UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Second Practical Assignment: Design of a multiagent system

Submitted by:
Bonifacio Marco Francomano
Marc Monfort
Mario Rosas
Michelangelo Stasi

Date:
May 29th, 2024

Course/Subject:
Multiagent System Design

Instructor:
Javier Vazquez Salceda

Institution:
Universitat Politecnica de Catalunya

Introduction

In this report, we outline the design of our multiagent system using the ***Prometheus software methodology***. Our system, the ***Alien Trading Corporation***, is designed to facilitate interstellar trade between an alien planet and Earth. This report details the specification phase of the Prometheus methodology, which is followed by the *system specification phase, architectural design, detailed design, and implementation phases*.

Chapter 1: System Specification Phase

High-Level Description of the System scenario

The Alien Trading Corporation operates as an interstellar entity specializing in the sourcing, manufacturing, and selling of unique products from an alien planet to Earth-based customers. The corporation is responsible for the entire trade process, from extracting raw materials on the alien planet to delivering finished products to customers on Earth. The system manages various activities, including exploration, manufacturing, trade negotiations, and logistics management, to ensure efficient and profitable operations.

Identification of Actors

The primary actors in our system are the ***Alien Trading Corporation*** and ***Earth-based Customers***. The Alien Trading Corporation represents the organization responsible for sourcing, manufacturing, and selling alien products. Earth-based Customers are the individuals or entities on Earth who browse and purchase products offered by the corporation.

Top-Level Scenarios

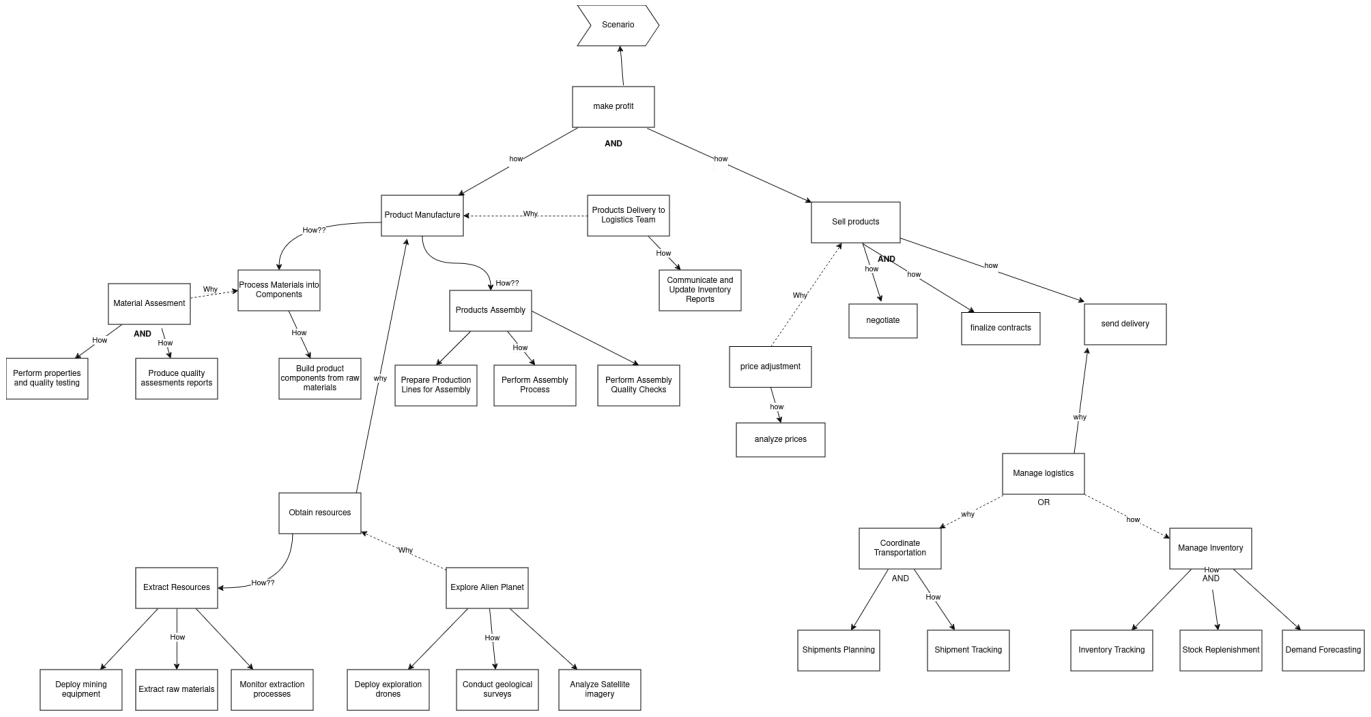
For the Alien Trading Corporation, the top-level scenarios include ***resource extraction, manufacturing, trade, and logistics***. In the *resource extraction scenario*, the corporation sources and extracts raw materials from the alien planet. This involves inputs such as exploration data and extraction plans, and results in the output of extracted raw materials. The *manufacturing scenario* sees the corporation transforming these raw materials into finished products. Inputs here include the extracted raw materials and manufacturing specifications, leading to the output of finished products.

In the *trade scenario*, the corporation negotiates trade agreements with Earth-based entities. This involves market analysis reports and negotiation offers as inputs, resulting in trade agreements as outputs. The *logistics scenario* focuses on managing the transportation and delivery of products to Earth. Inputs such as transportation info and delivery schedules result in delivered products and transportation status updates.

For Earth-based Customers, the top-level scenarios include product selection and purchasing. In the product selection scenario, customers select a product from the catalog of alien products offered by the corporation. The purchasing scenario deals with customers buying alien products, with inputs including selected products and payment information, and outputs like purchase confirmations and delivery schedules.

Goal overview diagram

In the next step of the specification phase, we identified goals for each sub scenario by creating four goal sub-diagrams—one for each top-level scenario. These diagrams captured the specific objectives within resource extraction, manufacturing, trade, and logistics. Subsequently, we integrated these individual diagrams into a ***unified goal diagram*** for the entire system. This unified representation contains all aspects of the Alien Trading Corporation's objectives, and it's needed to guide the subsequent phases of system development.



To construct the unified goal diagram, we began by establishing the overarching goal of "*make profit*" as the root. This global objective served as the foundation, reflecting the **primary aim** of the Alien Trading Corporation's operations.

Next, to ensure a balanced distribution of abstraction levels across the diagram, we strategically organized the sublevels under the root goal. We identified "*sell*" and "*product manufacture*" as two key sublevels closely tied to achieving profitability. "Sell" represents the objective of marketing and distributing products, while "produce" encompasses the process of obtaining and manufacturing goods.

Under the "produce" sublevel, we further refined the hierarchy by including "*obtain resources.*" This goal highlights the important step of acquiring raw materials necessary for manufacturing. Simultaneously, under the "sell" sublevel, we integrated "*logistics*" connected with the "s*end delivery*" goal by a "why" arrow. Additionally, the "send delivery" subgoal directly connects to the "sell" scenario, since it is one of the modality ("*how*") with which we achieve the sell goal . By structuring the

diagram in this manner, we achieved a coherent and hierarchical representation that effectively captures the diverse objectives and interdependencies within the system.

Functionalities of the system and development of sub-scenarios

After defining the goals for each sub-scenario, the next step was to identify the functionalities of the system. We designed these functionalities to accomplish all the subgoals of each sub-scenario. For example, in the Resource Extraction Scenario, we managed exploration and exploitation activities on the alien planet. This involves functionalities such as exploring the planet to identify resource-rich areas and extracting resources using mining techniques.

In the Product Manufacturing Scenario, the system functionalities include processing raw materials into usable forms and assembling finished products. These functionalities are needed for the transformation of raw materials sourced from the alien planet into final products.

Under the Trade Negotiation Scenario, functionalities include establishing trade agreements and setting pricing strategies based on market demand and competition. This enables negotiation and decision-making to maximize the profit.

In the end, in the Logistics Management Scenario, functionalities involve coordinating transportation and managing inventory levels. This ensures timely delivery of products to Earth-based customers while maintaining the control of inventory levels.

For each scenario, detailed steps were developed to provide a better understanding of how these functionalities are utilized. These detailed scenarios and functionalities provide a complete overview of the system's operations, facilitating a deeper understanding of the roles of agents and the utilization of functionalities in achieving the goals of each sub-scenario.

FUNCTIONALITIES:

Resource Extraction **Scenario**:

Explore Alien Planet

Description: Conduct exploration activities to identify potential resource-rich areas on the alien planet.

Percepts: Geological surveys, mineral composition data, terrain mapping.

Actions: Deploy exploration drones, conduct geological surveys, analyze satellite imagery.

Data Used: Geological data, satellite imagery, environmental sensors.

Interaction: Collaboration with geologists, coordination with exploration teams.

Extract Resources

Description: Utilize mining and extraction techniques to gather raw materials from the identified resource-rich areas.

Percepts: Resource availability, mining site conditions, equipment status.

Actions: Deploy mining equipment, extract raw materials, monitor extraction processes.

Data Used: Resource availability reports, equipment performance data, environmental impact assessments.

Interaction: Coordination with mining operators, communication with extraction teams.

Overview:

An agent in charge of resource extraction on an alien planet, oversees the exploration and exploitation of minerals and other valuable resources. This involves identifying potential sites through extensive surveys and then extracting the materials efficiently and responsibly.

