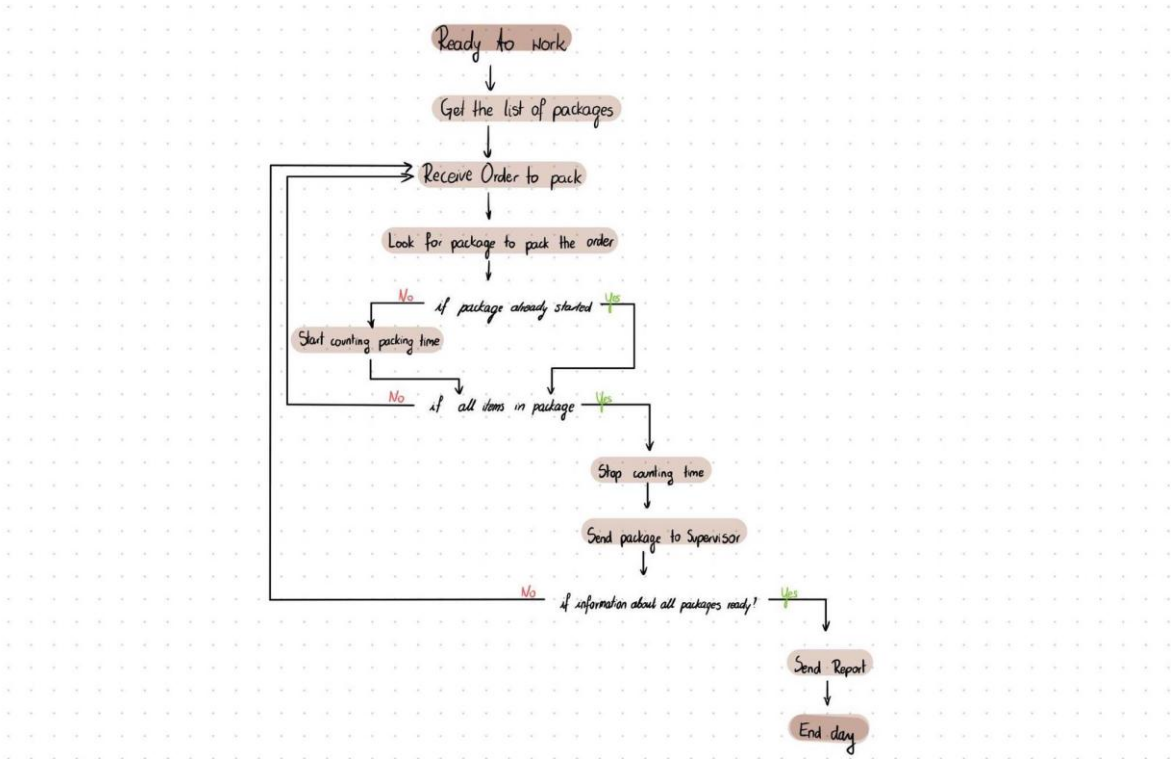
In the **WaitAndPackOrders**, there are two messages for which the agent is waiting: GoodToPack, which provides the order to be packed by Baker, and AllPackagesReady send by Supervisor, which states that all packages have been packed and the work is done, with activation of **WaitEndDay**.

During the execution process, if an order/s has been provided, Packer will check to which package this good should be packed. The order ID is not relevant. The only relevant information is whether the item matches the good to be packed in the package. If this is the first order being packed for a package, Packer starts timing the preparation period.

Following the completion of the packing process, the Packer verifies the package's readiness and the presence of all requested items. If so, the timing for packing the package is stopped, and the finished package is sent to the Supervisor.

At the end of the day – **WaitEndDay,** as the response to the message EndOfDay, Packer sends the report with the list of tuple: packageId and the time spend on packing such package.

Basic work flow sketch*:



* The purpose of the sketch is to outline the order in which a single agent operates and to show roughly the order in which it performs its tasks and generates results. However, it is important to remember that in a running program, several tasks may be happening at the same time.

