Università di Genova | DIBRIS DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

Validation of UML-based Family of Modelling Methods for Business Process

Jukanna Nityanand

Master Thesis

Università di Genova, DIBRIS Via Opera Pia, 13 16145 Genova, Italy
https://www.dibris.unige.it/

Università di Genova

MSc Computer Science

Data Science and Engineering

Validation of UML-based Family of Modelling Methods for Business Process

Jukanna Nityanand

Advisor: Gianna Reggio Examiner: Maurizio Leotta

July, 2022

Abstract

The focus of this thesis is to study the family-based UML methods and evaluate them by modelling a collection of case studies covering all aspects of the business process. The evaluation is done by comparing the results within the family-based UML models. With the help of UML modelling features, all the aspects of the business process are precisely defined. Concerning the case studies, we have chosen the business processes modelled in BPMN (Business Process Modelling Notation). BPMN uses many constructs to model a business process. While modelling a complex process with BPMN; at some point, the diagram becomes unreadable and challenging to understand. The UML methods solve this issue by modelling each business process precisely using only a few constructs. The BPMN shows the workflow of the business process in a single view. Whereas our UML models not only show the workflow view but also allows the user to see the same business process with different views by increasing the understandability of the process.

To understand how the BPMN works, we have followed the official BPMN documentation (https://www.omg.org/spec/BPMN/2.0/PDF). After acquiring the knowledge of BPMN constructs, we have started modelling the business process by mapping the BPMN constructs to the constructs that were defined in our UML models.

While modelling the BPMN case studies with our UML models, we have discovered the problems in BPMN related to the inconsistency and ambiguity in defining the tasks, and we have solved them precisely.

A comparative study of the results has been done within the UML models against BPMN case studies concerning the various values such as producibility, readability, and understandability of the process.

Table of Contents

1 Introduction *8*

1.1 Motivation **8**

1.2 Objectives and Contributions **8**

1.3 Thesis Structure **8**

2 BPMN *8*

2.1 Introduction to BPMN **8**

2.2 Components of BPMN **9**

2.2.1 Elements 9

2.2.2 Activity Markers 12

2.2.3 Artifacts 12

2.2.4 Connecting Objects 13
2.2.5 Gateways 13
2.3 Pools and Lanes 14
2.4 Types of BPMN 14
2.4.1 Process Diagram 14
2.4.2 Collaboration Diagram 15
2.4.3 Choreography Diagram 15
2.4.4 Conversation Diagram 15
3 Holistic view of UML Methods 16
3.1 UML for modelling holistic models 17
3.1.1 Dealer network description 18
3.2 UM1 (Hard-Abstract tasks) Method 18
3.2.1 Entity view 19
3.2.2 Organization view 20
3.2.3 Process Overview 20
3.2.4 Task View 21
3.2.5 Workflow View 21
3.2.6 Rule view 22
3.3 UM2 (Hard – Executable tasks-Interaction) Method 23
3.3.1 Entity View 24
3.3.2 Interaction View 24
3.3.3 Organization View 24
3.3.4 Workflow View 25
3.4 UM3 (Hard- Executable task) Method 25
3.4.1 Entity view 25
3.4.2 Workflow View 26
3.5 Disciplined Modelling Methods (UM4 and UM5) 26
3.6 UM4 (Disciplined-abstract tasks) Method 26
3.6.1 Entity View 27
3.6.2 Task View 27
3.6.3 Workflow View 28
3.7 UM5 (Disciplined-Executable tasks) Method 28
3.7.1 Entity view 29
3.7.2 Workflow View 29
3.8 Selecting a best suitable model for business process 29
3.8.1 Shifting among the UML Models 30
4 BPMN Case studies 31
4.1 Shipment Process 31
4.1.1 The case study 31
4.1.2 UM1 model of the Shipment Process 32
4.1.3 UM4 model of the Shipment Process 33
4.1.4 Lessons learned 34
4.2 Pizza Selling 35
4.2.1 The case study 35
4.2.2 UM2 model Pizza Selling process with four participants 35
4.2.3 UM5 model Pizza Vendor process with four participants 38
4.2.4 UM2 model Pizza Selling process with two participants 39
4.2.5 UM5 model Pizza Selling process with two participants 41
4.2.6 Lesson learned 42
4.3 Incident Management Process 43
4.3.1 The case study 43
4.3.2 UM1 model of the Incident Management Process 44
4.3.3 Lessons Learned 45

4.4 Incident Management Detailed **45**
4.4.1 The case study 45
4.4.2 UM2 model Incident Management process 46
4.4.3 Lesson learned 47
4.5 Incident Management as Choreography **47**
4.5.1 The case study 48
4.5.2 UM1 model Incident Management Process (Way 1) 48
4.5.3 UM1 model Incident Management Process (Way 2) 51
4.5.4 Lesson learned 52
4.6 The Nobel Prize Process **53**
4.6.1 The case Study 53
4.6.2 UM2 model Nobel Prize Process 54
4.6.3 Lesson learned 55
4.7 Amazon Fulfilment **56**
4.7.1 The case study 56
4.7.2 UM2 model Amazon Fulfilment 57
4.7.3 UM1 model Amazon Fulfilment 58
4.7.4 Lesson learned 60
4.8 Recruitment and Selection **60**
4.8.1 The case study 60
4.8.2 UM1 model Recruitment and Selection Process 61
4.8.3 Lesson learned 65
4.9 Incorporation of New Employee in Hospital **65**
4.9.1 The case study 65
4.9.2 UM2 model Incorporation of New Employee process 67
4.9.3 Lesson learned 70
5 Conclusions ***70***
6 Bibliography ***70***

Table 1 HTML Version of Case studies

Shipment Process	Pizza Selling	Incident Management	Incident Management Detailed	Incident Management as Choreography	The Nobel Prize Process	Amazon Fulfilment	Recruitment and Selection	Incorporation of New Employee in Hospital
1)UM1 Model		1)UM1 Model Figure 6.1		1)UM1 Model Figure 6.3 2)UM1 Model Figure 6.3 Precise		1)UM1 Model Amazon Fulfilment	1)UM1 Recruitment and Selection	1)UM2 Model INE
	1) UM2 model with 4 participants 2) UM2 model with 2 participants		1)UM2 Model Figure 6.2		1)UM2 Model Noble Prize Process	1) UM2 Model Amazon Fulfilment		
1)UM4 Model								
	1)UM5 model with 4 participants 2)UM5 model with 2 participants							

Table 2 BPMN Issues (summary)

Issue	BPMN means	Negative	Positive	UML methods	Positive	Negative
Process participants	Pool + lanes	• Not always clear the difference between the participant rendered with pools and those with lanes • No-way to represented structuring of the enterprise enacting the process at three or more levels • Relative to the single process, no way to share explicitly the same structuring among different processes	Immediate visual presentation in the process/collaboration diagrams	Worker/system/organization unit classes in the entity view + organization view	• Defined at the level of the enterprise enacting the process • Could be shared by all the processes enacted by the same enterprise	• Cannot be rendered visually in the abstract-task methods (not always possible to use the swimnlanes)
Task participants	Seen in pools/lanes and in the tasks	• Not always clear, sometimes the names are confusing since they are named using verbs in active voice	• Rendered inside the task and modelled in natural language fragments		• Precisely defined and introduced separately in the entity view • Easy identification of task participants, while reading the workflow and since they are explicitly represented • Stereotype of participants give details description of their state (i.e., in or out)	• Views become cluttered when there are many participants and also modifications of participants become complex
Data and business objects		• Data and objects are rendered inside the task and sometime it is difficult to identify them • Not very clear overall definition of the business objects/ data.	• Easy to understand since they are modelled by textual annotations		• Easy identification of all objects/data and they are explicitly defined in the entity view	• data/business objects are not modelled in the same diagram containing the workflow view
Data flowing among tasks		• Tracking of data flowing in larger diagrams gets difficult since all the data is not represented explicitly	Data flowing is seen through constructs related to data (e.g. data objects)		• Data flowing among the tasks can be explicitly seen in workflow view	• Understanding data flowing might become difficult, when we use swimlanes with high number of active participants
Workflow layout		• obliges to twist the flow more frequently than a vertical flowing • makes difficult to understand the process behaviour	• Preferred the horizontal flow to depict the time flowing • Try to save space by putting the flows in any order		• Easier to understand the flowing of the process activities along the time	• Larger diagram

Condition for the decision nodes	Condition on the arcs leaving the decision nodes	Sometime they are not without side-effects but they are actions performed by someone	Sometimes are the arrive of the events	Conditions without side effects	Oblige to add decision actions	Only conditions on data

1 Introduction

1.1 Motivation

UML (Unified Modelling Language) is a standard language for modelling and visualizing business processes. UML offers a large number of diagrams for modelling. Using UML, it is possible to model all the aspects of the business process.

The holistic views of UML methods allow the modeller to model the business process precisely. Our UML methods are composed of different views such as workflow view, entity view, task view, and organization view. These views help to grasp the business process precisely. Considering our UML models for the business process will result in a higher precision level. Our models are easily understood by people from technical and non-technical backgrounds. Our UML models have the capability of showing the same business process in different views, and this will result in higher comprehension. Our UML models offers higher flexibility. For example, one might consider our models for showing the business process to stakeholders and also converting the model to code for simulation and analysis. Our UML models are produced with only a few contracts, and this helps the modeller to save his time and increase focus on creating better models, instead of deciding which construct to use in the model. When modelling complex business process, the views of our models precisely shows the activities and participants that were part of the process. This helps to understand the process without any confusion.

This thesis evaluates holistic UML methods by practically modelling them using BPMN case studies and investigating the main features and advantages of family-based UML models.

1.2 Objectives and Contributions

The focus of this thesis is to produce UML models by studying the holistic views of UML methods. We have studied the features of BPMN components to model BPMN case studies using UML methods. We used the "Visual Paradigm" tool to model case studies in UML.

We have analyzed the results of BPMN and family-based UML models. We have produced the table summering the results of BPMN and UML models explaining why one might consider our UML methods for modelling a business process. We also produced a table explaining which UML methods have been applied to the BPMN case studies and attached an HTML reference of produced UML models.

1.3 Thesis Structure

Section 2 gives a brief introduction to BPMN and all the BPMN constructs, that were used in the BPMN case studies.

Section 3 introduces the family-based UML models and explains all the important features of our UML models.

Section 4 explains each BPMN case study and how the modelling has been done using our UML models covering all features. For each case study, there is a desired section called "lessons learned", representing the lessons learned while modeling the case studies and the BPMN errors that were discovered while modelling.

Finally, in section 5, all the main points of the thesis work are summarized.

2 BPMN

2.1 Introduction to BPMN

The business process model and notation (BPMN) is a graphic representation of the business process. It shows, in flowchart form, the sequence of steps in a business process from start to end, providing a visual aid for decision-making, collaboration, and optimization. It makes business processes standardized and understandable. BPMN is supported by OMG (Object Management Group), an organization that establishes standards for information systems.

The primary goal of BPMN is to support business process management for both business and technical users. Its notation is intuitive to business users and able to represent complex process semantics easily.

BPMN provides a way to quickly diagram business functions. The visual model will be translated quickly and easily into software that will run the process. BPMN works for any kind of management, operation, and support process. By developing a model with BPMN, you can improve communications with decision-makers about the nature and health of a process; you can collaboratively initiate improvements and you can move toward automating those improvements.

2.2 Components of BPMN

- Elements
- Activity Markers
- Connecting Objects
- Gateways
- Pool and lanes

2.2.1 Elements

BPMN diagrams are made of elements or shapes that comprise its visual language.
There are three main Elements which is used in BPMN.

- Events
- Activities
- Artifacts

2.2.1.1 Events

An event represents something that "happens" or triggers during the course of a process.

Types of Events

- START EVENT: Start Events are those which occur at the beginning of the process. Every process starts with a Start event. A Process can have more than one Start Event, but an Event Sub-Process only has one Start Event. Start events have no incoming flows.
- NON-INTERRUPTING EVENT: A Non-Interrupting Event is shown by a dashed circle. It is used to initiate an Event Sub-Process without interfering with the main process flow.
- INTERMEDIATE EVENT: Intermediate Events occur in the middle of the process (between activities) (thin double line). It indicates something that occurs or may occur during the course of the process, between Start and End. Intermediate Catching Events can be used to catch the event trigger and can be in the flow or attached to the boundary of an activity. Intermediate Throwing Events can be used to throw the event trigger.
- NON-INTERRUPTING BOUNDARY: Non-interrupting Boundary Events can be attached to the boundary of an activity. When they are triggered, the flow will be generated from them, but the source activity will continue to be performed.
- END EVENT: End Events occur at the end of the process. (Thick single line) and every process has at least one end event. A Process can have more than one end. The Process ends when all active paths have ended. End Events have no outgoing sequence flows.

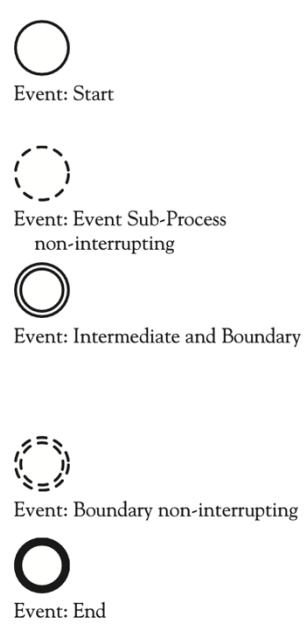


Fig 1. BPMN Events

- MESSAGE EVENT (RECEIVE): A Receive Message Event is used to start a Process or in the middle of a Process, either in the flow or attached to the boundary of an activity. Message events are commonly used in collaboration diagrams.
- MESSAGE EVENT (SEND): A Send Messages Event is used to send a message in the middle or at the end of a Process path. It is used by process participants to indicate communication
- TIMER EVENT: A Timer Event is always of catch type and used to specific time condition to evaluate to true, which will start a Process, start an Event Sub-Process, wait in the middle of a flow, or wait as a Boundary Event.

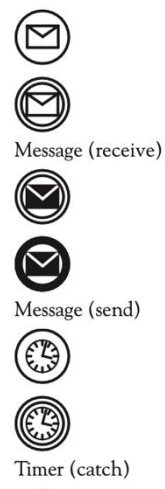


Fig 2. BPMN Message & Timer Events

- ESCALATION EVENT(CATCH): An Escalation Event used to handle escalation conditions, triggering the start of an Event Sub-Process or a Boundary Event.
- ESCALATION EVENT(THROW): A throw Escalation Event will cause the escalation conditions that will trigger the catch Events.

Fig 3. BPMN Escalation Events

- ERROR EVENT(CATCH): An Error Event is used to capture errors and to handle errors. This event is used as the start an Event Sub-Process or as a Boundary Event. These events can catch errors thrown by the throw Error Events
- ERROR EVENT(THROW): A throw Error Event is used set an error to be handled. This event can only be used as an End Event (i.e., not used as an Intermediate Event).

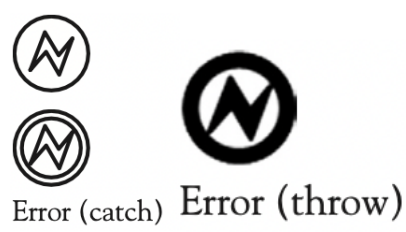


Fig 4. BPMN Error Events

- CANCEL EVENT(CATCH): A Cancel Events can only be used in the context of the transactions. The catch Cancel Events are used as Boundary Events for the transaction Sub-Process and will trigger the roll back of the transaction (i.e., the Activities of the Sub-Process).
- CANCEL EVENT(THROW): A Cancel Events can only be used in the context of the transactions. The throw Cancel Events are only used within a transaction Sub-Process.
- CONDITIONAL EVENT: Conditional Events are used to determine whether to start (or continue) only if a certain condition is true. Like the Timer Event, the Conditional Event can only exist as a catching event. They can be used at the start of a Process or an Event Sub-Process, in the middle of the flow, or as a Boundary Event.
- COMPENSATION EVENT (CATCH): A Compensation Event is used to handle compensation in the process. The catching Compensation Event be triggered as an Event Sub-Process Start Event, or as a Boundary Event.
- COMPENSATION EVENT (THROW): A Compensation Event is used to handle compensation in the process. The throwing Compensation Event can be used in the middle or end of a Process path.

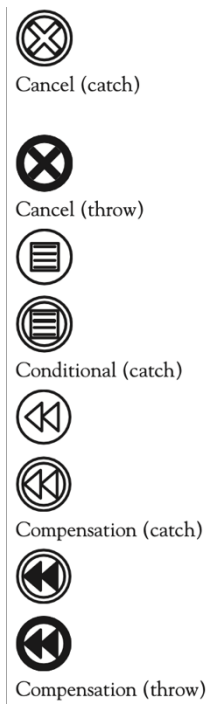


Fig 5. BPMN Cancel, Condition and Compensation Events

- SIGNAL EVENT (CATCH): Catching Signal Events are used for receiving signals. They are a generic, simple form of communication and exist within pools (same participant), across pools (different participants), and across diagrams. They can be used at the start of a Process or an Event Sub-Process, in the middle of the flow, or as a Boundary Event.
- SIGNAL EVENT (THROW): Throwing Signal Events are used for sending signals. They are a generic, simple form of communication and exist within pools (same participant), across pools (different participants), and across diagrams. They can be used in the middle or end of a Process path.

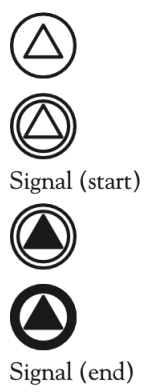


Fig 6. BPMN Signal Events

- MULTIPLE EVENT (CATCH): The Multiple Event is used to summarize several event types with a single symbol. The event is triggered if any one of those types is satisfied. They can be used at the start of a Process or an Event Sub-Process, in the middle of the flow, or as a Boundary Event.
- MULTIPLE EVENT (THROW): The Multiple Event is used to summarize several event types with a single symbol. When this is event is reached, then all the event types are thrown. They can be used in the middle or end of a Process path.
- PARALLEL MULTIPLE (CATCH): The Parallel Multiple Event is used to summarize several event types with a single symbol. The difference between this event and the Multiple Event is that the Parallel Multiple is only triggered if all of those types are satisfied. They can be used at the start of a Process or an Event Sub-Process, in the middle of the flow, or as a Boundary Event.
- TERMINATE (THROW): The Terminate End Event is the "stop everything" event. When a Terminate End Event is reached, the entire process is stopped, including all parallel activities.

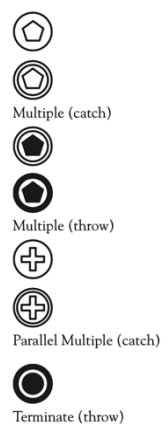


Fig 7. BPMN Multiple, Parallel Multiple and Terminate Events

2.2.1.2 Activities

Activities are Building blocks of BPMN, it shows specific task performed by a person or system within the business process.

Types of Activities

- TASK: A task is a single action that occurs in the business process, and which cannot be broken down further. We can also use task icons to show the specific kind of task. Whether it is a user task, manual task, and other tasks.
- USER TASK: A User Task is a typical "workflow" task in which a human performer performs the task with the assistance of a software application and could be scheduled through a task list manager of some sort.
- MANUAL TASK: A Manual Task is a task that is expected to be performed without the aid of any business process execution engine or application.
- SERVICE TASK: A Service Task is a task that uses some sort of service, which could be a web service or an automated application.
- SEND TASK: A Send Task is a simple task that is designed to send a message to an external participant (relative to the process).
- SCRIPT TASK: A Script Task is executed by a business process engine. The modeler or implementer defines a script in a language that the engine can interpret. When the task is ready to start, the engine will execute the script. When the script is completed, the task will also be completed.

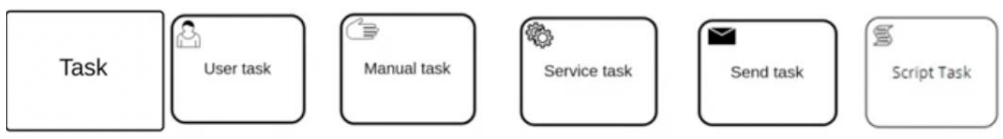


Fig 8. Different types of BPMN tasks

- SUBPROCESS: A Subprocess is a group of tasks that fit together particularly well. For example, the task "Collect Money" and "Fetch Items" are grouped in a Sub Process

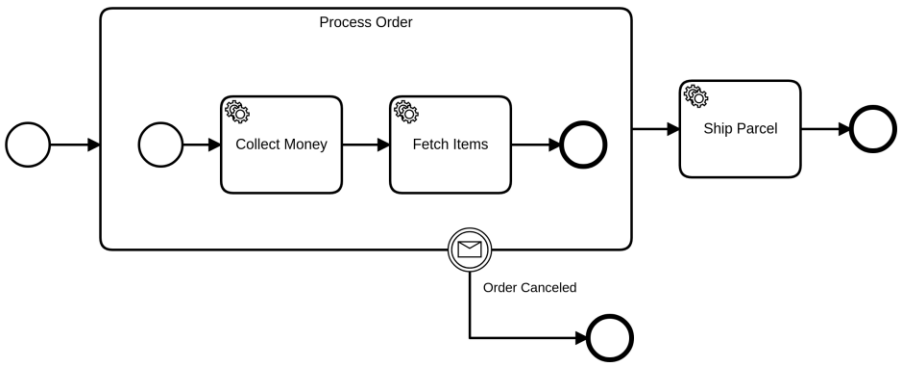


Fig 9. BPMN Sub-Process diagram

2.2.2 Activity Markers

- LOOP MARKER: A Loop Marker is used to represent an activity that will be executed multiple times until the condition is satisfied. The condition can be validated either at the start or end of the activity.
- PARALLEL MULTIPLE INSTANCE MARKER: A Parallel Multi-Instance Marker is used to represent an activity that can be executed as multiple instances performed in parallel. The number of instances will be determined through a condition expression that is evaluated at the start of the activity. All the instances will start in parallel, and each instance can have different input parameters. The activity, as a whole, is completed after all the instances are completed. However, another expression, if it becomes true, will stop all instances and complete the activity.
- SEQUENTIAL MULTIPLE INSTANCE MARKER: A Sequential Multi-Instance Marker represents an activity that is similar to a Parallel Multi-Instance activity, but its instances will be executed in sequence. The second instance will wait until the first instance is completed and so on.
- AD-HOC: The Ad-hoc is a tilde symbol and used to mark a Sub-Process for which the normal sequence patterns are relaxed and its activities can be performed in any order at the discretion of the users. Tasks can start at any time without any direct dependency on other tasks.

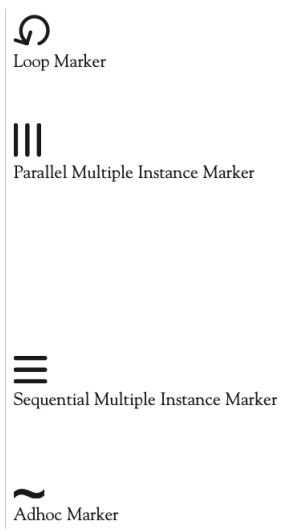


Fig. 10 BPMN Markers

2.2.3 Artifacts

BPMN Artifacts can be used to include additional information about a process.

Types of artifacts

- DATA OUTPUT: A Data Output is the data result of the entire process. It is a kind of output parameter.
- DATA STORE: A Data Store is a place where the process can read or write data (e.g., a database or a filing cabinet). It persists beyond the lifetime of the process instance.
- Collection of Data Objects: A Collection of Data Objects represents a collection of data elements related to the same data entity (e.g., a list of order items).
- ANNOTATION: An annotation provides further explanation to a part of the diagram. This is used to show or highlight some parts of BPMN diagram

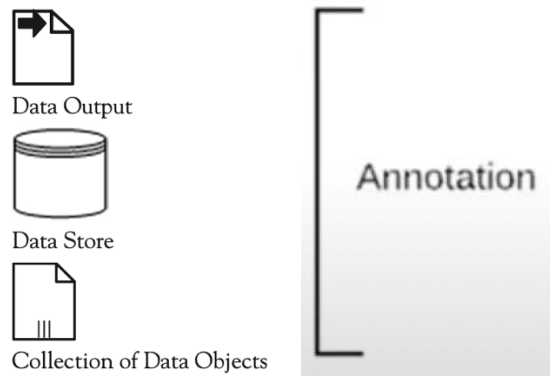


Fig 11 BPMN Artifacts

2.2.4 Connecting Objects

Connecting Objects are the lines present in every BPMN Diagram. They are used to show the order of activities to be performed.

Types of connecting objects

- Sequence Flow: It shows the Main order of activity to be performed
- Message Flow: it shows Message flow across the pools
- Association: it shows relationship between artifacts and flow objects

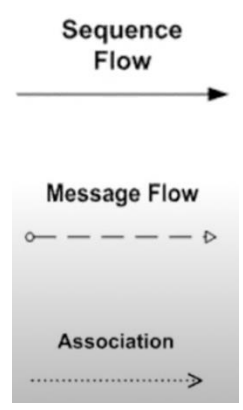


Fig 12. BPMN Connecting Object

2.2.5 Gateways

The BPMN gateways are decision points that can adjust the path of a flow in process based on certain conditions. They are represented by a Diamond shape; they dictate the flow of the process. There are seven types of Gateways in BPMN, in which five of them are most commonly used.

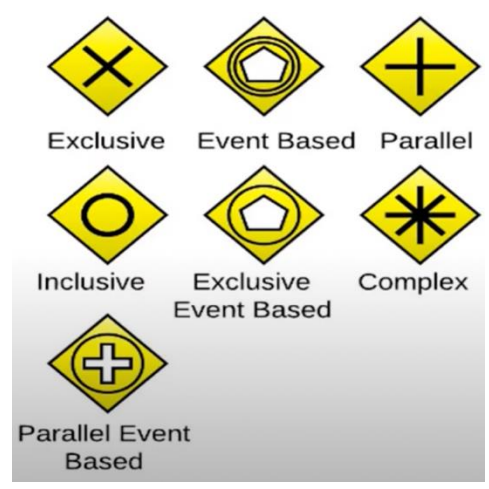


Fig 13. BPMN Gateways

- Exclusive: An exclusive gateway is used when only one alternative path can be taken in the process based on the condition. The gateway can be displayed with 'X' marker.
- Event-based: An event-based gateway is similar to an exclusive gateway, where only one path can be taken, but a path is taken based on an event not based on the condition
- Parallel Gateway: Parallel gateways are used to represent two concurrent flows. It is similar to the fork construct in the UML
- Inclusive: An inclusive Gateway specifies that one or more of the available paths can be taken.
- Exclusive event Based: An exclusive event-based gateway is used to branch a process when alternative paths are determined by events (various messages or signals) rather than by conditional flows.

- Complex gateway: The Complex Gateway defines behaviour that is not captured by other gateways. Expressions are used to determine the merging and splitting behaviour.
- Parallel Event Based: The Parallel Event Gateway is only used for starting a Process. It is configured like a regular Event Gateway, but all the subsequent events must be triggered before a new process instance is created.

2.3 Pools and Lanes

Pools represent a process participant superior to the lanes, who coordinates the process flow within the Pool. They cover the entire process and assign the included tasks to the responsible Lanes. Pools are usually further subdivided into Lanes. Lanes are used to organize and categorize activities within a pool. They are commonly used for allocating activities to roles systems or the organization's departments. Lanes can be nested, for example, in a pool there could be an outer set of lanes for company departments, and then an inner set of lanes for roles within a department. At a glance both swim lane elements look very similar however they have completely different meanings. A pool represents a participant in a business collaboration and may act as a container for a single process, whereas a lane represents an activity classifying mechanism.

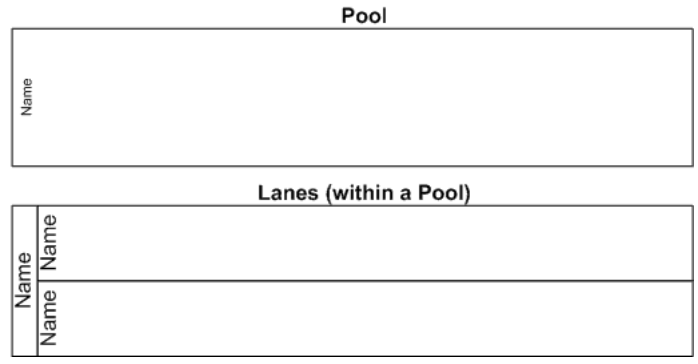


Fig 14. Pool and Lanes

2.4 Types of BPMN

BPMN can represent Business Models through four kinds of diagrams

- Process Diagram
- Collaboration Diagram
- Choreography Diagram
- Conversation Diagram

2.4.1 Process Diagram

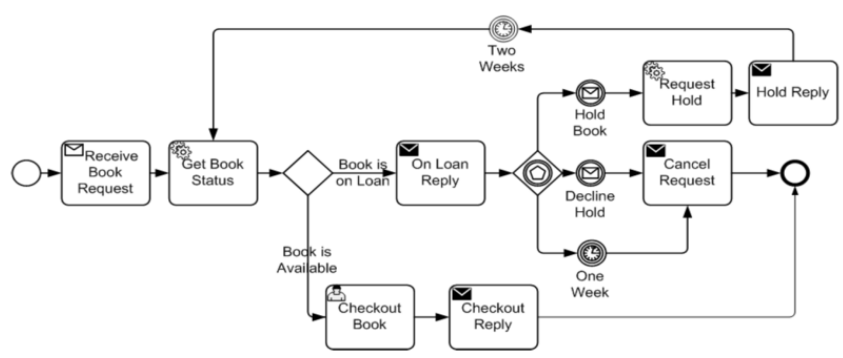


Fig 15. BPMN Process diagram

A Process diagram describes a sequence or flow of Activities in an organization with the objective of carrying out work. Process is depicted as a graph of Flow Elements, which are a set of Activities, Events, Gateways, and Sequence Flows that define finite execution semantics. Processes can be defined at any level from enterprise-wide Processes to Processes performed by a single person. Low-level Processes can be grouped together to achieve a common business goal. BPMN uses the term Process specifically to mean a set of flow elements.

2.4.2 Collaboration Diagram

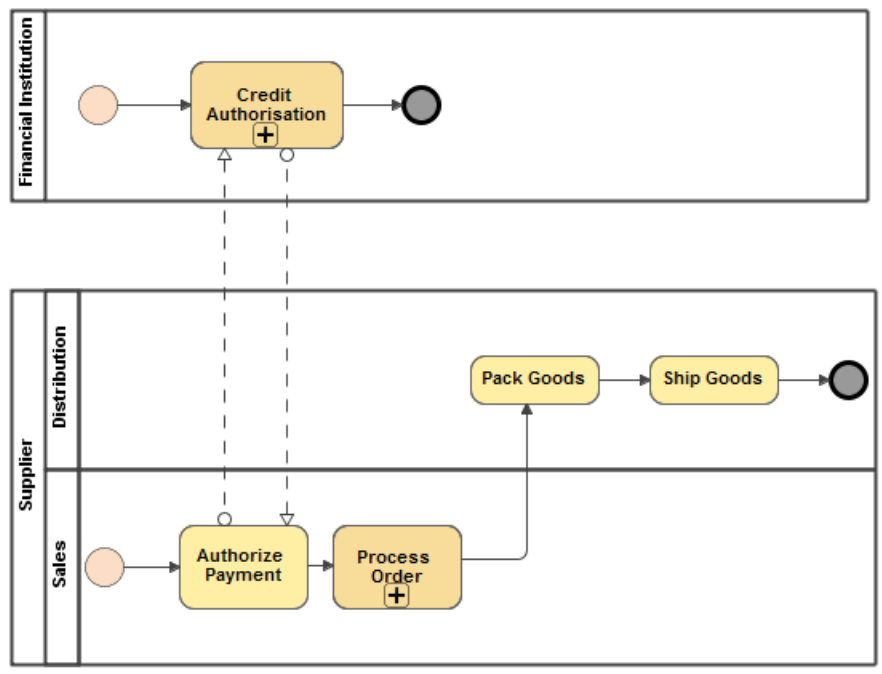


Fig 16 BPMN Collaboration diagram

Fig 16 represents a Collaboration diagram showing the interactions between two pools. Similar to process diagrams, a collaboration diagram consists of activities, sequence flows, and gateways. The message exchange between the Participants is shown by a Message Flow that connects two Pools (or the objects within the Pools). The Messages associated with the Message Flows can also be shown graphically. All combinations of Pools, Processes and a Choreography are allowed in a Collaboration.

2.4.3 Choreography Diagram

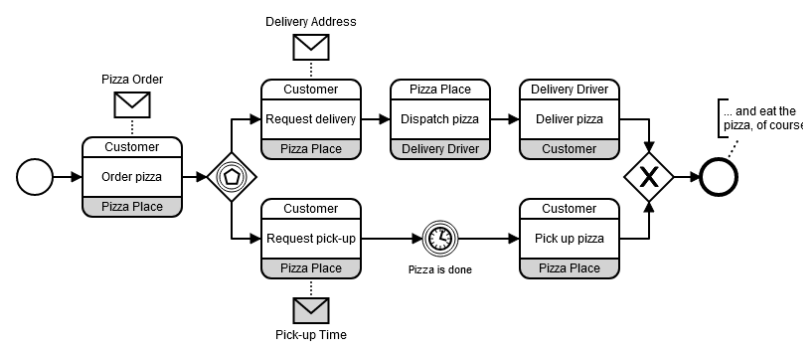


Fig 17. BPMN Choreography diagram

Choreography is a different kind of diagram, and it has a different behaviour compared to other BPMN diagrams. Its purpose is to show the interaction between participants in a different format and focusing only on message flow instead of the individual detailed tasks of a process. Each step in the choreography involves two or more participants and it is defined outside of a pool. The choreography activity is depicted as a rectangle the two bands (I.e., one at the top and one at the bottom (see fig. 17)) represent the parties involved in the interaction captured by the activity. A white band is used for the initiating party and the dark band is used for the recipient. The envelope represents a message sent by the corresponding party return message envelopes of a two-way interaction are darkened to establish an order dependency between two interactions.

2.4.4 Conversation Diagram

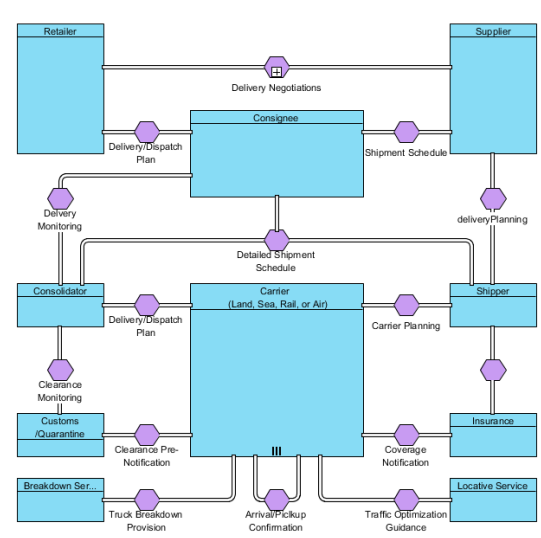


Fig 18. BPMN Conversation Diagram

conversation diagrams represent a top-level view of collaboration diagrams, and they are useful for representing process landscapes and high-level interactions between involved participants (I.e., representing an overview of a network of partners and how they communicate with each other). The goal is to present a summary of the messages that flow between a group of participants in a domain. We group sets of messages into conversations thus hiding the complexity of the choreography of the underlying messages. The conversations (hexagonal elements) are connected to the participants that engage in the conversation. This creates an overview of how the participants interact together. Just as in human speech it is possible simply to say that two (or more) parties engaged in a conversation about a topic without going into the detail of what was said.

3 Holistic view of UML Methods

Every business process consists of activities. These activities (tasks) are grouped in to chunks to achieve a business goal. Note that we will use the term **enterprise** which denoted organizations such as banks, universities, government agencies, departments etc.

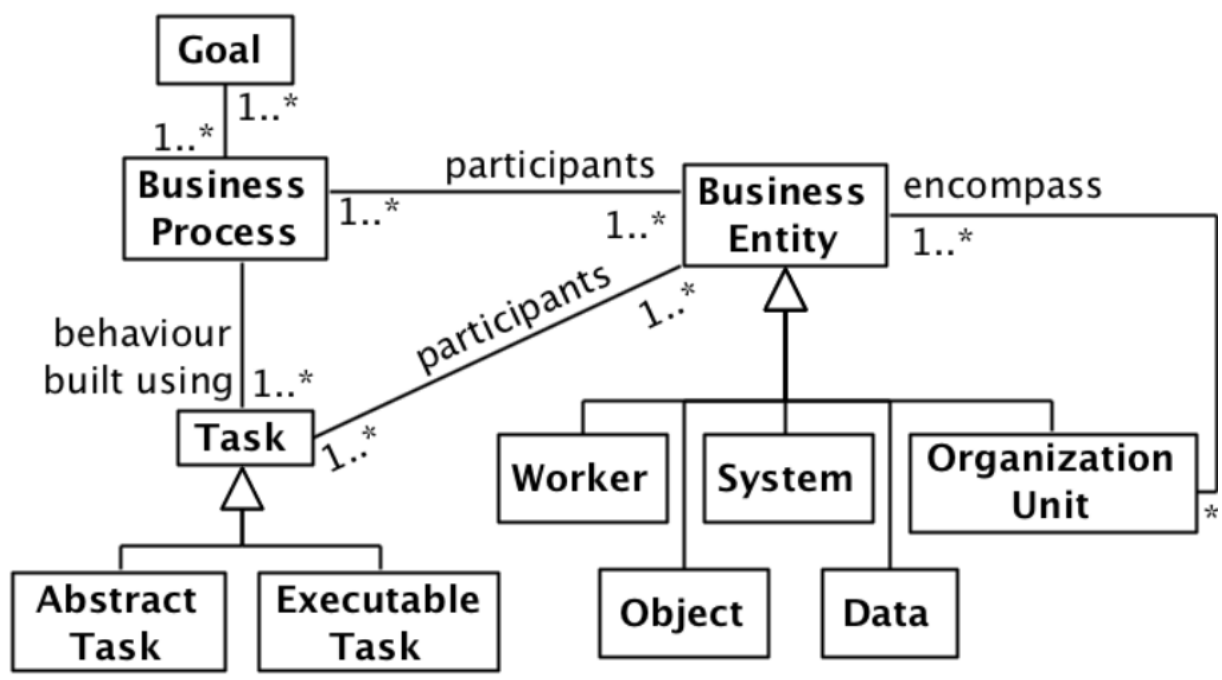


Fig. 19 Business Process Conceptual View

Fig 19 represents the conceptual view of the business process, which is built with entities that perform some activity or are manipulated by other activities. These entities are called Business Entities and we define them in the following.

- worker: Activities that are performed by humans in the enterprise (e.g., a clerk, an employer, a buyer)
- system: hardware/software systems that perform some activities in the enterprise (e.g., an ATM machine, PayPal)
- business object: digital and physical things that are handled and used in the enterprise (e.g., a bank account, a product, a computer, a post label)
- data: data represents a plain text used in the process (e.g., credit card number, amount of money, phone numbers)
- organization units: organizational unit represents the structure of the enterprise. For example, it could be defined the parts of organization such as departments, sub-branches, or offices. Note that organization units do not perform independent activities as worker and systems do.

- TASK: A task is a general concept, and it refers to whatever kind of activity is done in the business process. A task is expressed by some participants and by its behaviour. The task is distinguished into two parts in our models (i.e., abstract task and executable task).

1. Abstract Task: It represents a complex activity in which (at least one active participant) takes part and shows how the active participant contributes to the whole activity. For example, if we consider the dealer network (we are referring to the general trading network) "Seller and buyer making an agreement on a price with the help of mediator" is an abstract task. But in this case, we only know that both the parties (i.e., seller and buyer) make an agreement with the help of the mediator, but we do not know how the agreement is made and what are the private actions and interaction took place to make agreement (i.e., We do not know the detailed explanation of the process). Notice that if we consider a task as abstract, we do not care much about details (i.e., what exactly participant does)
2. Executable Task: it represents an action made by the active participant itself (i.e., private action) or interaction between active participants. Consider the seller-buyer example discussed above. Using executable task, we can see, which private actions and interactions took by active participants. For example, the seller proposes a price" based on the seller price, the buyer proposes another price," and "the mediator proposes another price (i.e., third price)". Notice that here we can clearly see the agreement made by both the parties (i.e., seller and buyer). Notice that executable tasks are much more precise compared to abstract task.

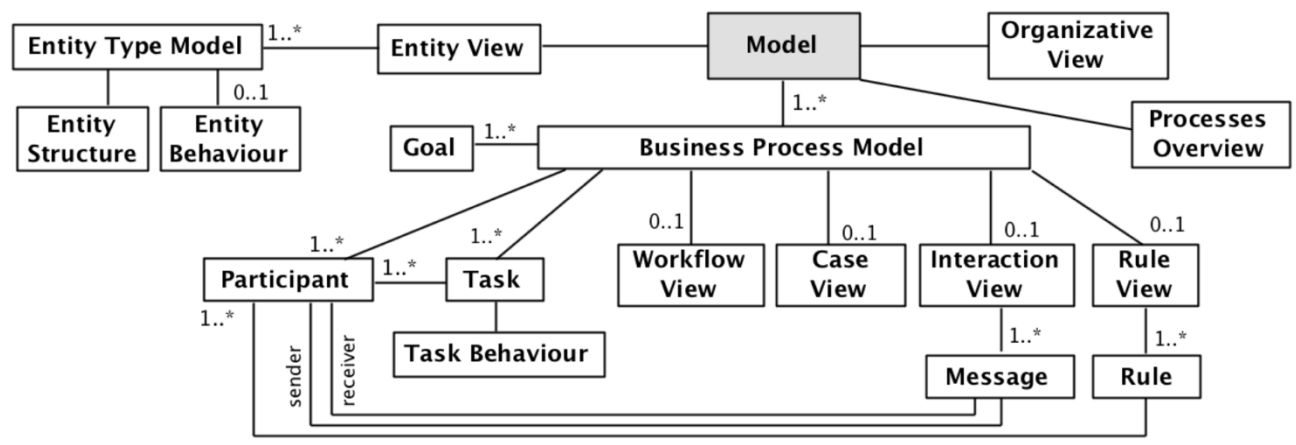


Fig. 20 Business Process Model Conceptual Structure

Fig. 20 represents the conceptual structure of business process showing possible view of a model. Notice that fig 20 is different compared to fig 19.