Context:

Assumes the alien planet has been initially surveyed, and the environment is suitable for deploying drones and mining equipment. The ecosystem's impact is considered with every action taken.

Steps:

1. EVENT SiteSurvey (-> Satellite Imaging)
    a. ACTION DeployDrones (Satellite Imaging -> Exploration Team)
    b. Data read: Geological data, environmental sensors
    c. Data write: Survey reports, mineral composition data
2. ReviewSurveyData (Exploration Team -> Analysis Department)
    a. Data read: Survey reports
    b. Interaction: Geologists analyze the data for potential resource-rich sites
3. EVENT ResourceIdentification (-> Analysis Department)
    a. ACTION PlanExtraction (Analysis Department -> Operations Team)
    b. Data read: Mineral composition data
    c. Interaction: Operations Team plans the extraction process
4. ACTION DeployMiningEquipment (Operations Team -> Site)
    a. Data read: Resource-rich site coordinates
    b. Data write: Equipment deployment logs
5. ACTION StartExtraction (Site -> Resource Collection)
    a. Data read: Equipment status, mining site conditions
    b. Data write: Resource extraction logs, environmental impact assessments
6. MonitorExtractionProcess (Resource Collection -> Operations Team)
    a. Data read: Extraction progress, environmental impact data
    b. Data write: Continuous impact assessments, process optimization data

7. ACTION AnalyzeExtractionData (Operations Team -> Analysis Department)
    a. Data read: Resource extraction logs
    b. Interaction: Analysis for efficiency and sustainability
8. EVENT ExtractionCompletion (-> Operations Team)
    a. ACTION ReportExtractionSuccess (Operations Team -> Headquarters)
    b. Data read: Final extraction reports
    c. Interaction: Communicate successful completion and next steps

Product Manufacturing **Scenario**

Process Raw Materials

Description: Convert the extracted raw materials into usable forms suitable for manufacturing.

Percepts: Raw material inventory, processing capabilities, quality control metrics.

Actions: Crushing, milling, refining, chemical processing.

Data Used: Raw material composition analysis, processing equipment specifications, quality assurance standards.

Interaction: processing engineers, coordination with quality control inspectors.

Assemble Products

Description: Combine processed materials to manufacture finished products ready for sale.

Percepts: Component inventory, production schedules, assembly line efficiency.

Actions: Assembly line operations, quality control checks, packaging.

Data Used: Product designs, assembly instructions, inventory management systems.

Interaction: Coordination with assembly line workers, communication with quality assurance,

Overview:

Transforming raw materials sourced from the alien planet into finished goods. This process involves multiple manufacturing stages, from initial material processing to final assembly and quality checks. Agents in this scenario work closely with agents from Resource Extraction and Trade Negotiation scenarios to ensure materials are appropriately processed and products are assembled according to the demands set by market dynamics and customer requirements.

Context:

Assumes alien raw materials are already on-site and have been tested for quality and suitability. It operates under the assumption that manufacturing facilities are equipped with advanced machinery capable of handling various material types and complexities. The environment prioritizes sustainability and efficiency, closely monitoring waste production and energy consumption.

Steps:

1. EVENT: MaterialAssesment (--> ResourceExtraction reports)
    a. ACTION: AssessMaterialQuality (Raw Material Inventory -> Quality Assurance)
    b. Data read: Raw material composition analysis
    c. Data write: Quality assessment reports
2. ACTION: ProcessMaterials (Quality Assurance -> Processing Department)
    a. ACTION: BuildComponents (QualityAssesmentReports -> Manufactured Components)
    b. Data read: Processing equipment specifications
    c. Data write: Processed materials logs, quality metrics
3. ACTION: PrepareForAssembly (Processing Department -> Assembly Line Prep)
    a. Data read: Processed materials logs
    b. Data write: Assembly readiness reports
4. EVENT: InitiateAssembly (-> Assembly Department)
    a. ACTION: AssembleProducts (Assembly Line Prep -> Production Line)
    b. Data read: Product designs, assembly instructions
    c. Data write: Finished product logs, quality control reports
5. ACTION: PerformQualityChecks (Production Line -> Quality Control)
    a. Data read: Finished product logs
    b. Data write: Quality approval, packaging instructions
6. ProductsReadyForDelivery (-> Logistics Management Scenario)
    a. ACTION: TransferToLogistics (Quality Control -> Logistics)
    b. Data read: Quality control reports, packaging instructions
    c. Data write: Shipping orders

Trade Negotiation Scenario

Establish Trade Agreements

Description: Negotiate and finalize trade agreements with Earth-based entities for the sale and distribution of alien products.

Percepts: Market demand, competitor pricing, negotiation offers.

Actions: Negotiating terms, set_price, finalizing contracts.

Data Used: **MarketDB, negotiation_history**.

Interaction: Communication with (Earth) customers

Set Pricing Strategies

Description: Determine appropriate pricing strategies for the alien products based on market demand and competition.

Percepts: Market trends, price ranges, cost of production.

Actions: Pricing analysis, competitor benchmarking, price_adjustment.

Data Used: **MarketDB, competitor_data**

Interaction: sales_info

Overview: The customers ask for a product. A market analysis is performed. The manager compares the offer with the report and decides to confirm or propose another price. The negotiation history is updated until the two parts agree with the price.

Context: Assume the products were already assembled and they are ready to be sold

Steps:

1. EVENT ProductOrder (->Communication with customers)
2. AvailabilityCheck(transport_manager -> Communication_with_customers)
3. NotifySalesEXperts(Communication with customers ->sales_info)
4. ACTION Analyze Market (sales_info)
   data read: marketDB, competitor_data
   writes_data: marketDB
5. ACTION Negotiate_terms(Communication with customers)
   data read: marketDB, negotiation_history
   writes_data: negotiation_history
6. ACTION set_price(Communication with customers)
   data_read: negotiation_history
7. EVENT ReceiveFeedback(->Communication with customers)
8. emailTransportTeam(communication with_customers -> transport_manager)

Variations: If at step 7 we receive another offer from the customers, we GOTO step 5 again.

Logistics Management **Scenario**

Coordinate Transportation

Description: Organize the transportation and delivery of manufactured products from the alien planet to Earth.

Percepts: Shipping companies, information of order to process.

Actions: Shipments planning, shipment tracking.

Data Used: Shipment manifests, transportation schedules,orders info.

Interaction: Coordination with shipping companies, communication with the customers (notifications on the status).

Manage Inventory

Description: Monitor and manage inventory levels to ensure adequate stock availability for timely delivery to customers.

Percepts: Inventory levels, demand forecasts, stock replenishment needs.

Actions: Inventory tracking, stock replenishment, demand forecasting.

Data Used: Inventory management systems, sales data, demand forecasts.

Interaction: Collaboration with procurement teams, coordination with production schedules

Overview: When a new order is processed, transport_manager does this : the product is deleted from the inventory and the forecasting is updated. If this type of product needs to be replenished, it is done by communicating with the manufacturing_manager.
With the information given by the trade_manager, the transport_manager coordinates transportation by adding the product to a shipping order. After having planned the route, the transport_manager gets in touch with a shipping company to coordinate the order. The customer is notified about the status of the delivery.

Context: It is assumed the consistency of the information given by the trade_manager about the order. It is also assumed that it's always possible to get in touch and coordinate a delivery with an external shipping company. The Logistics Managers Agents can have two possible roles : inventory_manager and transport_manager.
(**Two possible events, here only the longest one is described**)

Steps:

1. OrderToShip(communication_with_customers)
2. AddOrder(communication_with_customers)
   - data write : OrdersDB
3. TrackInventory(prod_id):
   - data read : OrdersDB
4. UpdateForecast(prod_id):
   - data read : forecasts_per_prodDB
   - data write : forecasts_per_prodDB
5. CheckAvailability(prod_id):
   - data read : ready_to_deliver_products_DB

If the replenishment of the given product is necessary :

6. ACTION replenishProd(prod_id -> manifacturing_manager)
   End_if
7. CreateShipping(communication_with_customers -> shipping_info)
8. LookForAShippingCompany(-> shipping_company)
   - read data : shipping_manifestsDB
9. ACTION AskForAShipping(shipping_info -> shipping_company)
10. CheckForAvailableShipping(shipping_company -> shipping_info)

If the shipping is not available for this company, get back to 7 , else :

11. ACTION shipment_tracking(shipping_info -> customers)

Chapter 2: Architectural Design Phase

Data Coupling Diagram

In this phase, we develop a Data Coupling Diagram to organize system functionalities based on their association with common domains of action or shared databases.

Purpose

The Data Coupling Diagram serves to:

1. **Group Functionalities**: Organize functionalities related to similar actions or databases.
2. **Identify Agent Types**: Clearly identify agent types responsible for specific tasks.
3. **Promote Modularity**: Facilitate system maintenance and updates by promoting modularity.

Methodology

We group functionalities identified in the previous phase based on their domain of action or shared databases.

In the Data Coupling Diagram, the grouping of functionalities associated with the ***Trade Manager*** was done carefully to ensure coherence and effectiveness. Specifically, the "***Negotiate Terms***" and "***Market Analysis***" functionalities were linked because they both access two crucial databases: ***marketDB*** and ***competitor_data***. Both of these functionalities read from and write to these databases to gather market insights and competitor information, essential for effective negotiation strategies.

Additionally, the "***Finalize Contract***" functionality was grouped with these two because it reads from the ***negotiation history database***. This grouping was based on the understanding that the responsibility of finalizing contracts aligns closely with the negotiation process, which is overseen by the manager.