Main knowledge of agent:

- Baker: book containing recipes for various goods, including bread, buns and cookies, if the Baker is marked as "normal". If the Baker is marked as "private", the book contains recipes for cake and cupcakes. Each Baker has its own stock of ingredients.

```
property bread = Good(
    "bread",
    [
        IngredientQuantity("flour", 2),
        IngredientQuantity("water", 2),
        IngredientQuantity("yeast", 1)
    ],
    "PT20S" as duration,"PT10S" as duration,6
)
property bun = Good(
    "bun",
    [
        IngredientQuantity("flour", 2),
        IngredientQuantity("water", 2),
        IngredientQuantity("yeast", 1)
    ],
    "PT25S" as duration,"PT15S" as duration,8
)
property cookies = Good(
    "cookies",
    [
        IngredientQuantity("flour", 2),
        IngredientQuantity("water", 1),
        IngredientQuantity("eggs", 2),
        IngredientQuantity("sugar", 2)
    ],
    "PT15S" as duration,"PT20S" as duration,14
)
```

```
property cake = Good(
    "cake",
    [
        IngredientQuantity("flour", 4),
        IngredientQuantity("water", 2),
        IngredientQuantity("eggs", 3),
        IngredientQuantity("sugar", 2)
    ],
    "PT30S" as duration,"PT20S" as duration,1
)
property cupcakes = Good(
    "cupcakes",
    [
        IngredientQuantity("flour", 4),
        IngredientQuantity("sugar", 2),
        IngredientQuantity("eggs", 3)
    ],
    "PT10S" as duration,"PT15S" as duration,6
)
property recipeBook as list of Good

property flour = IngredientQuantity("flour",15)
property sugar = IngredientQuantity("sugar",20)
property eggs = IngredientQuantity("eggs",20)
property yeast = IngredientQuantity("yeast",20)
property water = IngredientQuantity("water",20)
property stock = [flour,sugar,eggs,yeast,water]
```

- Supplier: both Suppliers offer a range of ingredients, each with specific products available depending on the type of Supplier. While supplierType1 can provide flour and

eggs, supplierType2 can only provide sugar, yeast and water. To obtain the necessary ingredients, it is essential for each Baker to contact with both Suppliers

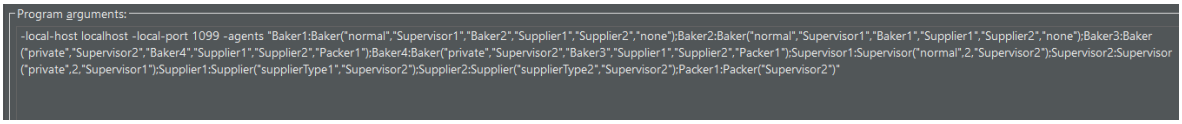
```
property flour = IngredientQuantity("flour",10)
property sugar = IngredientQuantity("sugar",10)
property eggs = IngredientQuantity("eggs",10)
property yeast = IngredientQuantity("yeast",10)
property water = IngredientQuantity("water",10)
property supplies1 = [flour,eggs]
property supplies2 = [sugar,yeast,water]
```

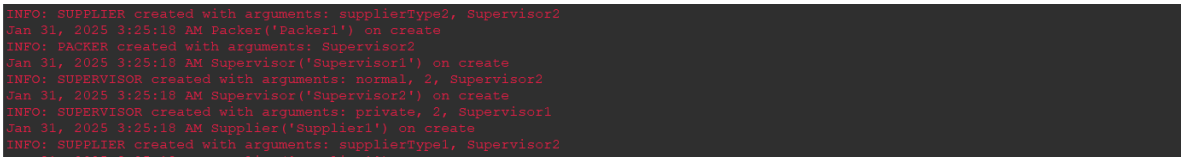
- Packer: initial knowledge is not specified, but Packer is responsible for acquiring the list of packages from the supervisor at the beginning of execution.
- Supervisor: list of orders to be made, including the number of normal orders and the number of packages to be packed, as well as the number of private orders to be included in the package