- Entity View: Entity view contains the collection of business entities that are part of enterprise enacting the process. Business objects in each model will define a static structure and its behaviour. For example, the structure of business objects that are used in the process is defined as their state, and the transformation of business objects from one state to another state is defined as their behaviour.
- Organization view: organization view defines how the business entities (i.e., entities that are already defined in the entity view) are structured in the enterprise. For example, the communication between the worker entities (i.e., seller, buyer and mediator) in the enterprise can be seen through organizational view. Notice that we can also see the structure of organizational units (i.e., departments and sub-branches) in the organization view.
- Process Overview: Process overview shows the goals of the enterprise, and a set of processes use to achieve the organization's goal.
- Workflow view: The Workflow view shows the process behaviour of the enterprise with the help of executing tasks. Notice that in many cases tasks are just defined in natural language. Using workflow view, the user can also grasp the flowing of the participant activities and objects/datatype used in the workflow.
- Interaction View: interaction view shows the interactions (i.e., sequence of message exchange) taken between active participants. The interaction view allows user to see both the process behaviour and interactions that took place between the process participants in one place. This enables the user to comprehend the whole process's behaviour in less time.
- Rule View: Rule view shows the business rules that are determining the process behaviour. The business rules are demonstrated in structured natural language. Note that one can use strict OMG standards for defining vocabularies and business rules using natural language based on ontologies, but they can also define by formal notations.
- Task view: task view shows the tasks done by active participants separately from the workflow view. The behaviour of each task is shown using class diagram. Note that sometimes the workflow view of the complex process can be cluttered with many constructs (i.e., creating confusion in understanding the tasks done by active participant). The task view helps the user to see the objects/data type and the task participants (i.e., worker participants) separately from the workflow view with the help of class diagram.

3.1 UML for modelling holistic models

UML is a rich visual modeling language for the architecture, design, and implementation of complex software systems both structurally and behaviorally. UML allows the modeler to model business process without forgetting textual notations (i.e., OCL and Action language). Using the UML, we can model all the possible aspects of business process. For example, the workflow is modelled by the means of activity diagram, the entity view is modelled using class diagram, the interaction view is modelled using sequence diagram. We can produce both incomplete (i.e., attributes type in class is not defined and the behavior of the operations in a class is not defined) and complete models (transformation of UML models to code) using UML. Both the models (i.e., complete and incomplete) are useful depending on the business process. The incomplete models are also known as loose methods. Note that the modeler needs less effort and time to produce incomplete models and these types of models are easy to understand by anyone without requiring knowledge on UML contracts. The UML textual parts (i.e., guards and the transition of state machines) are also express in natural language instead of using strict OCL and action language. This could also be seen as a disadvantage of incomplete models since they use disciplined natural language for modelling business process. For example, an action in an activity diagram will be described by a sentence having form "name verb object complements", where name belongs to the list of the process participants. Notice also that this leads to the differences with respectively to the levels of precision.

To produce a precise model, one must consider using the complete models (models having complete list of the features of a class, the signature of each feature is complete, the behavior of each operation is and active class is defined) expressing the textual part using OCL and action language. The complete models are also known as hard methods. Complete models are used to transform models to code or it can be used as input for simulation/analysis tools. Modeler needs more effort and time to produce complete models compared to incomplete models. To understand complete models, one must be familiar with all the UML constructs. Complete models are quite precise when compared to incomplete models, since they express parts of the activity diagram using OCL and action language. The disadvantage of complete models is completeness. Complete models do not allow modeler to skip irrelevant details of the business process.
Note that having an incomplete model (loose method), we can refine the rendering until we obtain complete model (hard method) with good precision level, and similarly, complete models can also be rendered to incomplete models by making it less selectively and less precise (i.e., disciplined model by replacing OCL and actions by natural language fragments).

We have five different UML models based on abstract and executable tasks and they are listed below.

- UM1 (Hard-Abstract tasks)
- UM2 (Hard-Executable tasks-Interaction)
- UM3 (Hard-Executable tasks)
- UM4 (Disciplined-Abstract tasks)

- UM5 (Disciplined-Executable tasks)

Note that above five methods are based on specific UML profiles. A UML profile will be defined by:

1. the used subset of the UML diagrams and constructs
2. a set of stereotypes, and
3. the form of its models.

3.1.1 Dealer network description

We have used the dealer network example to demonstrate our UML methods and their features. The Dealer network consists of three primary members: the dealers, the manufacture, and the shippers. All the three members are independent and we assume that they will work together. The manufacture sells a unique kind of product (e.g. milk). The dealer buys the product from the manufacture. The shipper delivers the product to the dealer. The goal of the dealer network is to ensure working together conveniently. The dealer places and order to manufacture. The order includes the quantity of product (i.e., number of liters of milk). The manufacture check for the product availability, when he receives an order from dealer. If there is no availability, the manufacture cancels the order and inform the dealer about the availability. When there is availability of order, the manufacture informs the dealer that order is accepted and the dealer pays for the order. Once the order is paid, the manufacture requests the shipper to deliver the product to the dealer. If there a failure in dealer payment, the manufacture cancels the order. After receiving the shipping request from the manufacture, the shipper will send the shipping request conformation to the manufacture. In the Dealer Network it is also important to define and handle the reputation information of the members in order to minimize the risks. The dealer network members can express their satisfaction on provided service. For example, the dealer can express his satisfaction about the service and the product provided by the manufacturer. There is also a satisfaction monitoring member named Authority. The authority is responsible for collecting the satisfaction reports (report containing truthfulness) from the dealer network members after the deal has been completed. the authority makes reports (i.e., reliability report) available to access in dealer network.

The Dealer Network enacts three business processes: buying some quantity of products, collecting the feedbacks, and providing reliability reports.

Although this case study is small and simple, it illustrates many relevant features. For example, many different kinds of participants such as business objects, data, and a family of correlated processes.

3.2 UM1 (Hard-Abstract tasks) Method

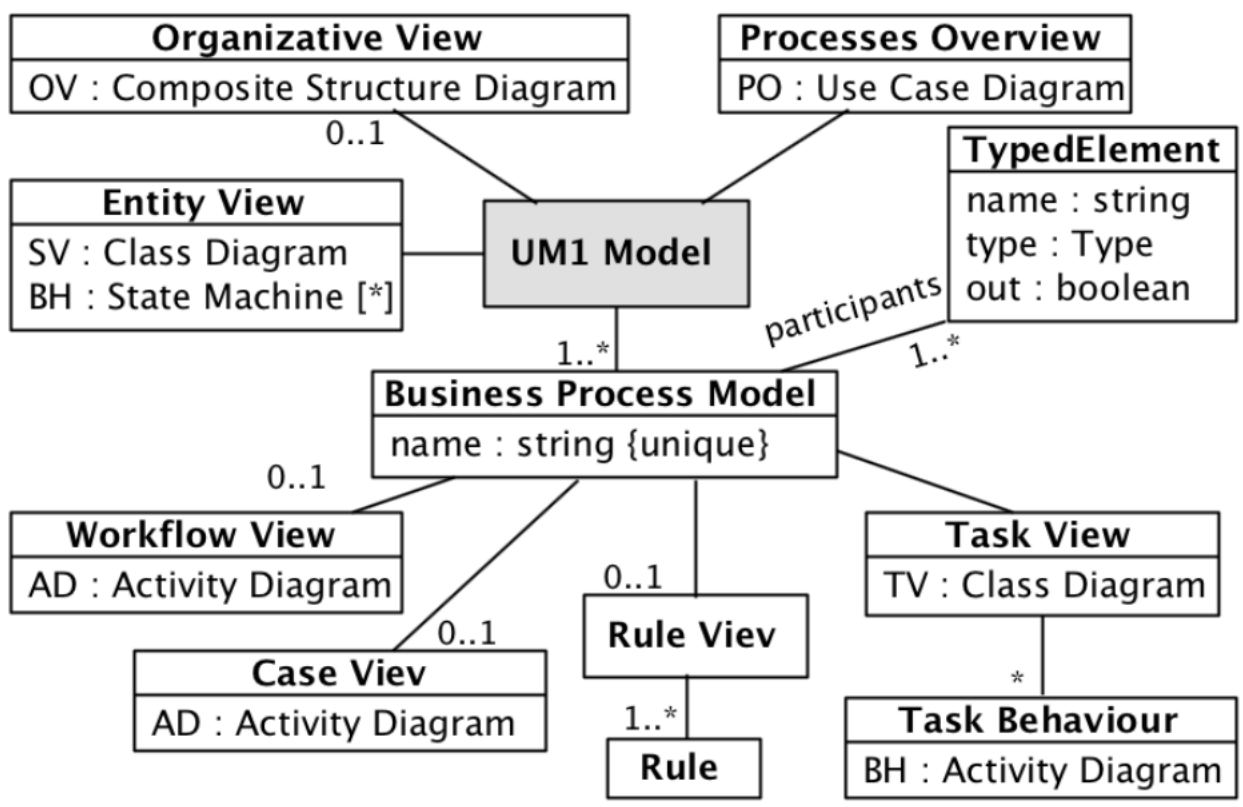


Fig. 21 UM1 model structure

Fig. 21 shows the conceptual version of UM1 model represented in prescribed form of model and it is expressed by means of UML class diagram. The conceptual view of UM1 also shows the possible views that are used with the model.

Process participants: Process participants are roles used in the process (i.e., they are introduced in the entity view) and they are represented by typed element (class/datatype) in the entity view. The process participants may be classified as <<in>> or <<out>>. An in (for input) participant will exists before the process may start and it may be modified during the process, also deleted. An out (for output) participant does not exists before the start of the process and it will be created inside the process.

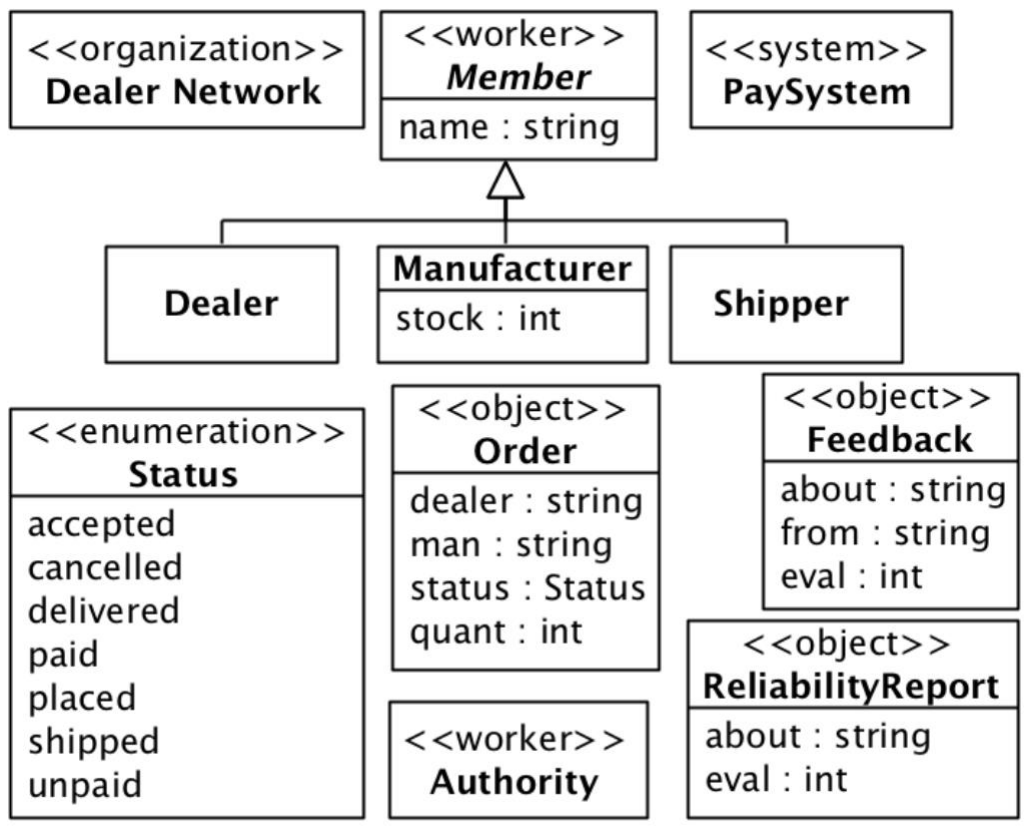


Fig. 22 Dealer Network UM1 Model: Entity View

3.2.1 Entity view

The entity view shows the business objects/data type used in the business process in the form of UML class diagram. The entities are modelled using associations to show mutual relationships. The non-predefined data needed to described the business entities are defined by means of datatypes appearing in the entity view.

Each entity may consist of attributes and operations, which are used to model relevant aspects of business process. Each business entity class is identified with stereotypes.

The following are the stereotypes used in the entity view:

- <<worker>>: basic actions, that are done by humans in the process.
- <<system>>: hardware/software components used in the process
- <<object>>: physical/ digital elements that are used in the process.
- <<organization >>: organization units of the enterprise enacting the process. The relationship between the organization unit and the sub parts of organization unit (i.e., branches or departments) are shown using aggregation or composition relationship

The behavior of active entities (worker and system) and the business objects are defined by state machine. The operations and the data type of any class can be defined using OCL pre/post conditions and methods. Data types are defined by defining their operation. The behavior of business objects is optional since the level of modelling in hard models are relaxed.

Each entity in the entity view, must be stereotyped as <<worker>> or <<system>> or <<organization>> or <<object>>. Each entity in the entity view must be defined using the above stereotypes.

We will be using a Dealer Network case study to represent all the views of the further models.

The business entities (types) of Dealer Network are:

<<worker>>

Manufacture who sells the unique product (e.g. milk) in the network, the **Dealer** who buys the product from the manufacture, the **Shipper** who delivers the product from the manufacture to the dealer, and **Authority** who collects the feedbacks about the service and make them available in the network.

<<system>>

PaymentSystem, who supports the handling of payments.

<<object>> (i.e., business object)

the **orders** sent by the dealers to the manufacturers, **Feedback** (i.e., the satisfaction provided by the dealer network members to the authority at the end of a deal) and **Reliability Report** (i.e., the documents provided by the authority about the performance of the members)

Rules of Entity view:

Let (CD, SMs) be an entity view:

- Each class appearing in CD must be stereotyped by either <<object>> or <<worker>> or <<system>> or <<organization>>
- CD must include at least a class stereotyped either by <<worker>> or by <<system>>, and a class named as the enterprise enacting the processes stereotyped by <<organization. >>
- The context class of each state machine in SMs must be a class stereotyped either by <<worker>> or by <<system>> appearing in CD.

3.2.2 Organization view

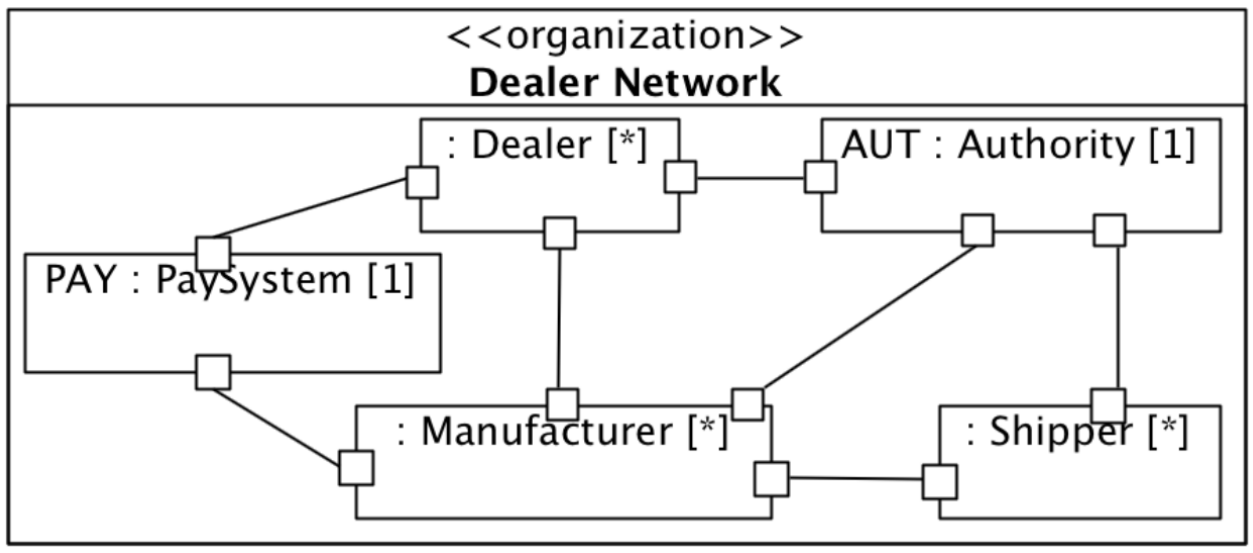


Fig. 23 Dealer Network UM1 Model: Organizative View

The organization view is a composite structure made of exactly one (structured) class corresponding to the enterprise enacting the processes (an organization unit) that contains parts typed by active entities or organization unit classes (i.e., department or branch's). It illustrates the top-level structuring of business process including the active entities and organization units. It also shows the possible communication in the organization between active entities with the help of ports and connectors.

Note that the organization view is quite useful when there is a complex business process (i.e., includes many business entities) showing the possible communications in the organization. Fig 23 represents the organization view of dealer network. Notice that the active entities (Authority, PaySystem, Dealer, Manufacturer and Shipper) are modelled with ports are connectors to represent communications. For, example the shippers will never communicate with the pay-system whereas the authority communicates with dealers, manufacturers and shippers.

Rules of Organization view:

Let OV be an organizative view:

- OV must consist of a structured class corresponding to the enterprise enacting the processes.
- All parts appearing in OV must be typed by a class corresponding to active entities or to organization units defined in the entity view.
- For each class corresponding to active entities or to organization units defined in the entity view there must exist at least one part in OV typed by it.
- OV must contain at least a part with multiplicity greater or equal than 1.
- At most one connector may link two parts contained in OV.

3.2.3 Process Overview

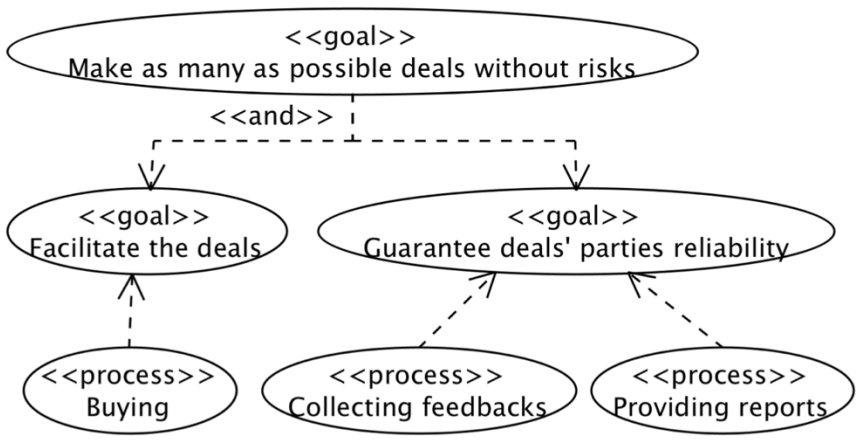


Fig. 24 Dealer Network UM1 Model: Processes Overview

Fig. 24 represents the process overview of Dealer network. The process overview is similar to UML usecase diagram but it does not contain actors. The process overview represents the goals of the business process and the use cases are stereotyped as <<goal>> or <<process>>. The process also contains dependency relationships among the goals and they should be stereotyped by <<and>> or by <<or>>. Note that there exists a dependency relationship (relationship linking business processes with business goals) between processes(<<process>>) and goals (<<goals>>) (i.e., if process P depends on goal G, modification in goal G leads to change in process P).

Rules of Process view:

Let PO be a processes overview:

- A goal in PO cannot depend in modality "or" on a unique subgoal.
- A goal in PO cannot depend in modality "and" on some subgoals and in modality "or" on others.
- The transitive closure of the dependency relationship in PO between goals, i.e. of the relationship to be a subgoal, (of the dependency relationship between processes) must be antireflexive.
- Each process in PO must depend at least on one goal.
- For each goal not decomposed in subgoals appearing in PO and there must be at least one process depending on it.

3.2.4 Task View

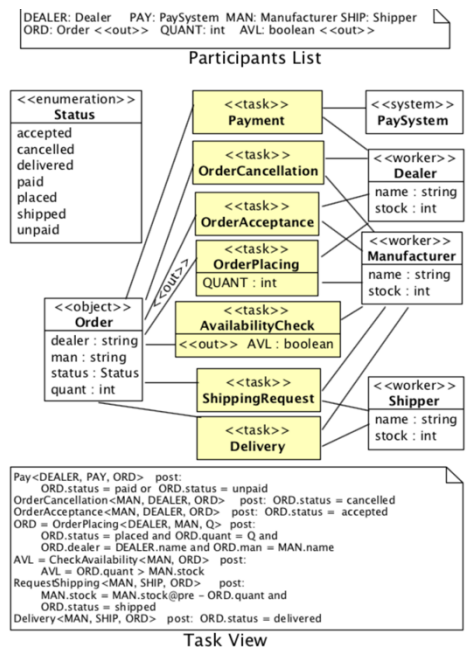


Fig. 25 Dealer Network UM1 Model: Task view

Task view shows the basic tasks in the business process and it is modelled using class diagram. Each task in task view is stereotypes as <<task>>. Each task is connected to task participants using association (where the participants type is already defined in the entity view). The attributes of the task class represent the datatype used in the task. the task participants are stereotyped similar to process participants (i.e., <<in>> and <<out>>). the task behavior is defined by OCL pre/post conditions (i.e., showing which conditions to be satisfied before and after the task executions) or by operations used in activity/ sequence diagram detailing the task behavior, using the operations on the task participants' classes.

Rules of task view:

Let TV be a task view:

- A class in TV is either stereotyped by <<task>> or has been introduced in the entity view;
- An association may link a task class only with an entity class, and the attributes of a task class are typed by pre- defined data or datatypes introduced in entity view; a task class is linked by an association with at least one active entity class (i.e. a class stereotyped either by <<worker>> or <<system>>);
- Each non-abstract task class in TV must be used to type at least one action node of either the workflow view or the case view or appear in a rule in the rule view.

3.2.5 Workflow View

The workflow view of the business process is produced by the means of activity diagram. The UML activity diagram consist of high number of constructs, and usability of most of the constructs depends on specific use cases. The UM1 model suggests modeler to use only few constructs instead of spending time on deciding which constructs may be used in business use case. This helps modeler to save time and improve focus on producing better model. Below are the few constructs suggested by the UM1 workflow.

- Action node, accept (call and time) accept event actions,
- Control flows,
- Control nodes (i.e., decision node, merge node, activity final node, initial node, fork node, join node)
- Interruptible regions
- Rake constructs
- Swimlanes

The action nodes are used to model the basic actions/activity of the process. Each action will be labelled by an instance of a task class.
Decision node and merge nodes are used to describe the condition in the process. The fork node and join node are used to model parallel flows in the process. The initial node is used to start the process and activity final node is used to terminate the process. The control flow is used to show the sequential behavior of the activity. The interruptible regions are used to model activities whose execution may be interrupted. The rake construct is

used to model the decomposition of complex and large activity diagrams. The participants are represented explicitly represented inside each task. UM1 does not use swimlanes, but one might consider using swimlanes since it shows what is going in various organizational units (i.e., departments and branch's). hence the use of swimlanes in UM1 model is optional.

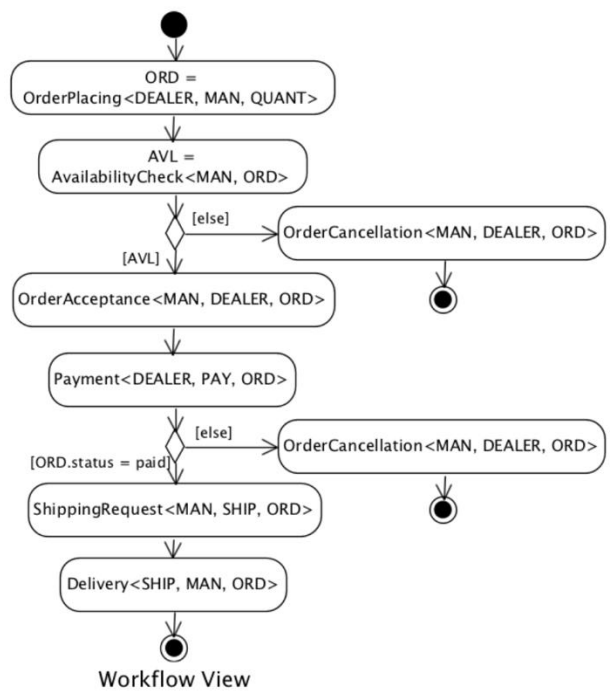


Fig. 26 Dealer Network UM1 Model: Workflow View (Buying)

Fig. 26 represents the buying process of the dealer network. Swimlanes in UM1 model are optional since the entities performing the activity are explicitly represented inside each task. The modeler might consider using swimlanes in UM1 when the process has sub-organization units or departments and branches. This helps to clearly see what is done by organization and components of organization (i.e., departments and branches).

The UM1 model also suggests not to use object nodes since we explicitly define the things used in the process as business objects and data. We can also track the flowing of the data/objects among the various tasks. For example, see fig.26, the object ORDER was produced in the task "orderplacing" and we can see the object flowing among various tasks. The usage of object nodes in the workflow view should be done for the purpose. For example, assume the dealer network workflow using five object nodes representing object flow between different activities. Thus, workflow will become unreadable and difficult to understand. Moreover, the size of the diagram will also increase. This leads to ambiguities and will make the model not suitable for code transformation or analysis. The style indication of UM1 model also helps to understand the flowing of objects easily (see fig. 26).

Rules of Workflow View:

Let AD be a workflow view:

AD is built using only action nodes, control nodes (i.e. initial, decision/merge, fork/join, flow and activity final), control flows, time and accept events, the rake construct, and swimlanes; and moreover: has exactly one initial node;

- Each action node is labelled by an instance of a task class introduced in TV;
- An initial/action node has must have exactly one leaving flow;
- A decision node/fork node has at least two leaving flows;
- Exactly one f low leaving a decision node is labelled with the "else" guard ([else]);
- A guard on a flow leaving a decision node is labelled by a well-formed OCL expression of type bool without side effects, where only the participant names appear freely;
- For each decision node there must be a matching merge node, and similarly for any fork node there is a matching join node (exception can be made whenever all flows except one leaving the decision/fork node end with a final node). Notice that it is however possible to have a merge (join) node without a corresponding decision (fork) node, e.g. in the case when different exclusive accept messages may start the process.
- Each non-final node (i.e. that is neither an activity final nor a flow final) must have at least an existing flow;
- Each node that is neither an initial node nor an accept event nor a timed event must have at least an entering flow;
- No flow enters the initial node and no flow leaves a final node;
- Any participant must appear at least in one task instance labelling an action node or in a guard labelling a flow leaving a decision node;
- Any activity recalled by means of the rake construct must be defined by an activity diagram.

3.2.6 Rule view

Rule view represents the behavior of business process by listing all the set of rules followed by the participants. The rules are expressed in a precise vocabulary without ambiguities. The UM1 entity view operations and the task view attributes and task names are made using quite formal vocabulary. UML uses OCL for expressing textual notions and constraints. However, it is not suitable to express rules related with the behavioral aspects, thus UM1 proposes to use also combinators inspired by the temporal logic and the behavioral diagrams of UML.

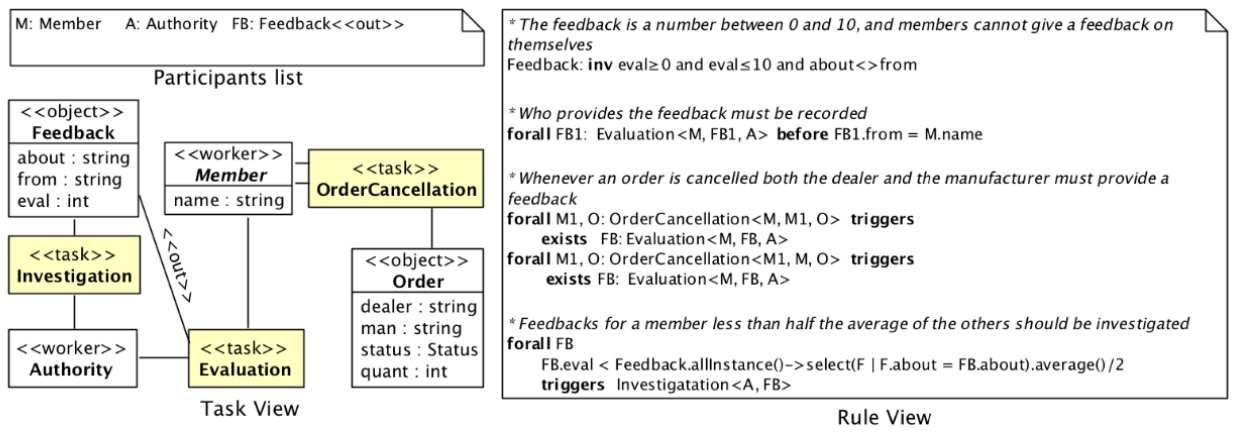


Fig. 27 Dealer Network UM1 Model: Rule View (collecting feedbacks)

Fig .27 represents the rule view of the Dealer network. Notice how the rules are expressed using precise language. Below the rules and they are expressed in natural language.

the metrics of evaluations are between 1 and 10, and the participant cannot provide a feedback on themselves.

- Feedback: inv eval≥ 0 and eval <10 and about<>from

The dealer and manufacture must go through the evaluation when there is an order cancellation

- **forall** M1, O: OrderCancellation <M, M1, 0> triggers
  exists FB: Evaluation <M, FB, A>
  forall M1, O: OrderCancellation <M1, M, O> triggers
  exists FB: Evaluation<M, FB, A>

the name of the person who evaluated must be recorded in the feedback

- **forall** FB1: Evaluation <M, FB1, A> before FB1.from = M.name

when evaluation metric of some member is less the half average compared to others members, then it should be investigated

- **forall** FB

  FB.eval < Feedback.alllnstance(->select(F | F.about = FB.about).average() /2

  triggers Investigation <A, FB>

3.3 UM2 (Hard – Executable tasks-Interaction) Method

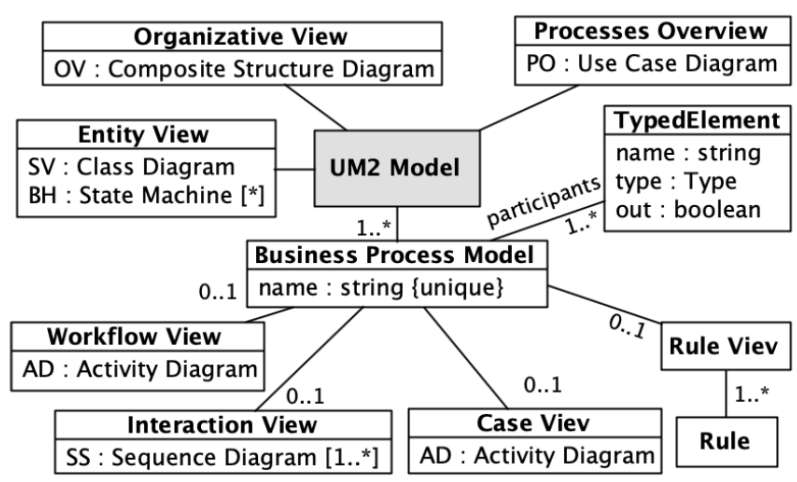


Fig. 28 UM2 model Structure

All the business process tasks correspond to executable tasks in the UM2 model. Executable tasks correspond to the execution of a private action (i.e., actions performed by active entity class on them self) of an active participant or to a communication from one active participant to another active participant. UM2 method precisely models private actions (what exactly participants do) and interactions between them. Notice that the structure of UM2 model and UM1 model quite similar. The main difference between UM2 and UM1 is the kind of task used to build the business process (i.e., executable tasks) and absence of the task view. UM2 model does not require task view since the process tasks will be modelled by the means of operations in the entity view. The private actions of an active participant are stereotypes as <<priv>>. The operations (stereotyped as <<priv>>) of an active participant and datatypes (introduced in entity view) can be defined using the pre/post conditions and methods. The usability of swimlanes in UM2 model is mandatory since to express who is calling the operation modelling the executable tasks uses swimlanes. Each active participant will have an allocated swimlane and their activities are placed inside the swimlane. The modeler should take care about the swimlanes ordering and lanes layout since they might become intricate.

UM2 model uses send signal icon

to represent communication between the active participants and it will easily differentiate the private actions with communications in workflow. UM2 model prescribes to model an executable task in the following way.

Let AP and AP1 be two active participants of a business processes typed respectively by the classes AC and AC1.

- A private action of AP is modelled either by AP.pop(e_1, . . . , e_n), where pop(T_1,. . . , T_n) is an operation of AC stereotyped by <<priv>> or by X = AP.pop(e_1, ..., e_n), where pop(T_1,. . . , T_n): T stereotyped by <<priv>> and X is a participant.

- An interaction of AP with AP1 is modelled by the AP1.op(e_1, . . . , e_n) performed by AP, where op(T_1,. . . , T_n) is an operation of AC1.

where T, T_1 , . . . , T_n are passive entity classes (i.e. either datatypes or classes stereotyped by <<object>>) and e_1 , . . . , e_n are OCL expressions in which only the process participants names freely appear.

3.3.1 Entity View

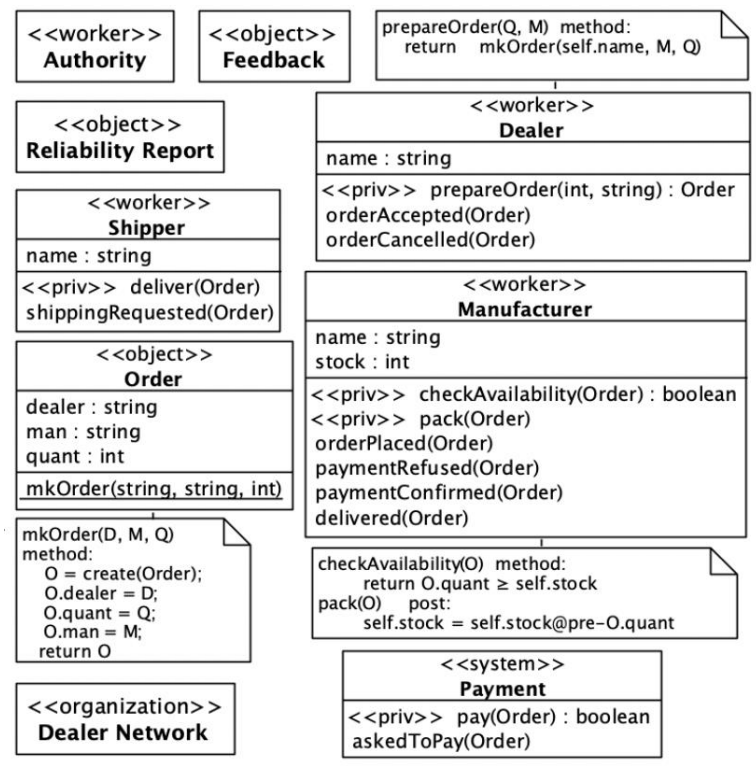


Fig. 29 Dealer Network UM2 Model: Entity view

Fig. 29 represents the entity view of the Dealer Network UM2 model. Notice that the entity view of UM1 (see Fig. 22) is different from this one (the differences are related to the operations). Participant operations are seen through the task view in UM1. Whereas in UM2 they are modelled in the entity view as operations of active participant in the active participant entity. Notice also that operations of an active participant in this model are defined using OCL pre/post constrains.

3.3.2 Interaction View

An interaction view is produced by the means of sequence diagram. Interaction view in UM2 represents the message exchanged between active participants. each active participant has a lifeline named as participant itself. The messages are modelled as call of operations representing communication. The modeler has to take care about the ordering of the messages (in accord with the flowing of the actions in the workflow) else the methods user might not understand the sequence of ordering messages.

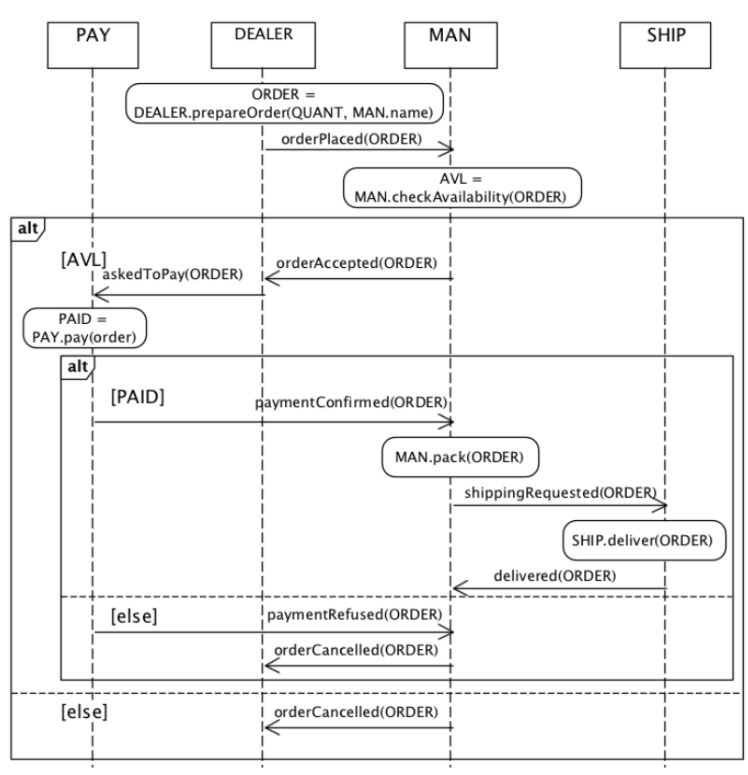


Fig. 30 Dealer Network UM2 Model: InterAction View

Fig. 30 represents the UM2 model Dealer Network interaction view. Notice that the interaction view not only shows the communications but also flowing of the participant activities. Thus, the method user easily understands the messages and activities performed by the active participant.

3.3.3 Organization View

The UM2 model of the Dealer Network has the same organization view and the same processes overview of the UM1 model.

3.3.4 Workflow View

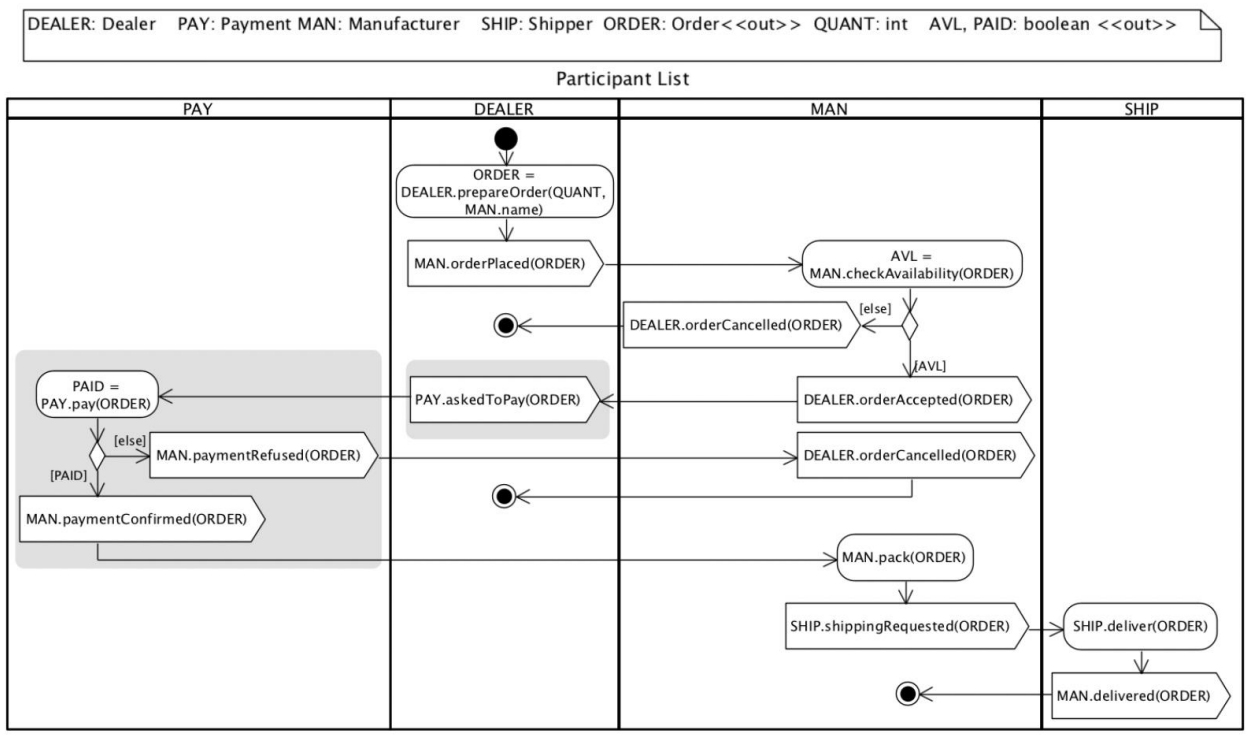


Fig. 31 Dealer Network UM2 Model: Workflow View

Fig. 31 represents the workflow view of UM2 model Dealer Network. Notice that workflow view in UM1 (see fig 26) is different to this one (the differences are related to transformations of the abstract tasks to executable tasks and the communications between active participants). Consider the abstract task **Payment** with four tasks (highlighted in the picture) transformed in to executable illustrating the dealer requesting pay-system to pay. The payment system examines the request and send the result (confirmed/refused) to manufacturer.

Comparing the two models (UM1 and UM2) results in different abstraction level. Both methods are useful. For example, one can use UM1 model to discuss about the process with stake holders and UM2 can be used to provide software support since all the details are explicitly and precisely defined.

3.4 UM3 (Hard- Executable task) Method

The UM3 model is slightly different compared to UM2 Model (the differences are related to the executable tasks that are modelled using entity view class operations) but the model structure is same as UM2. An executable task performed by an active participant of a business processes will be represented by a self-call of an operation of its class. The workflow view, organization view, process view is similar to the UM2 model. Differently from the UM2, the usability of swimlanes are optional. UM3 model prescribes to model an executable task in the following way.

Precisely, let AP and AP1 be two active participants of a business processes typed respectively by the classes AC and AC1.