The result is straightforward and intuitive. By grouping these functionalities together under the Trade Manager, it becomes evident that this agent type is responsible for overseeing all aspects of trade negotiation, from market analysis and negotiation to contract finalization. This clear delineation makes it easy to assign a name to this type of agent and understand its role within the system.

Concerning the ***Logistic Manager***, the data and activities grouped represent the core functions necessary for efficient logistics and supply chain management.

Orders are the starting point for logistics activities. Managing and processing orders efficiently ensures that customer demands are met. Therefore, Process Order functionality is directly tied to the orders received, since the manager needs to ensure orders are processed correctly to avoid delays and errors in the supply chain.

The planning of a shipment and its related data are crucial for the logistic process, since they represent the actual movement of goods. It ensures that the manager can organize and schedule shipments in a way that aligns with order processing and inventory levels.

As we have just said, it is important to plan the shipment based also on the inventory levels; keeping track of inventory is important for knowing stock levels, avoiding stock outs, and ensuring that there is enough product to meet demand. **Even if this functionality reads from the "*Ready to deliver products DB*", it is quite straightforward that the Logistic manager is responsible for tracking inventory, since it also reads from other 3 db of the other functionalities.** This functionality is linked to a lot of data since the manager needs real-time data on inventory to make informed decisions about order processing and shipment planning.

Sales per Product data is critical for understanding demand patterns and trends; therefore it is strictly connected to the forecasting of the products.

Finally, the Logistic Manager is responsible for coordinating these activities to ensure the smooth flow of goods from suppliers to customers.

The **Resource Extraction Manager** was done carefully to ensure coherence and effectiveness. Specifically, the "Site Exploration" and "Resource Extraction" functionalities were linked because they both access crucial data sources: geological surveys and mineral composition data. Both of these functionalities read from and write to these databases to gather insights on resource-rich areas and monitor extraction processes, essential for efficient resource extraction.

Additionally, the "Environmental Impact Management" functionality was grouped with these two because it reads from the environmental sensors data and extraction logs. This grouping was based on the understanding that the responsibility of assessing and mitigating environmental impacts aligns closely with the exploration and extraction processes, which are overseen by the Resource Extraction Manager.

The result is straightforward and intuitive. By grouping these functionalities together under the Resource Extraction Manager, it becomes evident that this agent type is responsible for overseeing all aspects of resource exploration, extraction, and environmental management. This clear delineation makes it easy to assign a name to this type of agent and understand its role within the system.

Keeping track of resources is important for knowing stock levels, avoiding stock outs, and ensuring that there is enough raw material to meet manufacturing demands. Even if this functionality reads from the “Ready to deliver resources DB,” it is quite straightforward that the Resource Extraction Manager is responsible for tracking extracted resources, since it also reads from other databases of the other functionalities. This functionality is linked to a lot of data since the manager needs real-time data on resource levels to make informed decisions about extraction and transport planning.

Finally, the Resource Extraction Manager is responsible for coordinating these activities to ensure the smooth flow of raw materials from the extraction sites to the manufacturing facilities.

The role of the **Manufacturing Manager** Agent is designed to meet the corporation's product manufacturing needs. The pace and thoroughness of their work are tied to the production requirements generated by sales activities carried out by the Trade Manager. However, their work is also dependent on coordination with the Resource Extraction Manager, who ensures the availability of sufficient materials for manufacturing the requested orders.

To perform their tasks, the agent must conduct various validations and maintain communication and coordination with other agents to successfully fulfill their responsibilities.

The product manufacturing process consists of two main subprocesses:

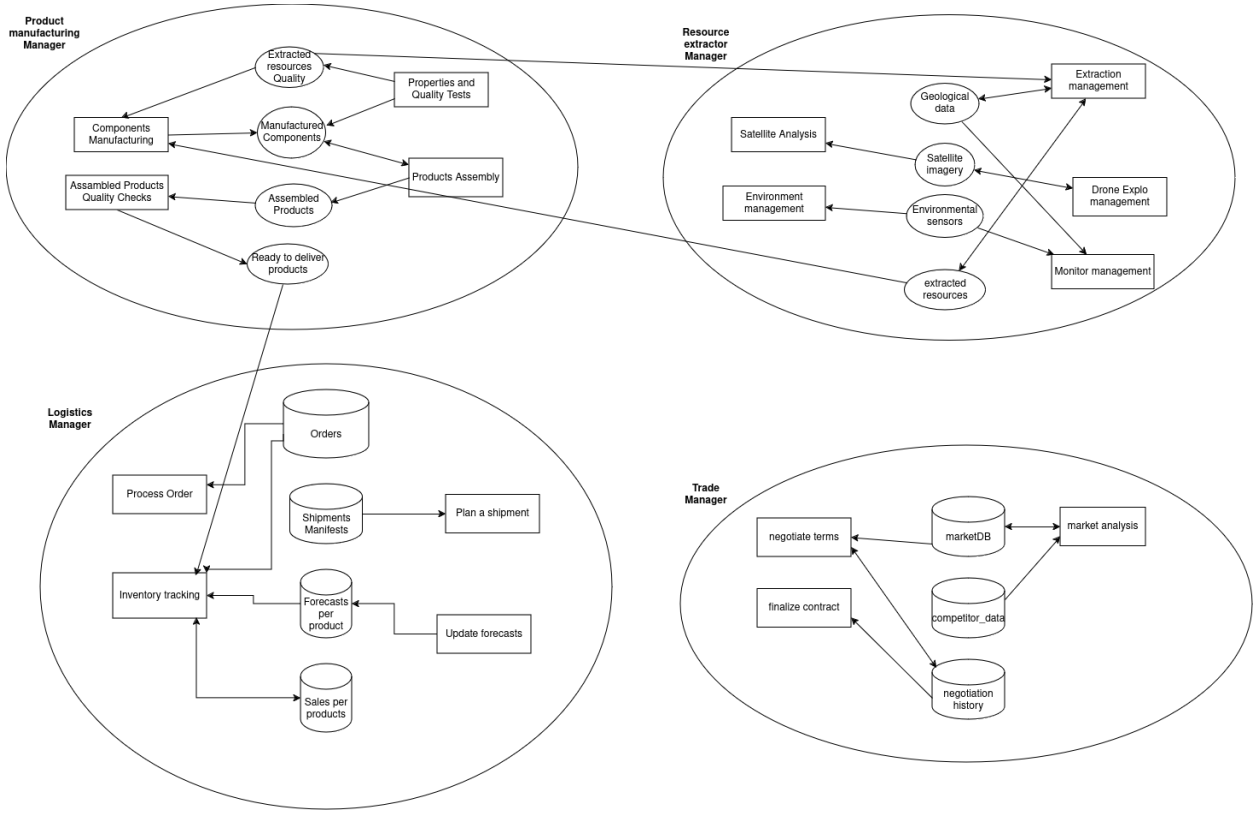
- Component Manufacturing: This initial stage involves creating necessary components from resources extracted by the Resource Extraction Manager.
- Product Assembly: Once the components are manufactured, they are assembled to form the final product.

Before commencing component manufacturing, the agent ensures the quality of the extracted resources and verifies their suitability for the intended products. Upon receiving the extracted resources, the agent conducts quality tests and records the results in a dedicated database for subsequent analysis.

Once a product or a batch of products is manufactured, the agent records it in the database of manufactured components. Quality validations are also conducted on these components. After validating the quality, the final product assembly takes place, and the assembled products are registered in a separate database.

Following the registration of newly assembled products, final quality checks are performed. The product's record then moves from the assembled products database to the ready-for-delivery database. This step is crucial as it serves as a communication channel with the Logistics Manager, who prepares the product for customer delivery.

The data recorded in the quality testing database for extracted resources is vital as it serves as an indirect communication channel with the Resource Extraction Manager. This manager consults the quality test results to make necessary adjustments in the extraction processes, if needed.



Agent descriptors (roles):

From the descriptions provided, we have generated four straightforward agent descriptors:

Resource Management Group:

- **Agent Type:** Resource Management Team
- **Description:** Coordinates resource extraction and processing, ensures sustainability, and manages the supply chain.
- **Cardinality:** One team per extraction site.
- **Lifetime:** Instantiated at the beginning of resource exploration, remains active throughout extraction.
- **Demise:** Deactivated upon resource depletion or termination of operations.
- **Initialisation:** Configures parameters based on survey data, connects to central database.
- **Functionalities Included:** Resource Extraction, Processing Raw Materials.
- **Uses Data:** Geological surveys, mineral composition data, environmental impact reports.

- **Produces Data:** Extraction schedules, processed material logs, sustainability reports.
- **Goals:** Efficient extraction, environmental responsibility, steady supply, safety compliance.
- **Interactions:** Environmental Monitoring, Extraction Equipment, Market Analysis, Logistics.

Manufacture Management Group:

- **Agent Type:** Production Management Team
- **Description:** Supervises raw material processing and product assembly, coordinates with resource and logistics.
- **Cardinality:** One team per manufacturing facility.
- **Lifetime:** Instantiated at production onset, remains active with available resources.
- **Demise:** Deactivated upon resource depletion or cessation of operations.
- **Initialisation:** Configures protocols, connects to manufacturing database and logistics.
- **Functionalities Included:** Product Manufacturing, Assemble Products.
- **Uses Data:** Processed material logs, product designs, production schedules.
- **Produces Data:** Manufacturing reports, quality documents, inventory updates.
- **Goals:** Efficiency, product quality, meeting targets, adapting to market.
- **Interactions:** Resource Management, Quality Assurance, Logistics, Trade Management.