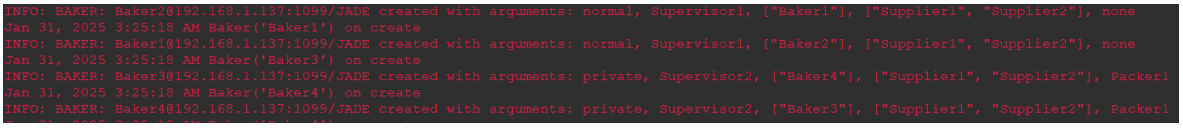
```
property orderBread = OrderQuantity("bread","normal",6)
property orderBun = OrderQuantity("bun","normal",4)
property orderCookies = OrderQuantity("cookies","normal",5)

property orderCake1 = OrderQuantity("cake","private",1)
property orderCupcake1 = OrderQuantity("cupcakes","private",2)
property orderCake2 = OrderQuantity("cake","private",2)
property orderCupcake2 = OrderQuantity("cupcakes","private",1)

property package1 = PackageOfGoods(
    "Pcg1",[orderCake1,orderCupcake1],"pending"
)
property package2 = PackageOfGoods(
    "Pcg2",[orderCake2,orderCupcake2],"pending"
)
property privateOrders = [package1, package2]
```

4. Project Execution

Initially, I was uncertain how to efficiently run the program. The JadeScript guide[4] only mentions building a program by running containers in the correct order. However, I observed in the provided materials from previous years that a different approach was used: running the program from the Run Configuration window, where all agents can be easily predefined and the program can be easily executed. Primary settings are as follows:





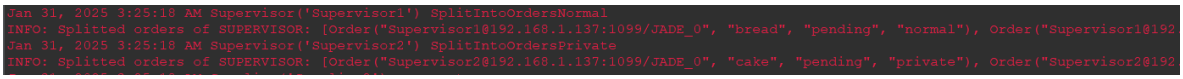


[4] Jadescript language guide: https://github.com/aiagents/jadescript/blob/main/README.md

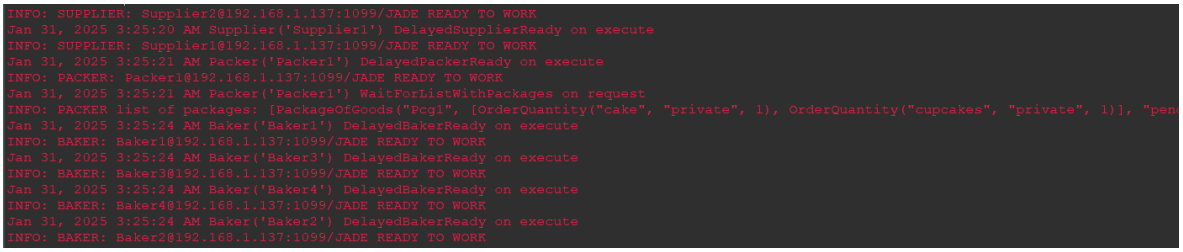
Arguments of agents:

- Baker: type of Baker, name of Supervisor, names of both Suppliers, name of Packer if Baker should have a contact with
- Supervisor: type of Supervisor, number of Bakers to take care of, name of the other Supervisor
- Supplier: type of Supplier, name of Supervisor
- Packer: name of Supervisor

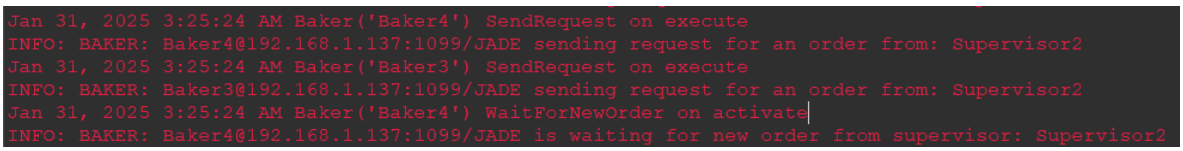
At the start of the process, both suppliers divide the list into individual orders, each containing a specific quantity of goods.



All employees have confirmed their availability to work, and the supervisor has provided the list of packages to the packer.

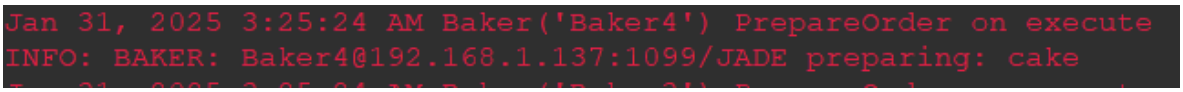


Once all team members have confirmed their readiness to perform, Baker will proceed to initiate the request for orders and await a response.