- A private action of AP (as in UM2) is modelled either by AP.pop(e_1, ..., e_n), where pop(T_1,..., T_n) is an operation of AC stereotyped by <<priv>>
  or by X = AP.pop(e_1, ..., e_n), where pop(T_1,..., T_n): T is an operation of AC stereotyped by <<priv>> and X is a participant.
- An interaction of AP with AP1 is modelled by AP.op(AP1, e_1, . . . , e_n) where op(AC_1, T_1, . . . , T_n) is an operation of AC

where T, T_1, . . . , T_n are either datatypes or passive entity classes (i.e. classes stereotyped by <<object>>) and e_1 , . . . , e_n are OCL expressions in which only the processes participant names freely appear.

3.4.1 Entity view

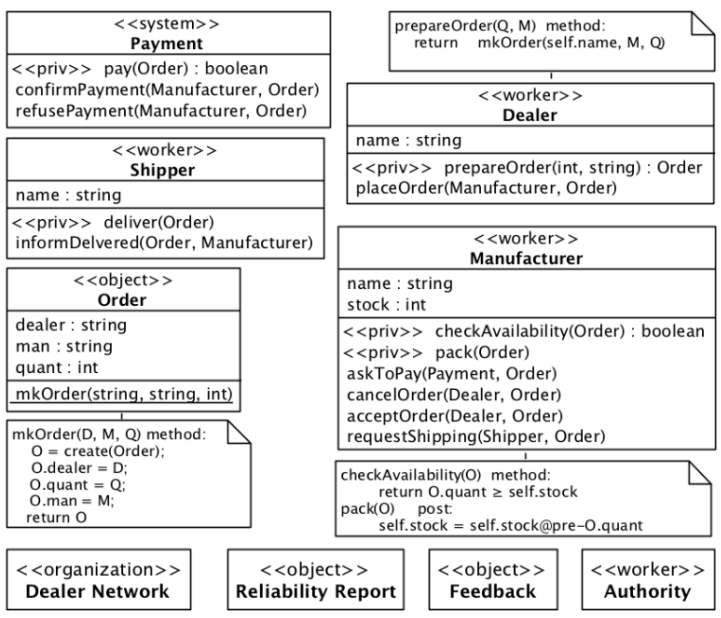


Fig. 32 Dealer Network UM3 Model: Entity View

Fig. 32 represents the entity view of the UM3 Model Dealer Network. Notice that the entity view in this model is slightly different compared to UM2 (see fig. 29). The differences are related to the communications that are modelled as operations in the communication initiator class (active participant class that starts communication belonging to the active participant class since he is initiator of the message).

3.4.2 Workflow View

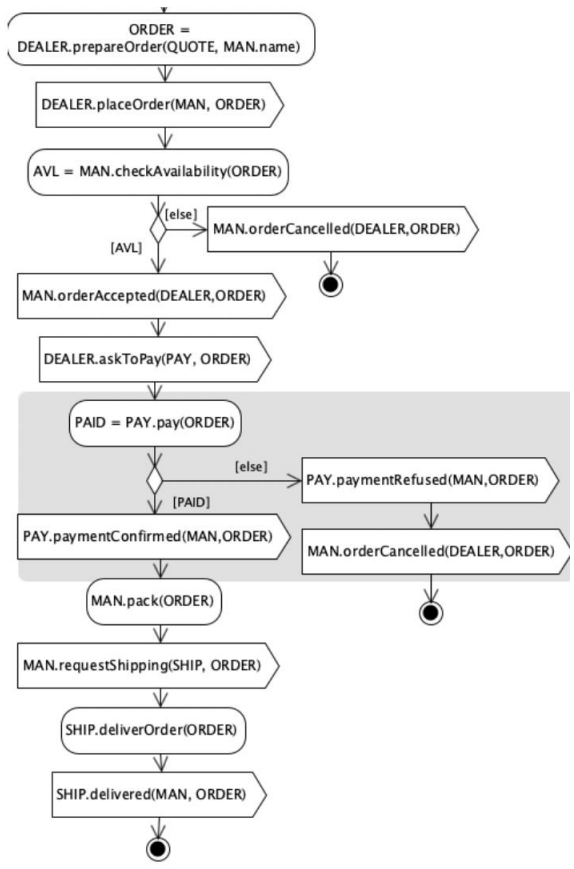


Fig. 33 Dealer Network UM3 Model: Workflow View

Fig. 33 represents the workflow view of the UM3 mode Dealer Network. Notice that the rendering of participants in the workflow view are similar to UM1 model (i.e., they are explicitly defined in the task)

3.5 Disciplined Modelling Methods (UM4 and UM5)

The disciplined models (loose models) are different compared to UM1, UM2 and UM3 (hard models). Disciplined models take the advantage of representing business process in plain natural language fragments instead of expressing with UML constructs. The modeler requires less effort and time to produce disciplined models. These models are also easily understandable to the people having minimum knowledge of UML constructs. The level of precision also differs when disciplined models are compared to Hard Models (UM1, UM2 and UM3). Maintaining a balance quality, it is still possible to model and represent who is taking part in the tasks, flowing of business objects among the different tasks.

Disciplined Modelling are further classified as two types

1. UM4 Model (corresponds to abstract tasks)
2. UM5 Model (corresponds to executable tasks)

3.6 UM4 (Disciplined-abstract tasks) Method

UM4 model is quite similar to the UM1 model except the usage of UML constructs. The structure is also same as UM1, but many operations of active entities are defined in natural fragments without using UML constructs.

3.6.1 Entity View

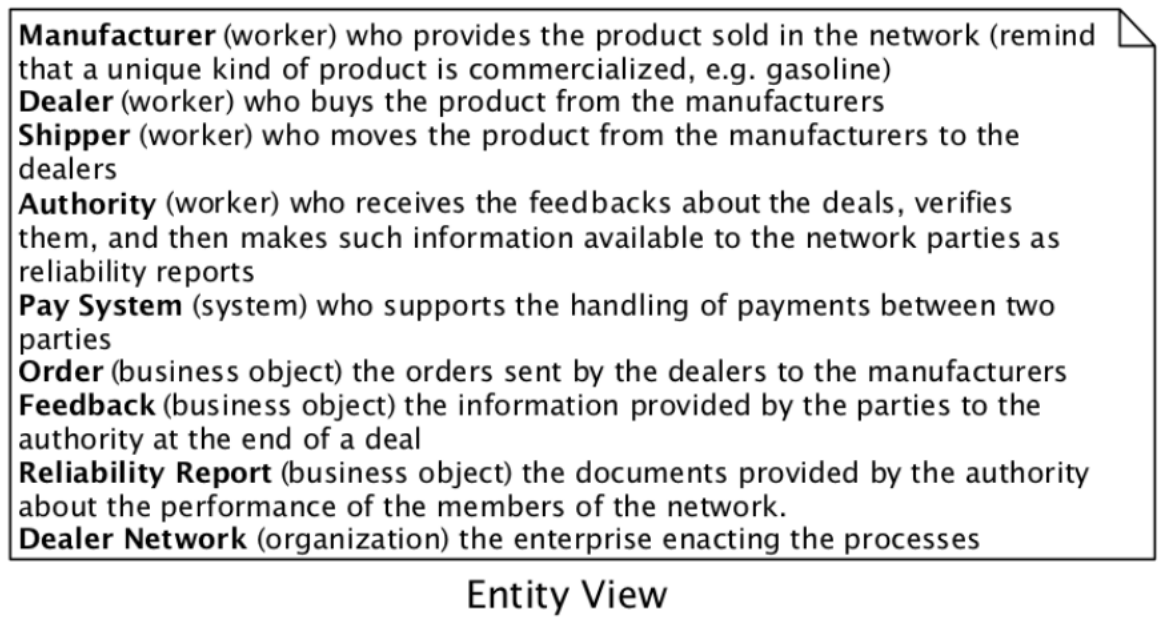


Fig. 34 Dealer Network UM4 Model : Entity View

Fig. 34 represents the entity view of UM4 model Dealer Network. The entity view is slightly different here compared to other models. the entity view with business entities that are part of the enterprise enacting the processes are defined in the following form:

EntityCategory1 (kind1) description1 ...
EntityCategory$_n$ (kind$_n$) description$_n$

where for i = 1, . . . , n EntityCategory $_i$ is a string starting with a capital letter, kind$_i$ belongs to {worker, system, business object, data}, and description$_i$ is an optional natural language fragment. For example, "**Dealer** (worker) who buys the product from the manufacturers". Dealer corresponds to entity category and the kind is written as (worker) and the description is written in natural text

3.6.2 Task View

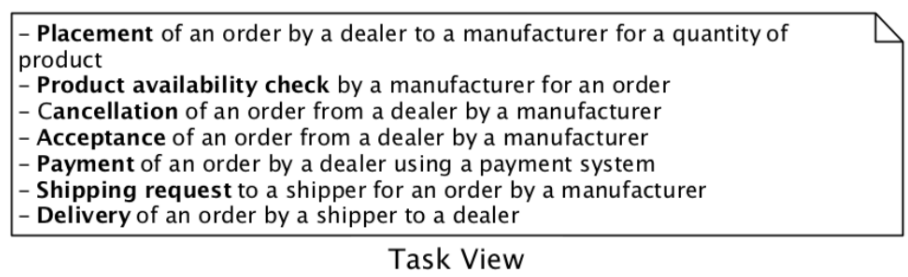


Fig. 35 Dealer Network UM4 Model :Task View

Fig 35 represent the task view of UM4 Dealer Network. The task view is produced using natural language and its structure is given below.

action description1 comment1 ...
action description$_n$ comment$_n$

where for i = 1, . . . , n action description I natural language fragment where the substantives should be generic elements of the entity types listed in the entity view (e.g. "A client pays an order to a seller ", "A client, a seller and a shipping agent agree on the date and the price of the delivery of an item", and at least one substantive must correspond to a generic element of an active entity type; description$_i$ is an optional natural language fragment).

3.6.3 Workflow View

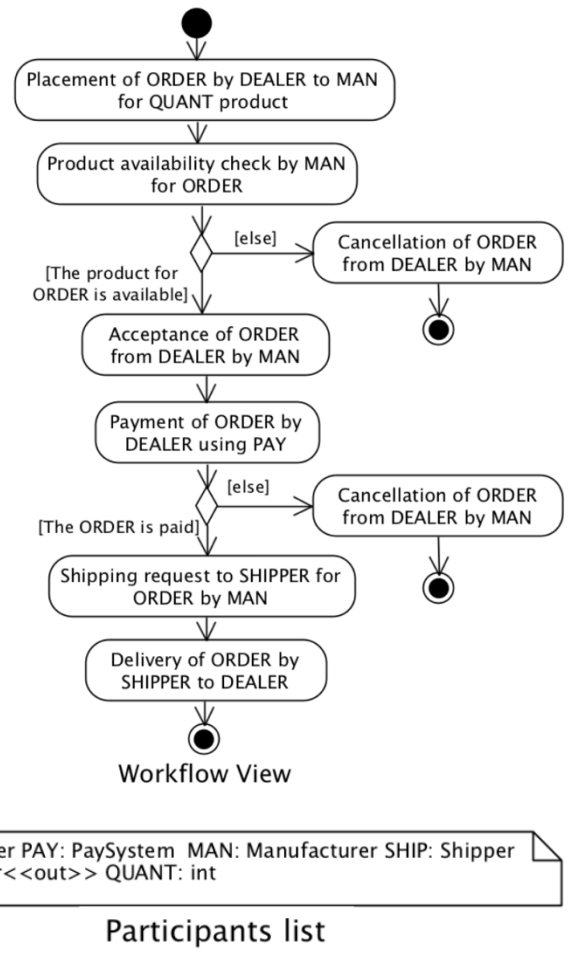


Fig. 36 Dealer Network UM4 Model :Workflow View

Fig 36 represents the workflow view of the UM4 model Dealer Network. The workflow is defined in the natural language fragments. The name of the action node is labeled by natural fragments corresponding to the tasks in the task view. The names of the Participant are written in capital letters (using generic names is avoided since we need to clearly highlight the process participants). For example, "The CLIENT pays the ORDER to the SELLER", "The CLIENT, the SELLER, and the SHIPPER agree on the date and the price of the delivery of the ITEM". The guards on the condition must also be written using natural language describing the condition. Notice that workflow in fig. 36 is easy to understand.

3.7 UM5 (Disciplined-Executable tasks) Method

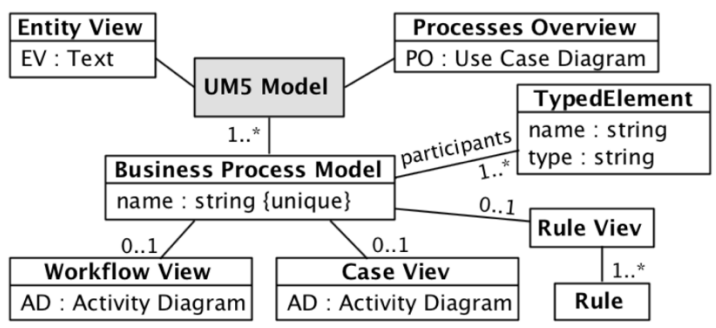


Fig. 37 UM5 Model Structure

The difference between the UM4 and UM5 are the tasks. All the tasks in the UM5 model are executable tasks and they correspond to private actions and communications similar to the UM2 model. The private actions are expressed in natural language sentences having a specific form. The operations of entity view are defined in the following form.

action description1 comment1 ...
action $description_n$ $comment_n$

where for i = 1, . . . , n action description $_i$ is a sentence where the subject must a generic element of ActCat, the substantives must be generic elements of the entity types listed in the entity view, and at most one substantive must correspond to a generic element of an active entity type (worker or system). $description_i$ is an optional natural language fragment. For example, "A client pays an order to a seller " represents a communication between two active entities, "A seller compute the pickup date and the price of an item", instead, is a private activity of an active entity.

3.7.1 Entity view

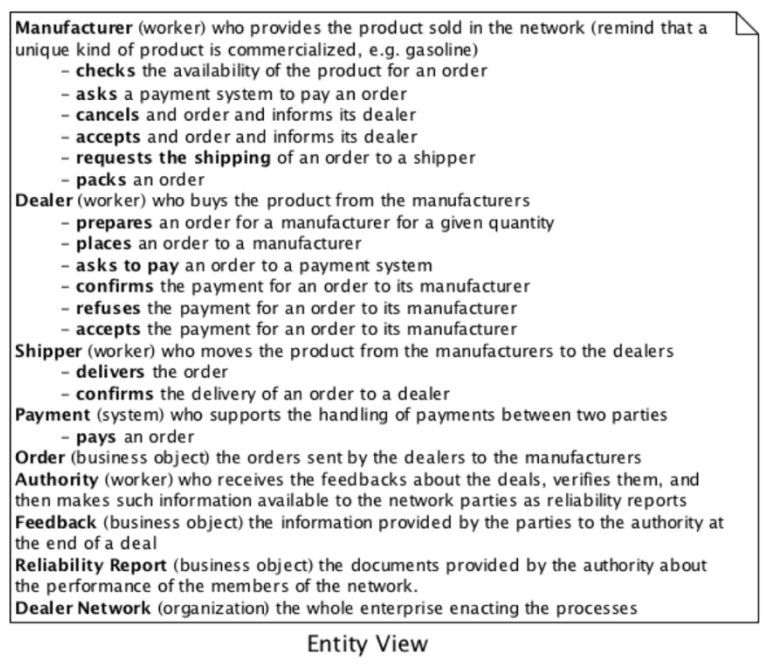


Fig. 38 Dealer Network UM4 Model :Entity View

Fig.38 represents the entity view of UM5 Dealer Network. Notice that all the operations of active entities are also modelled in natural language.

3.7.2 Workflow View

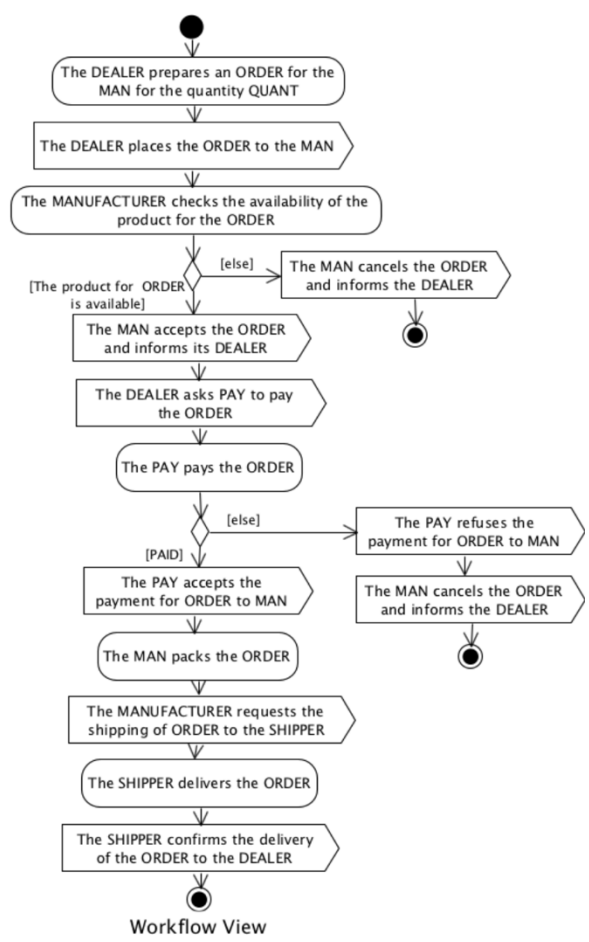


Fig. 39 Dealer Network UM4 Model : Workflow View

Fig. 39 represents the workflow view of the UM5 Dealer Network. The action node is labeled using natural language. The description on conditions leaving the decision node are also labeled in natural language. Notice that UM4 also uses send signal icon to represent communication similar to the UM2 model.

3.8 Selecting a best suitable model for business process

We have five UML models and their selections depends on modelers choice concerning the business process and modelling requirements. However, to be more specific, one might select best suitable model from the table below where we have briefly discussed characterizing features of each model.

	Features									
Method	**Task granularity**	**Formality**	**Workflow View**	**Swimlanes in workflow view**	**Case View**	**Rule View**	**Interaction View**	**Mockups**	**Complex data structures**	**Computations**
UM1 (Hard-Abstract tasks)	abstract	hard	yes	optional	yes	yes	no	no	yes	yes
UM2 (Hard-Executable tasks-Interaction)	executable	hard	yes	mandatory	yes	yes	yes	yes	yes	yes
UM3 (Hard-Executable tasks)	executable	hard	yes	optional	yes	yes	no	yes	yes	yes
UM4 (Disciplined-Abstract tasks)	abstract	disciplined	yes	optional	yes	yes	no	no	no	no
UM5 (Disciplined-Executable tasks)	executable	disciplined	yes	optional	yes	yes	no	yes	no	no

Fig. 40 UML models with their characteristic features

Business process with interaction: when we have a business process with communications, one might consider using the UM2 model since it has an interaction view and it provide an additional description of a business process in terms of message exchanges among its active participants. But notice that the use of swimlanes in the UM2 model is mandatory and sometimes siwmlanes may hinder the readability of the workflow. For example, consider the case where the business process has a many number of active participants and they interact with many other. In such case choosing a model that does not use swinmlane or a model where the usability of swimlane is optional is preferred.

Business process with complex data structures and computations: Complex data structures can be defined using hard methods since they allow defining the data structures (by classes and datatypes in the entity views) together the operations for manipulating them (by pre-post conditions or methods) with a varying degree of precision. Modelling computations of a business process with hard methods allow to precisely define computations over data and business objects. For example, computing the monthly payment of a mortgage or the tuition fees for a master that depend on many factors, by means of pre/post conditions or methods without cluttering the workflow view. The modeler also has to consider the fact that to produce hard models takes high effort and time. One might consider using the disciplined methods since they just define relevant details of business process in natural language text, but again disciplined models has different level of precision. Thus, depending on the level of precision and usability views used in the business process, one might pick the model.

3.8.1 Shifting among the UML Models

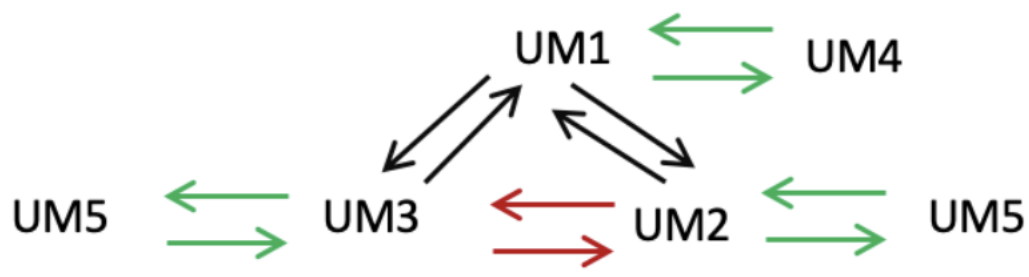


Fig. 41 Transformations among the different business process models

Fig. 41 represents the possible transformations among five UML methods. The green arrow in the picture shows the level of formality of the models (i.e., how they are relatable with the relevant features). The black arrows show the passing of model based on abstract task to executable task. The red arrows show the different kinds of models based on executable tasks. All the transformations are simple and easy to realize since the conceptual view of models are same.

Abstract tasks to Executable tasks: The modeler can start producing the model in abstract task (the behaviour of the process is defined in workflow view) and transform it to executable task. The result of transforming not only produces an executable model but also allows the modeler to concentrate and understand the behaviour of the process by analysing the basic actions of participants. The modeler might consider transformation to executable model, if the business process needs an interaction view.

Executable tasks to Abstract tasks: The modeler can to start producing the model using executable tasks where the behaviour of process is defined in the workflow using executable tasks. The transformation of executable tasks to abstract tasks allows the modeler to decompose the business process (remember when the task is transformed to abstract task, we do not have clear details about the task) and helping to identify the problematic issues.

4 BPMN Case studies

BPMN case studies are taken from the book "Business Process Model and Notation (BPMN)" version 2.0. This book provides a number of BPMN 2.0 examples, which are non-executable BPMN 2.0 models conforming to the Process Modelling Conformance class as defined in the OMG specification Business Process Model and Notation (BPMN) Version 2.0. The BPMN case studies are in the form of Collaboration diagrams, Process diagrams, and Choreography diagrams. Few case studies are also taken from the BPMN Incubator website (https://www.businesprovessincubator.com/category/type/templates/).

4.1 Shipment Process

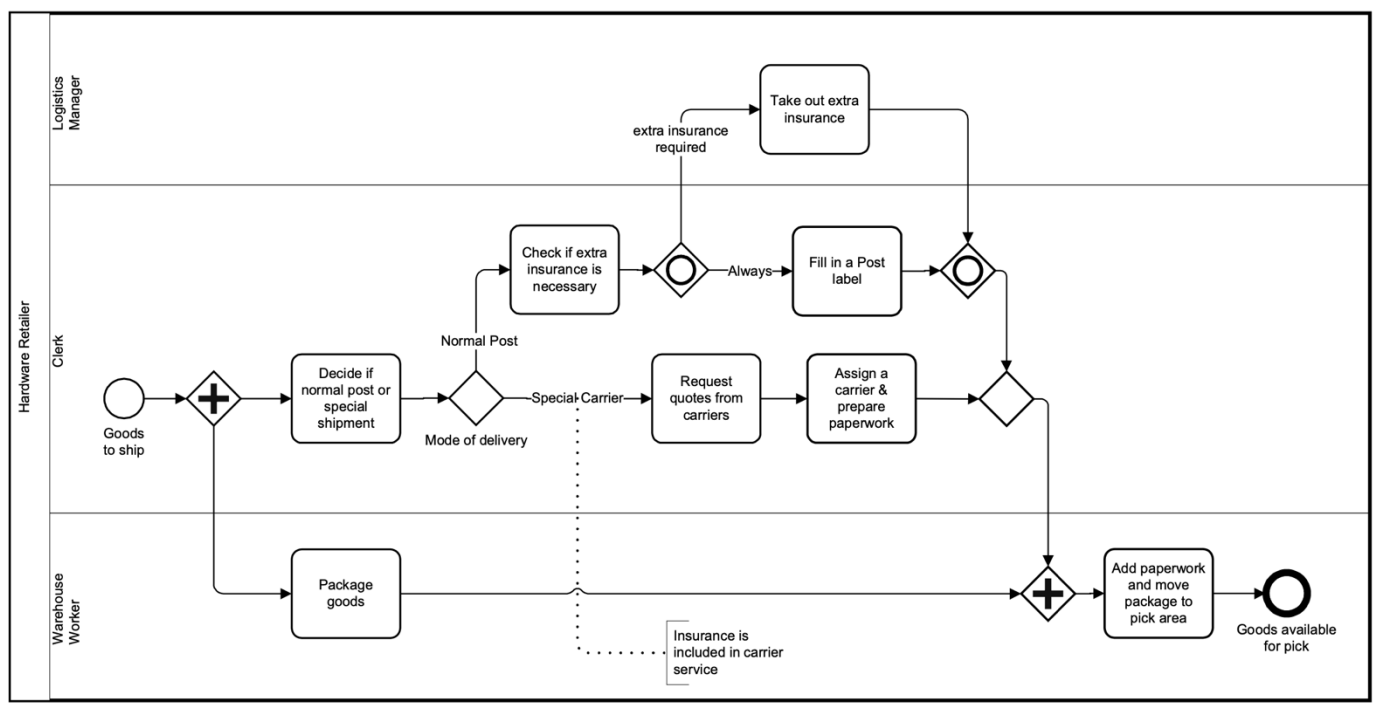


Fig. 42 Shipment Process: BPMN Model (process diagram)

4.1.1 The case study

Fig.42 represents the BPMN model of the Shipment Process (a process diagram).

While the clerk decides whether a normal post or a special carrier is needed, the warehouse worker packages the goods. If the goods need a special carrier, the clerk requests quotes from different carriers, then assigns the delivery to a carrier and prepares the paperwork. In the meantime, the clerk will fill a postal label for the shipment, and if the goods are sent by normal post, then they check if an extra insurance is required. If the goods need extra insurance, the logistics manager will take out the insurance and finally, the warehouse worker adds the paperwork and moves the goods to the pickup area.

The BPMN diagram in Fig.42 consists of one single pool defining the main organizational unit (i.e., the Hardware Retailer) subdivided in three lanes grouping the activities performed by the participants (i.e., Clerk, Logistic Manager, and Warehouse Worker). This model does not represent any communication between the participants.

The diagram contains three kinds of gateways (i.e., ⊕ Parallel Gateway, ◎ Inclusive Gateway, and ◇ Exclusive Gateway). A Parallel Gateway is used to synchronize (combine) parallel flows and to create parallel flows. A Parallel Gateway creates parallel paths without checking any conditions; each outgoing Sequence Flow receives a token upon execution of this Gateway. For incoming flows, the Parallel Gateway will wait for all incoming flows before triggering the flow through its outgoing Sequence Flows" (see [1, sect 10.5.4]). The inclusive gateway specifies that one or more of the paths leaving will be taken. The Exclusive gateway specifies that exactly one path will be taken from the available paths. The workflow starts at the Start Event ○ in the clerk lane and ends when the flow reaches the End Event ○ in the warehouse worker lane.

The BPMN process diagram in Fig 42 illustrates the tasks performed by the three workers (i.e., clerk, logistic manager, and warehouse manager). This diagram does not contain detailed information about such tasks. For example, the clerk decides if the package is a normal or a special shipment, but we do not know on what factors the decision is made.

This BPMN tasks in this model may be considered as abstract tasks in our UML based methods (i.e., UM1 Hard-Abstract tasks and UM4 Hard-Abstract tasks). The UM1 method leads to model all the details of the process precisely. For example, the task view in the UM1 model may show the behaviour of the tasks precisely. The UM4 method leads to model the process using the natural language, which increases the understandability of the model. Thus, the Shipment Process will be modelled using the UM1 and the UM4 methods.

4.1.2 UM1 model of the Shipment Process

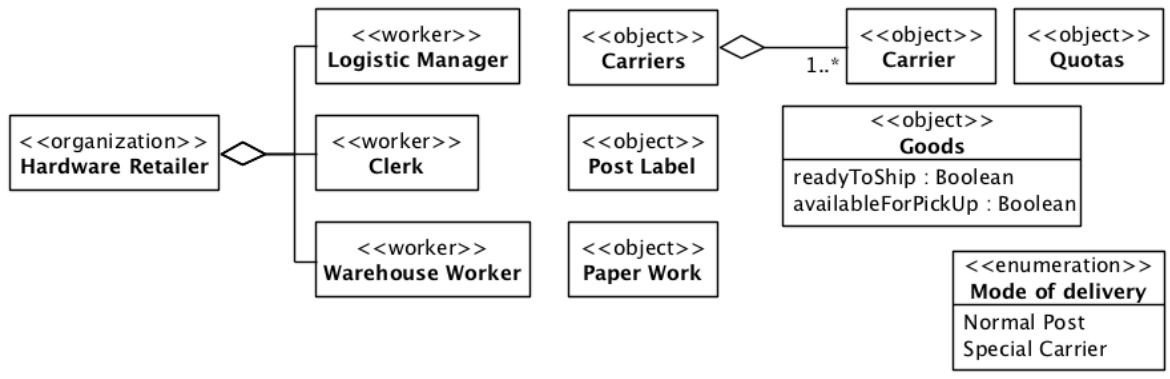


Fig. 43 Shipment Process UM1 Model: Entity view

All the entities involved in the Shipment Process are explicitly defined in the Entity view, see Fig.43. The Hardware Retailer entity class is stereotyped as organization since it corresponds to the whole enterprise enacting the process. The Hardware Retailer is composed of one logistic manager, a clerk, and a warehouse worker. The structuring is modelled by the aggregation relationships between the Hardware Retail class and the other worker classes, and by the Organization view, see Fig.44. The Organization view also represents the possible communications between the Hardware Retailer components. Notice that we used port and connectors to show communications between participants.

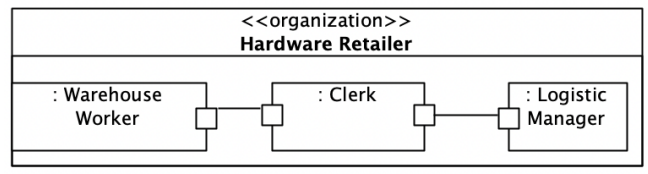


Fig. 44 Shipment Process UM1 Model: Organization View

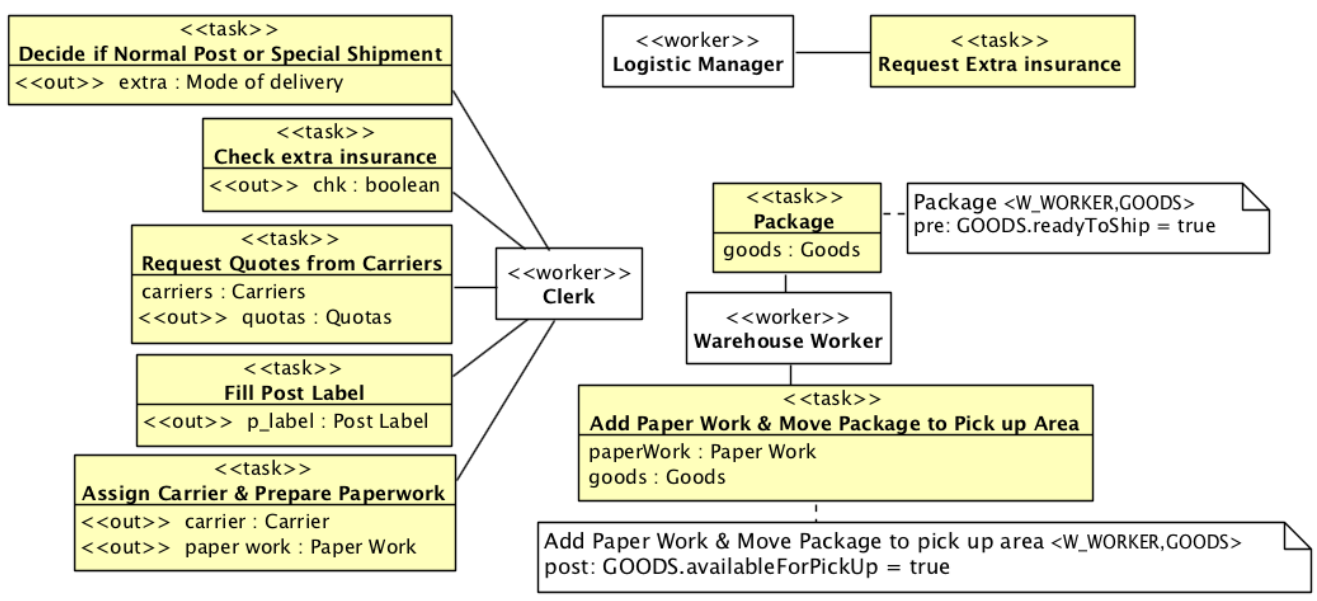


Fig. 45 Shipment Process UM1 Model: Task View

The UM1 model contains a Task view, see Fig. 45, that illustrates the tasks performed by the process participants. All the tasks are represented by classes stereotyped by <<task>> and they are connected to the participants entity classes using associations. Some OCL pre/post conditions are used in the task view to represent the behavioural aspects of tasks expressing which conditions should be satisfied before and after their executions.

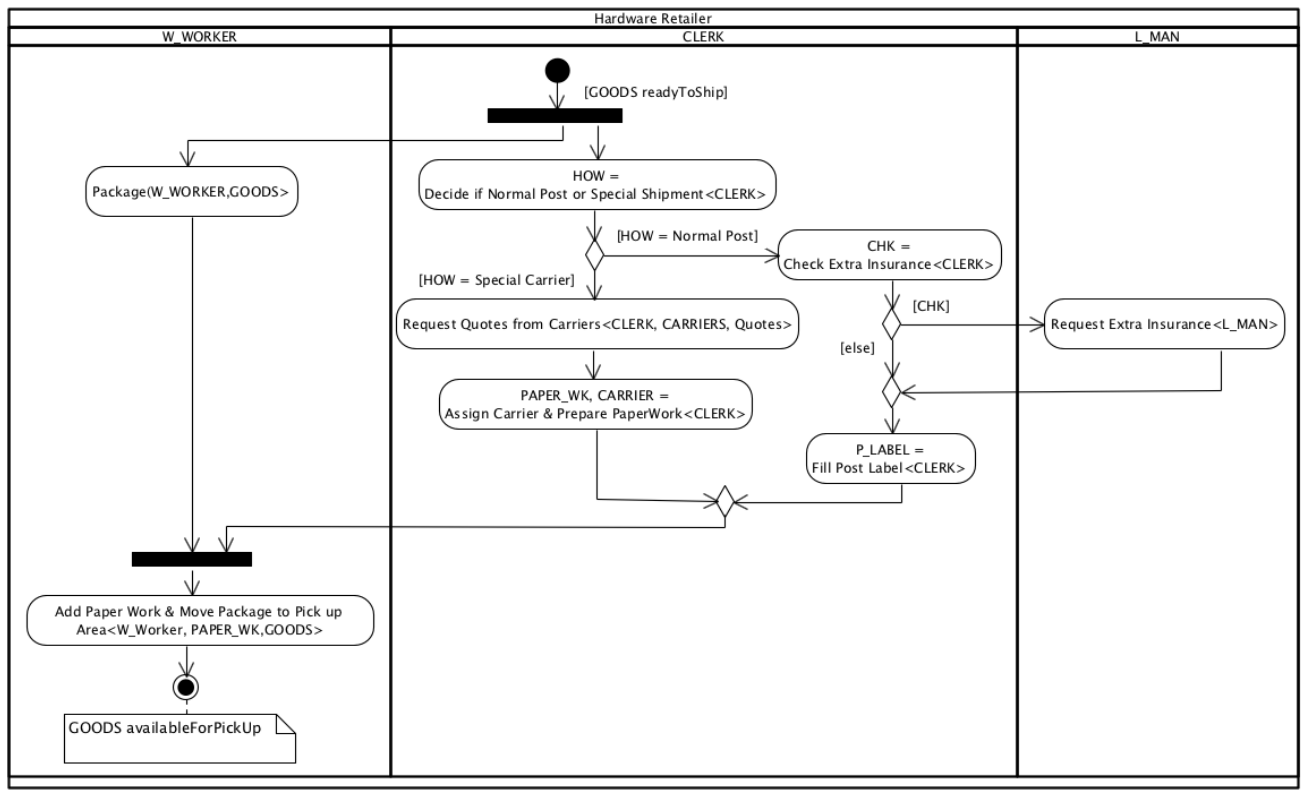


Fig. 46 Shipment Process UM1 Model: Workflow View

The Shipment Process workflow view part of the UM1 model is an activity diagram using swimlanes. We decided to use the swimlanes, also if UM1 does not enforce their use. Notice that the BPMN model uses pools and lanes to show the process participants. We wanted to produce a model similar to the BPMN model, thus we have modelled the workflow view using (nested) swimlanes.

4.1.3 UM4 model of the Shipment Process

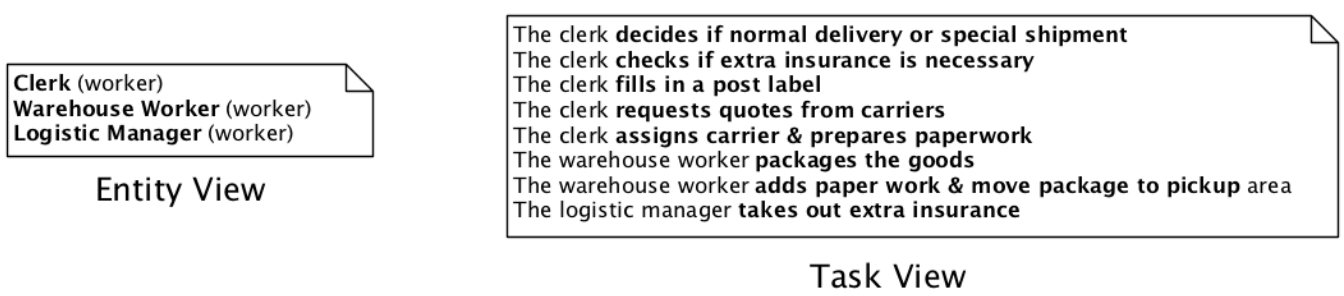


Fig. 47 Shipment Process UM4 Model: Entity & Task View

Notice that the entity view in this model is slightly different compared to UM1 model. The objects are explicitly defined in the entity view of UM1 model, whereas, here in the UM4 model, they are rendered in the task view using natural text.

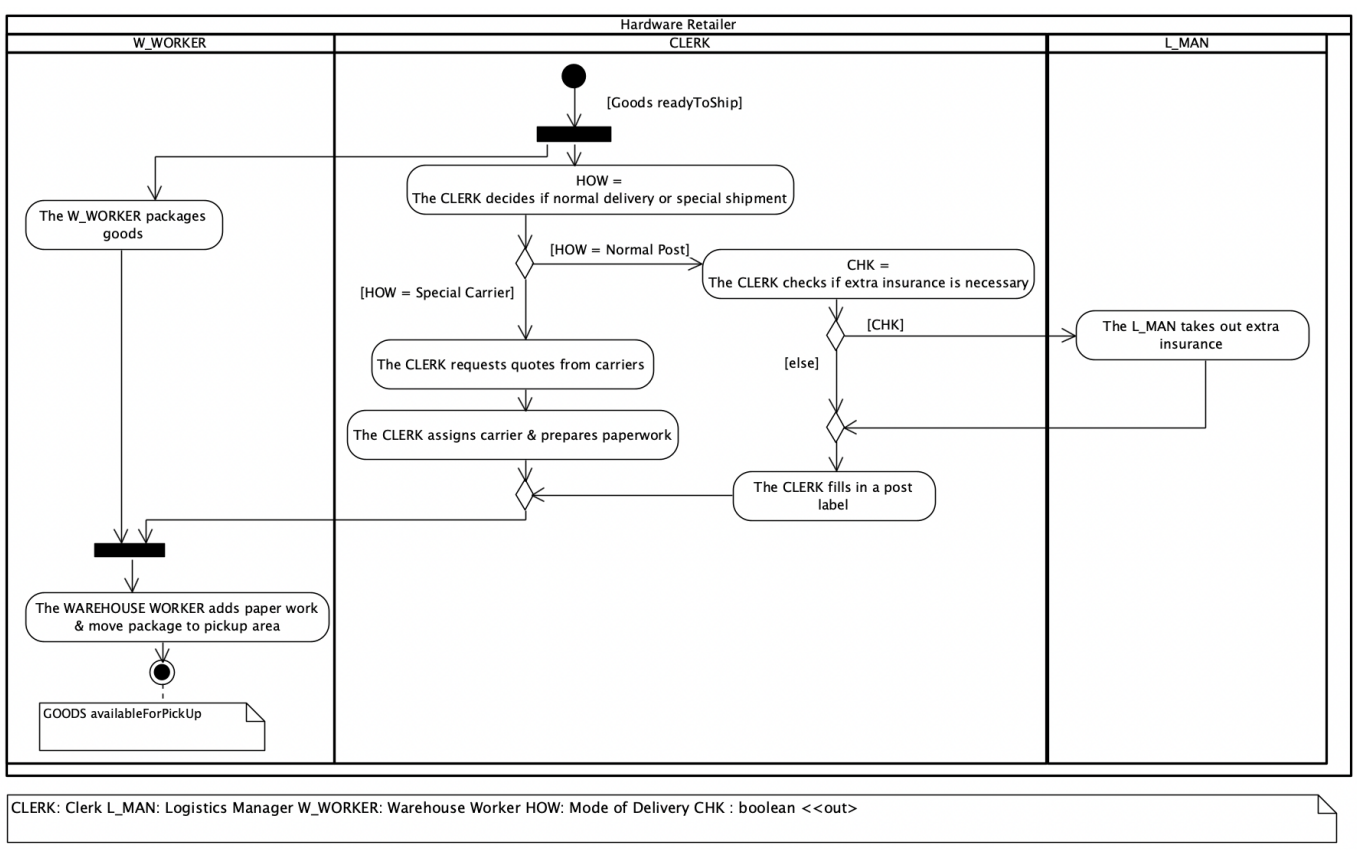


Fig. 48 Shipment Process UM4 Model: Workflow View

The workflow view of the UM4 model is similar to the one of the UM1 model (see, fig. 46).