Trade Management Group:

- **Agent Type:** Trade Manager
- **Description:** Conducts market analysis and negotiates with customers.
- **Cardinality:** One per customer.
- **Lifetime:** Instantiated upon customer request, demises after trading.
- **Functionalities Included:** Trade Negotiation, Set Pricing Strategies.
- **Uses Data:** MarketDB, competitor_data, negotiation_history.
- **Produces Data:** Market reports, negotiation queue, offers.
- **Goals:** Optimal selling, data updates, negotiation facilitation.
- **Interactions:** Earth-based customers, Logistics Management.

Logistics Management Group:

- **Agent Type:** Logistic Manager
- **Description:** Manages inventory and plans order shipments.
- **Cardinality:** One per order.
- **Lifetime:** Instantiated upon event, demises upon order receipt.
- **Functionalities Included:** Coordinate Transportation, Manage Inventory.
- **Uses Data:** OrdersDB, forecasts_per_prodDB, shipping_manifestsDB.
- **Produces Data:** OrdersDB, forecasts_per_prodDB.

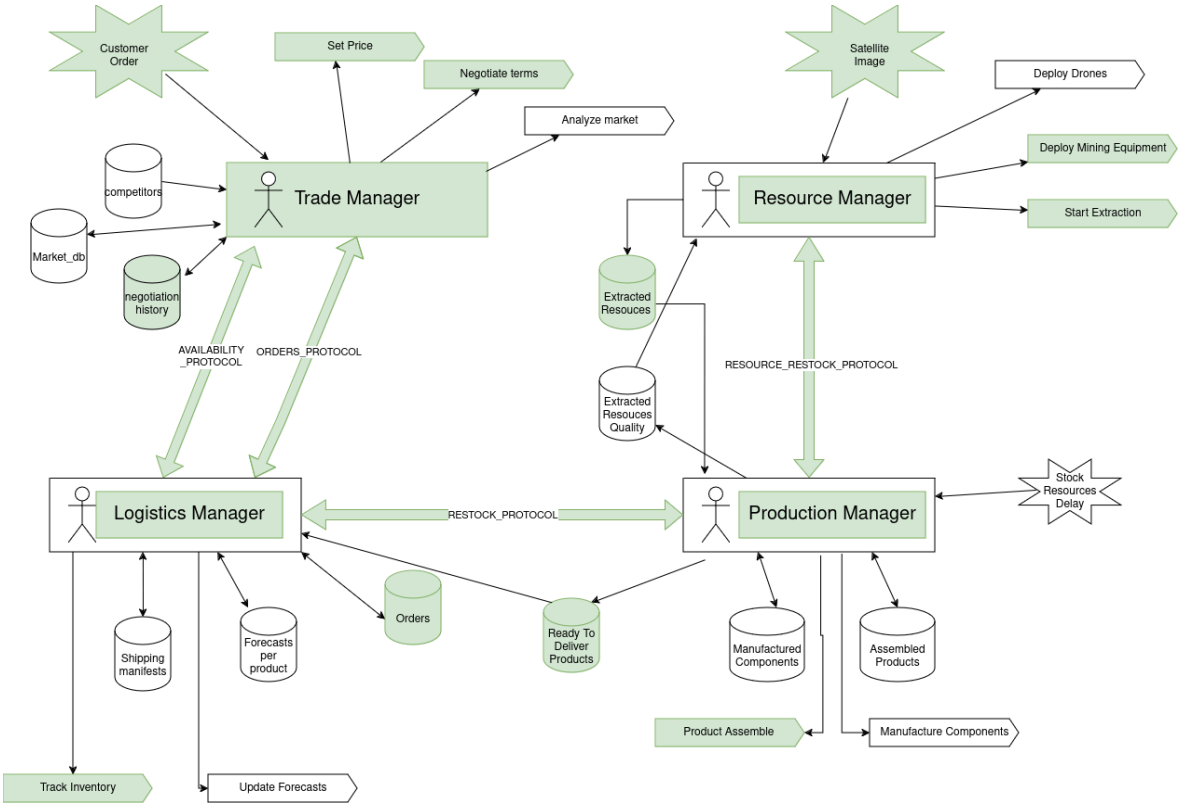
- **Goals:** Inventory management, shipment planning.
- **Interactions:** Earth-based customers, Production Management, Shipping Companies.

System overview diagram

The next step was to realize the System Overview Diagram, by putting together agents, databases, external action on the environment, perceptions, and protocols to make the agents communicate. Each protocol was chosen in relation to a subgoal: the ***orders protocol*** and the ***availability protocol*** between trade and logistic manager serves to accomplish the "*send delivery*" subgoal. the ***restock protocol*** between the logistic manager and manufacturing manager for the "*manage inventory*" subgoal.The **resource_restock_protocol** between the Manufacturing Manager and the resource manager is to accomplish the "*product manufacture*" subgoal. The resource manager and the Manufacturing Manager don't need to communicate directly with the trade manager. It's important to notice that the subgoals that are related to the protocols are the ones that unify the goal sub-diagrams of each scenario.

The choice of the **subpart of the system to implement for the demo** is slightly different to the proposed design in the data coupling diagram, indeed the Manufacturing Manager reads directly from the *Extracted resources database* **with the action *assemble products,* instead of passing though all the steps of manufacturing** (in principle the *component manufacturing* functionality should read from it).

(In **green** the subpart of the system that we decided to implement for our demo)



Chapter 3: Detailed design

Capability Descriptors of the Resource Extraction Manager Agent

Capability Descriptor: Site Exploration

Name: Site Exploration

External Interface to the Capability:

- Events Used/Produced: SiteSurvey, DeployDrones, ReviewSurveyData, ResourceIdentification.

Natural Language Description: This capability manages the exploration activities on the alien planet to identify resource-rich areas. It involves conducting site surveys, deploying exploration drones, and analyzing survey data to pinpoint potential extraction sites.

Interaction with Other Capabilities:

- This capability interacts with the Resource Extraction capability to provide data on potential extraction sites.
- It also communicates with environmental monitoring systems to ensure compliance with environmental regulations.

Data Used/Produced by the Capability:

- Data Used: Geological surveys, mineral composition data, satellite imagery.
- Data Produced: Survey reports, mineral composition data, potential extraction site coordinates.

Inclusion of Other Capabilities:

- This capability may trigger actions in the Resource Extraction capability to plan and initiate extraction operations based on the identified sites.

Capability Descriptor: Resource Extraction

Name: Resource Extraction

External Interface to the Capability:

- Events Used/Produced: PlanExtraction, DeployMiningEquipment, StartExtraction, MonitorExtractionProcess, AnalyzeExtractionData, ExtractionCompletion.

Natural Language Description: This capability oversees the extraction of resources from identified sites. It involves planning extraction operations, deploying mining equipment, and continuously monitoring and optimizing the extraction processes to ensure efficiency and environmental compliance.

Interaction with Other Capabilities:

- This capability interacts with the Site Exploration capability to obtain data on potential extraction sites.

Data Used/Produced by the Capability:

- Data Used: Resource availability reports, equipment performance data, environmental impact assessments, site coordinates.
- Data Produced: Extraction logs, equipment deployment logs, environmental impact assessments, final extraction reports.

Capability Descriptor: Environmental Impact Management

Name: Environmental Impact Management

External Interface to the Capability:

- Events Used/Produced: MonitorExtractionProcess, ContinuousImpactAssessment.

Natural Language Description: This capability manages the assessment and mitigation of environmental impacts during the resource extraction process. It involves monitoring extraction activities, assessing environmental impacts, and implementing measures to minimize negative effects on the alien planet's ecosystem.

Interaction with Other Capabilities:

- This capability interacts with both the Site Exploration and Resource Extraction capabilities to ensure environmental considerations are integrated into exploration and extraction activities.

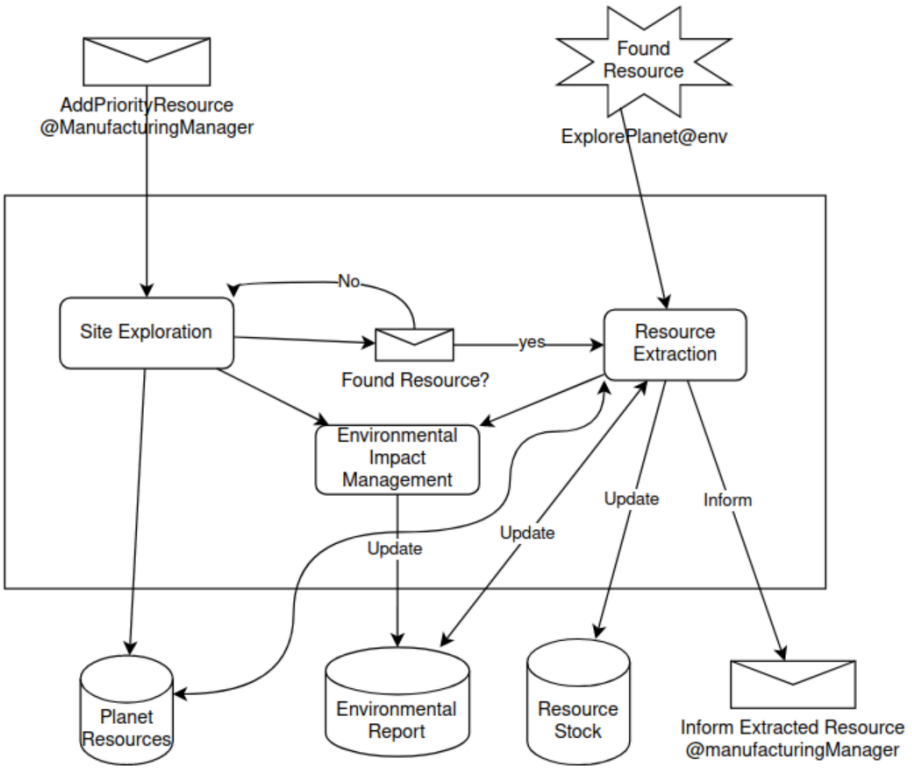
Data Used/Produced by the Capability:

- Data Used: Environmental sensors data, extraction logs, site conditions.
- Data Produced: Continuous impact assessments, environmental impact reports, process optimization data.