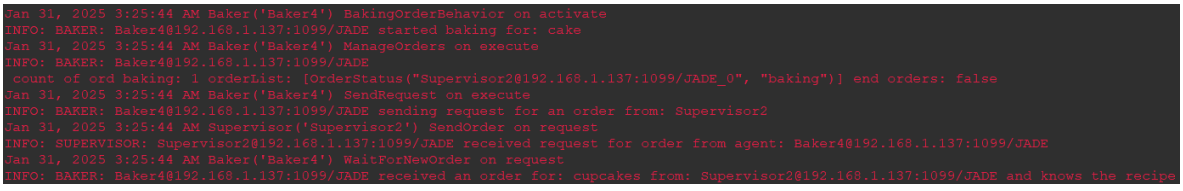


Following acknowledgement of the request for the order, the matter is then passed to the Baker.

Once the order has been assigned, Baker begins to prepare the good.



The next step is to initiate the baking process. Once all the goods currently being prepared by the Baker are in the baking stage, he will be able to request another order.



Once the necessary ingredients have been utilized, the Baker is able to contact the Supplier if they are running low on supplies.

```
Jan 31, 2025 3:25:49 AM Baker('Baker3') PrepareOrder on execute
INFO: LOW STOCK - try to restock supplies
```

```
Jan 31, 2025 3:25:51 AM Supplier('Supplier1') ProvidingRestock on request
INFO: Received NeedRestock from ( agent-identifier :name Baker3@192.168.1.137:1099/JADE  :addresses (sequence http://
Jan 31, 2025 3:25:51 AM Supplier('Supplier1') ProvidingRestock on request
INFO: SUPPLIER sends supplies: [IngredientQuantity("flour", 6)]
```

```
Jan 31, 2025 3:25:51 AM Baker('Baker3') HandleIngredientsShortage on request
INFO: Just got ingredients from: Supplier1@192.168.1.137:1099/JADE
Jan 31, 2025 3:25:51 AM Baker('Baker3') HandleIngredientsShortage on request
INFO: Fixed stock: [IngredientQuantity("flour", 9), IngredientQuantity("sugar", 14), IngredientQuantity("eggs", 11), IngredientQu
```

The following message may be received when attempting to restock supplies with the supplier (it indicates the moment of supplier restocking its own supplies).

```
Jan 31, 2025 3:26:11 AM Supplier('Supplier1') CurrentlyRestocking on request
INFO: Sorry but SUPPLIER: Supplier1@192.168.1.137:1099/JADE is restocking supplies
```

When a Baker of the “private” type requests a new order, but it is the end of the “private” orders period, the Supervisor creates a new list of orders that includes the standard orders and notifies the Baker. The Baker's type is then changed to “normal”.

```
Jan 31, 2025 3:25:59 AM Baker('Baker4') SendRequest on execute
INFO: BAKER: Baker4@192.168.1.137:1099/JADE sending request for an order from: Supervisor2
Jan 31, 2025 3:25:59 AM Supervisor('Supervisor2') SendOrder on request
INFO: END OF PRIVATE ORDERS. Lets do some normal orders.
Jan 31, 2025 3:25:59 AM Supervisor('Supervisor2') SplitIntoOrdersNormal
INFO: Splitted orders of SUPERVISOR: [Order("Supervisor2@192.168.1.137:1099/JADE_4", "bread", "pending", "normal"), Order("Supervisor2@192
Jan 31, 2025 3:25:59 AM Supervisor('Supervisor2') SendOrder ChangeTypeOfAgent
INFO: BAKER: WorkerReady(( agent-identifier :name Baker4@192.168.1.137:1099/JADE  :addresses (sequence http://DESKTOP-3CC6MK8.lan:7778/acc
Jan 31, 2025 3:25:59 AM Baker('Baker4') ManageOrders on inform
INFO: BAKER: Baker4@192.168.1.137:1099/JADE is changed into normal: normal
```