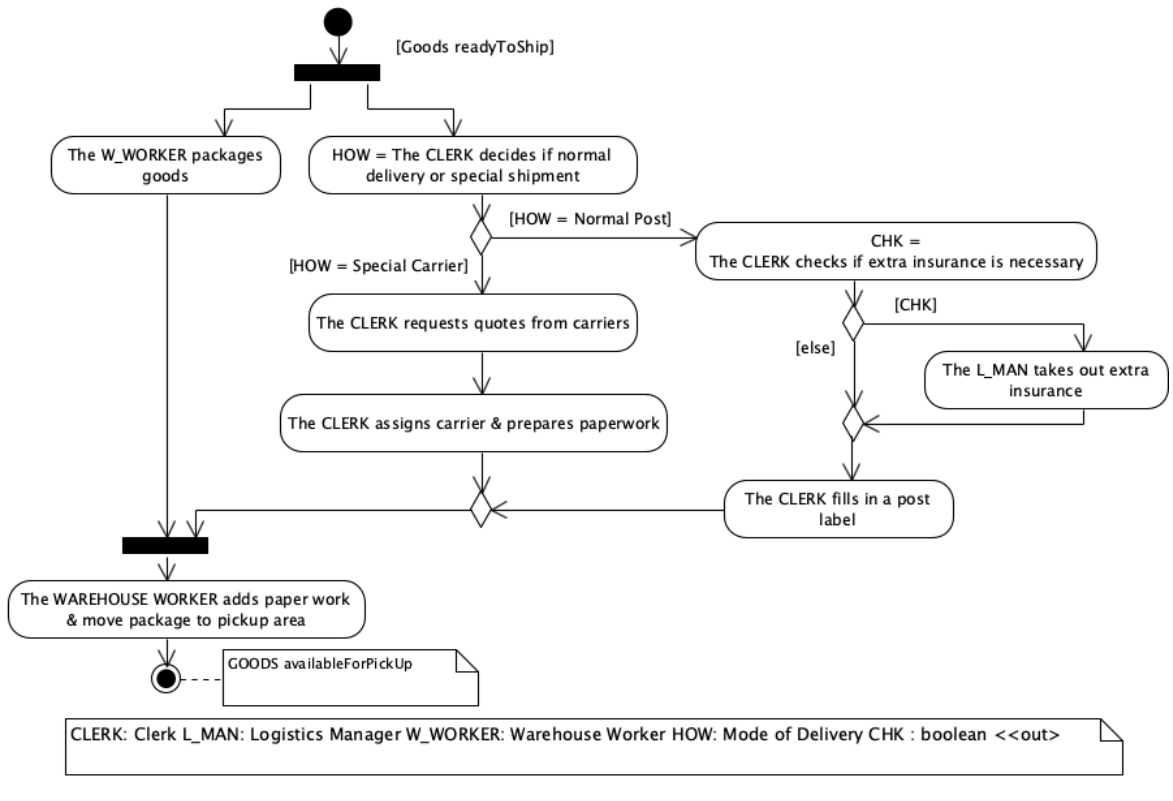


Fig. 49 Shipment Process UM4 Model: Workflow View without Swimlanes

Fig. 49 we present also the workflow without swimlanes, in this case the task participants are simply expressed by the text labelling the task.

4.1.4 Lessons learned

- In the BPMN model the task participants are rendered inside the task names, but not precisely/systematically defined. For example, the task participant (object/datatype in our models) "Post Label" is rendered inside the task "Fill in a Post Label" whereas, in our models we have explicitly defined the task participants.
- The BPMN tasks are named using verbs in active voice, precisely, the person or thing represented by a lane performs the action represented by the verbs appearing in the tasks inside such lane. The names of the tasks are explicitly declared in the entity view and by listing the task participants, and named following a uniform style (names of the tasks should be named by action names e.g. Packaging. Post label filling, but instead we have preferred following the BPMN names)
- Using the UM1 and UM4 methods, the Shipment Process is modelled much more precisely then using the BPMN. For example, in the BPMN there is no view to show which entities are business entities and how their structure is organized. Using the UM1 method, all the entities involved in the business process are precisely defined in the entity view and also the structure of the enterprise enacting the process is defined in the organization view. Using the UM4 method, all the activities are modelled in natural language. Notice that the UM1 model is quite precise since it clearly defines behaviour of operation of the Shipment process. Whereas the UM4 model is less precise since the tasks are modelled in natural language.
- To understand the BPMN model, one must have a basic understanding of BPMN constructs. Whereas our models are built with few basic UML constructs thus the method user has less to learn/know with respect to the BPMN constructs
- The flow in the BPMN model is always misleading since it seems that it is related in some way with the point of extra insurance, in our models it is more clear (due to the fact that we don't have the BPMN construct, but we model the same thing using basic flows and choice) not creating confusion and made using limited UML constructs.

The errors in the BPMN model identified during the UML modelling are:

- The "Package" is mentioned in the warehouse worker pool, but it is not used later.
- The "Post Label" was created in the clerk pool, but it is not used later.
- The "Carrier" is used by the clerk, but it is not mentioned earlier in the workflow.

4.2 Pizza Selling

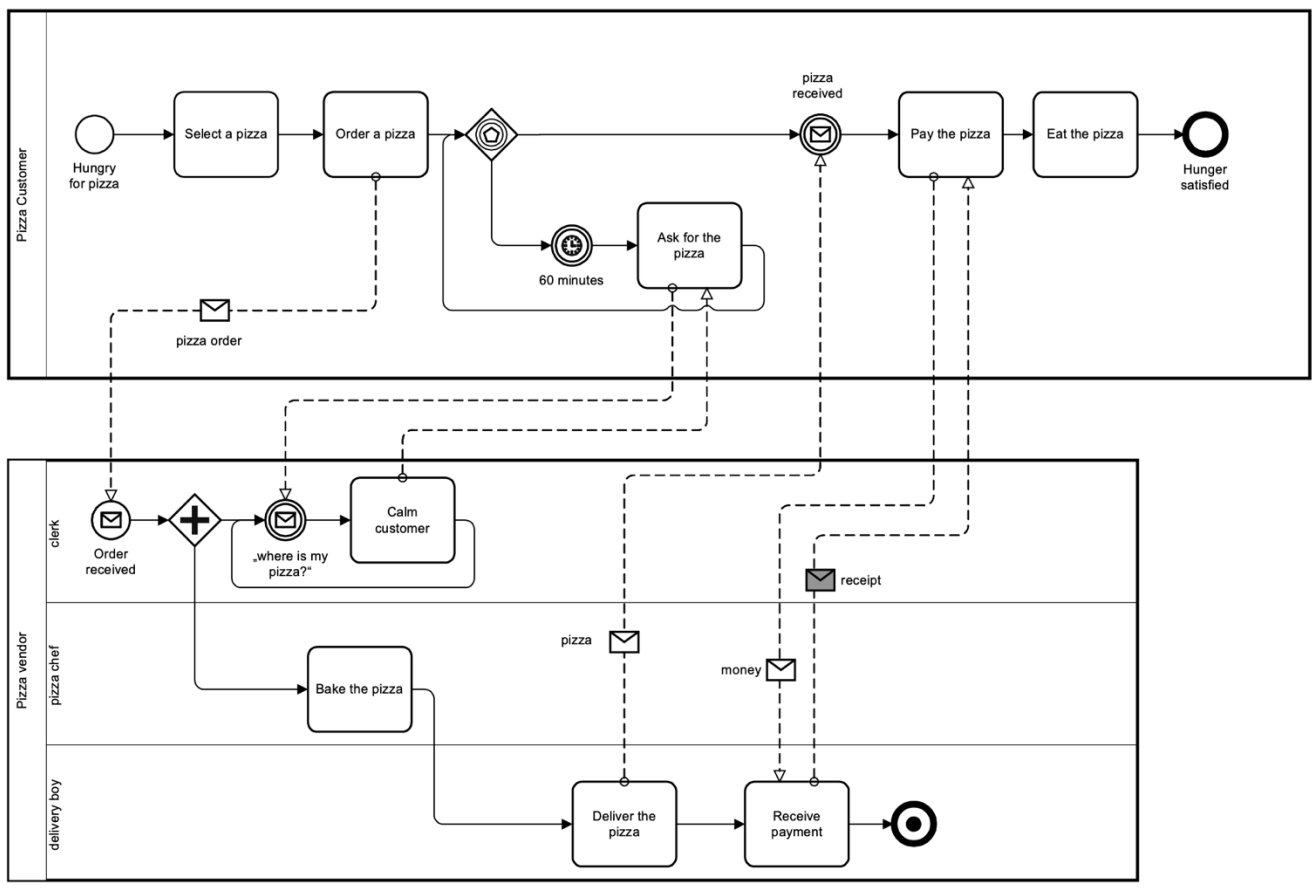


Fig. 50 Pizza Selling Business Process: BPMN model (collaboration diagram)

4.2.1 The case study

The BPMN collaboration diagram [3, sect. 5.2] reported in Fig. 50 demonstrates how the pizza gets delivered from a Pizza Vendor to a Customer.

The process starts when the Customer feels hungry, therefore they select a pizza and order a pizza. After ordering the pizza the Customer waits for the pizza to get delivered. If there is no delivery after 60 minutes, the Customer contacts the Clerk and asks about their pizza. The Clerk calms the Customer, and the Customer waits for the pizza again. In the meanwhile, after receiving the pizza order from the Customer, the Clerk requests the Chef to bake the pizza. After baking the pizza, the Chef requests the Delivery boy to deliver the pizza to the Customer. Therefore, the Delivery boy delivers the pizza, receives the payment, and gives a receipt to the Customer.

The collaboration diagram in Fig. 50 consists of two pools (i.e., Pizza Vendor and Customer). The pizza vendor pool is structured using three lanes modelling Clerk, Chef and Delivery Boy, respectively. Basing on the fact that BPMN pools are assumed to model the process participants, we also modelled the Pizza Vendor process with two participants. But, on the other hand, the use of the lanes suggests that the entities acting in this process are four (i.e., Pizza Customer, Clerk, Chef and Delivery Boy, where the last three are grouped inside the Pizza Vendor) thus it is also sensible to model the process with four participants. The diagram also demonstrates that the Pizza Vendor components communicate with the Customer, whereas the interactions among the last three participants are not communications.

The Pizza Selling business process is modelled using the method UM2 (Hard-Executable Tasks-Interaction). A UM2 model contains an interaction view, that represents the business process behaviour as a sequence of message exchanges done between the participants, and also shows the flowing of participant activities. The private actions performed by participants are clearly shown using the entity view and the workflow view. Compared to the other UML methods, UM2 is the most suitable to model the Pizza Selling business process since we can define and model the interactions and the private actions precisely.

The Pizza Selling business process is also modelled using the UM5 (Disciplined-Executable tasks) method since we wanted to see if a less precise modelling could be more similar to the BPMN model. In this case the executable tasks are expressed using natural language fragments instead of using UML constructs.

4.2.2 UM2 model Pizza Selling process with four participants

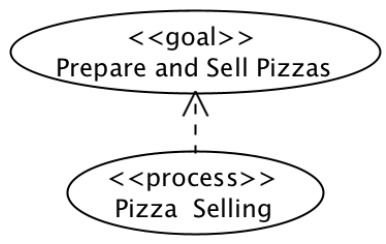


Fig. 51 Pizza Selling Process UM2 Model: Process Overview

The process overview represents the goal of the pizza vendor business process. Notice that the goals are not considered in the BPMN models, but our models have ability to show the goals of the business processes. The trivial goal of a Pizza Selling process is to deliver many pizzas to their customers. We have added the goals in the Pizza Selling model while we have overlooked them in the other models, since usually they are quite trivial, but more realistic processes need to represent their goals. The goals are stereotyped as <<goal>> showing goals of the business process and there always exists

a process to achieve a goal. The process is stereotyped as <<process>> showing what to be done to achieve a business goal. The process and the goal are linked through a dependency relationship.

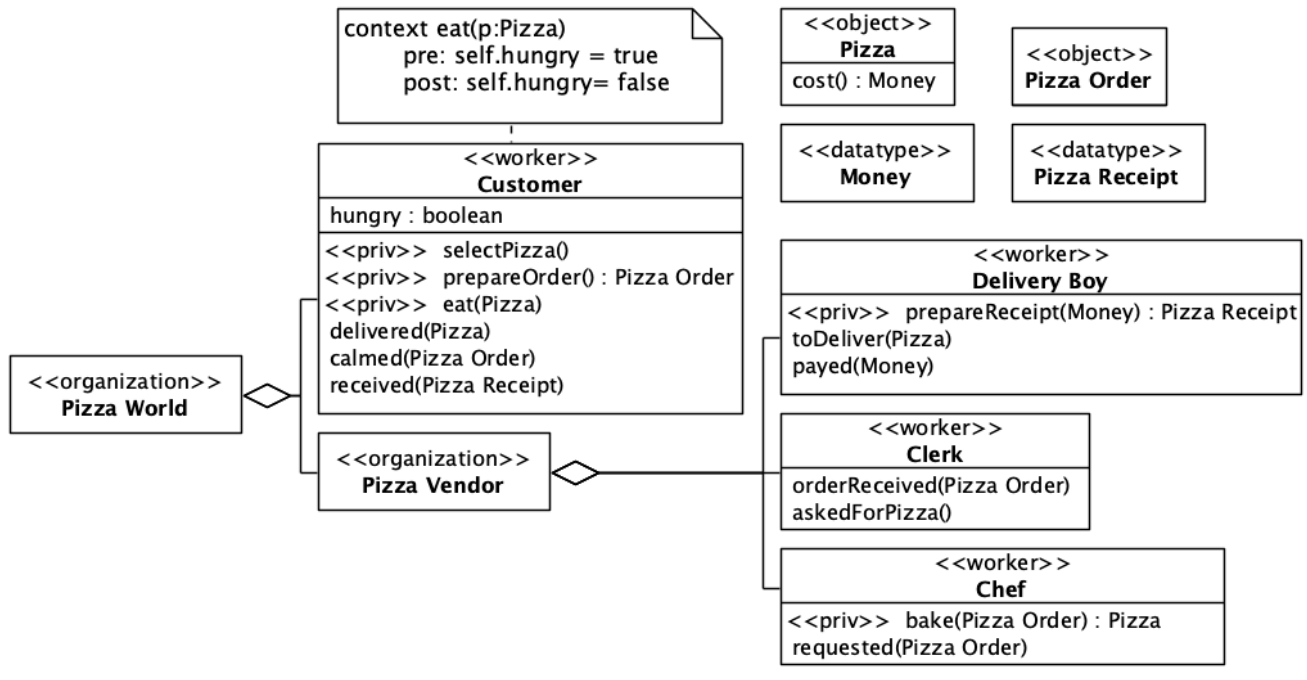


Fig. 52 Pizza Selling Process UM2 Model: Entity View

The entity view in Fig. 52 shows that the Pizza Vendor is composed one chef, one clerk, and one delivery boy, this structuring is modelled by the aggregation relationships between the Pizza Vendor class and their classes, and by the organization view, see Fig. 53.

Pre/post-conditions are attached to the operation eat of the Customer class showing which conditions have to be satisfied before/after the execution of the task that it represents.

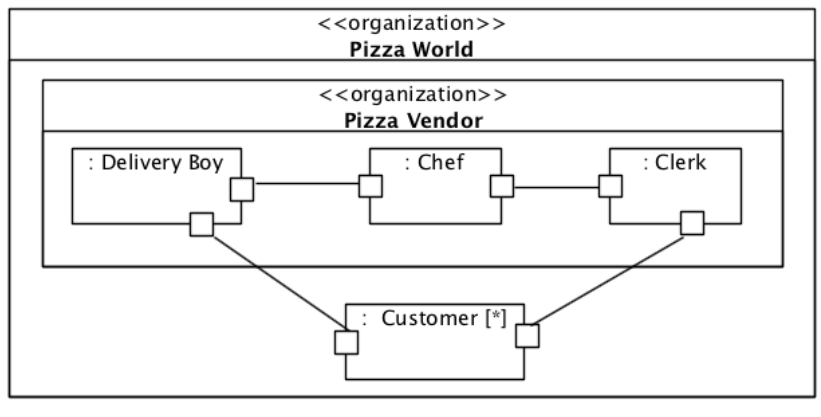


Fig. 53 Pizza Vendor Process UM2 Model: Organization View

Fig. 53 represents the organization view showing the possible communication paths among the worker participants.

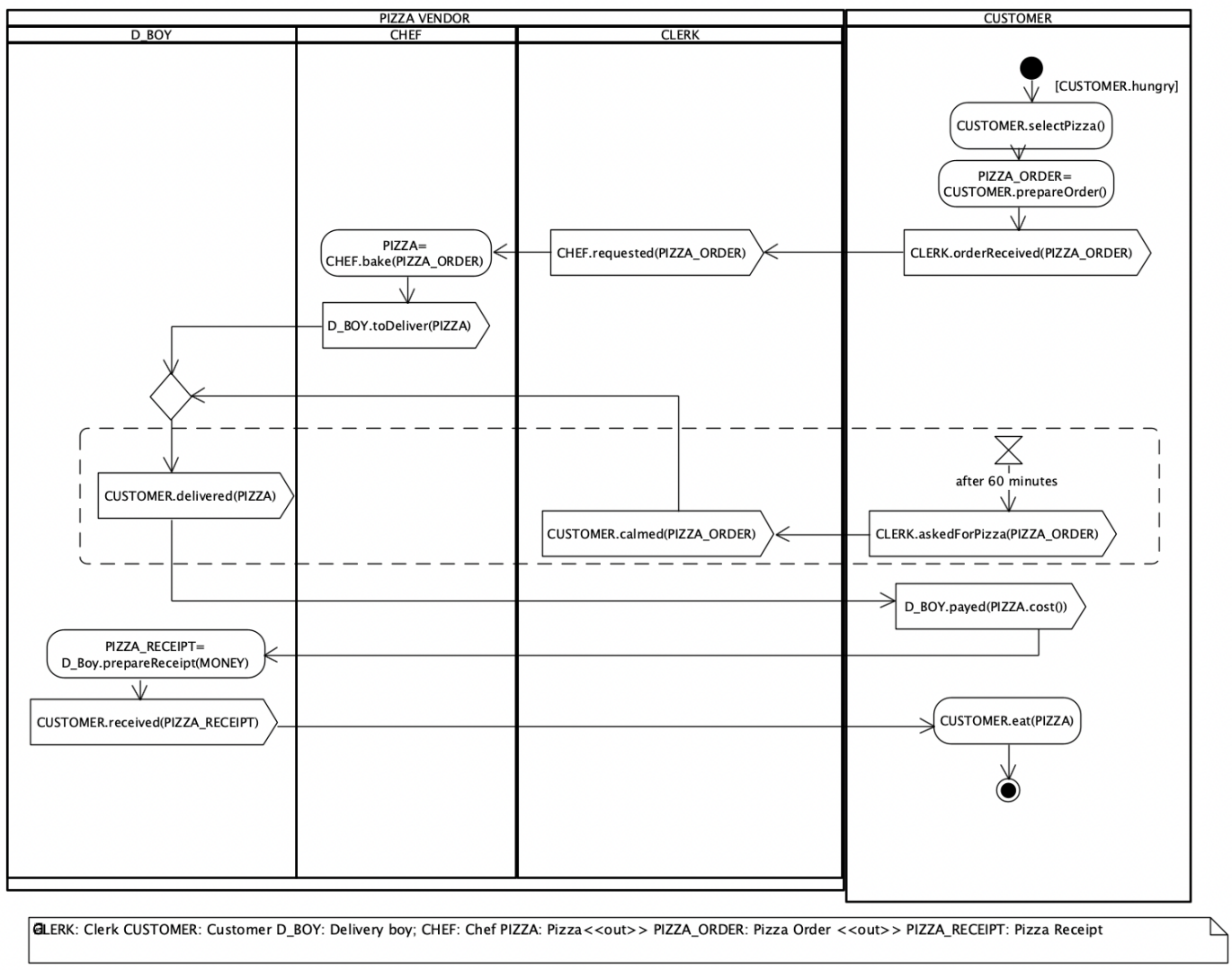


Fig. 54 Pizza Selling Process UM2 Model: Workflow View

The UM2 method provides a workflow view consisting in one activity diagram containing swimlanes. Swimlanes are used to show the process participant activities. Notice that, to increase the similarity of the workflow, we used multilevel swimlanes that are provided by UML and the methods will explicitly require them.

To model the time event appearing in the BPMN model, we used an interruptible activity region and an accept time event action to show how a certain action is performed after that a given time period is elapsed. An interruptible region contains activity nodes. When a flow leaves an interruptible region, all flows and behaviours in the region are terminated. In the above example after the Clerk calms the Customer, the interrupting edge flows to the merge node crossing the region boundary. Then all the activities inside the interruptible region are terminated.

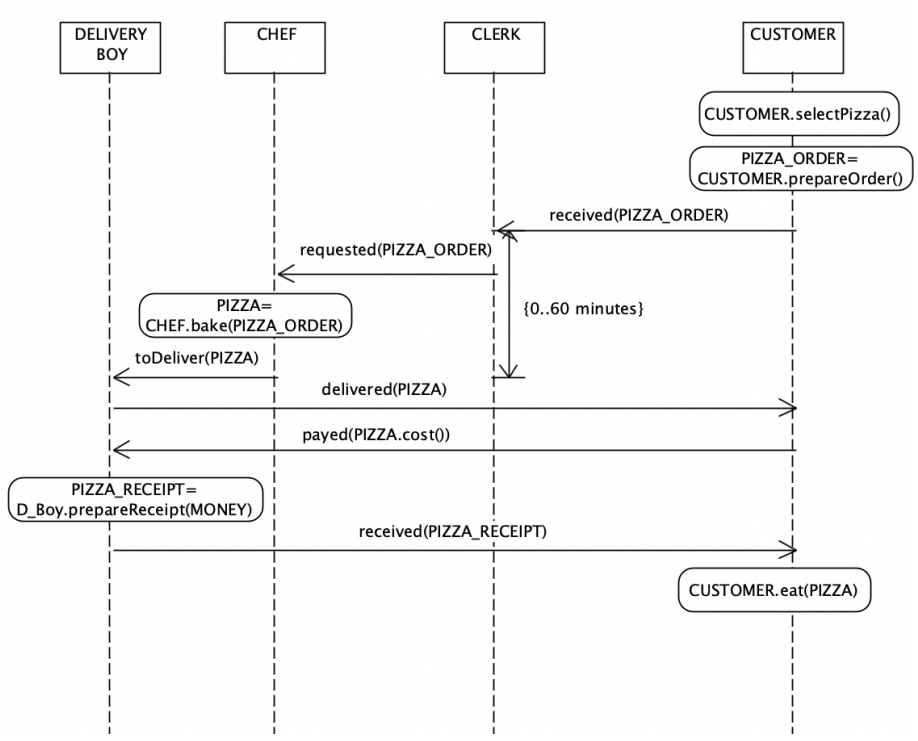


Fig. 55 Pizza Vendor Process UM2 Model: Interaction View

The UM2 method also provides an interaction view shown in Fig. 55. The interaction view shows the sequence of messages exchanged between the active participants. For example, the active participant Chef communicates with the Delivery boy to deliver pizza and the active participant Clerk request Chef to bake the Pizza for the Pizza order.

4.2.3 UM5 model Pizza Vendor process with four participants

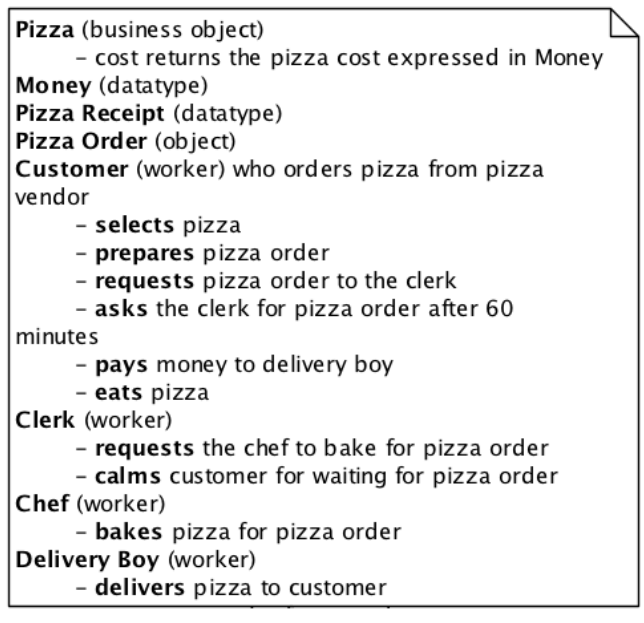


Fig. 56 Pizza Vendor Process UM5 Model: Entity View

Fig. 56 shows the Pizza Vendor process UM5 Model entity view. Notice that entity view in UM5 is different compared to entity view in UM2 in Fig.52, indeed now the entity classes and their operations are expresses using the natural language.

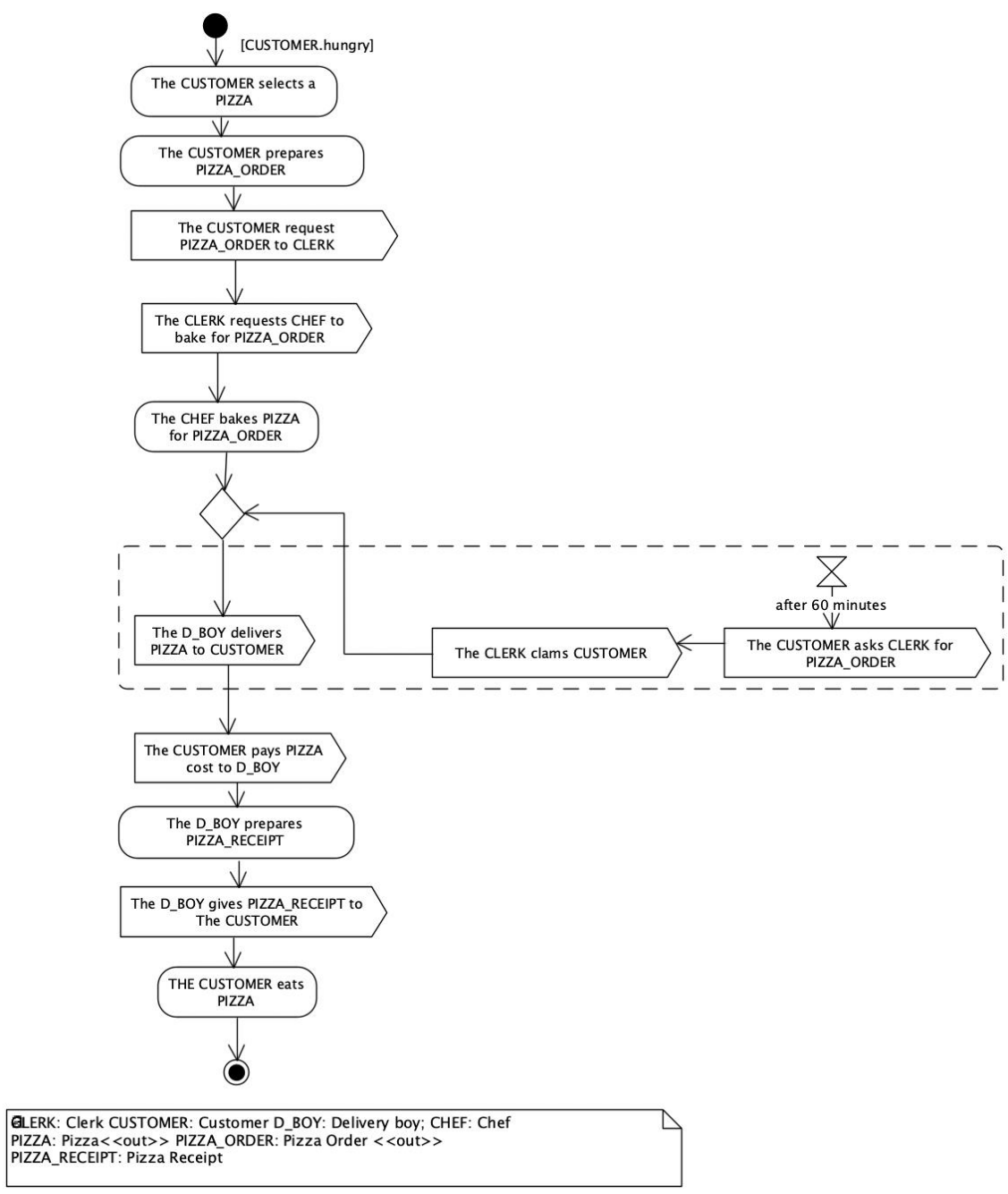


Fig. 57 Pizza Vendor Process UM5 Model: Workflow View

The workflow in the UM5 model is different compared to those of the UM2 model. The swimlanes in UM5 are optional. Yet, to produce a similar version to BPMN, the UM5 workflow is produced using swimlanes, see Fig. 58. We have also produced the workflow view, see Fig. 57, without swimlanes to show a unique way to render it, and this cannot be supported by BPMN.

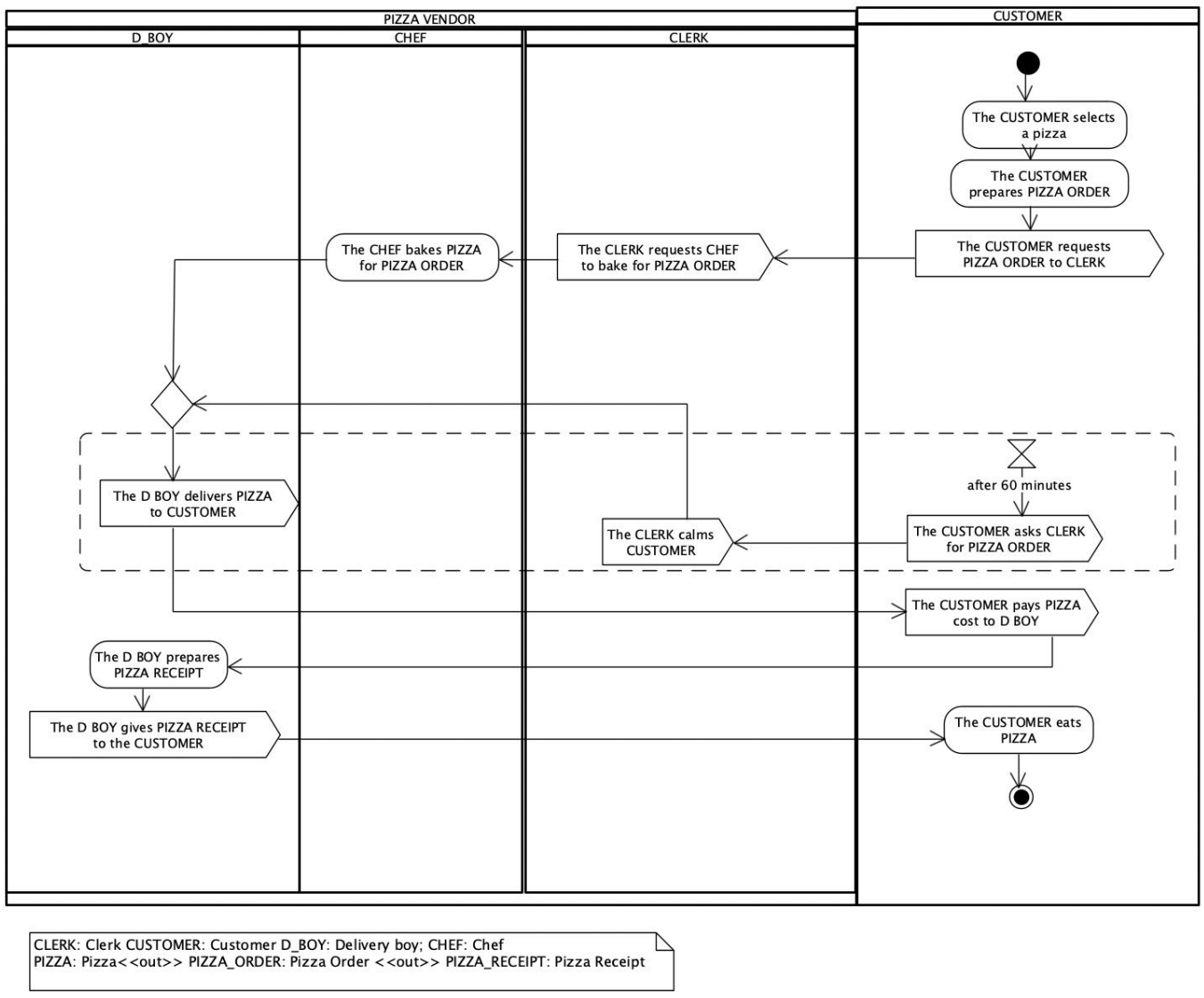


Fig. 58 Pizza Selling Process UM5 Model: Workflow View with Swimlanes

4.2.4 UM2 model Pizza Selling process with two participants

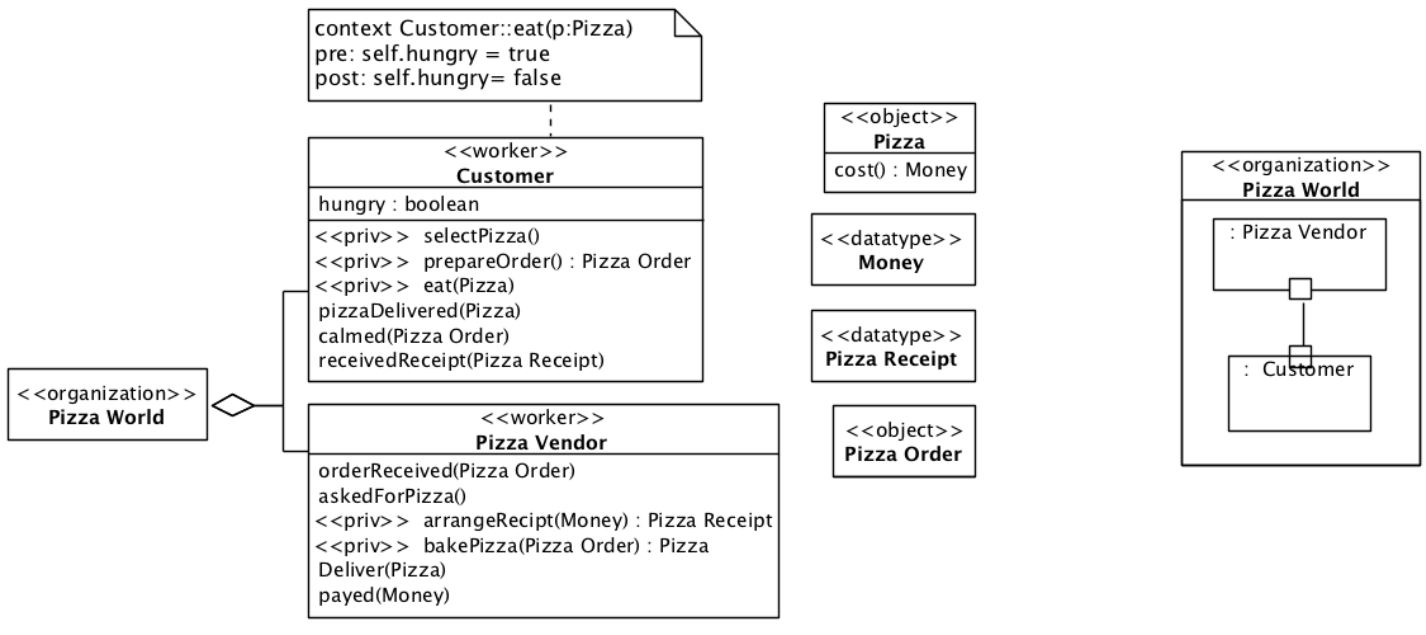


Fig. 59 Two Participants Pizza Selling Process UM2 Model: Entity & Organization View

The entity view of the two participants Pizza Selling model represents two active entities (i.e., Customer and Pizza Vendor). All the operations of the Pizza Vendor subcomponents (i.e., Clerk, Chef, and Delivery Boy) are now handled by Pizza Vendor. The aggregation associations and the organization view represent the fact that in the pizza world there are two entities (i.e., Customer and Pizza Vendor).

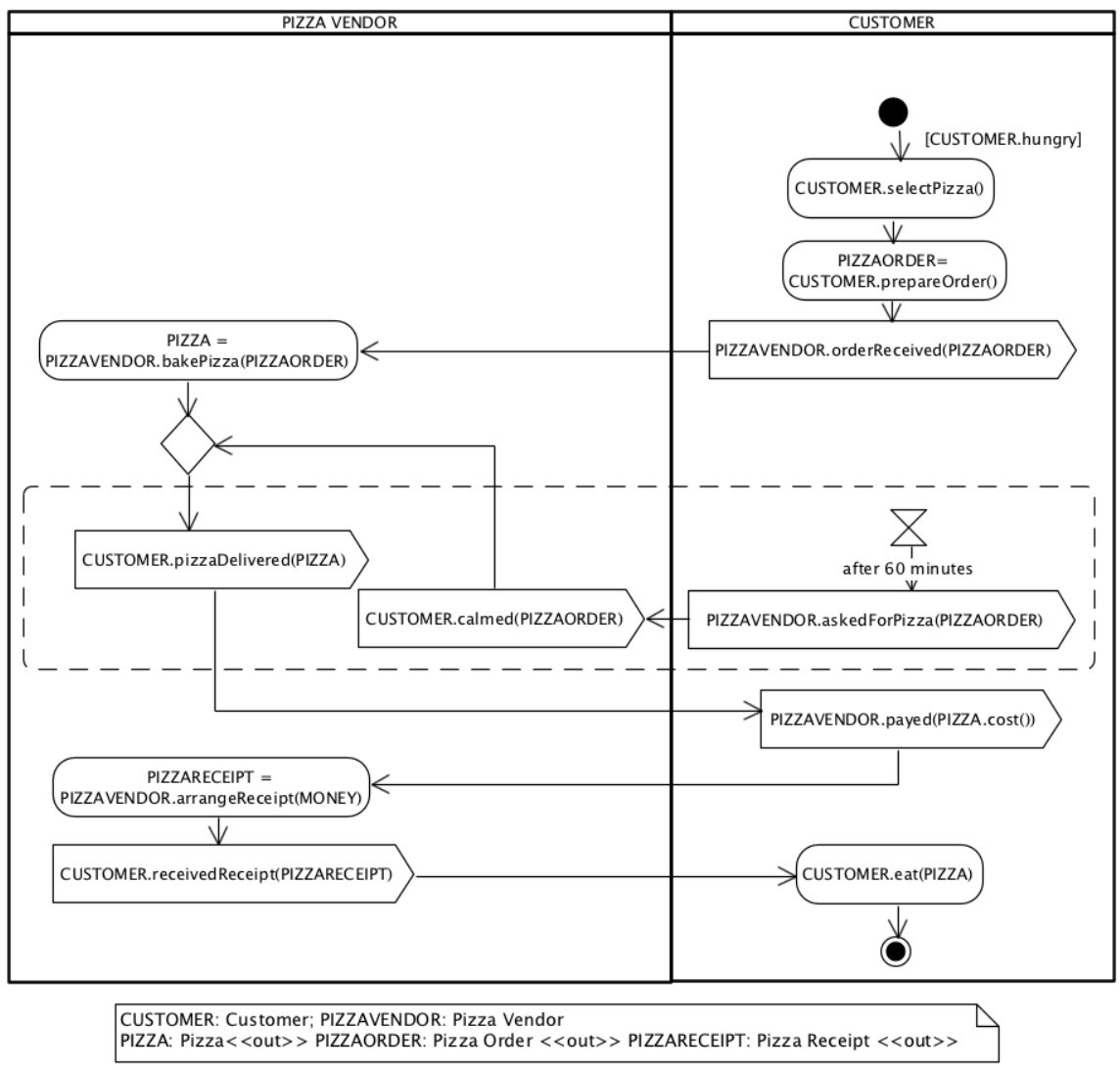


Fig. 60 Two Participants Pizza Vendor Process UM2 Model: Workflow View

The workflow view is slightly different compared to the UM2 model with four participants. All the activities handled by the Pizza vendor subcomponents (i.e., Clerk, Chef, and Delivery Boy) are now handled by a single active participant (i.e., Pizza Vendor). For example, the delivery boy delivers pizza to the customer in the four participants model, whereas in the two participants, pizza vendor delivers pizza to the customer.

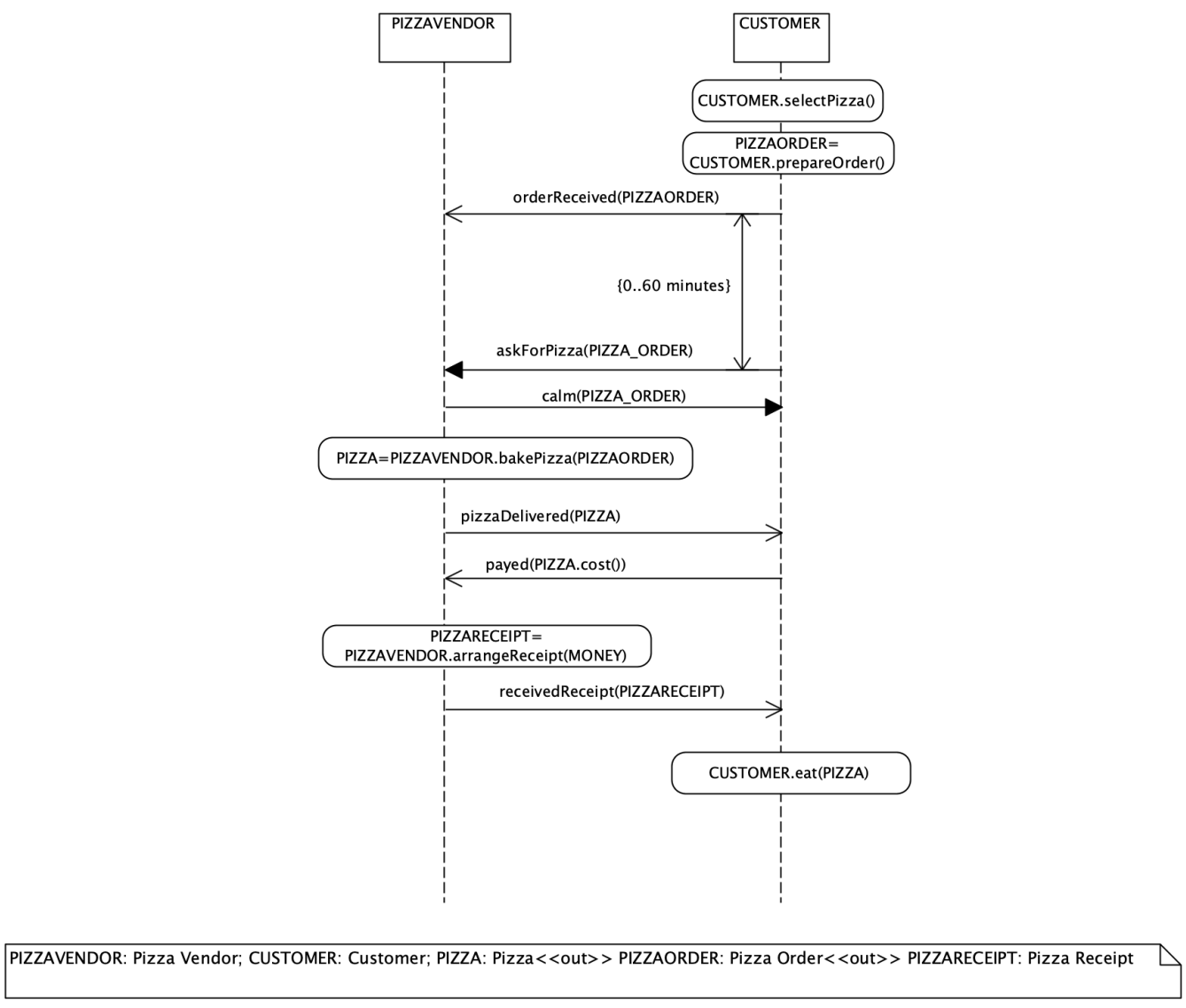


Fig. 61 Two Participants Pizza Selling Process UM2 Model: Interaction View

Notice that the interaction view in Fig. 61 does not represent all process executions modelled by the activity view. This diagram only shows a possible scenario, where the pizza delivered is made on time.