Inclusion of Other Capabilities:

- This capability may trigger actions in the Resource Extraction capability to adjust extraction methods based on environmental impact assessments.

Simplified Agent overview diagram



Plan Descriptor: Site Exploration

Name: Site Exploration

Natural Language Description: Manages the exploration of the alien planet to identify potential resource-rich areas, using advanced surveying techniques and data analysis.

Triggering Event Type: SiteSurvey

Plan Steps:

1. **DeployDrones**: Launch exploration drones to conduct surveys of the alien planet's surface.
    - **Data Used**: Satellite imagery, geological data.
    - **Data Produced**: Survey data.
2. **ReviewSurveyData**: Analyze the collected survey data to identify areas with potential resources.
    - **Data Used**: Survey data.
    - **Data Produced**: Mineral composition data, survey reports.

3. **ResourceIdentification**: Pinpoint and record coordinates of potential resource-rich sites.
    - **Data Used**: Survey reports.
    - **Data Produced**: Potential extraction site coordinates.

Context of Performing the Plan: Executed when a new site survey is initiated to explore and identify resource-rich areas on the alien planet.

Data Used/Produced:

- **Data Used**: Geological surveys, satellite imagery.
- **Data Produced**: Survey reports, mineral composition data, potential extraction site coordinates.

Plan Descriptor: Resource Extraction

Name: Resource Extraction

Natural Language Description: Manages the extraction of resources from identified sites, involving the deployment of mining equipment and continuous monitoring of the extraction process.

Triggering Event Type: PlanExtraction

Plan Steps:

1. **DeployMiningEquipment**: Set up mining equipment at the identified extraction sites.
    - **Data Used**: Site coordinates, equipment performance data.
    - **Data Produced**: Equipment deployment logs.
2. **StartExtraction**: Begin the extraction process at the designated sites.
    - **Data Used**: Equipment deployment logs.
    - **Data Produced**: Extraction logs.
3. **MonitorExtractionProcess**: Continuously monitor the extraction activities to ensure efficiency and compliance with environmental standards.
    - **Data Used**: Extraction logs, environmental impact data.
    - **Data Produced**: Environmental impact assessments, process optimization data.
4. **AnalyzeExtractionData**: Review extraction data to identify areas for process improvements.
    - **Data Used**: Extraction logs.
    - **Data Produced**: Process optimization data.
5. **ExtractionCompletion**: Finalize extraction operations and compile reports on the extracted resources.
    - **Data Used**: Extraction logs.
    - **Data Produced**: Final extraction reports.

Context of Performing the Plan: Executed when an extraction operation is planned and resources need to be extracted from identified sites.

Data Used/Produced:

- **Data Used**: Resource availability reports, equipment performance data.
- **Data Produced**: Extraction logs, equipment deployment logs, final extraction reports.

Plan Descriptor: Environmental Impact Management

Name: Environmental Impact Management

Natural Language Description: Manages the assessment and mitigation of environmental impacts resulting from resource extraction activities.

Triggering Event Type: MonitorExtractionProcess

Plan Steps:

1. **ContinuousImpactAssessment**: Monitor extraction activities to assess their environmental impact.
    - **Data Used**: Environmental sensors data, extraction logs.
    - **Data Produced**: Environmental impact data.
2. **AnalyzeImpactData**: Evaluate environmental impact data to identify potential risks and areas for improvement.
    - **Data Used**: Environmental impact data.
    - **Data Produced**: Environmental impact reports.
3. **ImplementMitigationMeasures**: Develop and apply measures to mitigate negative environmental impacts.
    - **Data Used**: Environmental impact reports.
    - **Data Produced**: Process optimization data, mitigation plans.

Context of Performing the Plan: Executed continuously to ensure that environmental impacts are assessed and mitigated during resource extraction operations.

Data Used/Produced:

- **Data Used**: Environmental sensors data, extraction logs.
- **Data Produced**: Environmental impact assessments, process optimization data.

Capability Descriptors of the Trade Manager Agent

Capability Descriptor: Product Order Handling

- **Name**: Product Order Handling
- **External Interface to the Capability**:
    - Events Used/Produced: ProductOrder, NotifySalesExperts, ReceiveFeedback.
- **Natural Language Description**:
    - This capability manages the processing of product orders received from customers. It involves notifying sales experts about incoming orders, and handling customer feedback regarding orders.
- **Interaction with Other Capabilities**:
    - This capability interacts with the Market Analysis capability to negotiate terms with customers and adjust pricing strategies based on market analysis and negotiation outcomes. It also communicates with the Logistic Manager to proceed with the delivery once the price is accepted.
- **Data Used/Produced by the Capability**:
    - Data Used: ProductOrder
    - Data Produced: NotifySalesExperts (regarding incoming orders), ReceiveFeedback (regarding order status and negotiation outcomes).
- **Inclusion of Other Capabilities**:
    - This capability may trigger actions in the Negotiation Management capability to adjust pricing strategies based on market analysis and negotiation outcomes.

Capability Descriptor: Market Analysis

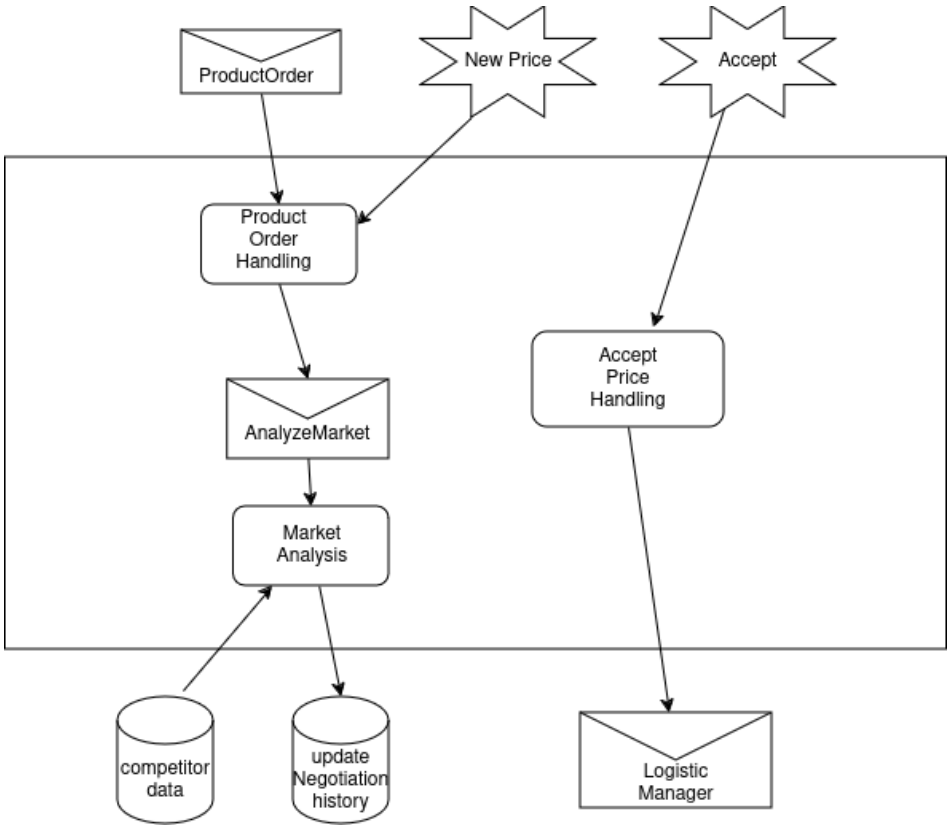
- **Name**: Market Analysis
- **External Interface to the Capability**:
    - Events Used/Produced: AnalyzeMarket, NegotiateTerms, SetPrice.
- **Natural Language Description**:
    - This capability determines pricing strategies for products based on market analysis and negotiation outcomes. It involves analyzing market data, negotiating terms with customers, and setting prices accordingly.
- **Interaction with Other Capabilities**:
    - This capability interacts with the Product Order Handling capability to adjust pricing strategies based on market analysis and negotiation outcomes.
- **Data Used/Produced by the Capability**:
    - Data Used: AnalyzeMarket, NegotiateTerms, negotiation_history.
    - Data Produced: SetPrice (updated pricing information).

- **Inclusion of Other Capabilities**:
    - This capability may trigger actions in the Product Order Handling capability to adjust pricing strategies based on market analysis and negotiation outcomes.