Packer got the item to put in the package (0 indicates that no further orders of this product are required for this package). The package has been completed a few seconds later (after the rest of the items have been placed inside), and if all packages have been completed, the Supervisor provides this information.

```
Jan 31, 2025 3:26:09 AM Packer('Packer1') WaitAndPackOrders on request
INFO: PACKER got an order to pack
```

```
Jan 31, 2025 3:26:09 AM Packer('Packer1') WaitAndPackOrders on execute
INFO: PACKER packed order: OrderQuantity("cupcakes", "private", 0) to package: Pcg1
```

```
Jan 31, 2025 3:26:14 AM Packer('Packer1') WaitAndPackOrders on execute
INFO: PACKER send FINISHED package: Pcg1
```

```
Jan 31, 2025 3:26:29 AM Supervisor('Supervisor2') WaitingForFinishedOrders on inform
INFO: ALL packages packed
```

Once all orders placed under the Supervisor have been completed or are currently in the process of being prepared, the Supervisor indicates that no work remains to be done.

```
Jan 31, 2025 3:26:19 AM Supervisor('Supervisor2') SendOrder on request
INFO: NO MORE ORDERS under SUPERVISOR: Supervisor2@192.168.1.137:1099/JADE
```

And at the end of the day Supervisors is waiting for the reports.

```
Jan 31, 2025 3:27:44 AM Supervisor('Supervisor2') FinishRaports on create
INFO: SUPERVISOR waiting for final reports
Jan 31, 2025 3:27:44 AM Supervisor('Supervisor1') FinishRaports on create
INFO: SUPERVISOR waiting for final reports
```

Reports:

```
INFO: ["REPORT OF: Supervisor1@192.168.1.137:1099/JADE for BAKER: Baker2@192.168.1.137:1099/JADE
 Num of successful necessary assistance of colleagues: 0
 Num of successful necessary restocks from suppliers: 0
 Orders done: 5 and orders needed to be redone: 0
 Stock left: [IngredientQuantity("flour", 5), IngredientQuantity("sugar", 16), IngredientQuantity("eggs", 16), IngredientQuantity("yeast",

", "REPORT OF: Supervisor1@192.168.1.137:1099/JADE for BAKER: Baker1@192.168.1.137:1099/JADE
 Num of successful necessary assistance of colleagues: 0
 Num of successful necessary restocks from suppliers: 0
 Orders done: 4 and orders needed to be redone: 3
 Stock left: [IngredientQuantity("flour", 7), IngredientQuantity("sugar", 18), IngredientQuantity("eggs", 18), IngredientQuantity("yeast",

"]
Jan 31, 2025 3:27:44 AM Supervisor('Supervisor1') FinishRaports on execute
INFO: END OF WORK
Jan 31, 2025 3:27:44 AM Supervisor('Supervisor2') FinishRaports on execute
INFO: ["REPORT OF: Supervisor2@192.168.1.137:1099/JADE for BAKER: Baker3@192.168.1.137:1099/JADE
 Num of successful necessary assistance of colleagues: 0
 Num of successful necessary restocks from suppliers: 0
 Orders done: 3 and orders needed to be redone: 1
 Stock left: [IngredientQuantity("flour", 7), IngredientQuantity("sugar", 14), IngredientQuantity("eggs", 11), IngredientQuantity("yeast",

", "REPORT OF: Supervisor2@192.168.1.137:1099/JADE for BAKER: Baker4@192.168.1.137:1099/JADE
 Num of successful necessary assistance of colleagues: 0
 Num of successful necessary restocks from suppliers: 0
 Orders done: 4 and orders needed to be redone: 1
 Stock left: [IngredientQuantity("flour", 1), IngredientQuantity("sugar", 14), IngredientQuantity("eggs", 12), IngredientQuantity("yeast",

", "REPORT OF: Supervisor2@192.168.1.137:1099/JADE for PACKER: Packer1@192.168.1.137:1099/JADE
 Packages with their prep time: PackerRaport([PackagePreparation("Pcg1", PT4.968S), PackagePreparation("Pcg2", PT15.016S)])"]
Jan 31, 2025 3:27:44 AM Supervisor('Supervisor2') FinishRaports on execute
INFO: END OF WORK
```

References

[1] Vivek Kumar, Dr. S. Srinivasan,2010. IJCSI International Journal of Computer Science Issues Vol.7,Issue 5, pp. 198.

[2] Colin Masson, 2025. Unlocking Supply Chain Potential with AI Agents and Multi-Agent Workflows.

[3] Giuseppe Petrosino 2022. Jadescript Programmer’s Guide Version 1.0.

[4] Jadescript language guide: https://github.com/aiagents/jadescript/blob/main/README.md