4.2.5 UM5 model Pizza Selling process with two participants

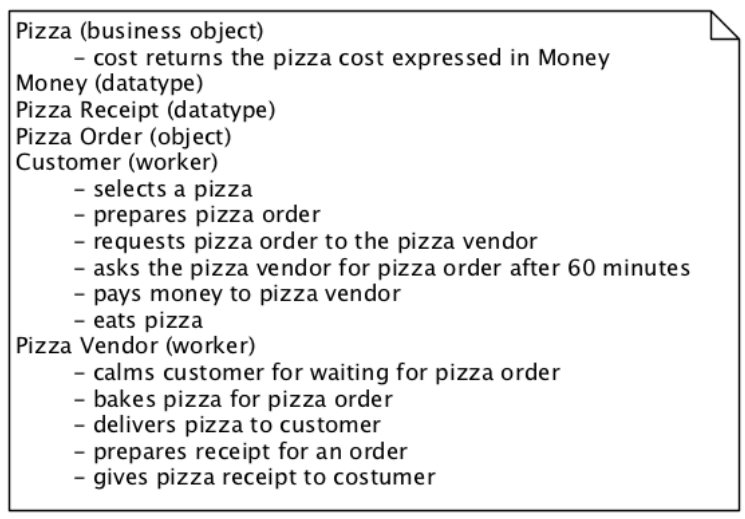


Fig. 62 Two Participants Pizza Selling Process UM5 Model: Entity View

All the operations in the entity view are expressed using natural language fragments.

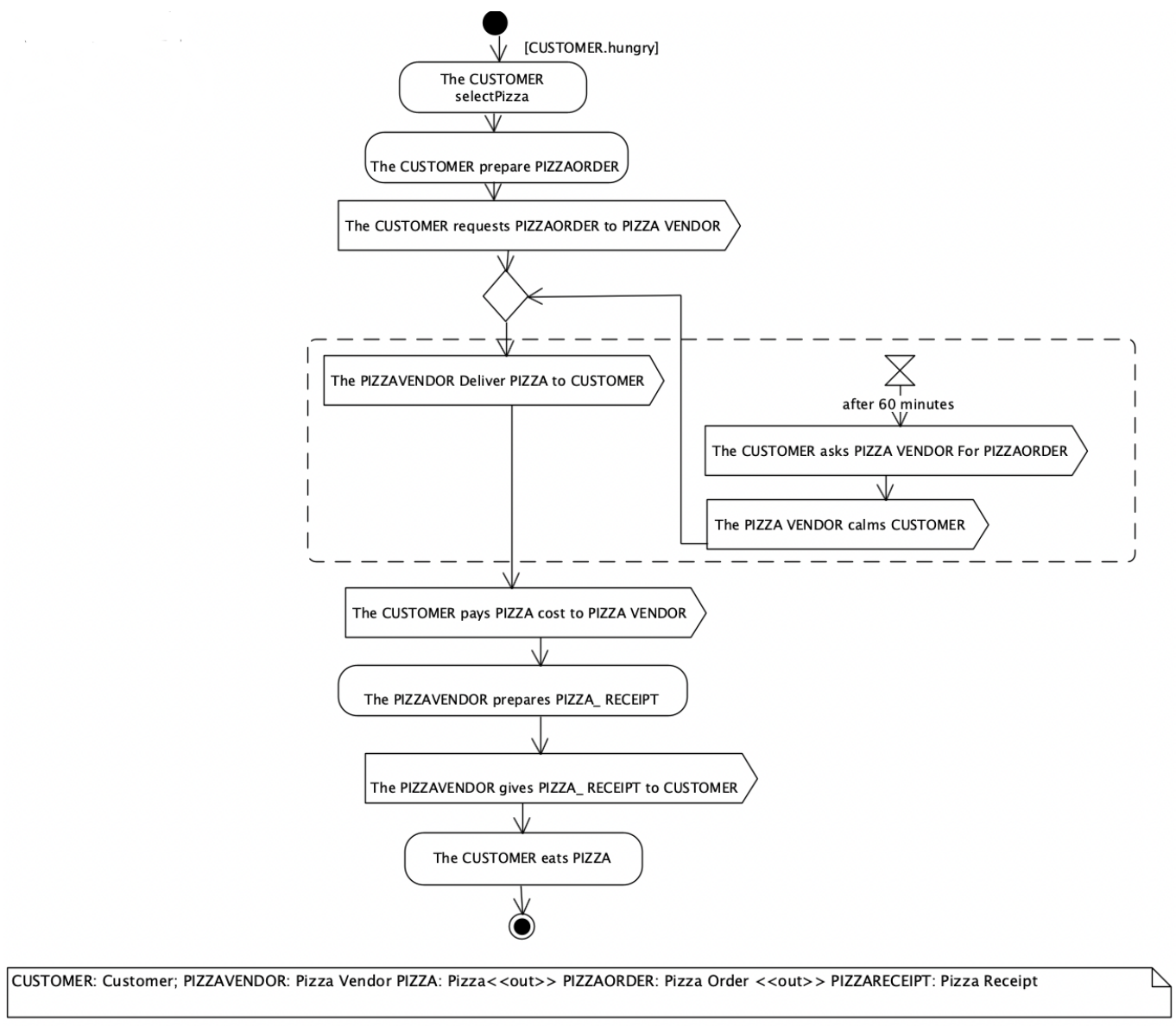


Fig. 63 Two Participants Pizza Selling Process UM5 Model: Workflow View

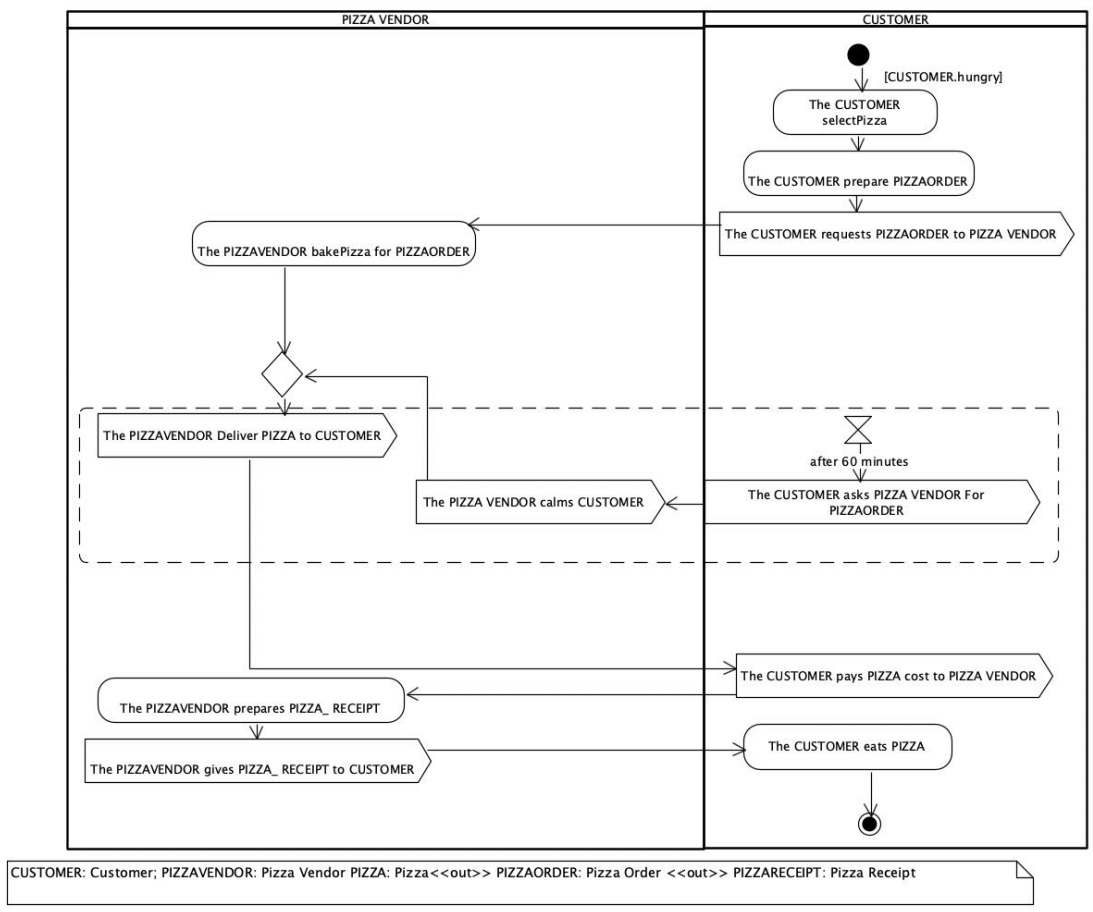


Fig. 64 Two Participants Pizza Selling Process UM5 Model: Workflow View with Swimlanes

4.2.6 Lesson learned

- Also, in this case that the object/data are not explicitly represented in BPMN. This issue was also discovered in the Shipment Process case study (see [sect 4.1.4]). Our models are precise compared to BPMN and we explicitly defined the objects and flowing of the objects between the tasks.
- The UM2 model increases the understandability of the business process in a precise way. For example, the price of the pizza in the BPMN model was just mentioned as a message exchange, but we do not know what the actual price of the pizza is. The operation modelling the payment needs to know the cost of the pizza. Using our models, we have been led to define the cost of the pizza precisely. We added a cost parameter to the pizza class, that made the payment task more detailed.
- To show the BPMN time event, we used a combination of interruptible activity region and accept time event action to show that if a certain action is not performed within a particular time, then something else is performed. UML does not provide constructs for this aspect of the process behaviour
- Notice that the BPMN uses pools to represent process participant, and these Pools are usually further subdivided into Lanes. Lanes are used to organize and categorize activities within a pool. The BPMN diagram (see fig 50) shows the process participants structured in the pool and also using the lanes. This is due to the fact that BPMN wants to show how the active participants are organized in a two-level structuring. Whereas, in our model we have the organization view which allows us to model all the possible structures. Notice also that we model active participants in the entity view by giving them a stereotype (i.e., <<worker>>, <<system>>).

4.3 Incident Management Process

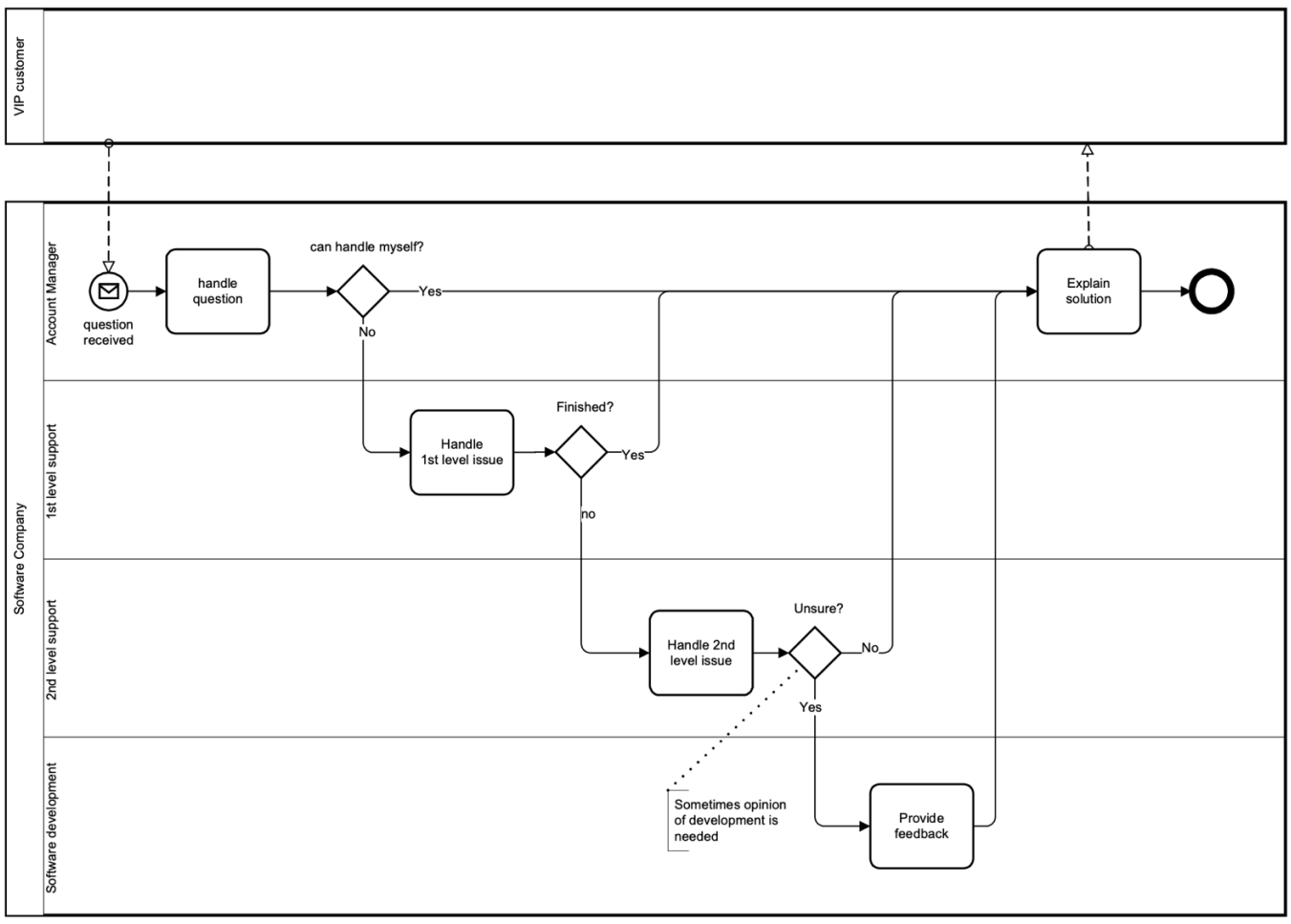


Fig. 65 Incident Management Process: BPMN Model (Collaboration diagram)

4.3.1 The case study

The Incident Management BPMN collaboration diagram model [3, sect 6] reported in Fig. 65 demonstrates how the VIP customer problems are handled and solved by a Software Company.

Notice that the Software Company has five participants. We want to model this process assuming that there are just two participants. This is due to the fact that BPMN uses pools to represent process participants and lanes are used to represent process sub participants.

The BPMN collaboration diagram in Fig 65 illustrates the tasks performed by the Software Company and the VIP Customer. Notice that the VIP Customer pool is empty (this means we cannot see what is inside the pool). The role of the VIP Customer in this example is limited. The other activities corresponding to the Software Company participants are task-oriented and they correspond to abstract tasks in our UML models. Therefore, this process will be modelled using the UM1 (Hard-Abstract tasks) method. Using other models like UM5 and UM2 does not produce an equivalent process to BPMN since they correspond to executable tasks which requires a direct communication between process participants.

4.3.2 UM1 model of the Incident Management Process

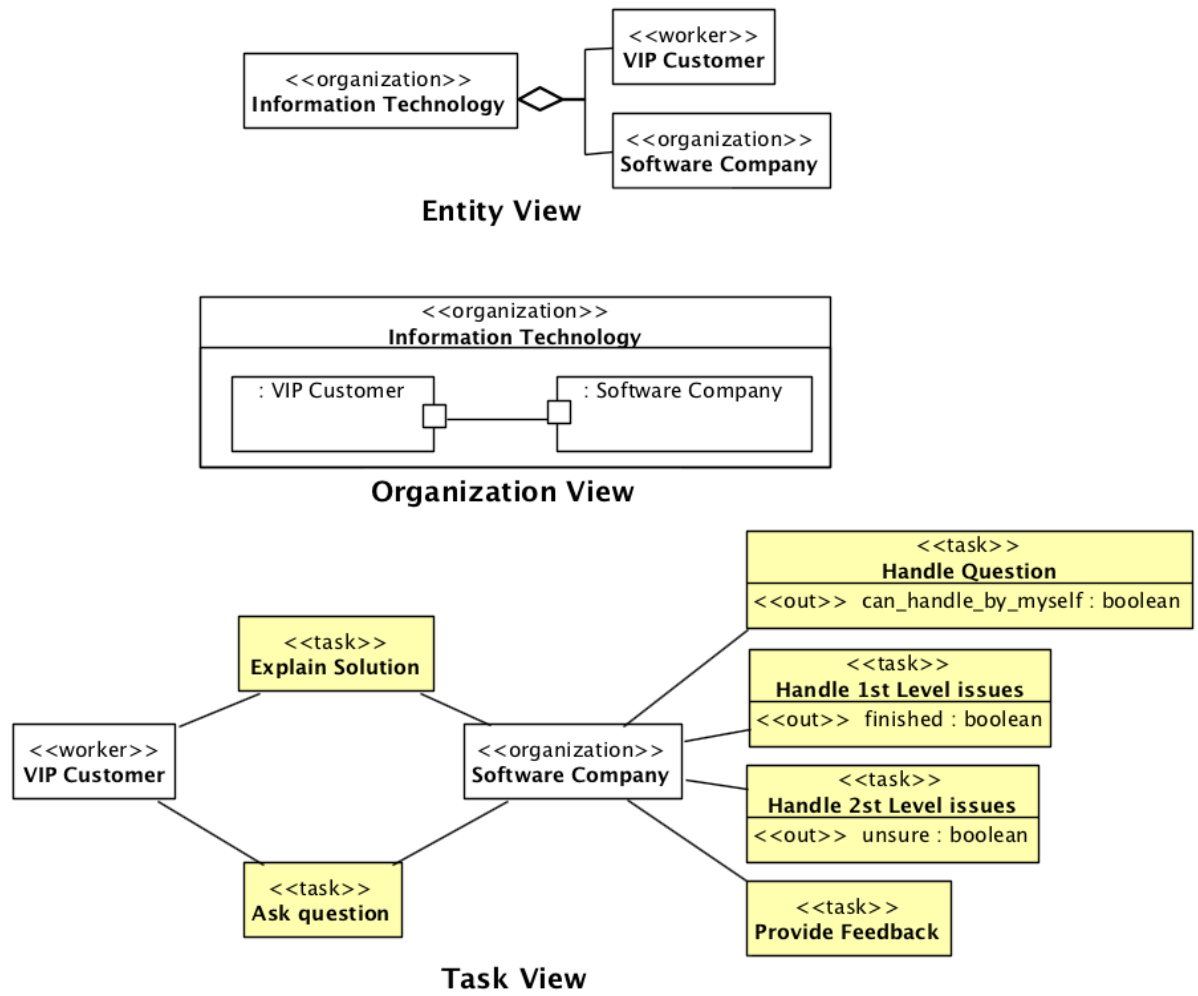


Fig. 66 Incident Management Process UM1 Model: Task, Entity & Organization View

The entities involved in the Incident Management Process are precisely defined in the Entity view. Notice that we have explicitly added an entity called "Information Technology" since the software company belongs to the information technology category modelling the enterprise enacting the process. The organization view in UM1 shows the structure of the Information Technology company describing the composing active entities and organization units.

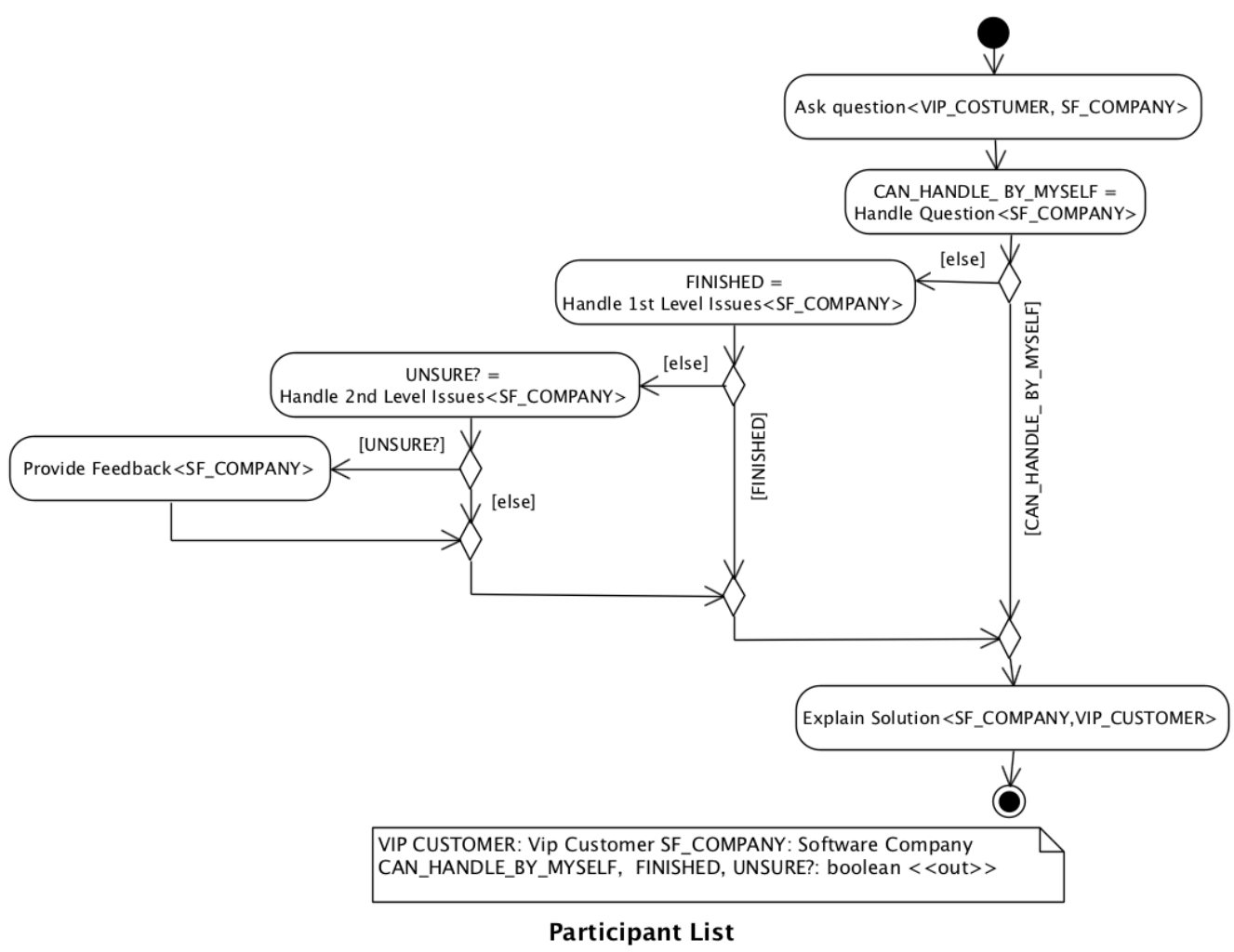


Fig. 67 Incident Management Process UM1 Model: Workflow View

Notice that, all the activities which were performed by Software Company sub-participants are now handled by the single participant Software Company

4.3.3 Lessons Learned

- We have modelled this process with two participants, but we may have also decided to have five participants, since we know that BPMN pools/lanes are used to represent process participants and we also solved this issue in Pizza selling case study.
- In this model we have lost the info about the four lanes, indeed later we have added the lane names into the task names

4.4 Incident Management Detailed

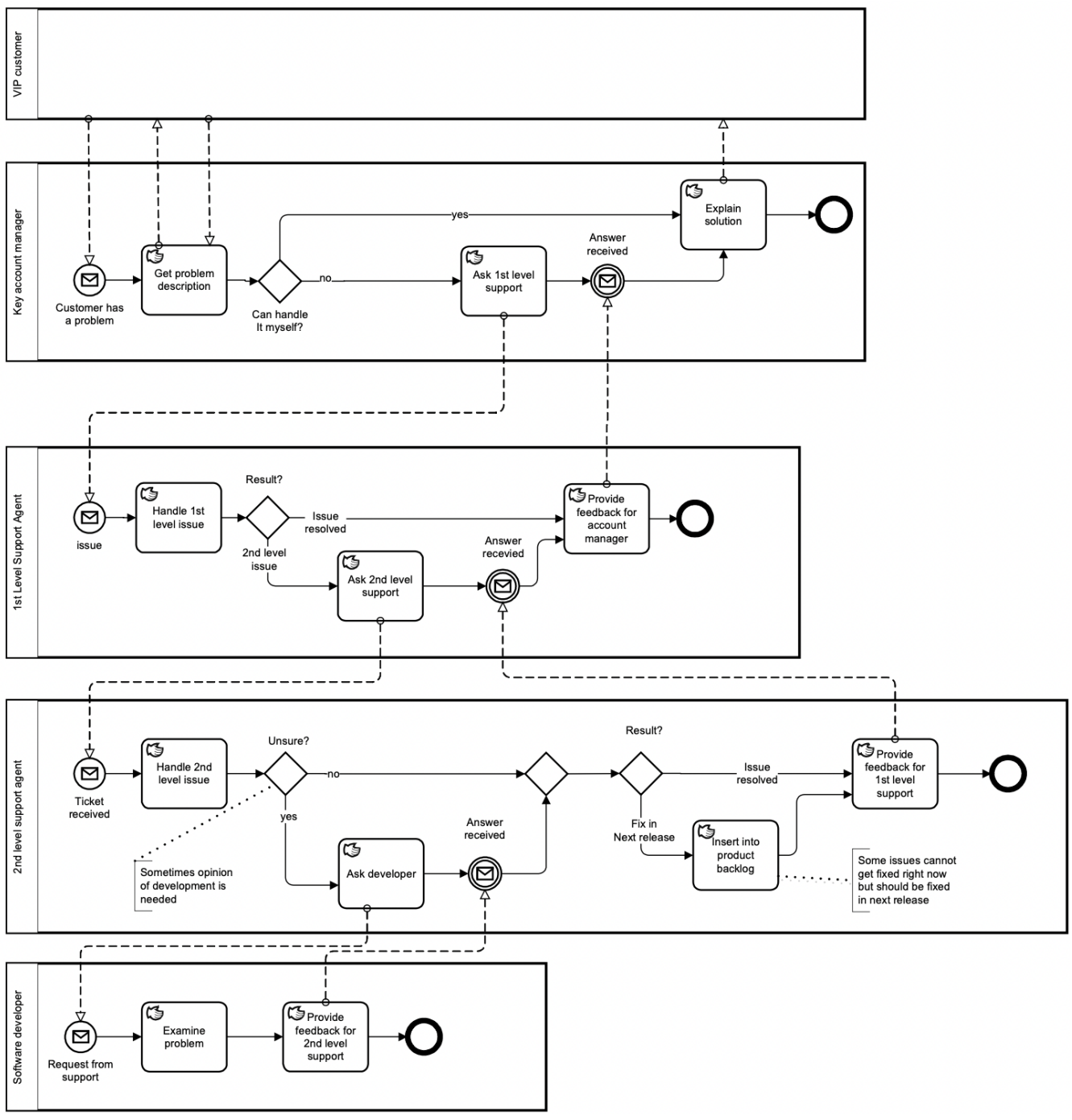


Fig. 68 Incident Management Process Detailed: BPMN Model (collaboration diagram)

4.4.1 The case study

The Incident Management BPMN model reported in Fig. 68 shows a detailed view of the process workflow. The collaboration diagram consists of five pools (i.e., VIP Customer, Key Account Manager, 1st Level Support Agent, 2nd Level Support and Software Developer). The diagram also shows that there exist communications between the various participants.

The Key Account Manager checks if they can handle the problem. In case of affirmative answer, then they resolve the problem and provide the solution directly to the VIP Customer. Otherwise, they forward the issue to the 1st Level Support Agent.

The 1st Level Support Agent checks if they can solve the issue. In case of affirmative answer, then they provide the solution to the Key Account Manager, that later provides the solution to the VIP Customer. If the 1st Level Support Agent cannot solve the issue, then they forward the issue to the 2nd Level Support Agent.

The 2nd Level Support Agent checks if he can solve the issue. In case of affirmative answer, then they provide the solution to the 1st Level Support Agent, otherwise, they forward the issue to the Software Developer. The Software Developer provides the solution to the 2nd Level Support Agent. The solution is finally provided to the VIP Customer.

The BPMN diagram consists of private actions and the communications performed by the participants. All the tasks in the BPMN model correspond to executable tasks in our UML methods. Thus, The BPMN Incident Management collaboration diagram is modelled using the method UM2 (Hard-Executable Tasks-Interaction).

4.4.2 UM2 model Incident Management process

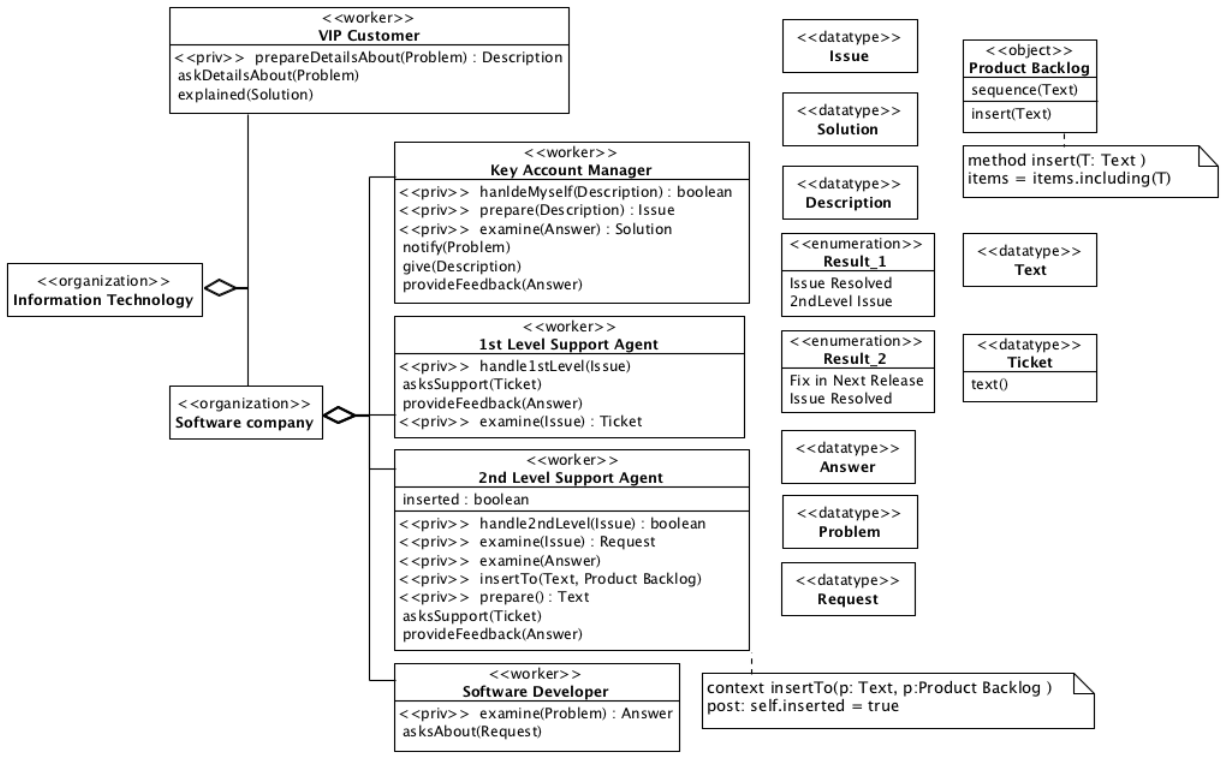


Fig. 69 Incident Management Process UM2 Model: Entity View

The Software Company entity class is stereotyped as organization. It is composed of a key account manager, a 1st level support agent, a 2nd level support agent, and a software developer, and this is modelled by the aggregation relationships and by the organization view in Fig. 70.

Pre/post-conditions are attached to some operations (e.g., insertTo of the 2nd Level Support Agent entity class). A method is attached to the operation insert of the "Product Backlog" entity class.

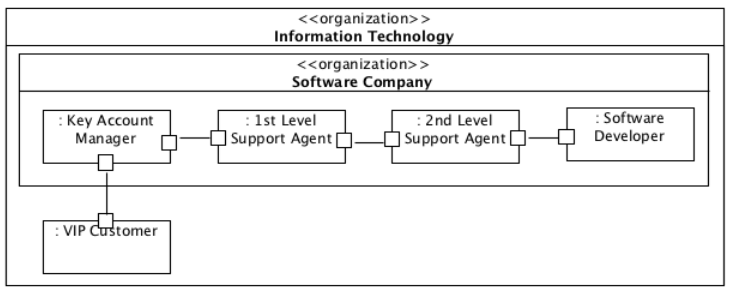


Fig. 70 Incident Management Process UM2 Model: Organization View

The organization view in the UM2 model represents the structure of the enterprise enacting the process. It also shows the possible communications between its components. Here it is clear that there is a chain of communication going from the VIP Customer to the Software Developer

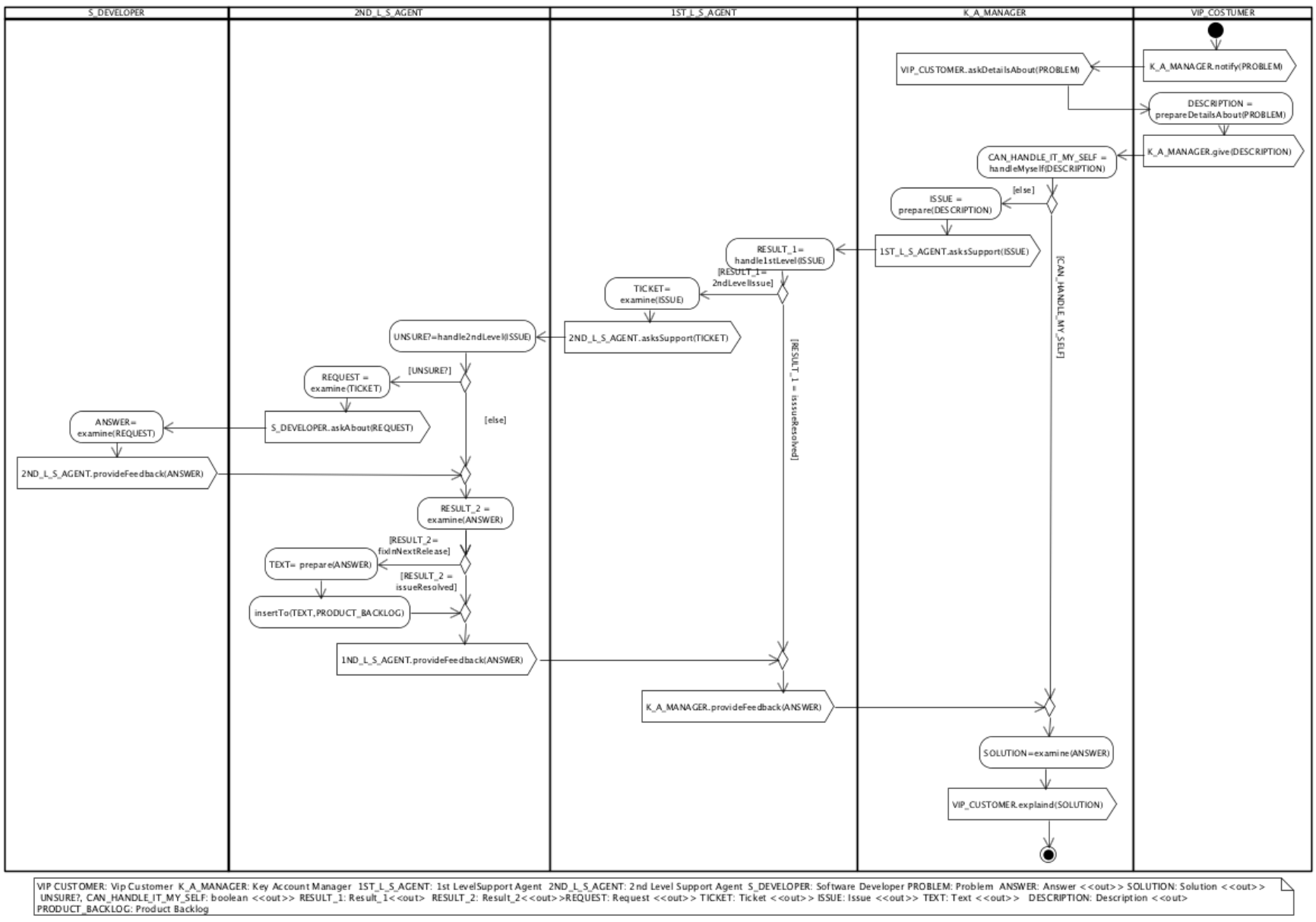


Fig. 71 Incident Management Process UM2 Model: Workflow View

Notice that in the BPMN diagram there exist an End Event at the end of each pool. This could create confusion not knowing where exactly the flow terminates. The workflow in the UM2 solves this issue by producing a single Final Node in the VIP Customer swimlane considering it as correct flow termination.

4.4.3 Lesson learned

- The task "Insert into Product Backlog" in the 2nd Level Support Agent pool does not specify what should be inserted. From the domain knowledge we know that there should be some text or comments about the issue, and it must be inserted in the Product Backlog. We explicitly introduced an object called "Text" assuming text was generated from the ticket, and it is added to the Product backlog considering the real-world case. Notice also that we have modelled the database (i.e., the Product Backlog) as an object in our model.

The errors in the BPMN model identified during the UML modelling are:

The names of the objects used in the BPMN diagram is confusing. For example, the Problem and the Issue seem to be synonyms. Also notice that, the names of the gateways in 2nd Level Support Agent and 1st Level Support Agent are similar (named both as Result?). The UM2 method solves this issue by modelling the process precisely (i.e., using different names)

4.5 Incident Management as Choreography

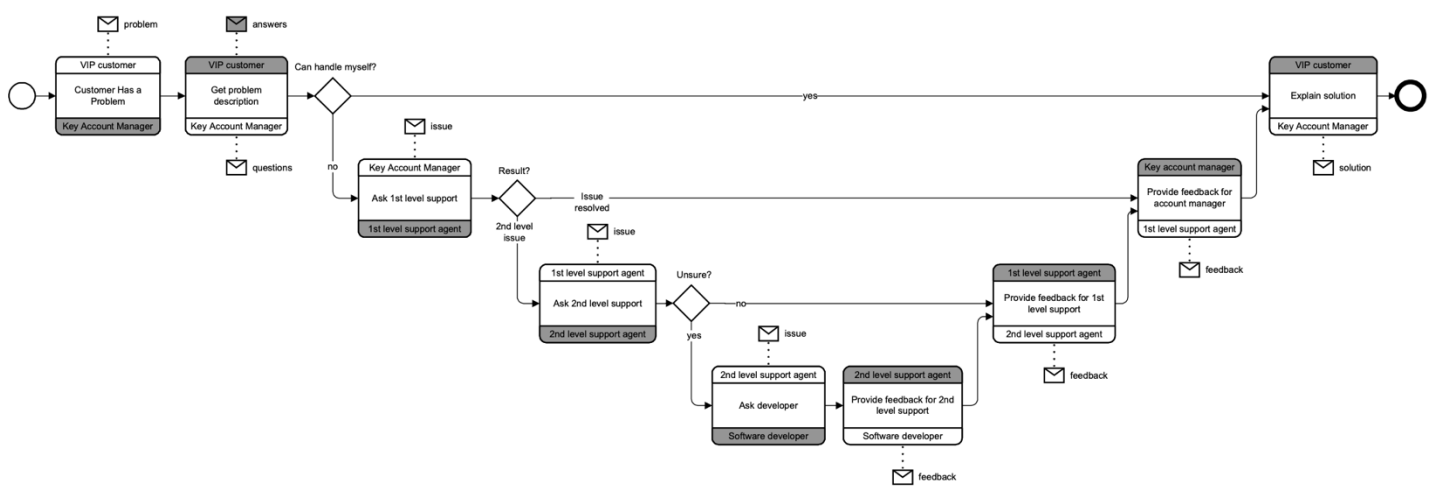


Fig. 72 Incident Management Process: BPMN model (Choreography diagram)

4.5.1 The case study

Fig. 72 shows the Incident Management Process Choreography diagram. A choreography diagram is an extended diagram type of a BPMN collaboration diagram focusing on the message and the sequence of the messages between the participants. A collaboration contains two or more pools, representing the participants in the collaboration and the message exchange between the participants is shown by a Message Flow that connects two pools (or the objects within the Pools [1, sect 9.1]). Whereas A choreography is a different kind of diagram. Its purpose is to show the interaction between participants in a different format and focusing only on message flow instead of the individual detailed tasks of a process. Each step in the choreography involves two or more participants and it is defined outside of a pool. The choreography activity is depicted as a rectangle the two or more bands (i.e., one at the top and one at the bottom) represent the parties involved in the interaction captured. A white band is used for the initiating party and the dark band is used for the recipient. The envelope represents a message sent by the corresponding party; return message envelopes of a two-way interaction are darkened to establish an order dependency between two interactions.

The BPMN choreography diagram reported in Fig. 72 does not represents exactly the same process modelled in Fig. 68 (the differences are relative to the data and object exchanged between the participants), so now we must decide if we model the process in Fig. 68 or the one in Fig. 72.