Capability Descriptor: Accept Price Handling

- **Name**: Accept Price Handling
- **External Interface to the Capability**:
    - Events Used/Produced: NegotiateTerms
- **Natural Language Description**:
    - This capability manages the acceptance of pricing terms by customers. It will send a message to the logistic manager advertising him to handle the ship of the product.
- **Interaction with Other Capabilities**:
    - This capability interacts with the Logistic Manager to proceed with the delivery once the price is accepted.
- **Data Used/Produced by the Capability**:
    - Data Used: negotiation_history
    - Data Produced: NotifyLogisticManager (regarding incoming orders),

Agent overview diagrams



Plan descriptors

Plan Descriptor: Product Order

- **Name**: Product Order
- **Natural Language Description**: Manages incoming product orders from customers and initiates subsequent actions to analyze the market, negotiate terms, and set prices.
- **Triggering Event Type**: ProductOrder
- **Plan Steps**:
    - **NotifySalesExperts**: Communicate with the sales information system to inform about the new order.
    - **AnalyzeMarket**: Assess market conditions, using marketDB and competitor_data, and update marketDB.
    - **NegotiateTerms**: Engage in negotiations with the customer, using negotiation_history, and update negotiation_history.
    - **SetPrice**: Determine the final price for the product based on negotiation outcomes, using negotiation_history.
    - **ReceiveFeedback**: Receive customer feedback on the proposed price and terms.
    - **EmailTransportTeam**: Notify the transport manager to proceed with delivery logistics if the price is accepted.
- **Context of Performing the Plan**: Executed when a new product order is received from a customer.
- **Data Used/Produced**:
    - Data Used: marketDB, competitor_data, negotiation_history.
    - Data Produced: marketDB (updated), negotiation_history (updated).

Plan Descriptor: Market Analysis

- **Name**: Market Analysis
- **Natural Language Description**: Conducts market analysis to determine current market trends, competitor pricing, and demand for products.
- **Triggering Event Type**: AnalyzeMarket
- **Plan Steps**:
    - **CollectMarketData**: Gather data from marketDB and competitor_data.
    - **AnalyzeTrends**: Analyze the collected data to identify market trends and pricing strategies.
    - **UpdateMarketDB**: Update marketDB with the latest market analysis results.
- **Context of Performing the Plan**: Executed when there is a need to assess market conditions as part of the product order handling process.

- **Data Used/Produced**:
    - Data Used: marketDB, competitor_data, negotiation_history
    - Data Produced: marketDB (updated), negotiation_history (updated)

Plan Descriptor: Accept Price

- **Name**: Accept Price
- **Natural Language Description**: Manages the acceptance of the negotiated price and initiates the delivery process.
- **Triggering Event Type**: AcceptPrice
- **Plan Steps**:
    - **ReceiveCustomerAcceptance**: Receive confirmation from the customer that they accept the final negotiated price.
    - **RecordAcceptance**: Update the negotiation history to reflect the customer's acceptance of the price.
    - **TriggerDeliveryCoordination**: Notify the logistics manager to begin the delivery process.
- **Context of Performing the Plan**: Executed when the customer accepts the negotiated price, signaling the need to proceed with order fulfillment.
- **Data Used/Produced**:
    - Data Used: negotiation_history.
    - Data Produced: Updated negotiation history, delivery initiation request.

Capability Descriptors of the Logistic Manager

Capability Descriptor : Track Inventory

- **Name:** Tracking Inventory
- **External interface to the capability:**
    - Events used : Order completed
    - Events produced : (eventually) Restocking request @manufacturingAgent
- **Natural language description:**
    - When the message from the Trade Manager concerning the order details is received, the inventory is tracked by looking at the data of the products, and, if it is necessary, the Logistic Manager will send a message to the Manufacturing Manager to notify the need of replenishment for a given product.
- **Interaction with other capabilities:** None
- **Data used/produced by the capability:** Inventory Data

- **Inclusion of other capabilities:**
    - This capability is also asking for replenishment when needed, therefore the capability of restocking the inventory is included.

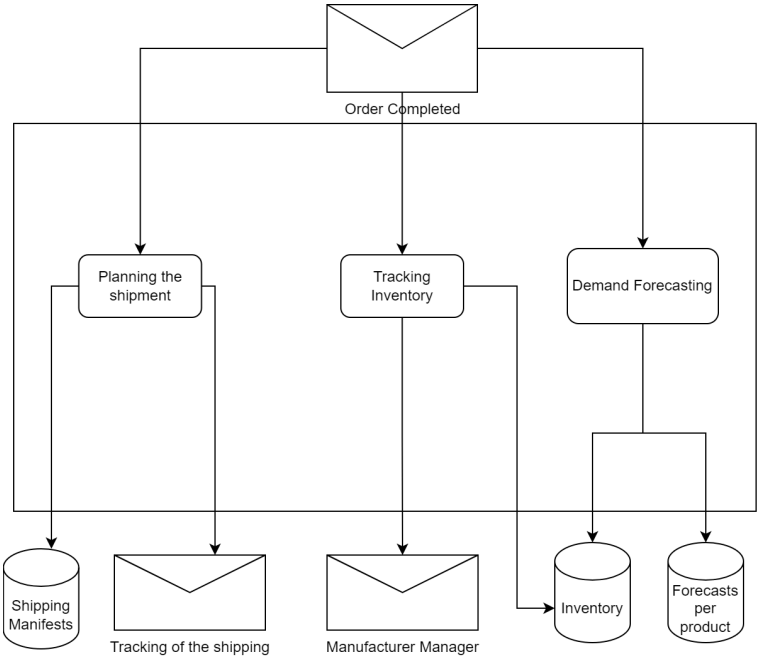
Capability Descriptor : Plan a Shipment

- **Name:** Planning a shipment
- **External interface to the capability:**
    - Events used : Order completed
    - Events produced : Shipment tracking
- **Natural language description:**
    - When the message from the Trade Manager concerning the order details is received, the Logistic Manager plans a shipment by getting in touch with a shipping company. Then, the Logistic Manager will notify the customer about the status of the shipment.
- **Interaction with other capabilities:**
    - This capability interacts with the environment to get the manifests of the shipping companies.
- **Data used/produced by the capability:**
    - It uses the data regarding the shipping companies and the order details.
- **Inclusion of other capabilities:** None

Capability Descriptor : Forecast

- **Name:** Demand Forecasting
- **External interface to the capability:**
    - Events used : Order completed
    - Events produced : Forecasts
- **Natural language description:**
    - Once the agent receives the details of the order, the agent can inspect the inventory data and can create demand forecasts for the products of the corporation and update the related database.
- **Interaction with other capabilities:** None
- **Data used/produced by the capability:**
    - Data used : It uses the data of the inventory.
    - Data produced : It produces new forecasts.
- **Inclusion of other capabilities:** None

Agent Overview Diagrams - Logistic Manager



Plan Descriptors of the Logistic Manager

Plan Descriptor : Tracking Inventory

- **Name :** Track Inventory
- **Natural language description :**
    - When the message from the Trade Manager concerning the order details is received, the inventory is tracked by looking at the data of the products, and, if it is necessary, the Logistic Manager will send a message to the Manufacturing Manager to notify the need of replenishment for a given product.
- **Triggering event type :** Order has been completed
- **Plan steps :**
    - Elaborate the order details
    - Update the inventory
    - If the new quantity of the product is under a certain threshold, notify the Manufacturing Manager
- **Context of performing the plan :**

    - Since the product related to the order has been sold, the inventory is not up-to-date.
- **Data used/produced :**
    - Data used : Order details and Inventory data
    - Data produced : Inventory updated

Plan Descriptor : Plan a Shipment

- **Name :** Plan a Shipment
- **Natural language description :**
    - When the message from the Trade Manager concerning the order details is received, the Logistic Manager plans a shipment by getting in touch with a shipping company. Then, the Logistic Manager will notify the customer about the status of the shipment.
- **Triggering event type :** Order has been completed
- **Plan steps :**
    - Elaborate the order details
    - Read the shipping manifest data
    - Find a shipping company
    - Finalize the shipment with the shipping company found previously
    - Notify the customer about the status of the shipment
- **Context of performing the plan :**
    - The order which has been completed has to be shipped to the customer.
- **Data used/produced :**
    - Data used : Order details and shipping manifest

Plan Descriptor : Demand Forecasting

- **Name :** Compute the demand forecasting
- **Natural language description :**
    - Once the agent receives the details of the order, the agent can inspect the inventory data and can create demand forecasts for the products of the corporation and update the related database.
- **Triggering event type :** Order has been completed
- **Plan steps :**
    - Elaborate the data of the order
    - Read the updated inventory and the demand forecasts for each product
    - Compute new analysis

    - Update the demand forecasting dats
- **Context of performing the plan :**
    - The order of a given product has been completed and therefore the analysis of the demand is not up-to-date.
- **Data used/produced :**
    - Data used : Order details
    - Data produced : Forecasts per product

Capability Descriptors of the Manufacturing Manager Agent

Capability Descriptor: Material Quality Assessment

- **Name**: Material Quality Assessment
- **External Interface to the Capability**:
    - Events Used: MaterialAssessmentReceived
    - Produced: QualityAssessmentCompleted
- **Natural Language Description**:
    - This capability involves assessing the quality of raw materials received for manufacturing. It ensures that materials meet specified standards before they proceed to the next stages of production. This involves testing material samples and evaluating them against quality metrics.
- **Interaction with Other Capabilities**:
    - This capability interacts with the Material Processing capability to ensure that only quality-assessed materials are used in production. It also publishes results to the Resource Planning capability to adjust supply chain actions based on material quality.
- **Data Used/Produced by the Capability**:
    - Data Used: Raw material samples.
    - Data Produced: Quality assessment reports

Capability Descriptor: Components Manufacturing

- **Name**: Components Manufacturing
- **External Interface to the Capability**:
    - Events Used: ComponentManufacturingEvent
    - Events Produced: ComponentManifacturingCompleted
- **Natural Language Description**:

    - This capability manages the transformation of quality-assessed raw materials into processed goods ready for component assembly. It involves operating machinery, scheduling production runs, and monitoring output to ensure efficiency and adherence to production standards.
- **Interaction with Other Capabilities**:
    - Interacts with the Component Building capability to provide processed materials necessary for component assembly. Also, it relies on data from the Material Quality Assessment to start processing.
- **Data Used/Produced by the Capability**:
    - Data Used: received resources, quality assessment reports.
    - Data Produced: Processed materials logs, manufactures components..

Capability Descriptor: Product Assembly

- **Name**: Product Assembly
- **External Interface to the Capability**:
    - Events Used: ProductAsemblyEvent
    - Produced: AssemblyPreparedEvent
- **Natural Language Description**:
    - This capability focuses on assembling processed components into final products. It involves detailed work to combine parts, often requiring precision and adherence to strict engineering specifications.
- **Interaction with Other Capabilities**:
    - It directly follows the Component Manufacturing capability to use the processed materials.
- **Data Used/Produced by the Capability**:
    - Data Used: Processed materials logs, component design specifications.
    - Data Produced: Assembled products inventory updates.

Capability Descriptor: Product Quality Checks

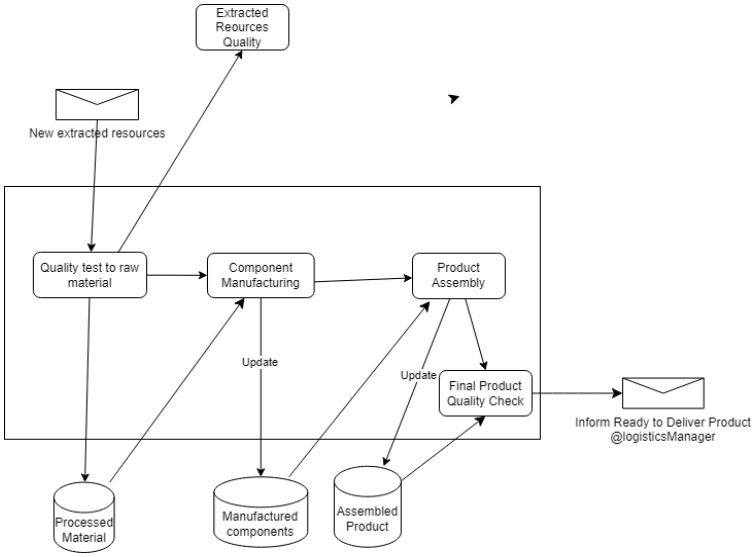
- **Name**: Product Quality Checks
- **External Interface to the Capability**:
    - Events Used: QualityCheckEvent
    - EventsProduced: PerformQualityChecks
- **Natural Language Description**:
    - This capability ensures that assembled products meet quality standards before they are cleared for logistics and delivery. It involves detailed

inspections, functional testing, and compliance verification with product specifications.

- **Interaction with Other Capabilities**:
    - This capability interacts with the Logistics Preparation capability to provide updates on product readiness and quality status. It is crucial for guaranteeing the final product quality before shipment.
- **Data Used/Produced by the Capability**:
    - Data Used: Product specifications, testing protocols.
    - Data Produced: Quality control reports, final product quality status.

Capability Descriptor: LogisticsDelivery

- **Name**: LogisticsDelibery
- **External Interface to the Capability**:
    - Events Used: QualityAssesedProductsEvent
    - Events Produced: TransferToLogistics
- **Natural Language Description**:
    - This capability manages the preparation for product delivery, ensuring that all logistics are in place for efficient distribution. It includes packaging, labeling, and coordination with transportation services.
- **Interaction with Other Capabilities**:
    - It is the final step in the manufacturing process, interacting with the Product Quality Checks to ensure only approved products are prepared for shipment.
- **Data Used/Produced by the Capability**:
    - Data Used: Quality control reports, shipping requirements.
    - Data Produced: Shipping orders, logistics schedules.



Plan Descriptor: Material Quality Assessment

- **Name:** Assess Material Quality
- **Natural Language Description:**
    - When raw materials are received, this plan is initiated to assess their quality. The Manufacturing Manager checks the materials against specified quality standards to ensure they meet production requirements. If the materials would not meet the required standards, the supplier is notified, and a request for replacement materials may be initiated.
- **Triggering Event Type:** Raw Materials Received
- **Plan Steps:**
    - Receive the raw materials.
    - Conduct quality tests as per the standards.
    - Update the material quality assessment record.
    - If materials fail the test, notify the supplier.
- **Context of Performing the Plan:**
    - This plan is activated upon the receipt of raw materials to ensure that only materials meeting quality standards proceed through the manufacturing process.
- **Data Used/Produced:**
    - **Data Used:** Raw material specifications, quality standards documents.
    - **Data Produced:** Material quality reports, supplier notifications (if needed).

Plan Descriptor: Material Processing

- **Name:** Process Materials
- **Natural Language Description:**
    - Upon successful material quality assessment, this plan processes raw materials into semi-finished goods suitable for assembly. The Manufacturing Manager oversees the operation of machinery and the workflow to ensure efficient material transformation.
- **Triggering Event Type:** Materials Approved for Processing
- **Plan Steps:**
    - Schedule the processing tasks.
    - Monitor the process for any discrepancies.
    - Update the materials processed records.
- **Context of Performing the Plan:**

    - This plan is executed once the raw materials have been approved, signaling readiness for processing into forms usable in further manufacturing stages.
- **Data Used/Produced:**
    - **Data Used:** Approved material quality reports.
    - **Data Produced:** Processed materials logs.

Plan Descriptor: Assembly Preparation

- **Name:** Prepare for Assembly
- **Natural Language Description:**
    - This plan prepares the processed materials for the assembly line. It involves setting up assembly stations, ensuring that all components and tools are ready, and scheduling assembly tasks. The Manufacturing Manager coordinates with the assembly team to streamline operations.
- **Triggering Event Type: Materials Processed**
- **Plan Steps:**
    - Set up assembly stations according to the product specifications.
    - Ensure all necessary components are available and ready.
    - Brief the assembly team on the product requirements and timelines.
    - Update the assembly preparation status.
- **Context of Performing the Plan:**
    - The readiness of processed materials triggers this plan, aiming to seamlessly transition them to assembly.
- **Data Used/Produced:**
    - Data Used: Processed materials logs, assembly instructions.
    - Data Produced: Assembly readiness reports.

Plan Descriptor: Quality Control

- **Name: Perform Quality Checks**
- **Natural Language Description:**
    - Following product assembly, this plan involves thorough quality checks to ensure that all products meet the predefined standards before they are cleared for shipping. The Manufacturing Manager supervises the quality control team in conducting tests and inspections.
- **Triggering Event Type:** Products Assembled
- **Plan Steps:**
    - Conduct functionality tests to ensure operational integrity.

    - Document the inspection results and update the quality control database.
    - If defects are detected, initiate corrective measures.
- **Context of Performing the Plan:**
    - This plan is crucial for maintaining product quality and is conducted after assembly, before the products are approved for delivery.
- **Data Used/Produced:**
    - **Data Used:** Assembly reports.
    - **Data Produced:** Quality final inspection reports.

Chapter 4: Implementation

We decided to implement a subset of the system, including all four roles. The implementation covers the following aspects:

- **Trade Manager**:
    - **Perception**: Customer orders injected by the environment randomly
    - **External Actions**: Set price and negotiate terms to complete negotiations.
    - **Databases:** Negotiation history
- **Logistics Manager**:
    - **External Action**: Track inventory.
    - **Databases**: "Ready to Deliver Products" DB and Orders DB to manage delivery statuses.
- **Manufacturing Manager**:
    - **External Action**: Product assembly.
    - **Database**: Writes to the "Ready to Deliver Products" DB.
- **Resource Extractor**:
    - **Databases**: Extracted Resources DB (shared with the Manufacturing Manager).
    - **External Actions**: Deploy extraction equipment and start extraction.
    - **Perception**: Satellite images from the environment.

Trade Manager Agent Implementation

In this section, we detail the refined implementation of the Trade Manager agent, focusing on its enhanced negotiation capabilities and improved handling of product sales.

Belief Updates

The belief updates have been adjusted to ensure proper termination of negotiations and enable the restarting of negotiations for previously sold products.

```
Unset
beliefupdates:
 {not sold(ProductID)} EndNegotiation(ProductID) {sold(ProductID)}
 {sold(ProductID)} SellAgain(ProductID) {not sold(ProductID)}
```

Additional Considerations

To maintain synchronicity between the agent's beliefs and actual negotiation history, the negotiation history database has been kept. This allows for more informed decision-making based on past negotiation experiences.

Plan Conditional Handling

The SellAgain(ProductID) plan condition addresses scenarios where previously sold products need to be sold again. This ensures effective management of products with existing negotiation histories.