If we choose to model accordingly to the Fig. 72, then we have to consider the fact that some messages are exchanged inside the BPMN choreography tasks, and they are rendered in the behaviour specification of the task classes modelled by sequence diagrams. Notice that, using our methods since the data and objects are modelled in the entity view, when we change views or methods data/objects are not changed (at most their models are adapted to the specific method/view). For example, adding some operation or attribute, but we cannot move from executable tasks to abstract tasks.

We have devised two ways for modelling the BPMN Choreography diagram in our UML methods.

- Way 1: We produce a model similar to Fig. 72 where each choreography task is modelled by an abstract task, and we add a behaviour specification for the various tasks. Such behaviour specifications will be modelled using some sequence diagrams showing that some message exchange will take place between the task participants. In this way, we have the same informative content as in the BPMN choreography model. The fact that a BPMN choreography task has attached some messages, means that some messages will be exchanged and there is the indication of who is the first initiator of the communication. This leads us to add some operations to the various worker classes.
- Way 2: we model the fact that there is a flowing of info between the various tasks. For example, in the task (prepare DESCRIPTION), the object DESCRIPTION has to be used in a successive task. This model will be similar to the one in fig. 66 Incident Management Process UM1 Model, but here there are different names for the info/data managed by the various tasks. This show that the entity view should be the same or similar in any method chosen among UM1to UM5.

The BPMN choreography tasks are rendered by abstract tasks. The BPMN may have various kinds of tasks that may use to represent both abstract and executable tasks, but the new choreography tasks are perfect examples of abstract tasks (see [2, sect 2.1]). Thus, we have decided to model the Incident Management Choreography adopting both the methods (i.e., Way 1 and Way 2) using the UM1 method since the tasks are rendered by abstract tasks.

4.5.2 UM1 model Incident Management Process (Way 1)

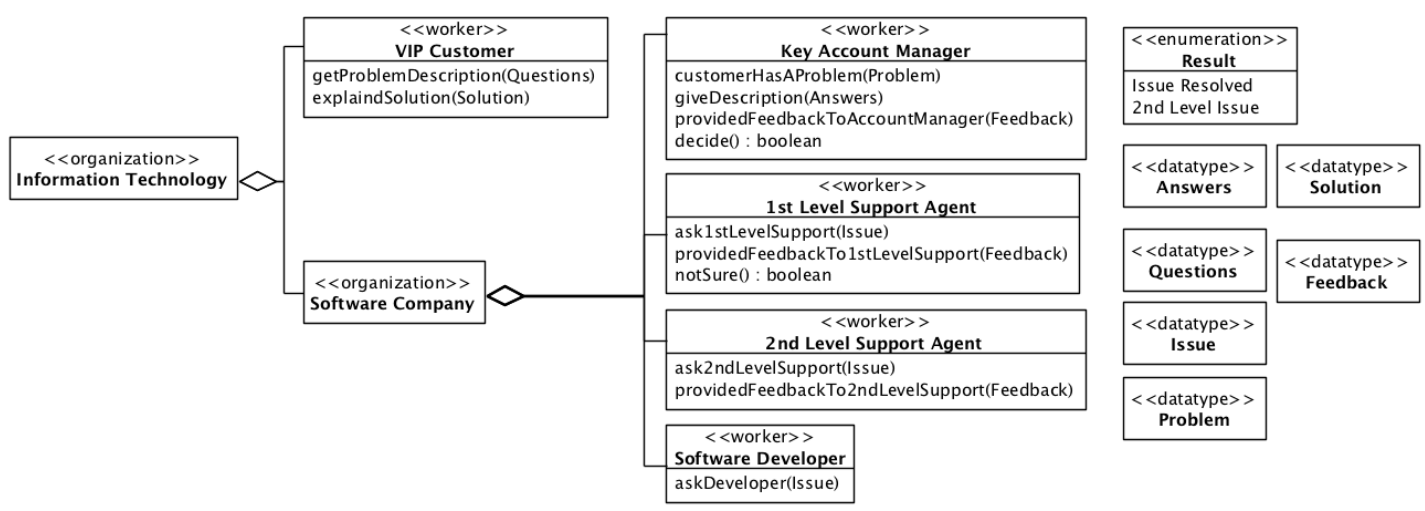


Fig. 73 Incident Management choreography UM1 Model Way 1: Entity View

Notice that the worker classes in the entity view have some operations. These operations are used to model the communications between the various participants.

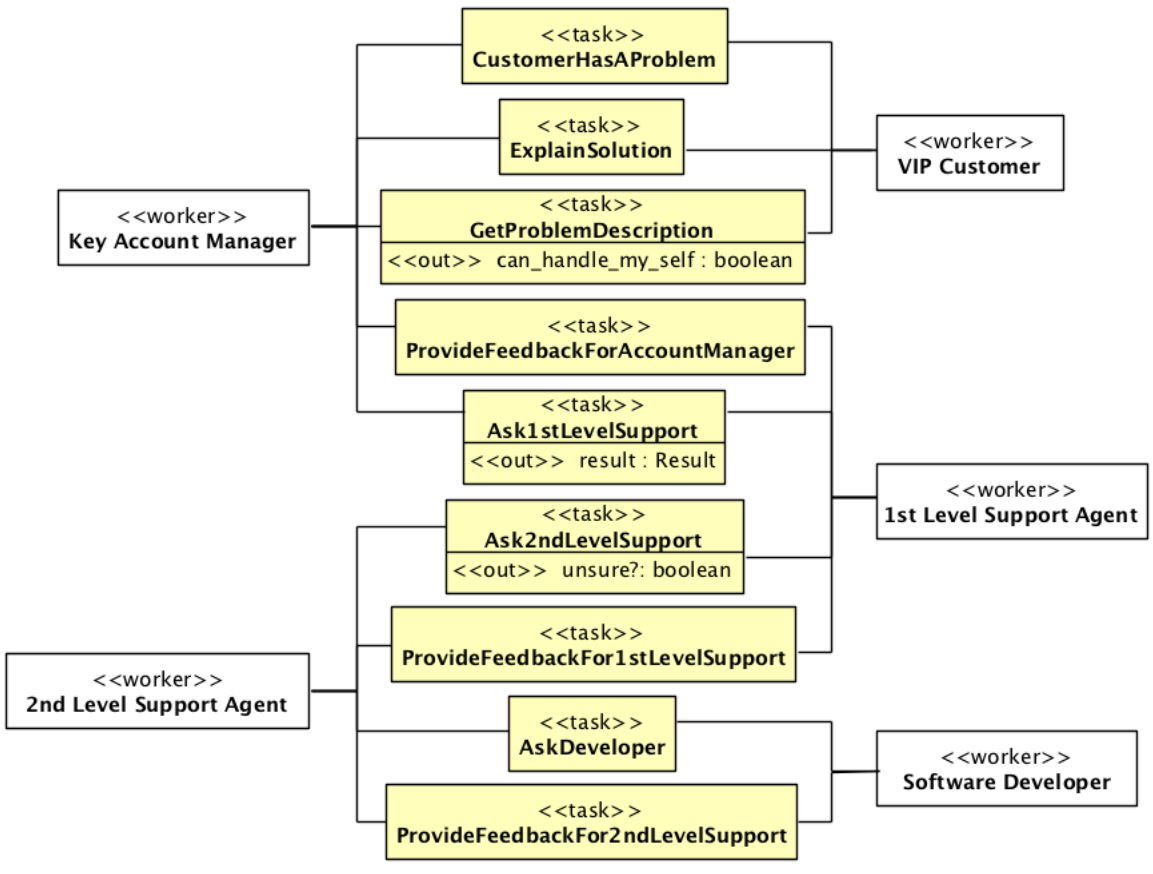
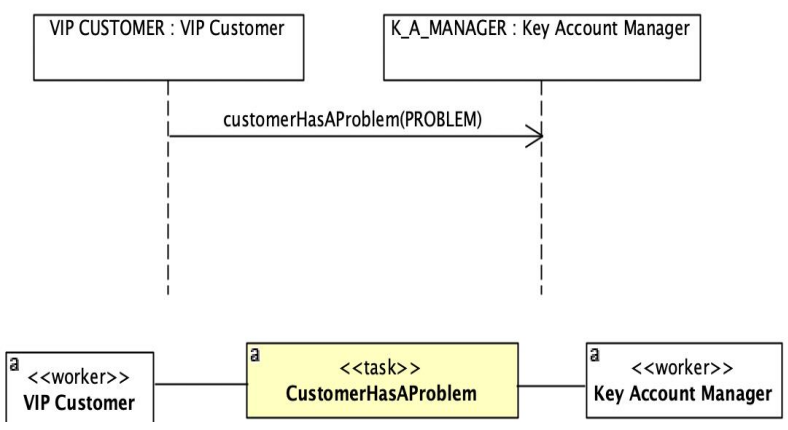
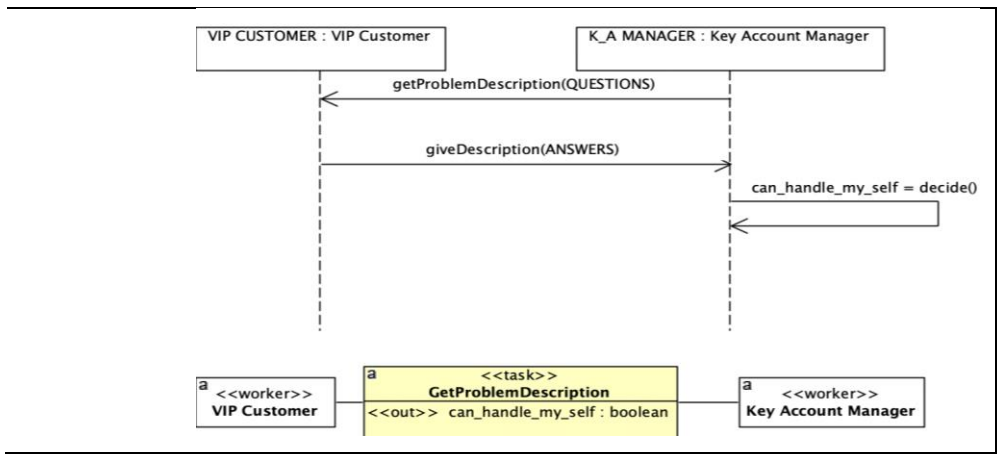
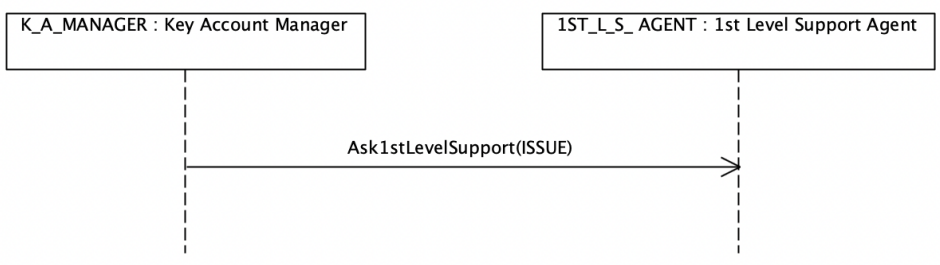
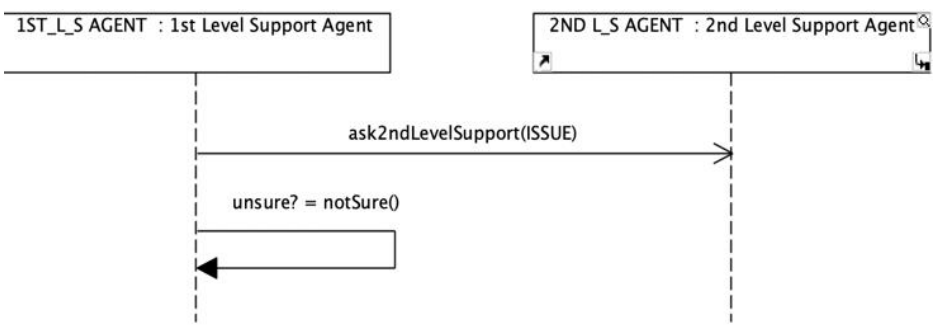


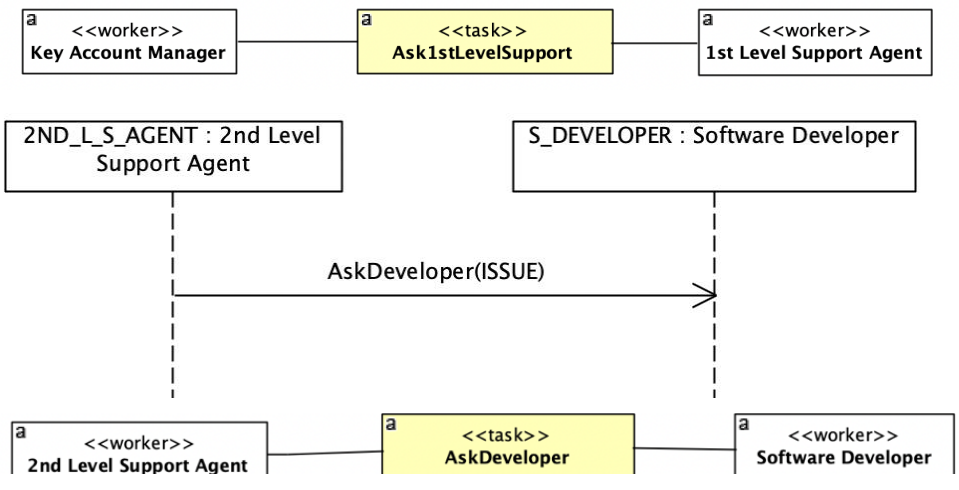
Fig. 74 Incident Management choreography UM1 Model Way 1: Task View

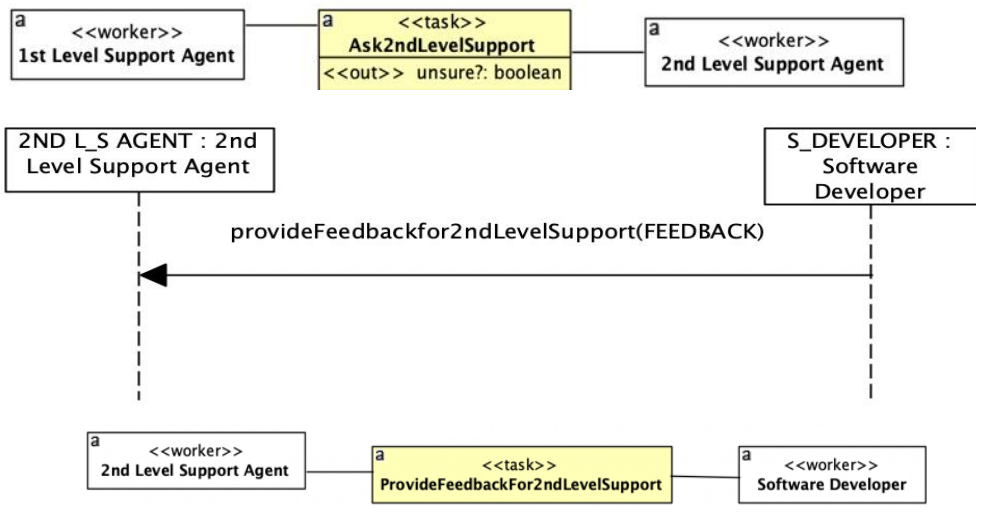












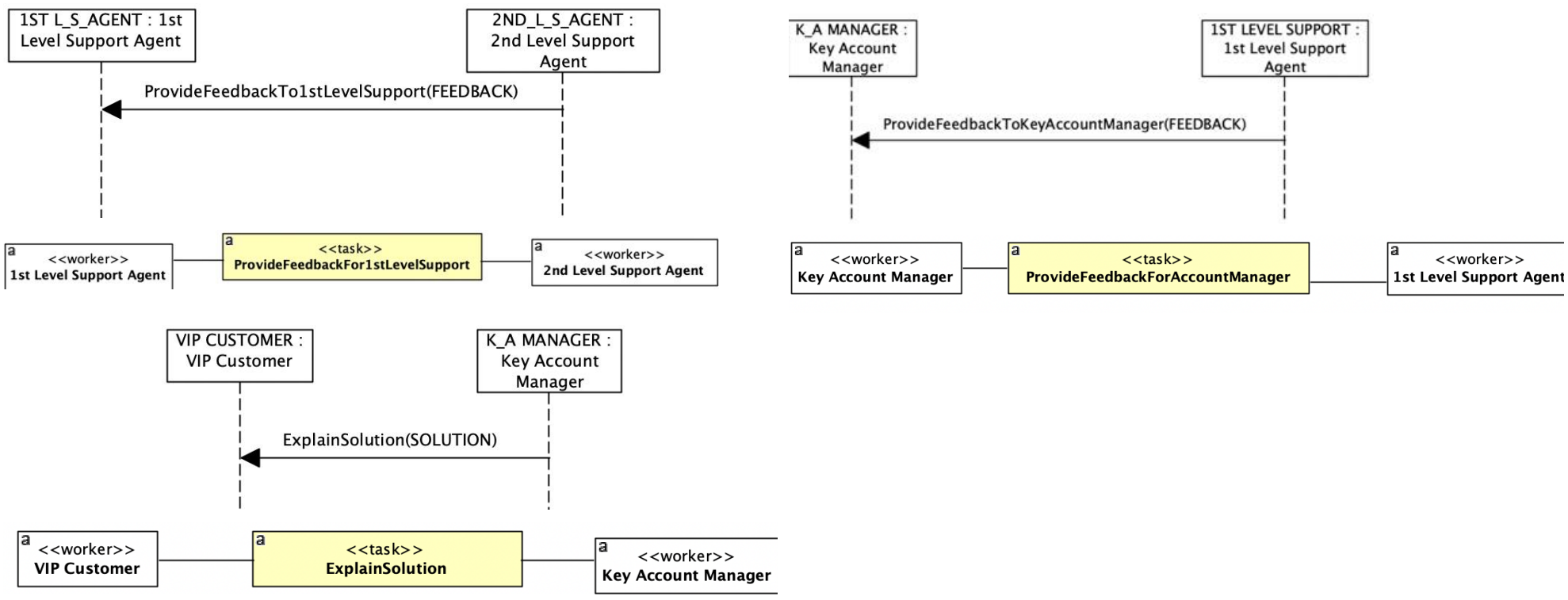


Fig. 75 Behaviour Specification of various tasks

Notice that the abstract tasks in fig.75 are different compared to fig.66. The BPMN invent tasks generalizing the action performed, and the names of the tasks are in natural language. Whereas in our methods we have to define the task in task view.

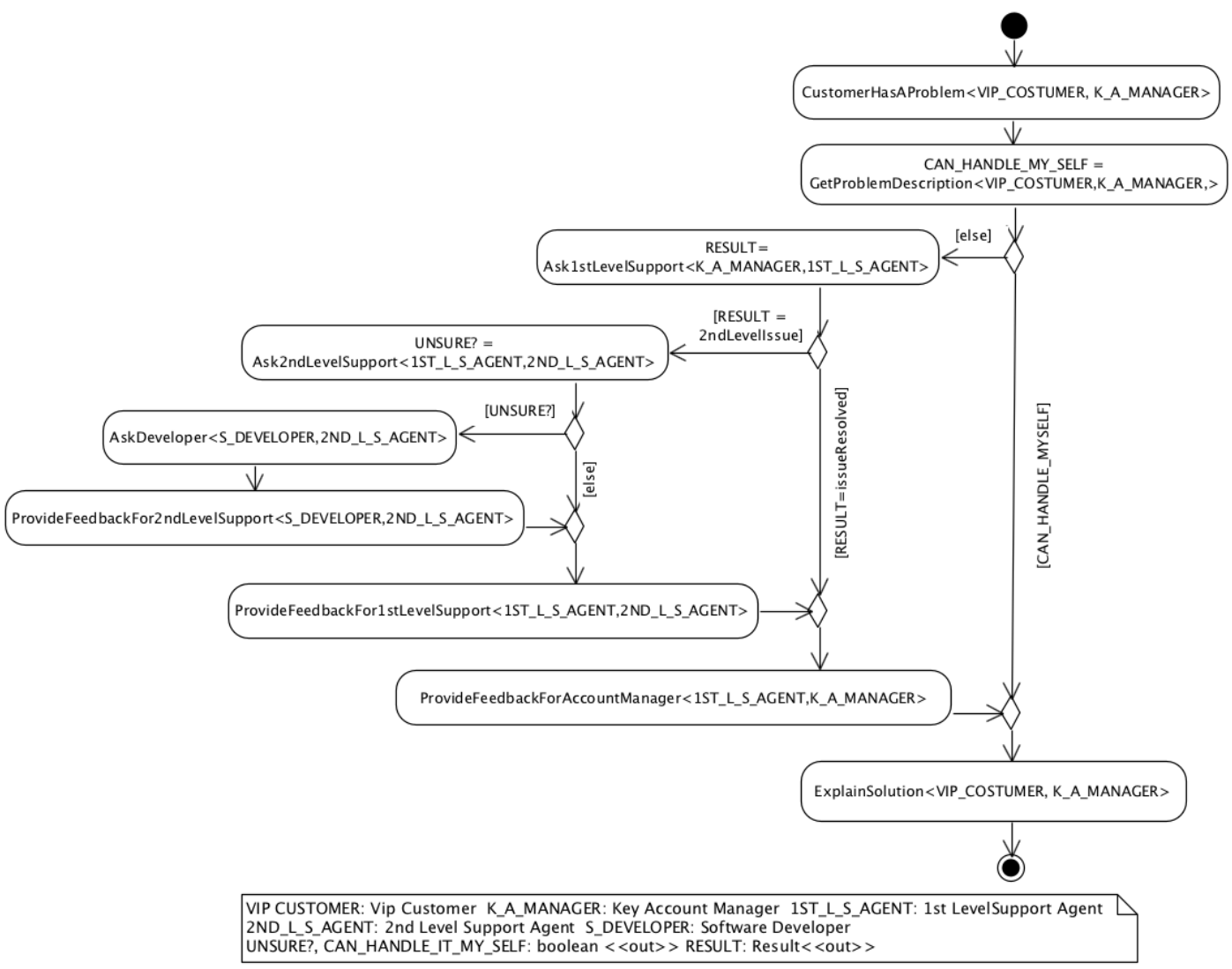


Fig. 76 Incident Management choreography UM1 Model Way 1: Workflow View

4.5.3 UM1 model Incident Management Process (Way 2)

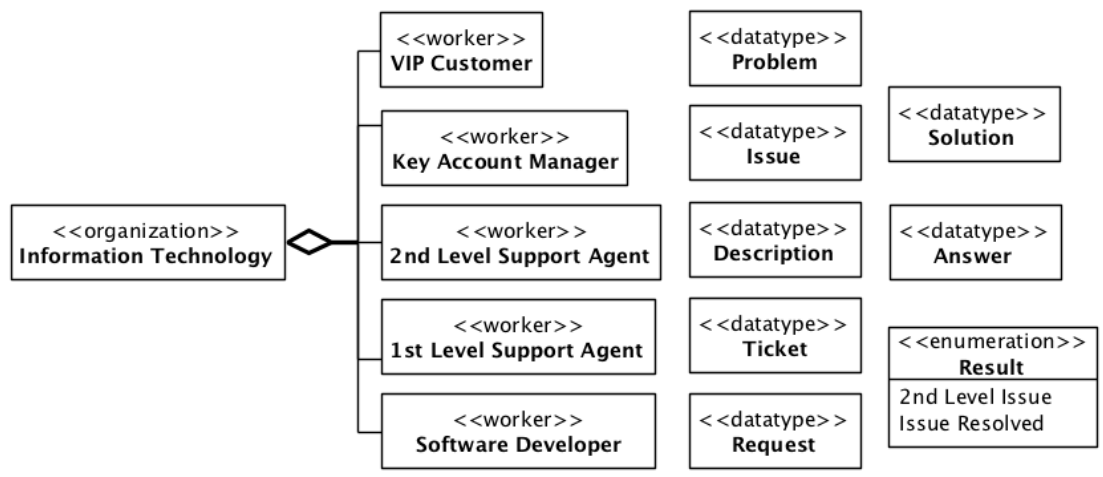


Fig. 77 Incident Management choreography UM1 Model Way 2: Entity View

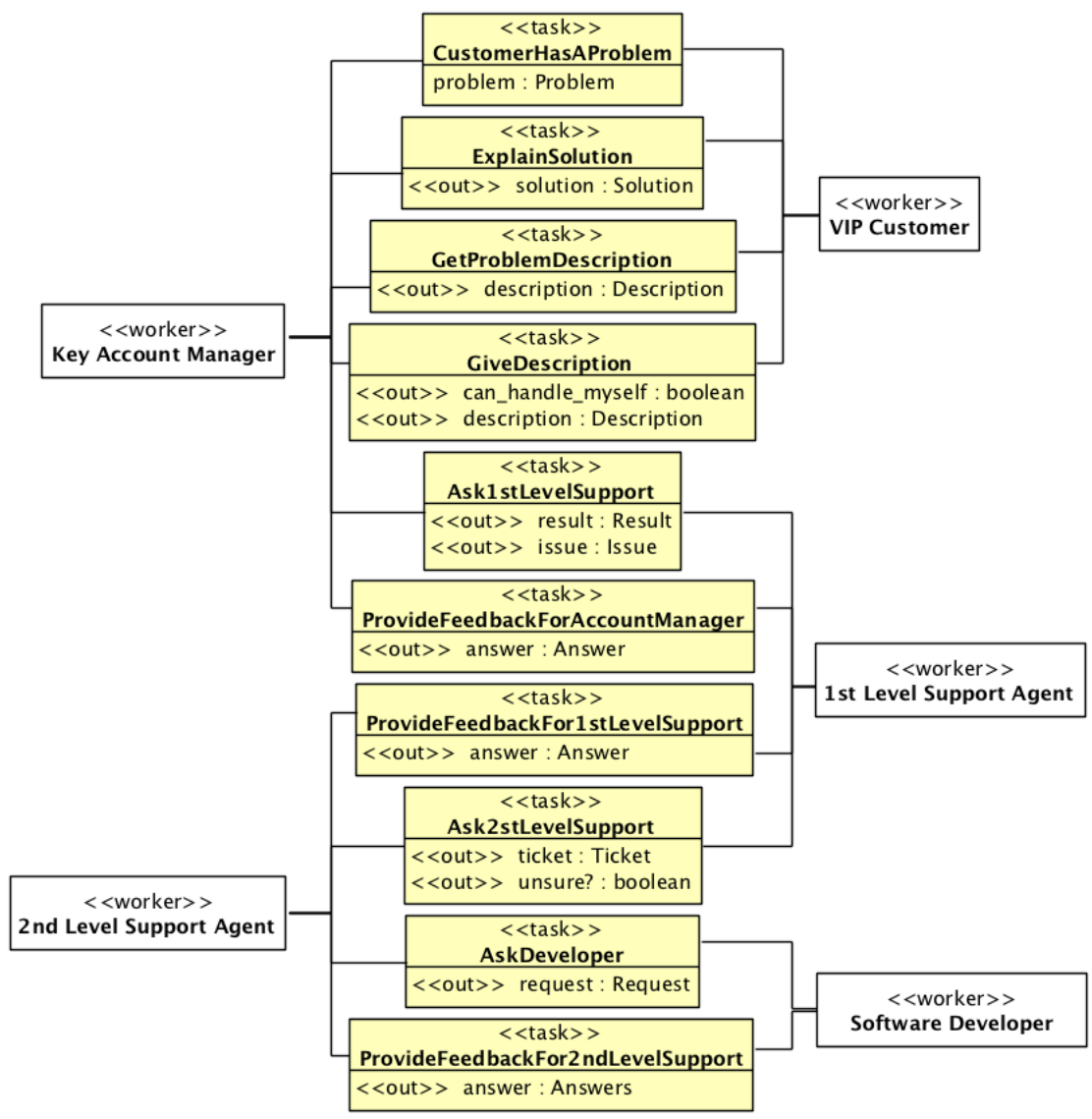


Fig. 78 Incident Management choreography UM1 Model Way 2: Task View

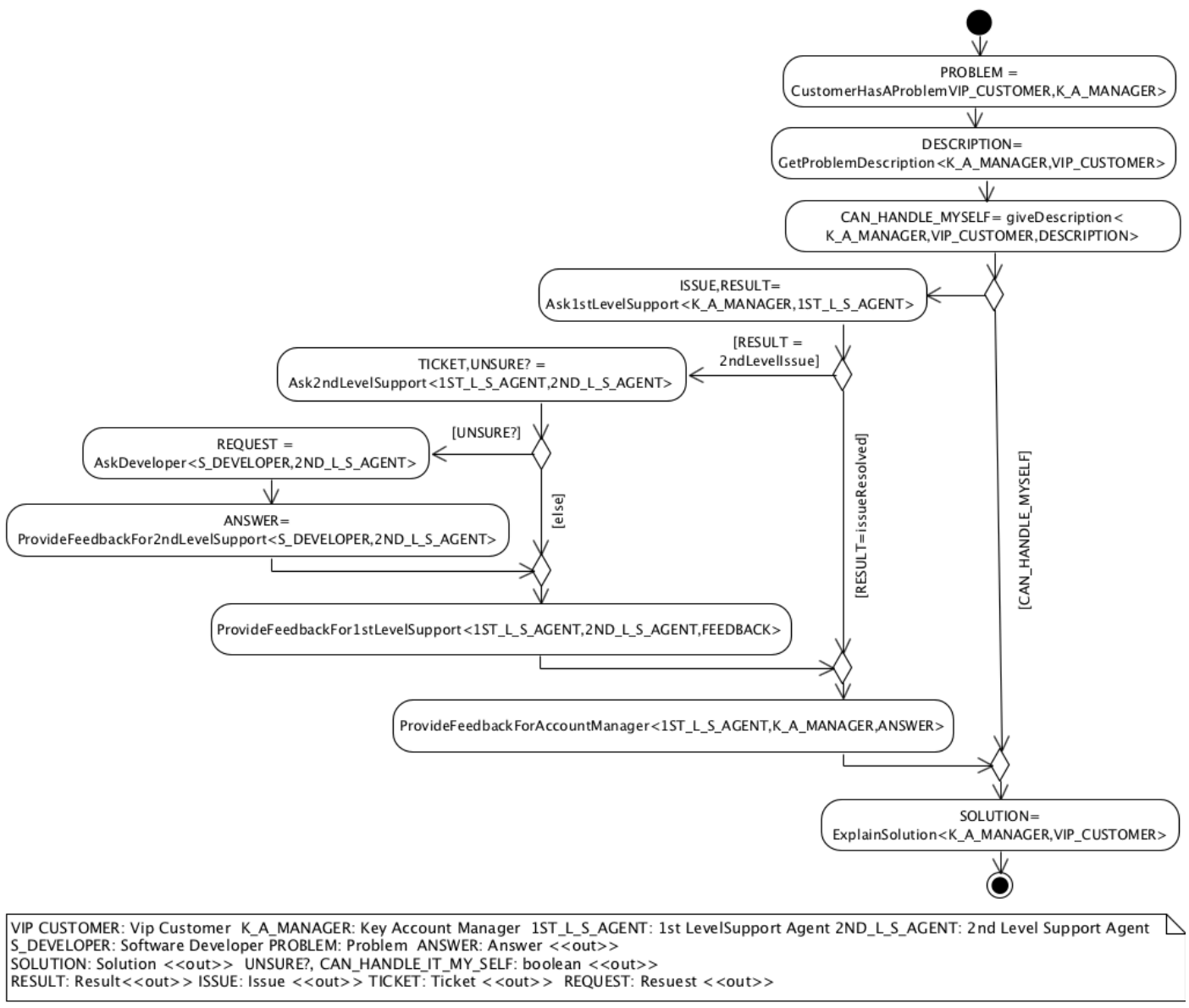


Fig. 79 Incident Management choreography UM1 Model Way 2: Workflow View

The abstract tasks are now similar to those of the BPMN model reported in Fig. 66. We assume that in BPMN repeating the same names means using the same entity that appear in several choreography tasks. For example, the object Feedback is used in the task "ProvideFeedbackToDeveloper" and also in successive tasks. The workflow is precisely defined, and it shows the flowing of data/objects between the various activities.

4.5.4 Lesson learned

- Notice that, in the UM1 model (Way 1) and in the BPMN Choreography model there is no indication of how the data flows among the various tasks, except the use of the same name in the BPMN model and of the same type in the UML model. A correct model of this process is following the idea to represent only the block activities and the interaction exchanged inside such activities should be rendered by adding several parameters to the task classes and correctly flowing among them.

The errors in the BPMN model identified during the UML modelling are:

- The answers and questions in the task (Get Problem Description) reported in Fig. 72 are a mistake, except that it is intended that one send a list of questions and receive a list of answers. Instead, the idea is that there is a dialogue made by a repeated exchange of a question and an answer, it could be modelled in our UML models but not in the BPMN

4.6 The Nobel Prize Process

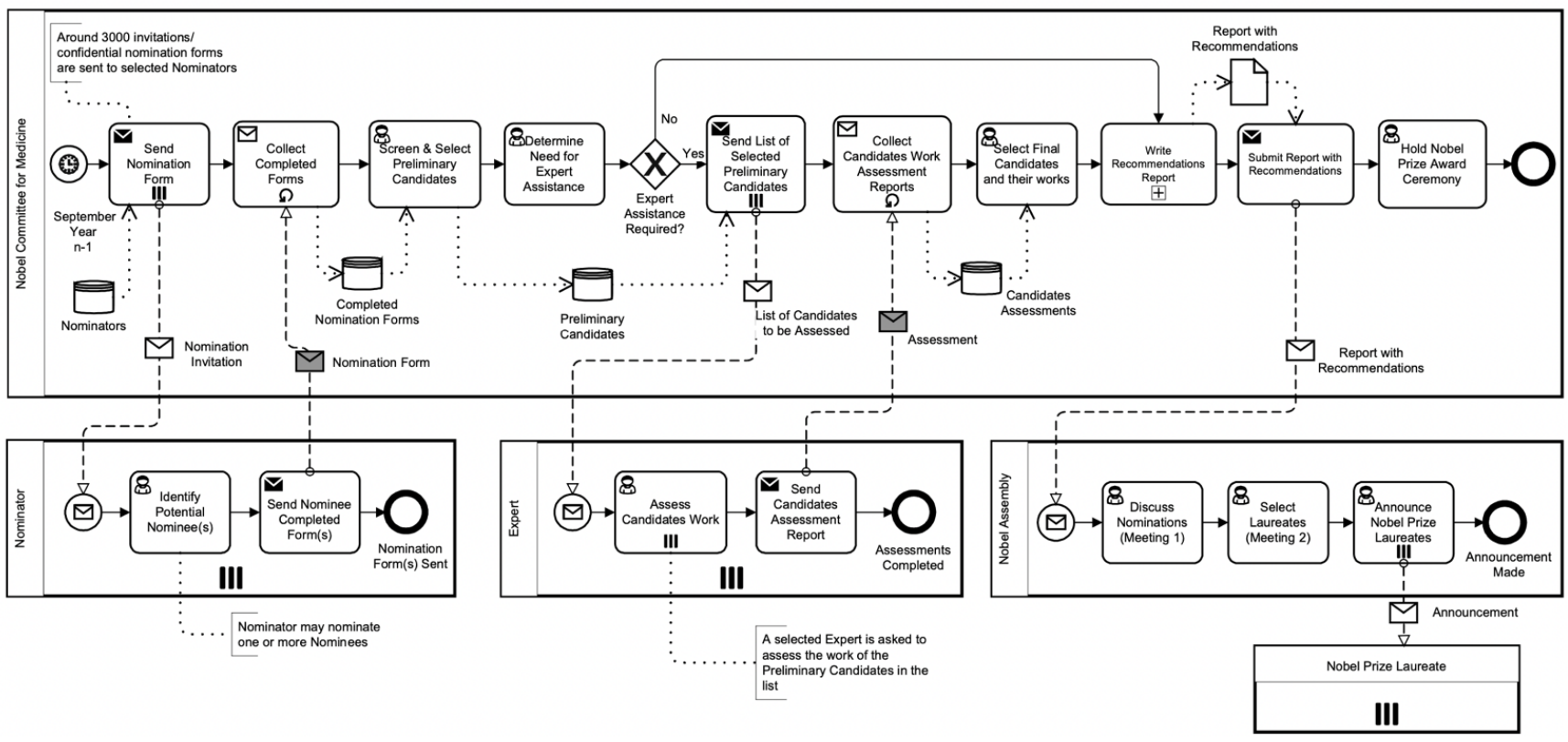


Fig. 80 Nobel Prize: BPMN Model (collaboration diagram)

4.6.1 The case Study

The Nobel Price BPMN collaboration diagram model [3, sect 8] reported in Fig. 80 demonstrates how the Nobel Prize in the field of medicine is awarded.

The process starts in September of the preceding year. The Nobel Committee for Medicine sends around 3000 confidential invitations/nomination forms to selected Nominators. These Nominators fill out the Nomination form selecting one or more nominees. The completed forms are sent back to the Nobel Committee for Medicine.

The Nobel Committee for Medicine stores the completed forms in a database, to perform screening and selection of preliminary candidates. After selecting the preliminary candidates, the Nobel Committee for Medicine may determine if they need any assistance from the Experts. If so, they will then send a list of preliminary candidates to experts.

The experts assess the candidate's work and send the candidate's assessment report to the Nobel Committee for Medicine. From this report, the Nobel Committee for Medicine selects the final candidates based on their work and writes a recommendation report. Finally, the Nobel Committee for Medicine submits the recommendation report to the Nobel Assembly and prepares Nobel Prize ceremony.

The Nobel Assembly makes a discussion and selects laureates. Finally, the Nobel Assembly announces Nobel Prize Laureates.

The Nobel Price BPMN collaboration diagram consists of private actions and communications between the participants. Notice that there is a flowing of objects in the form of message exchange between process participants, and the data flows using datastore. When compared to the other our methods UM2 is the best suitable since we can model interactions and private actions between active participants precisely.

4.6.2 UM2 model Nobel Prize Process

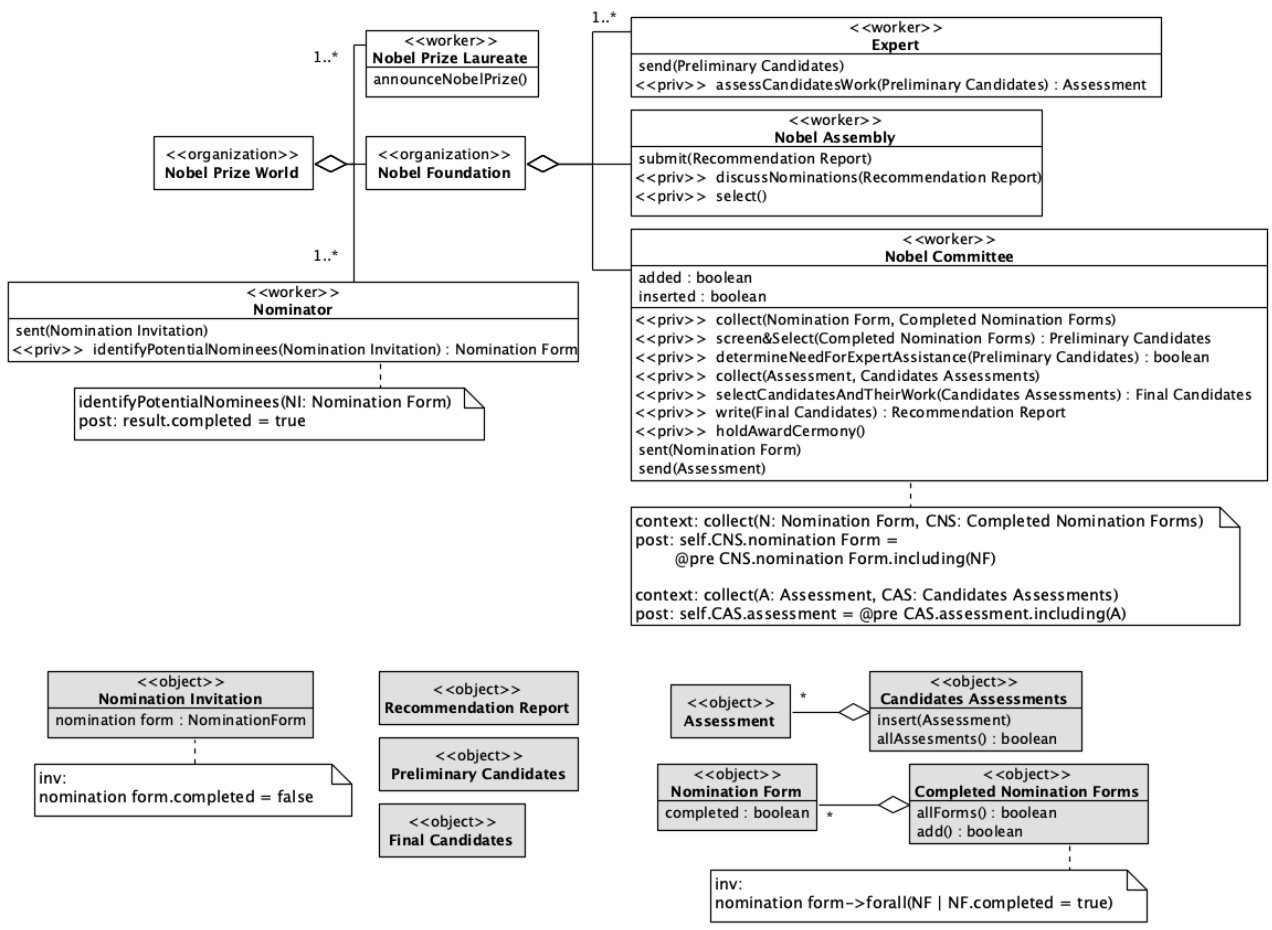


Fig. 81 Nobel Prize UM2 Model: Entity View

The Fig. 81 Represents the entity view of Nobel Prize process UM2 model. Notice that the BPMN model uses the datastore construct to represent a database used in the process, whereas in our model by default we use (business) objects to represent digital/ physical things in the process. Notice also that the multiplicity is 1..* for the associations linking some worker classes (i.e., Experts, Nominator and Nobel Prize Laureate) in order to represent that they are multiple members.

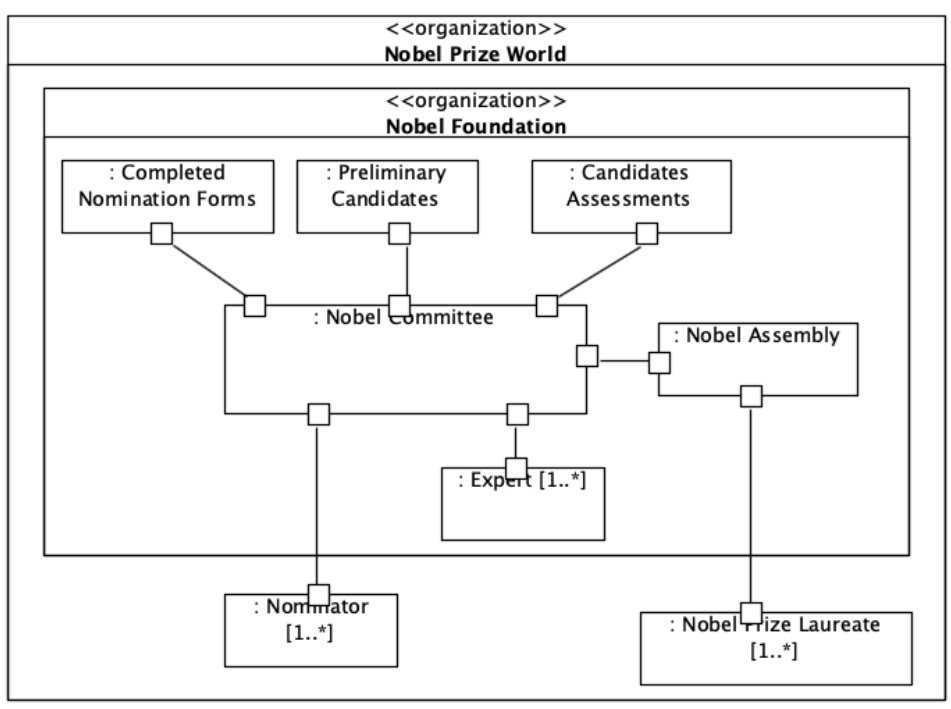


Fig. 82 Nobel Prize UM2 Model: Organization View

Fig. 82, the organization view, illustrates also the possible communication between the participants. Notice that the organization view is slightly different compared to previous case studies. The datastore constructs in BPMN diagram (see Fig. 80), which we modelled as objects in our model, were also introduced in the organization view showing which worker entities utilize and interact with them.

We believe that the Experts are part of Nobel foundation, whereas Nominators and Nobel Prize Laureates are external members. This is ambiguous in BPMN model, we have no way to know from it if this is true or they are external.

The multiplicities on the worker entities typed by Nominator, Nobel Prize Laureate and Expert represent the fact that they are multiple members.

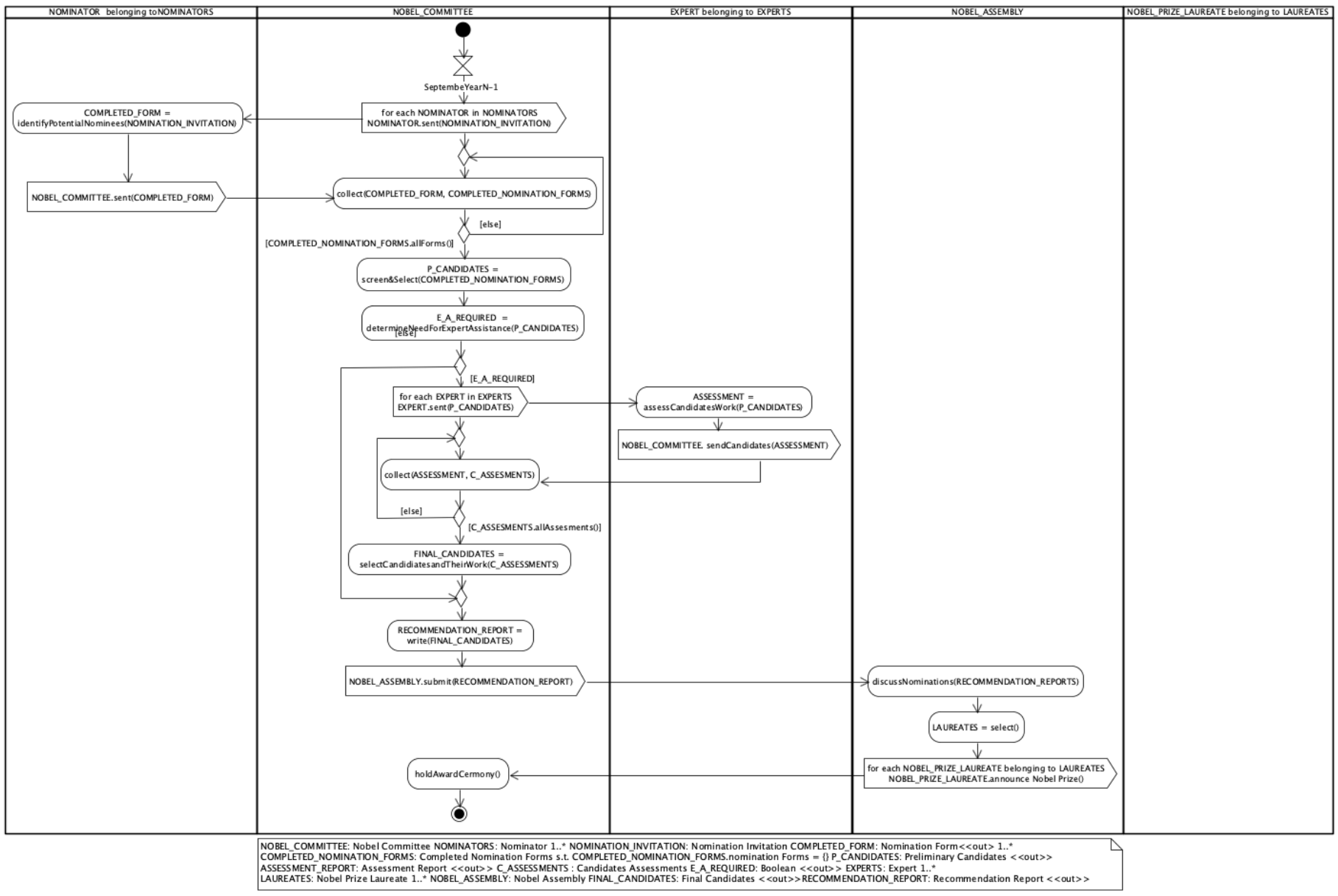


Fig. 83 Nobel Prize UM2 Model: Workflow View

The Fig. 83 represents the workflow view of the Nobel Prize process.

Notice that in the BPMN model the Nomination invitation is sent to multiple nominators. To represent the task "sent (Nomination Invitation)" to multiple nominators, we have modelled the operation as "for each NOMINATOR in NOMINATORS." This indicates that the invitation is sent to multiple nominators. Similarly, the task "sent (Preliminary Candidates)" has to be sent to multiple Experts. We have used naming conventions as "for each EXPERT in EXPERTS" to state Preliminary candidates are sent to multiple Experts.

Notice that the BPMN End events are placed in each participant pool to represent end of activity in each pool. This sometimes leads to confusion not knowing where exactly the process terminates. Using our model, we solved this issue by placing a one single final node in the Nobel Committee of Medicine pool showing correct flow termination.

4.6.3 Lesson learned

The errors in the BPMN model identified during the UML modelling are:

- In the Nominator pool there is some confusion, in one point it speaks about Completed Form(S) and in another point it speaks about a unique Form (i.e., Sending NOMINATION FORM to Nobel committee).
- In the Nobel Committee pool the task "Send Nomination Form" should actually send a nomination form, but instead we see that they send a "Nomination Invitation" which creates confusion to who read the model.
- In the BPMN XML file there is no way to express how many times a task is duplicated in parallel/a pool is duplicated and related to the duplication on the Nominator pool with the duplication of the task send nomination form.
- Both tasks "Write Recommendation Report" and "Submit Report with Recommendation" in the Nomination committee for medicine pool share the same object, but the names are synonymous. Similarly, Assessment and Assessment Report are also synonymous, and this creates confusion.
- The experts do the assessment in parallel performing several times the task "Assess Candidates Work." This is confusing and unusual, except the expert want to evaluate each time a group of candidates.
- The BPMN process states that the Experts have to Assess candidates work and send Assessment report to Nobel Committee. We do not know if all the Experts have to answer or only a certain percentage of experts.
- In the Nominator pool, Nomination forms are sent to the Nobel Committee. Here we have the same problem. We do not know if all the nominators have to send or only certain percentage of nominators. We do not know how to define such activity in our model.
- The names "Assessment Report" and "Assessment" seem to synonymous. To avoid the confusion, we always used "Assessment" in our model.
- The BPMN task "send Nomination form" actually has to send a nomination form but instead we see that nomination invitation has been sent to Nominators. To avoid confusion and simplify the process, we used only Nomination invitation. However, in entity view we solved this issue by adding a Nomination Form to the Nomination Invitations.
- Nomination form may be completed/not completed by the Nominators. We believe that if a Form is completed, we can extract from it a list of candidates.

4.7 Amazon Fulfilment

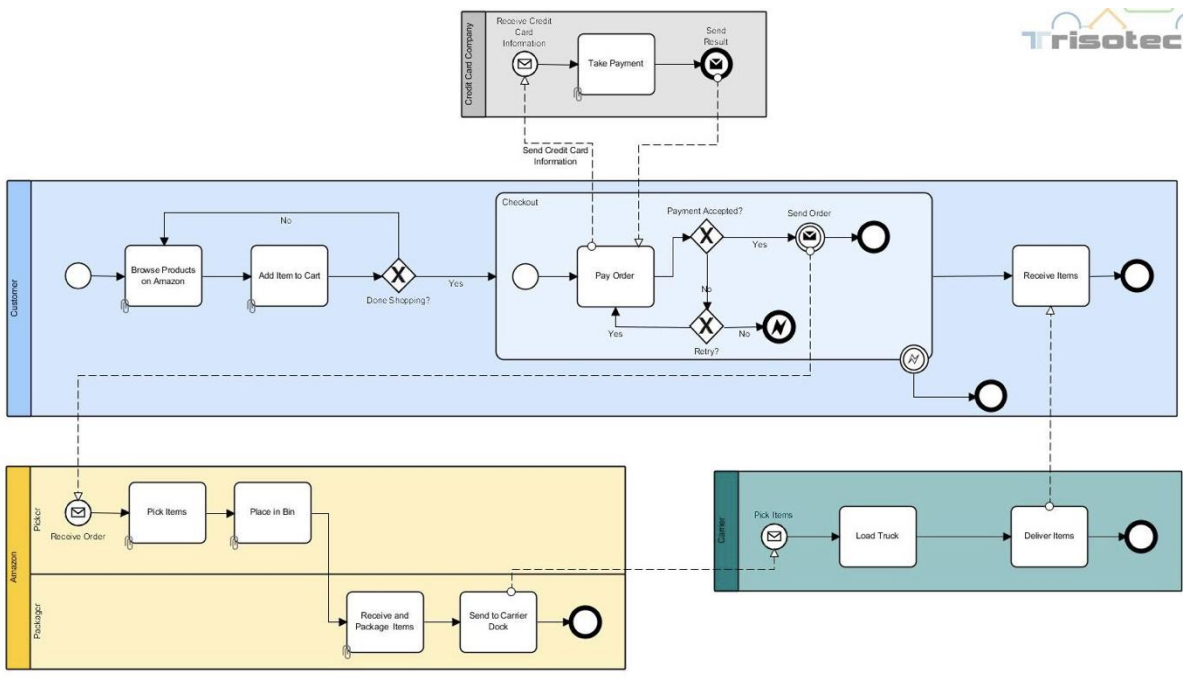


Fig. 84 Amazon Fulfilment BPMN model (Collaboration diagram)

4.7.1 The case study

The Amazon Fulfilment BPMN model [4] reported in Fig. 84 demonstrates how the Amazon orders are delivered to customers.

The process starts when a Customer browse the Amazon products and add them to the shopping cart until they finished shopping. After that a Customer finished shopping, they will proceed to the payment task. The payment task is placed in Sub-process. When the customer tries to pay with his credit card, the credit card information is sent to credit card company to take the payment and results are returned to the payment gateway. If the customer payment is accepted, the customer's order is sent to the Amazon warehouse. If the payment is not accepted, the Customer may decide to try again to make the payment.

After receiving the order from the customer, the Amazon warehouse Picker picks the ordered items and places them in a bin and later the items are sent for packaging. The Amazon warehouse Packager receives the items and prepares a package and sends it to the carrier dock. The Carrier picks the items and loads them in a truck, and, finally, they deliver the items to the Customer.

Notice that the worker participants in the BPMN diagram are not correctly modelled. For example, consider the real-world scenario, the customer interacts with Amazon web to place an order, and later, the Amazon web sends order conformation to the customer and sends the order to Amazon center. Whereas in the BPMN diagram, we see that the customer places order directly to the Amazon center. Notice also that in the BPMN diagram, the customer communicates directly with the Credit card company which is quite strange. Whereas in real world scenario, the Amazon web acts as a mediator between the customer and the Credit card company to perform the payment task.

Notice that, in the BPMN collaboration diagram we see a message exchange and private actions between worker participants. The UM2 method is then suitable for modelling this process, since we can model both the private actions and the messages exchanges.

We also wanted to model the Amazon Fulfilment process with UM1 methods since we want to extract the fuzziness from the BPMN model. Thus, the Amazon Fulfilment process will be also modelled using UM1 method using the abstract tasks.

4.7.2 UM2 model Amazon Fulfilment

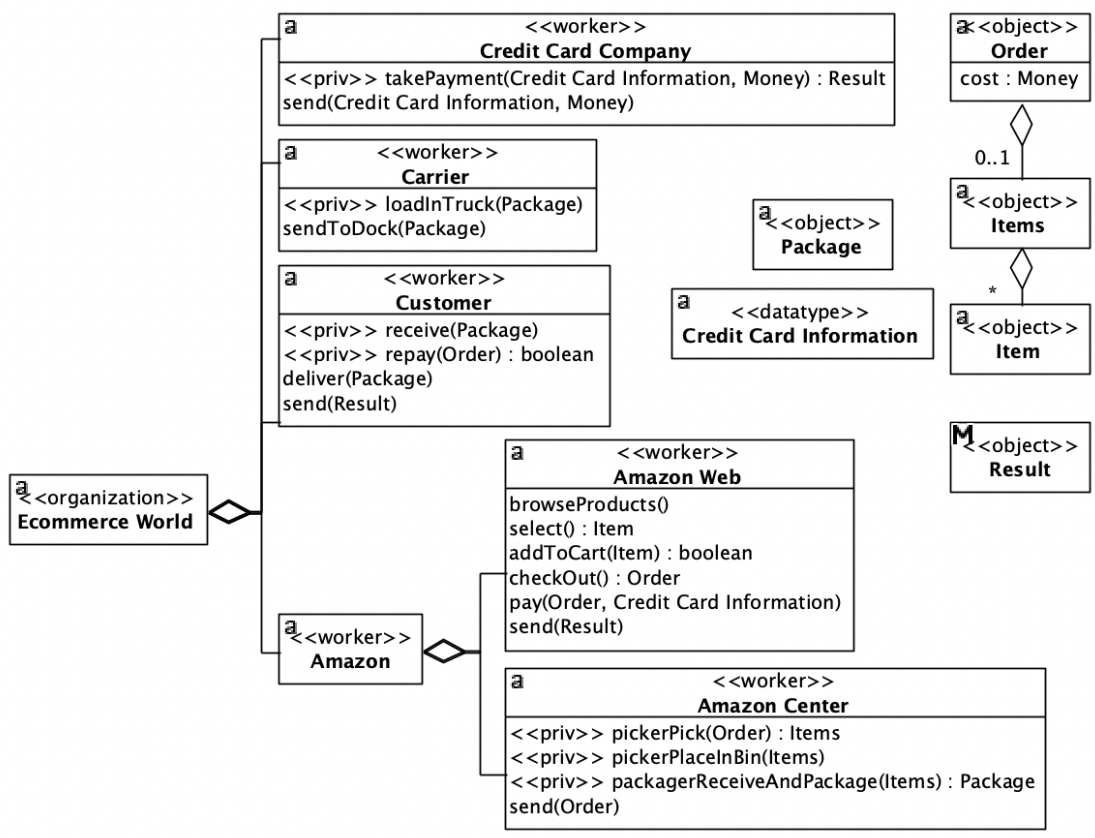


Fig. 85 Amazon Fulfilment Process UM2 model: Entity View

The Fig. 85 Represents the entity view of UM2 model Amazon Fulfilment process. Notice that we have introduced two other worker participants (i.e., Amazon Web and Amazon Center) differently from the original BPMN model, since we try to consider the real word case. Notice also that in the BPMN we see that the Amazon center uses lanes to show the worker participants (i.e., Picker and Packager). Whereas in our model we have rendered all the operations of picker and packager in the Amazon Center class simplifying the process. This is done due to the fact that our methods give the modeler the flexibility of choice in modelling. All the customer interactions with the Amazon web are precisely modelled.

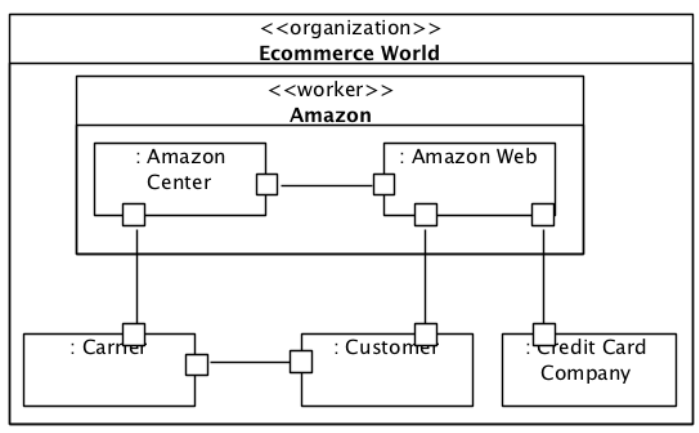


Fig. 86 Amazon Fulfilment Process UM2 Model: Organization View

Fig. 86 represents the Organization view of Amazon fulfilment Process showing also the possible communications between the process participants. As a consequence, in this model is immediately clear that the customer never directly communicates with the Credit card company. Notice also that the worker participants Amazon web and Amazon Center are placed inside the Amazon class (i.e., stereotyped as organization), showing the Amazon structuring.

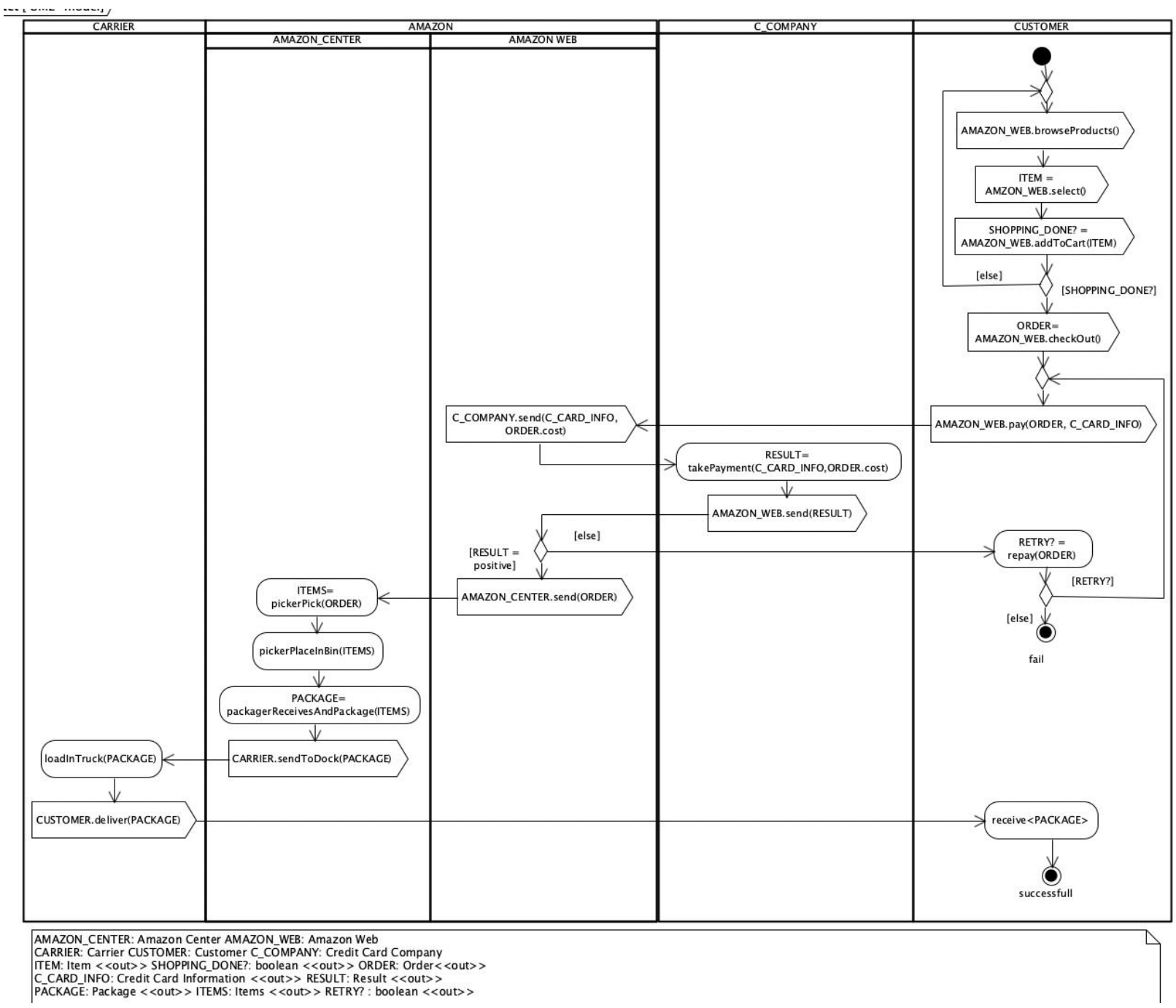


Fig. 87 Amazon Fulfilment Process UM2 Model: Workflow View

Fig. 87 represents the workflow view of the UM2 model of the Amazon Fulfilment process. All the actions of participants are precisely defined. For example, notice that few activities (i.e., browseProducts, addToCart, and Pay) in the customer pool are modelled as private actions. Whereas in our model we have correctly modelled these actions and they correspond to the interactions between the Amazon web and the customer. Notice also that we have used nested swimlanes to reflect the structuring of Amazon also in the workflow.

4.7.3 UM1 model Amazon Fulfilment

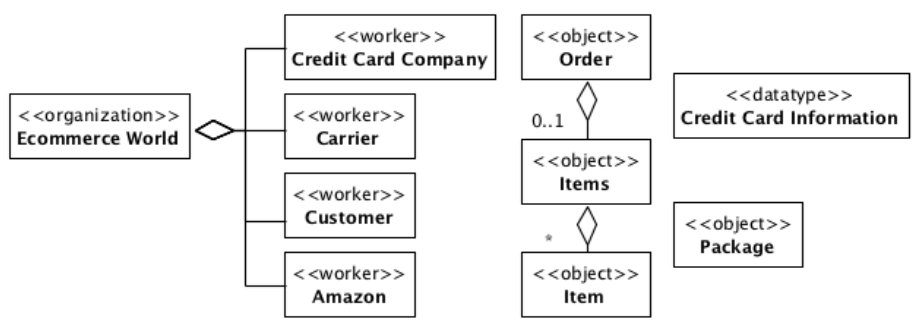


Fig. 88 Amazon Fulfilment Process UM1 Model: Entity View

Fig. 88 represents the entity view of the UM1 model of the Amazon Fulfilment Process. Notice that the entity view of the UM2 model in Fig. 85 (that is based on executable tasks) is different from this one. In this model, we have modelled participants similar to the original BPMN participants (i.e., the customer interacts directly with the credit card company. Whereas in UM2 entity view we see that the customer interacts with the Amazon web to perform payment.

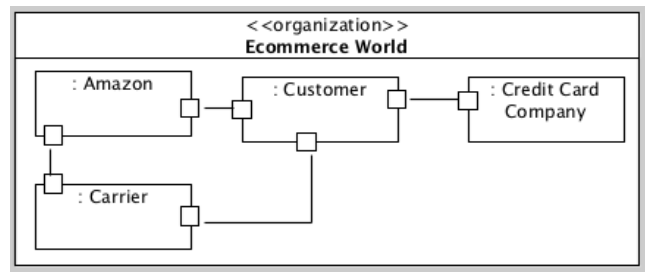


Fig. 89 Amazon Fulfilment Process UM1 Model: Organization View

Fig. 89 represents the Organization view of the UM1 model of the Amazon Fulfilment Process. Notice that here is clear that the customer communicates directly with the Credit card company without having a mediator (i.e., Amazon). The organization view helps also to spot the differences in the communications between process participants.

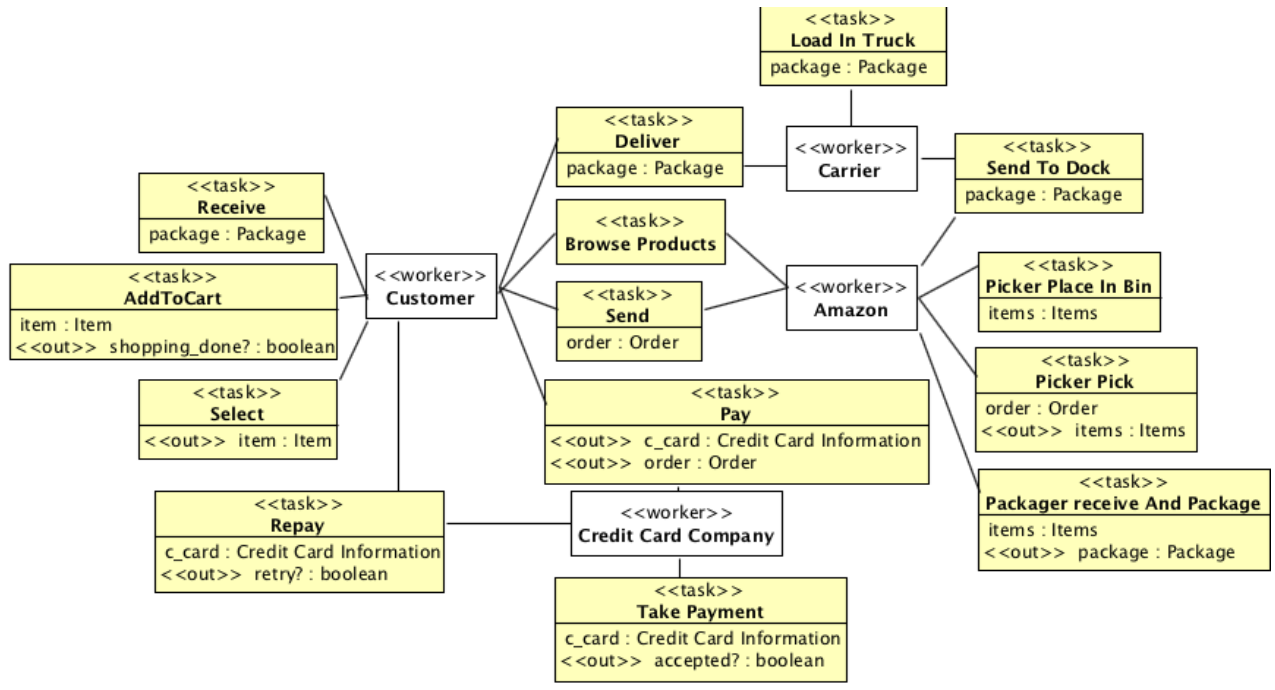


Fig. 90 Amazon Fulfilment Process UM1 Model: Task View

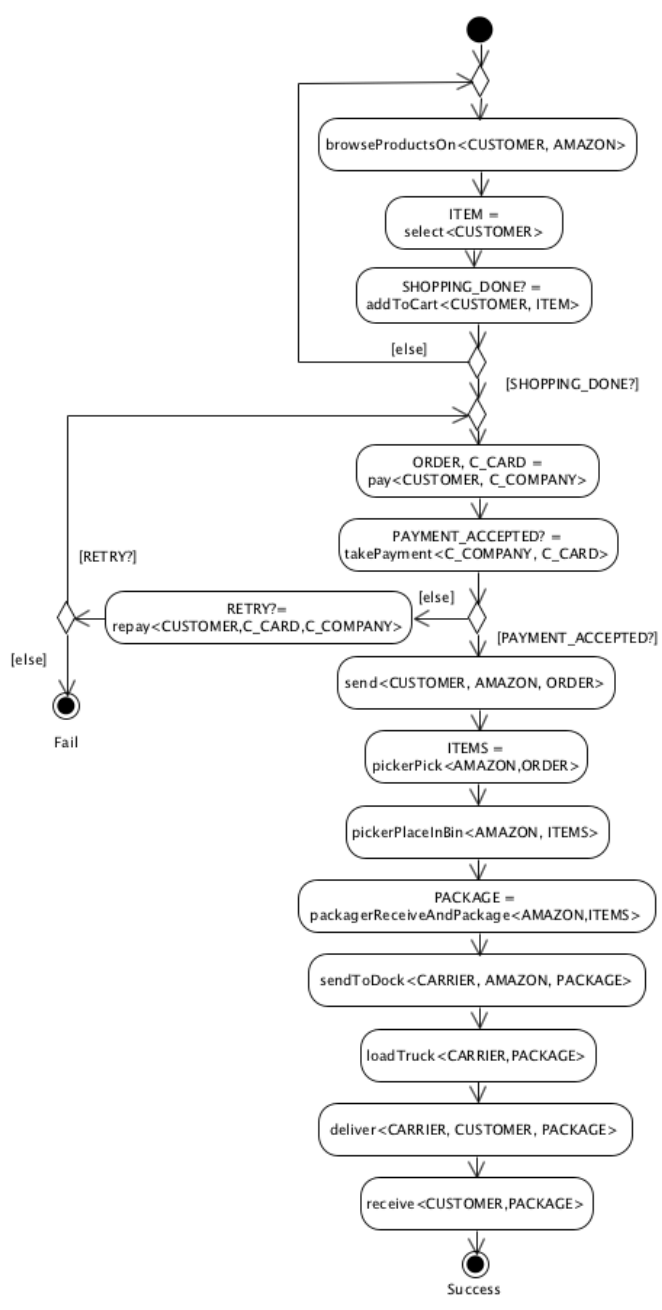


Fig. 91 Amazon Fulfilment Process UM1 Model: Workflow View

Fig. 91 represents Workflow View of the Amazon Fulfilment Process UM1 Model.

4.7.4 Lesson learned

- In the BPMN model the role of error event and subprocess (located in the customer pool) is that when the error is discovered, the process terminates with failure. We might model this case using interruptible regions, but since we want to make clear that process final states are not all the same type, we have modelled this case using activity final nodes labelling as failure and successful.
- Notice that the price of the order is not defined in the BPMN. The operation modelling the payment needs to know the cost of the order. we have been led to define the cost of the order precisely. We explicitly added a cost parameter for the order, that made the model more realistic, but we may have made otherwise.

The errors in the BPMN model identified during the UML modelling are:

- The task payment was performed without defining the amount of order.

4.8 Recruitment and Selection

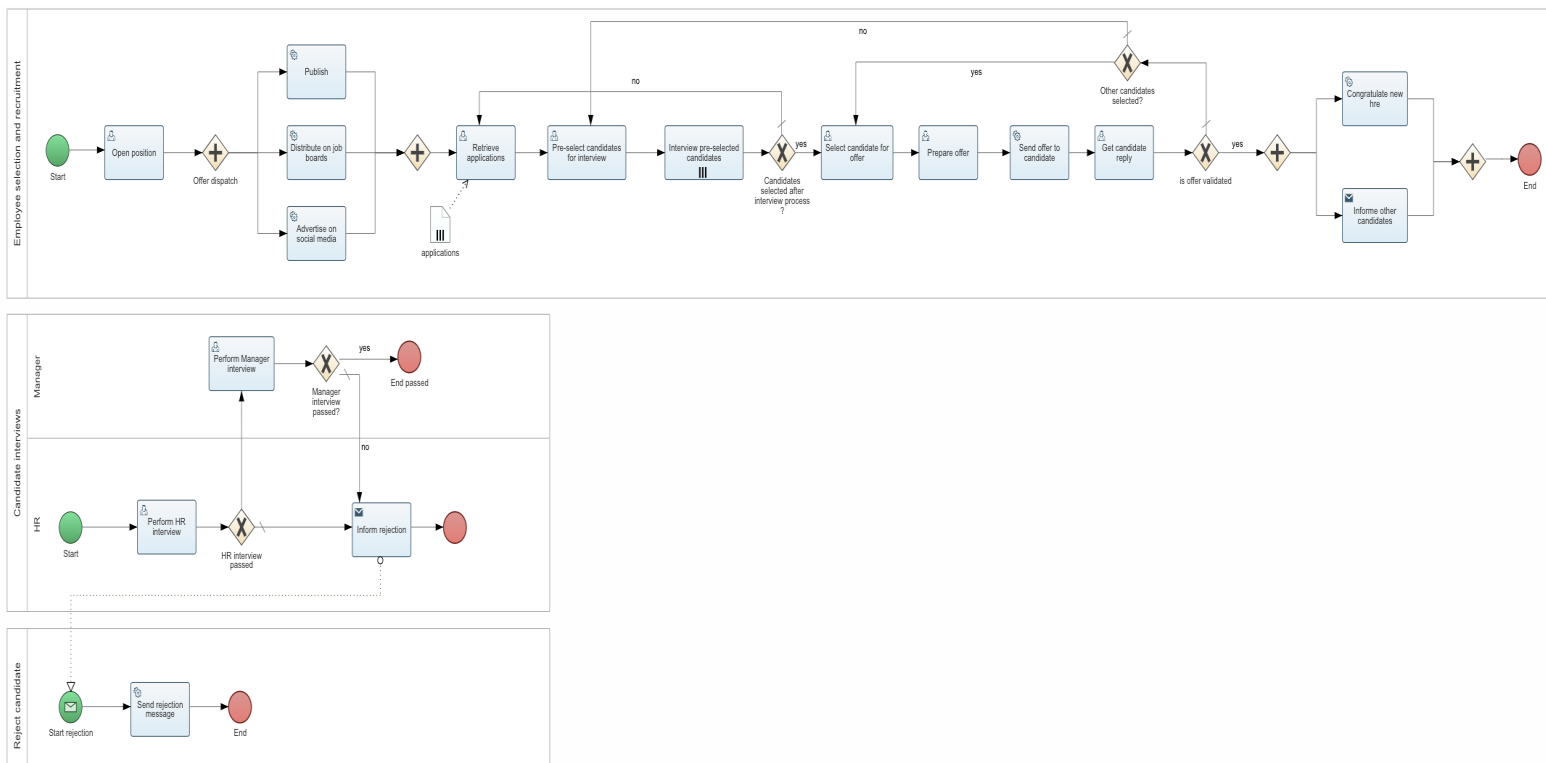


Fig. 92 Recruitment and Selection: BPMN Model (Collaboration diagram)

4.8.1 The case study

Fig. 92 represents the Recruitment and Selection Process BPMN collaboration diagram and this process was taken from [6]. The process starts at the employee selection and recruitment pool. Note that whenever we are use the term E_S_R Team, we are referring to the employee selection and recruitment pool. The E_S_R Team open positions. They also publish position, advertise the position on social media and distribute on jobs board. The E_S_R Team starts retrieving applications to pre-select candidates for interview. The pre-selected candidates are interviewed by the HR and the Manager (the HR and the Manager are located in the Candidates interviews pool). The candidate performs the HR interview. If the applicant fails the interview, the HR informs the applicant about the failing results. If the applicant passes the HR interview, the applicant is sent to the Manager to be further interviewed by the manager. If the applicant fails in the manager interview, the manager informs the applicant about the failing results and later the results are also shared with E_S_R_Team. Based on the results the E_S_R Team will check if all the candidates are selected after interview. For the resulting failing results, the E_S_R Team will retrieve applicants and for the passed results, candidates are selected and the E_S_R Team will prepare an offer. Later, the E_S_R_ Team will send offer to each candidate and wait for the candidate reply. If the offer is accepted by the candidate, The E_S_R Team will congratulate Applicant and inform other applicants. If the offer is not accepted by candidate, the E_S_R Team selects other candidates from the group of applicants who passed interview or select a candidate from the pre-selected candidates.

We have decided to use the UM2 method to model the recruitment process since we can model communications between participants precisely, but we discovered that there was a problem in this process (that most of the tasks in this process correspond to abstract tasks). We also considered modelling this process using the UM1 since that allows us to get the wanted information disregarding details in a precise way. After deeply analysing the process we have found that the interview actions in the recruitment process cannot be rendered with executable tasks and also in the whole BPMN model there is only just one communication (We believe that the authors of the recruitment process have used the concept of the pool to model different processes/subprocesses, and it is not clear at all how they are related. For example, the pool of Candidate interviews is the specification of an instance of the multiple task Interview of preselected candidates, and we may model tasks using sub-activities. We assume that the authors of the recruitment process model used the pools to simulate in some way the sub-activities). Hence, we dropped the UM2 method and chosen the UM1 method to model the recruitment process.

4.8.2 UM1 model Recruitment and Selection Process

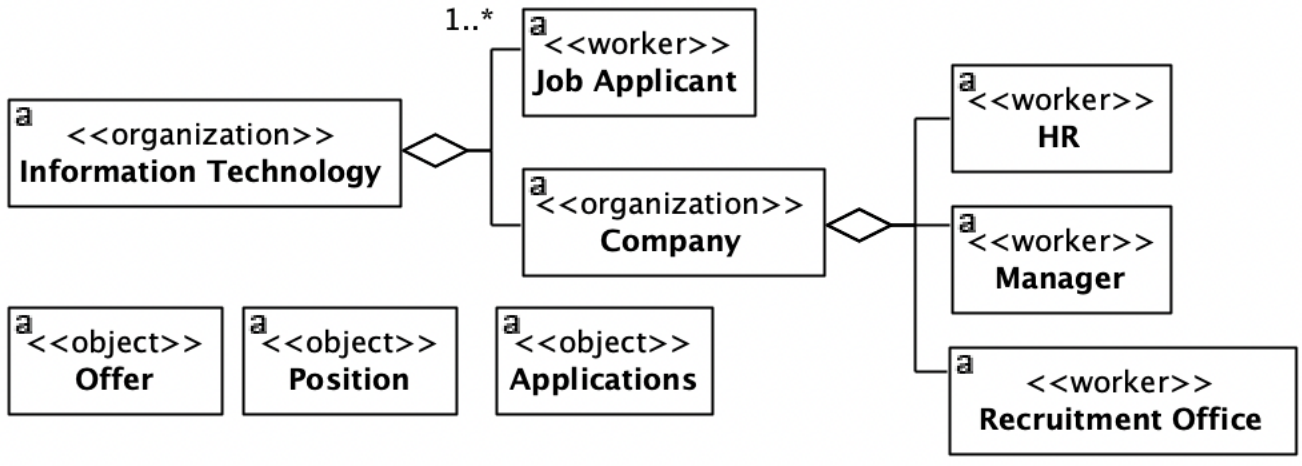


Fig. 93 Recruitment and Selection Process UM1 Model: Entity View

Fig 93 represents the entity view of the UM1 model Recruitment and Selection Process. In the BPMN diagram the applicant was modelled as "Reject candidate" which was a mistake and, in our model, we have explicitly modelled the applicant as "Job Applicant" and we added multiplicity as 1..* stating there may be one or multiple Job applicants.

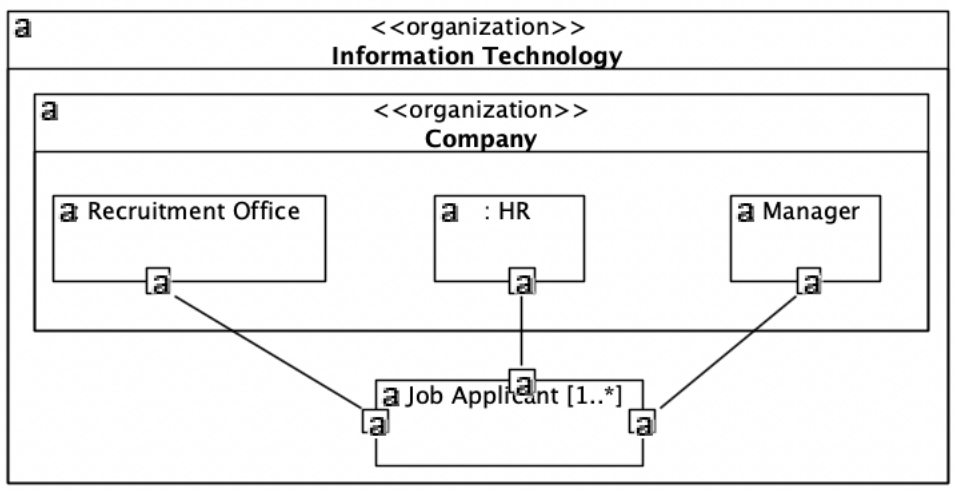


Fig. 94 Recruitment and Selection Process UM1 Model: Organization View

Fig .98 represents the organization view of the UM1 model Recruitment and Selection Process. Notice that in the BPMN diagram (see, fig. 92) there are no hints if the Manager and the HR belongs to the Organization and the structure of organization is not properly modelled. Whereas in our model we can clearly see that the Manager and HR belongs to the enterprise enacting the process (Company) and the structure is well organized.