Updated PC Rules

The PC rules have been modified to accommodate the revised implementation, ensuring appropriate responses to negotiation requests and acceptances.

```
Unset
pcrules:
 event(request(ProductID), env) <- true |
  if B(not sold(ProductID))
  {
   adopta(sold(ProductID));
  }
  else
  {
   SellAgain(ProductID);
   adopta(sold(ProductID));
  }

 event(accept(ProductID),env) <- not sold(ProductID) |
```

```
{
  EndNegotiation(ProductID);
  send(logisticManager, inform, sold(ProductID));
}
```

Implementation in Action

This updated implementation enhances the Trade Manager's negotiation processes and sales management capabilities. By incorporating a negotiation history database and refining plan conditional handling, the agent is better equipped to handle negotiation requests and product sales effectively.

Logistic Manager Agent Implementation

As seen before, at the end of the negotiation process, the Trade Manager will send a message to the Logistic Manager, which will start to operate when this message is received. In fact, in the pcrules the first rule starts when a message from the “tradeManager” is received. If the agent never sent this product, it will adopt the goal to send it and will track the inventory through the environment method. If it already sent this product, the belief is updated and it adopts the goal to send it. Once the inventory has been tracked, the environment could send to the agent a request concerning the stock out of this product; in this case, the manufacturing Manager is notified.

```
Unset
pcrules:
  message( tradeManager, inform, La, On, sold(ProductID)) <- true |
  if B(not sendproduct(ProductID)){
    adopta(sendproduct(ProductID));
    @env(trackInventory(ProductID), L);
  }
  else{
    -sendproduct(ProductID);
-,
    adopta(sendproduct(ProductID));
    @env(trackInventory(ProductID), L);
  }

  event(request(ProductID), env) <-true |
```

```
  {
    send(manufacturingManager, request, check(ProductID));
  }
```

Once the goal has been adopted, the agent will ship the product through the environment method and will update its belief.

```
Unset

beliefupdates:
  { not sendproduct(ProductID) } ProductSent(ProductID) {
sendproduct(ProductID) }

pgrules:
  sendproduct(ProductID) <- true |
  {
    @env( shipProd( ProductID ), L);
    ProductSent(ProductID);
  }
```

Resource Extractor Agent Implementation

The Resource Extractor Agent is responsible for exploring the alien planet, identifying resource-rich areas, and managing the extraction of these resources. The implementation involves updating beliefs based on events, adopting goals, and performing actions to achieve these goals.

```
Unset
beliefupdates:
      {availMiner} DeployMiners(Region) {miner(Region), not
availMiner}
      {miner(Region)} StopMiners(Region) {not miner(Region),
availMiner}

      {} FoundResource(ResourceID, Region) {resource(ResourceID,
Region)}
      {} ExhaustedResource(ResourceID, Region) {not
resource(ResourceID, Region)}
```

```
      {not ispriority} AddPriorityResource(ResourceID)
{priority(ResourceID), ispriority}
      {priority(ResourceID)} RemovePriorityResource(ResourceID) {not
priority(ResourceID), not ispriority}

beliefs:
      availMiner.

goals:
      explorePlanet.
      extractResources.
```

pgrules:

The plan generation rules (pgrules) for planet exploration dictate how the Resource Extractor Agent adopts plans based on its current beliefs and goals.

Unset

```
pgrules:

  explorePlanet <- not ispriority | @env( explorePlanet(), L);

  explorePlanet <- priority(ResourceID) | @env((ResourceID), L);

  mineResource(ResourceID) <- miner(Region) and resource(ResourceID,
Region) | @env( mineResource(ResourceID, Region), L);

  mineResource(ResourceID) <- resource(ResourceID, Region) and
availMiner | DeployMiners(Region);

  <- miner(Region) and not resource(_, Region) | StopMiners(Region);
```

pcrules:

The plan construction rules (pcrules) specify how the agent should respond to events and messages.

```
Unset

pcrules:

  event(foundResource(ResourceID, Region), env) <- true  |
FoundResource(ResourceID, Region);
adopta(mineResource(ResourceID));

  event(exhaustedResource(ResourceID, Region), env) <- true  |
ExhaustedResource(ResourceID, Region);
dropgoal(mineResource(ResourceID));

  message( manufacturingManager, request, La, On,
addPriority(ResourceID)) <- true | AddPriorityResource(ResourceID);

  message( manufacturingManager, inform, La, On,
removePriority(ResourceID)) <- true |
RemovePriorityResource(ResourceID);

  event( extractedQuantity(ResourceID, Quantity), env) <-
priority(ResourceID) | send(manufacturingManager, inform,
extracted(ResourceID, Quantity));
```

Manufacturing Agent Implementation

The ManufacturingManager agent is primarily responsible for overseeing the transformation of raw materials into finished products in a highly structured manufacturing environment. This includes managing quality assessments, processing materials, building components, preparing assemblies, and ensuring that finished products meet quality standards before they are dispatched for delivery.

```
Unset
beliefupdates:
    {not materialAssessed} receiveMaterialAssessmentReport
{materialAssessed},
    {not materialQualityAssessed} assessQuality
{materialQualityAssessed},
    {not materialsProcessed} processRawMaterials
{materialsProcessed},
    {not componentsBuilt} buildComponents {componentsBuilt},
    {not assemblyPrepared} prepareAssembly {assemblyPrepared},
    {not productsAssembled} initiateAssembly {productsAssembled},
```

pgrules:

The Procedural Goal Rules (PGRules) for the ManufacturingManager agent are crucial mechanisms that govern the sequential operations in the manufacturing process. These rules dictate how the agent transitions through various stages of manufacturing, from initial material assessment to the final preparation for logistics..

```
Unset
pgrules:

    assessMaterialQuality <- materialAssessed and not
materialQualityAssessed |

        {@env(assessMaterialQuality(ResourceID), L);}.

    processMaterials <- materialQualityAssessed and not
materialsProcessed |

        {@env(ProcessMaterials(), L);}.

    buildComponents <- materialsProcessed and not componentsBuilt |

        { @env(BuildComponents(), L);}.

    prepareForAssembly <- componentsBuilt and not assemblyReady |

        {@env(PrepareForAssembly(), L);}.

    assembleProducts <- assemblyReady and not productsAssembled |

        {@env(AssembleProducts(), L);}.

    performQualityChecks <- productsAssembled and not qualityChecked|

        { @env(PerformQualityChecks(), L);}.

    transferToLogistics <- qualityChecked and not readyForDelivery |

        {@env(TransferToLogistics(), L);}.
```

pcrules:

The plan construction rules (pcrules) specify how the Manufacturing Manager Agent should respond to events and messages.

```
Unset

pcrules:

 event(resourceAvaliable(ResourceID), env) <- true |

        {RegisterAvaliableMaterial(true);}.

    event(qualityAssessmentDone, env) <- true |

        {+materialQualityAssessed(true);}.

    message(logisticsManager, request, La, On,
productDeliveryRequest) <- readyForDelivery |

        {send(logisticsManager, inform, readyForDelivery); }.
```

Chapter 5 : Updated description of the Agent framework

Environment Java Class Definition

Our Environment (that we will call 'Env' to not confuse it with the class 'Environment' of 2apl java library) has to be a class which extends the 2apl 'Environment' class. This will give us several methods that we can use to communicate with the agents, and we can create methods that the agents can use to perform external actions.
To create the related jar file of the class 'Env', we need to add to the classpath the external jar of the 2APL Platform. Then, we can export the Java project as a runnable jar with main class the class 'Env'; therefore it is mandatory to have a main method in this class, since, event if we do not use this method, the jar file can point to this class as the main class, and this is only possible if the class contains main method.
Once created this file, we have to refer to it in the .mas file that specifies the components of the multiagent system. Remember that this file defines what environment to use, and what agents will exist.

Methods - addAgent(...)

This method is automatically called whenever an agent enters the MAS. Usually, one of the parameters is the name of the agent.

Objects

If we want to send information to a 2APL agent, we need to code this into special objects. We can then send these objects to the agent so that he can parse them correctly. All the objects extend the basic class "Term". We distinguish between the following objects:

- **APLNum** : this is equal to int and is for example instantiated by 'new APLNum(0)'
- **APLIdent** : this is equal to String, instantiated by 'new APLIdent("string")'
- **APLList** : this can be seen as a LinkedList and will be parsed as a Prolog list in 2APL
- **APLFunction** : it represents a function, where the arguments of the function again need to be Term objects. For example, the function: func(0) should be instantiated as new APLFunction("func", new APLNum(0))

External Actions

External actions of agents can be caught by defining methods that have a Term as return value. This method can be called by a 2APL agent as follows : \@env(func(0), X). X will now contain the return value.

Chapter 6 : Conclusions and last analysis

Analysis of the agent language

The 2APL documents approach this language from a very theoretical perspective ; therefore, one could think that the syntax of this language and its rules are not as strict as other languages. In the documents it is not specified how to create the environments and how to generate the associated JARs, which are fundamental to make the MAS work. However, looking at the examples, some guidelines were written directly in the code regarding the implementation of environments.

On the other hand, the syntax of the language was well explained in the documentation, but it has been difficult to understand when to use rules regarding the capitalization of the letters.
Another note is about the Eclipse Plug-In Editor, which is not working : moreover, one of the links in the documentation is not working. Even in the code of the examples it is said that the authors do not recommend the use of the Eclipse plugin since it is not completely bug-free.