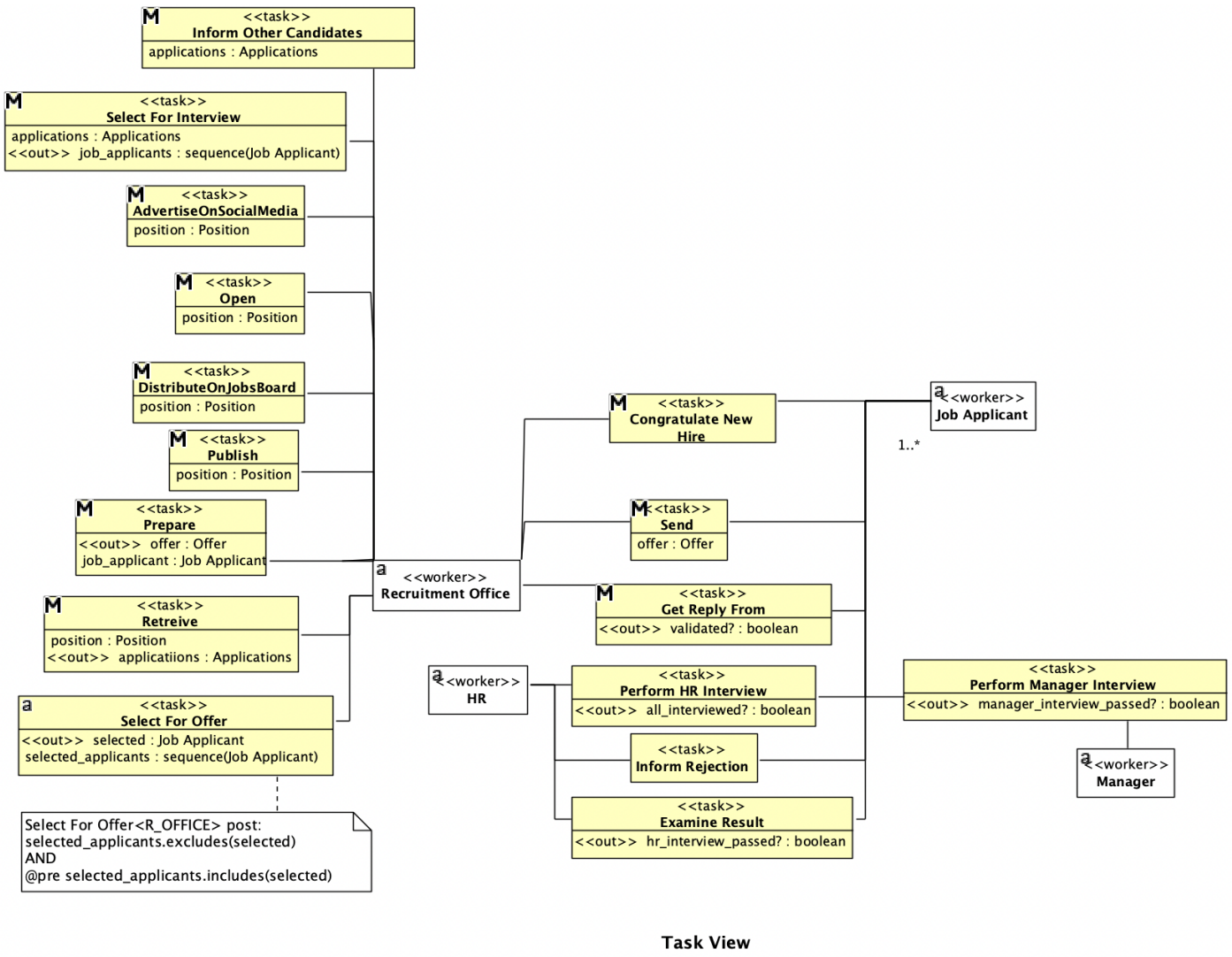


Fig. 95 Recruitment and Selection Process UM1 Model: Task View

Fig. 95 represents the task view of the UM1 model Recruitment and Selection Process. The task view is precisely defined. For example, the behaviour of the task "Select For Offer" is defined through the OCL constraints stating that the returned applicant of this task was belonging to the sequence before and after the task execution the selected applicants will no longer belong to the sequence. Notice that in the BPMN diagram (see, fig. 92) they have used a data object to represent the Applications. In our model if we try to define the meaning of this task we are obliged to introduce a store for the applications and may be introduce the actions of applying, if we do this we may be quite abstract disregarding this aspects (We may also have a very loose model that does not consider these aspects and so the task retrieve just return some applications and we do not know how they are determined).

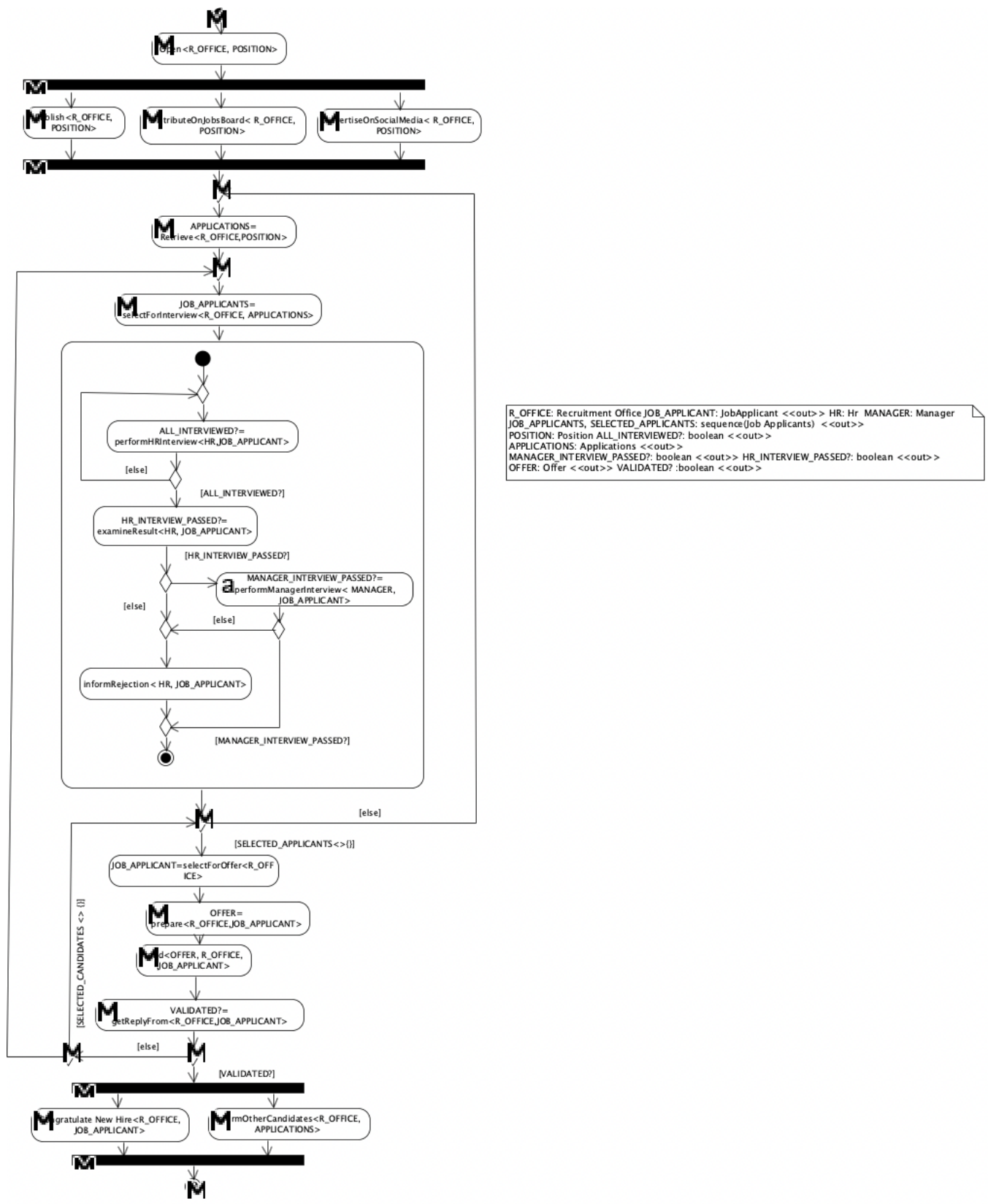


Fig. 96 Recruitment and Selection Process UM1 Model: Workflow View

Fig. 96 represents the Workflow of the UM1 model Recruitment and Selection Process. Notice that workflow in BPMN diagram (see, fig 92) is different compared to this model (the differences are related to the interview actions). The modelers of this process have used pool to model the interview actions and this mistake since there is no connection between the Recruitment and Selection pool and also it creates confusion in understanding the process. In our method we have modelled the interview actions precisely by modelling them using sub-activities. Notice also that in the BPMN diagram the interview actions represent hiring of single Job applicant (see Candidate interviews pool). Whereas in our model we have precisely modelled the interview actions. For example, notice that there is a decision node representing the loop (loop continues until all the Job Applicants are interviewed) after the task "performHrInterview".

Notice that in the BPMN there is task named as "interview pre-selected candidates" located in "Recruitment and Selection" pool, and this task was marked with parallel instance marker (parallel instance maker is used in BPMN to state that the task is executed multiple times). This is an ambiguity in BPMN not knowing if this task is for unique task or repeated task. If we assume that this as a unique task (i.e., non-repeated task), we need to model a detailed version of process with another activity diagram and task view diagram considering the interview actions were modelled for single Job applicant. Thus, we have also modelled a task view and workflow view for single Job Applicant.

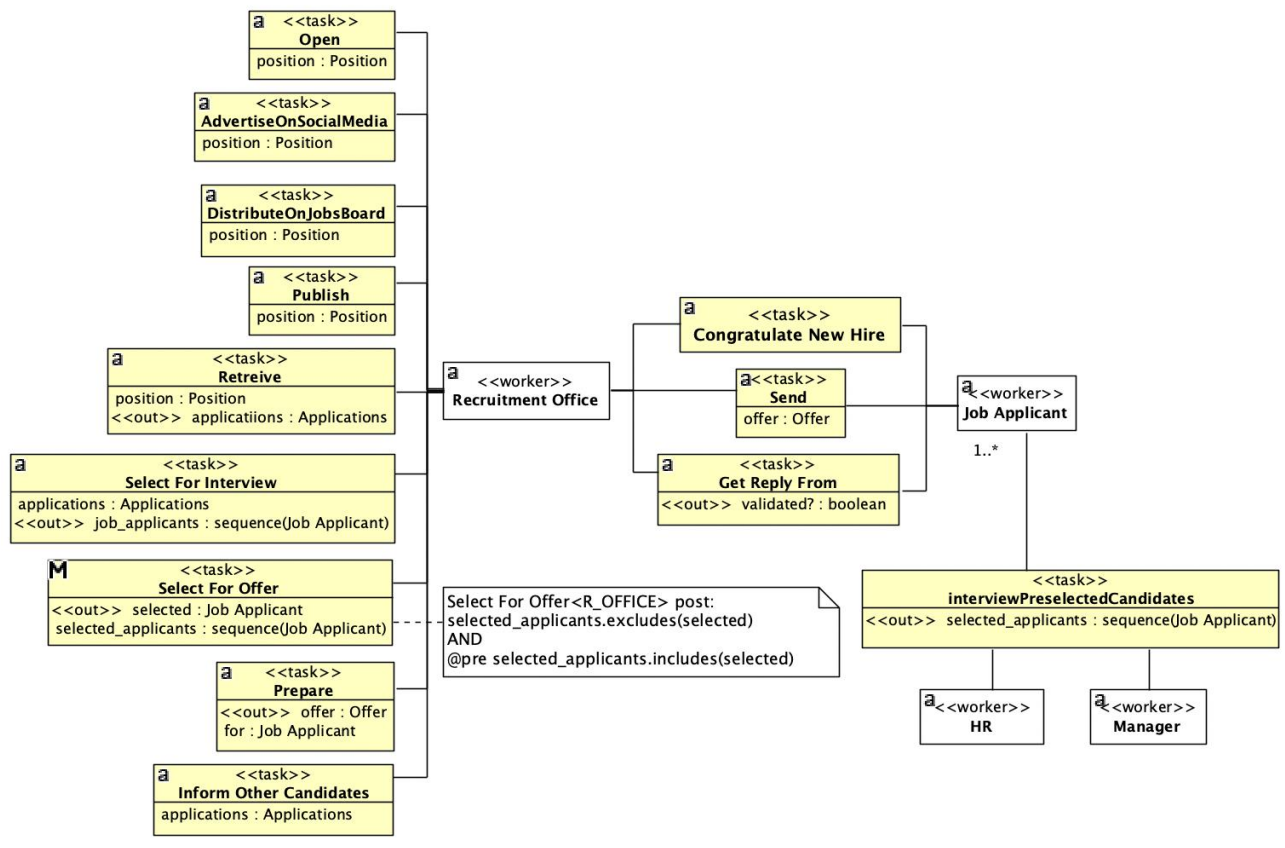


Fig. 97 Recruitment Process Unique candidate UM1 model: Task View

Fig. 97represents the task view of recruiting a unique candidate in the Recruitment and Selection process. Notice that this task view is different from the fig. 95 now the task view shows all the tasks corresponding to the unique Job applicant.

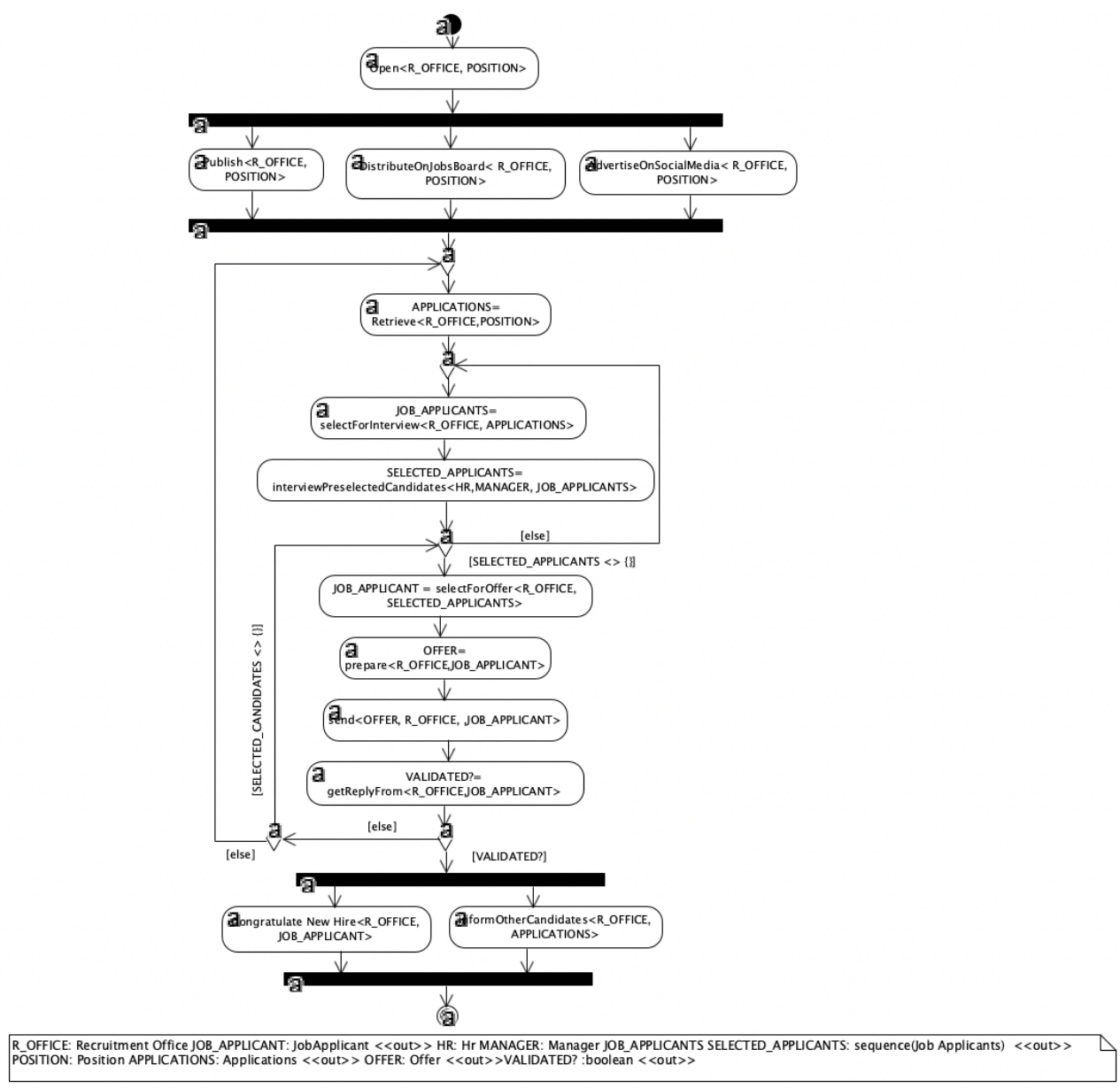


Fig. 98 Recruitment Process Unique candidate UM1 model: Workflow View

Fig 98. represent the workflow view of recruiting unique candidate in the Recruitment process. Notice that this workflow is different from the workflow defined in fig.96 This workflow clearly shows the decomposition of the activity of interviewing the candidate since now we only modelled interview actions for a single job. This workflow is precise and clearly understood since we did not use any sub-activity/many interview action nodes to explain the process.

4.8.3 Lesson learned

The errors in the BPMN model identified during the UML modelling are:

- The pool name in the BPMN diagram (see fig. 92) was modelled as "Reject candidate" and this is a mistake. The tasks inside the pool is completely strange and unable to understand.
- The pool name "Employee Selection and Recruitment Team" in the BPMN diagram has changed to "Recruitment Office". There are no hints in the model that this is a team, it seems that it is the office/clerk that mange the activities related with the recruitment.
- The multiple task labelled as "Interview preselected candidates" in the BPMN is an ambiguity. We modelled it as unique task and repeated task. For considering it a unique task we have created another activity diagram decomposing the BPMN "Candidates interview" pool as single task and for considering the repeated task we have modelled using sub activity in the activity diagram.
- The name of the choice node (is_Offer_Validated) in the BPMN diagram (E_S_R_Team pool) has been changed to "validated" since using the shorter names increases the readability and understandability of the process.

4.9 Incorporation of New Employee in Hospital

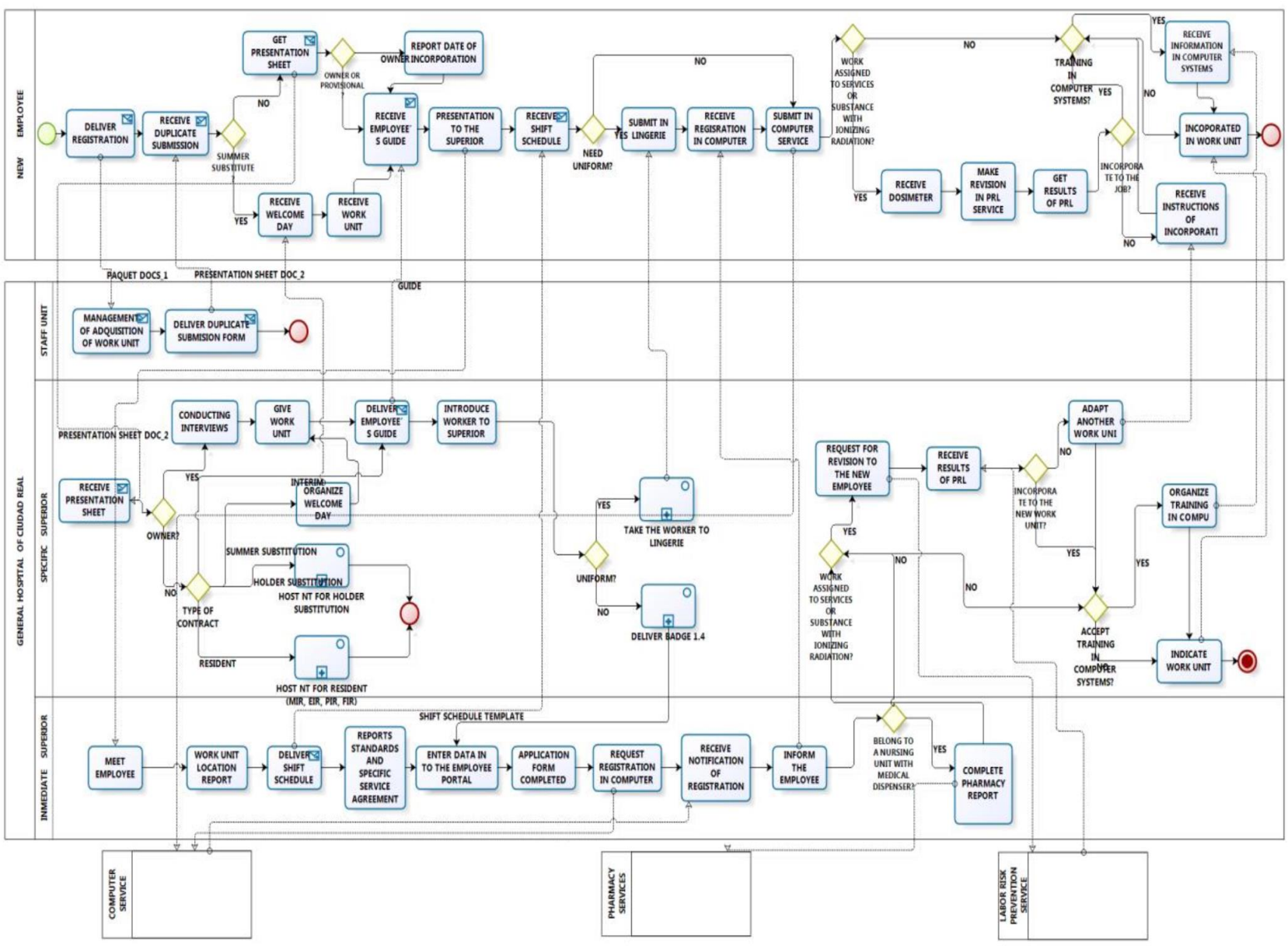


Fig. 99 Incorporation of the New Employee (INE) hospital process: BPMN (Collaboration diagram)

4.9.1 The case study

Fig. 99 represents the incorporation of the New Employee (employee hiring) process in General hospital of Ciudad real. This case study was taken from the [5] paper. This process is a problematic having a true spaghetti-workflow with very complicated messages flows. Note that the process is explained by mimicking the BPMN model. To simplify the process explanation, we decided to explain the process by a sequence of tasks in the pool since the flow is quite confusing and one might not clearly understand the process following the flow. The process starts in the New employee pool by sending a registration form to the staff unit. The staff unit examines the registration form and sends the presentation sheet to the new employee. The new employee receives a presentation sheet. We assume that now the new employee checks the type of contract (summer substitute). If the new employee contract corresponds to a summer substitute, he receives the welcome day from a specific superior. If the contract type is not a summer substitute, he sends the presentation sheet to the Specific Superior. If the new employee is the owner, he reports on the day of owner incorporation (we do not know to whom the owner is reporting). If the new employee is not the owner, he will receive the welcome day (employee joining formalities provided by the organisation) and workunit from the specific superior. Now the new employee receives an employment guide from the specific superior and sends a presentation sheet to the immediate superior. After sending the presentation sheet, the new employee receives the shift schedule from the immediate superior. If the employee needs a uniform, the specific superior takes the new employee for tailoring and the new employee submits in

lingerie (uniform). The new employee receives a reservation request from the computer service and submits his request to the computer service. We don't know the proper meaning of the Gateway (Worked assigned to services or substances with ionizing radiation), but we assume that it corresponds to the tools or work given to the new employee. In case of employee corresponds to the work assigned, he receives the dosimeter, makes a revision of plr service and finally gets the result of plr (we do not have any proper information about Plr). In case of work not assigned to the employee, the new employer has to decide if he needs training in computer systems. After getting the results of plr, the new employee has to decide if he is ready to incorporate the job. if the answer is no, the new employee will receive instructions from the specific supervisor about the incorporation. If the employee is ready to incorporate the job, he will take training in computer systems and receive information in computer systems. If the employee does not need training in computer systems, he will directly Incorporate the work unit and the process finally terminates in the employee pool.

After receiving the presentation sheet from the new employee, the specific supervisor has to check if the new employee is the owner. In case of the new employee being an owner, the supervisor will conduct an interview, give the work unit, deliver the employee guide to the new employee and later introduce the worker to the superior. In case of a new employee not being an owner, the specific supervisor checks the type of contract. If the contract corresponds to the resident, then he performs a sub-activity "Host NT for Resident" (We do not know the functionality of this task). If the new employee corresponds to the contract type "holder substitution" then he performs the sub-activity. "Host NT for Holder Substitution" (We do not know the functionality of this task too)". if the new employee corresponds to the contract "Summer Substitution", the specific supervisor organizes the welcome day. If the new employee corresponds to the contract "interim", the specific supervisor gives employer guide to the new employee. After introducing the worker to a superior, the specific supervisor has to decide if the new employee needs a uniform. If there is a need for uniform, the supervisor will take the worker to the tailoring, else he will deliver the badge to the immediate supervisor.

The new employer meets the immediate supervisor after sending the presentation sheet. The immediate superior performs the task work unit location report and send the shift schedule template to the new employee (We assume that this template contains details about the work timings). The immediate supervisor checks the report standards and specific service agreement, enters data in the employee portal and completes filling out the application form. After filling out the application form the Immediate superior will request registration in the computer and receive notification of registration from the computer service and also informs the employee about the registration. Now the immediate superior will check if the employee belongs to its nursing unit. In the case of an employee belonging to the nursing unit the immediate superior, he will complete the pharmacy report and send the report to pharmacy services, Else the work assigned to an employee is checked.

In case of work assigned to the employee, this specific supervisor will request for revision of the employee and receive the result of prl. In case of work not assigned, the specific supervisor has to go through another check regarding the training in computer systems. In case of no training in computer systems, the specific supervisor will indicate work unit, else he will organize training in computer systems. After receiving the results of prl, the specific supervisor has to check if the employee is incorporated into the new work unit. In case of the new employee not incorporated the new work, the specific supervisor suggests the new employee adopt another work unit, or else he is sent to training in the computer system.

We have discovered unusual aspects while modelling this process. It seems that the BPMN diagram (see fig. 99) has some deadlocks and other strange things. Based on our assumption a task with an incoming message has not terminated if the message has not arrived. For example, in the case of summer substitution we have a deadlock, it seems that the messages are also a kind of flows influencing and the ordering of the action making things complicated. Looking at the BPMN layout style, all the beginning elements in all the pool are at the same level and this makes the process still more complicated.

Notice that the flows in this process are misleading and the process is quite complicated to understand. We wanted to simplify the process by modelling this process using our UML models. Notice also that we can see the communication between process participants. Thus, the INE (incorporation of New Employee) process will be modelled using UM2 since we can represent communications and private actions precisely.

4.9.2 UM2 model Incorporation of New Employee process

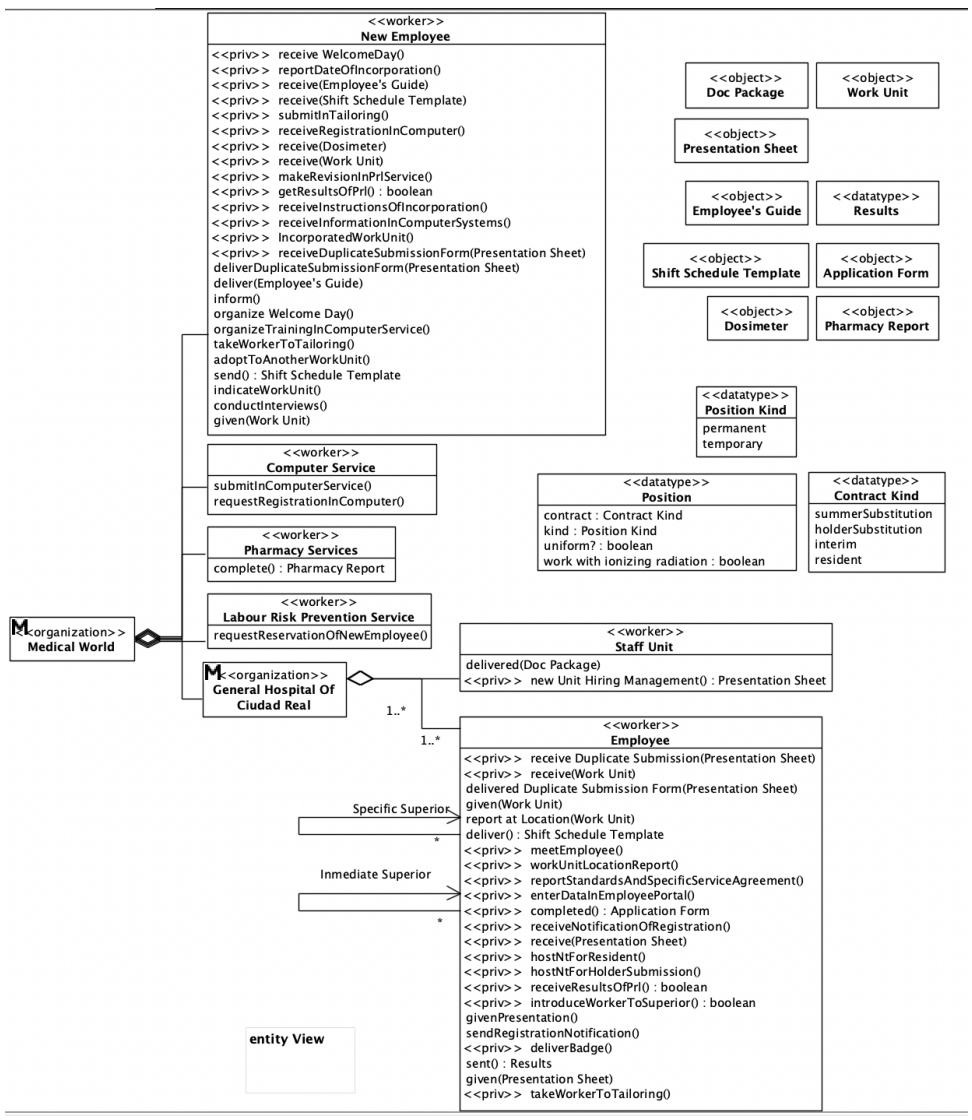


Fig. 100 Incorporation of New Employee Process UM2 Model: Entity View

Fig.100 represents the entity view of UM2 Model Incorporation of New Employee Process. The entity view is precisely defined. For example, notice how the workers are defined. We have explicitly defined an Employee entity holding all the operations of the Specific Superior and Immediate Superior. The workers (i.e., Specific Superior and the Immediate superior) are defined as roles of in employee class (in general the Specific superior and Immediate Superior are considered as roles played in the hospital). We have explicitly produced a few entities in the entity view. This is done due to the fact that explicit data modelling helps in understanding the process precisely since the process includes a lot of fuzziness. For example, the datatype Position, Position Kind and Contract kind are explicitly defined. The attributes of the Position Kind class represent the position type of employee (Temporary and Permanent). Whereas in BPMN, these attributes are defined as Owner and Provisional in the New Employee pool. The attributes of the contract kind class represent the contract type of the employee (four contract types i.e., summer Substitution, holder Substitution, Interim, Resident located in Specific Superior pool). There is no indication of multiplicities in the BPMN model (see fig. 99). Whereas in our model we have precisely defined the multiplicities and navigation indication on the associations of two participants (Specific superior and Immediate superior) and two entities (employee and the Staff Unit).

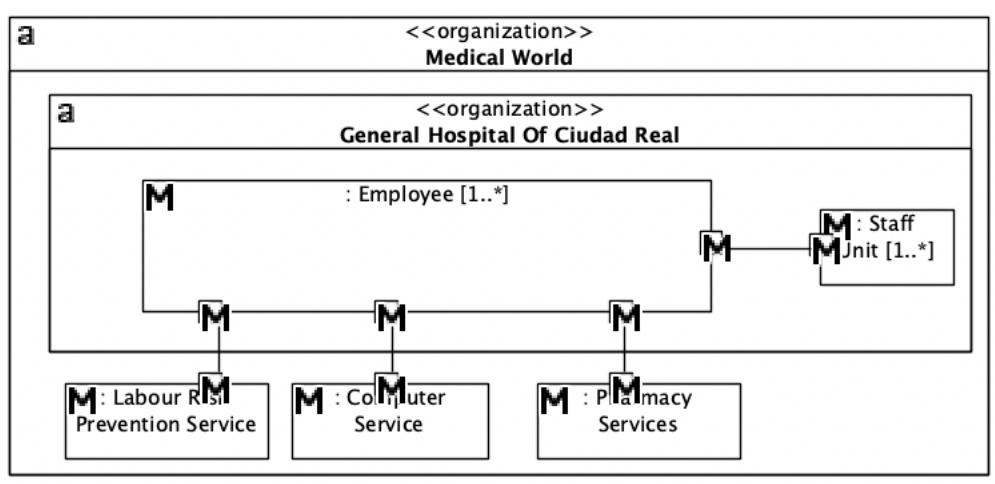


Fig. 101 Incorporation of New Employee Process UM2 Model: Organization View

Fig. 101 represents the organization view of the UM2 model INE (Incorporation of New Employee Process) process. Notice that the organization view is slightly different compared to the previous case studies. The specific superior and the immediate superior are not modelled, since they are considered roles. BPMN uses pools and lanes to identify the process participants and sometimes using this mechanism is not suitable to describe the process participants. For example, notice that the new employee in the BPMN diagram is outside the hospital so they can make communications, but when the new employee is inside the hospital pool, the communications are not captured. Whereas in the organization view we don't have this problem, we may model communications among the various participants though they are inside some organization units.

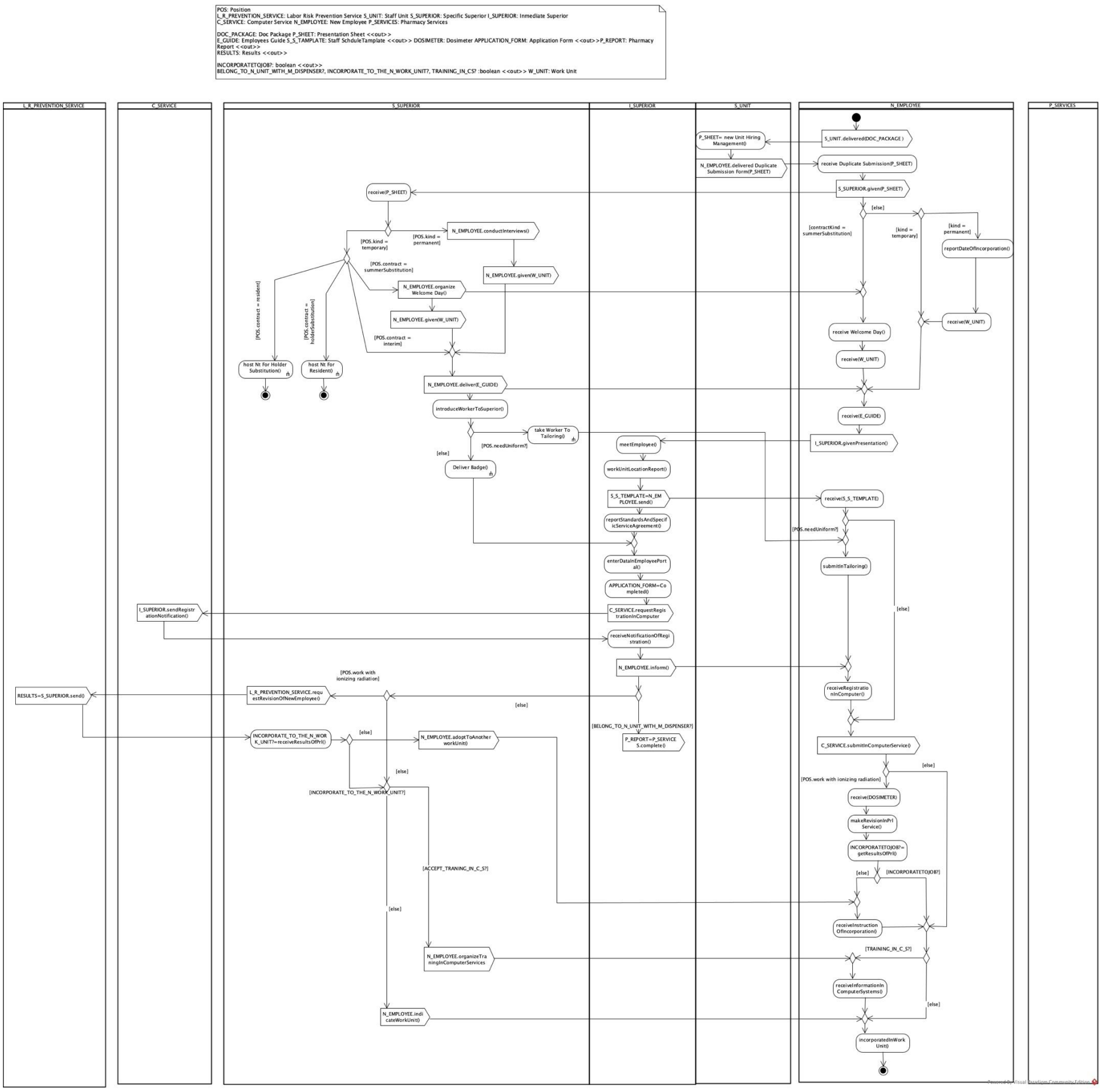


Fig. 102 Incorporation of New Employee Process UM2 Model: Workflow View

Fig. 102 represents the workflow view of UM2 model Incorporation of New Employee Process. Using the explicit data modelling we have been led to define workflow precisely. For example, the name of the choice node was labelled as owner or provisional in BPMN. In our model we explicitly changed the names of the choice node as temporary or permanent. This simplifies the process for better understanding. Notice the flows, for example in the BPMN (see fig. 99) the flow is confusing and looks like a spaghetti model. Whereas in our model the flows are clear and easy to read the model. There are a couple of choice nodes (one located in the New employer pool names as "NEED UNIFORM" and the other located in the specific superior pool named as "UNIFORM") that creates confusion while reading the BPMN model. Both the choice nodes correspond to similar functionality but due to change in names and it is difficult to read model. Whereas in our model we solved this issue by naming both the choice nodes with a similar name (i.e., UNIFORM) and we also defined it in the entity view based on the fact that having shorter names in the workflow view will improve the readability of the process. This helps the method user understanding the process precisely. This is better expressed in the entity view by having an explicit class and also not all the relevant details can be expressed through the workflow view. The task "Give Work Unit" in the specific superior pool should correspond to an interaction between the new employee and the specific superior. But in the BPMN model (see fig 99) it was modelled as a general task. Whereas using our method we have defined it as communication between the new employee and the specific superior.

Notice that the workflow defined in our model (see fig. 102) and in the BPMN model, uses many duplicate choice nodes (i.e., the functionality is the same but labelled with different names). We also wanted to produce the workflow by eliminating the duplicate choice nodes. This is done due to the fact that having many choice nodes creates confusion and makes workflow more complicated and unreadable. thus, the workflow is also produced eliminating duplicate decision nodes. The workflow in the figures 102 and 103 are not clearly seen since they are large in size and does not fit to this paper size. However, we provided a HTML version of this case studies in the beginning of the paper where these pictures are clearly seen.

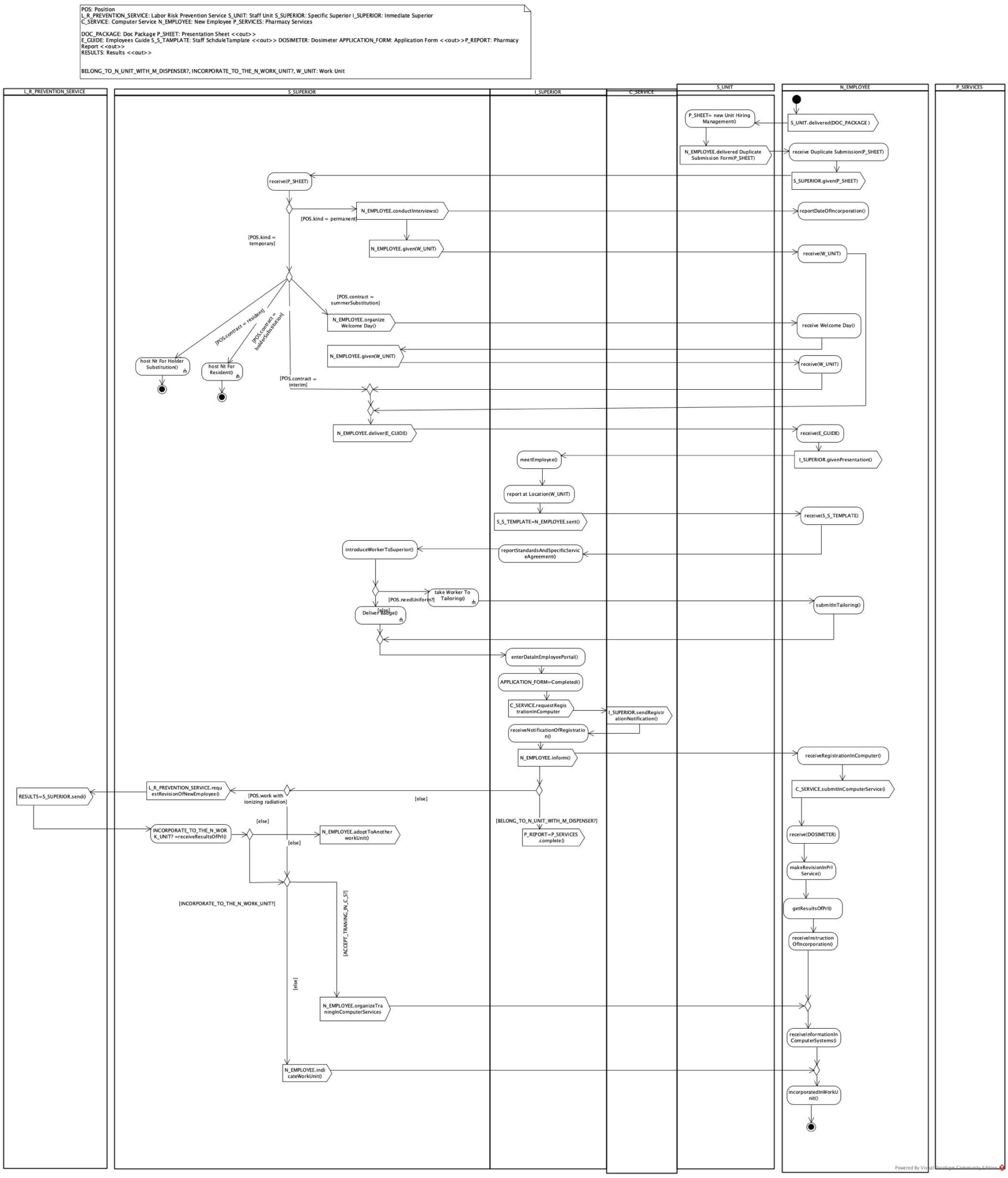


Fig. 103 Workflow View with Duplication of Choice Nodes

4.9.3 Lesson learned

The errors in the BPMN model identified during the UML modelling are:

- In the BPMN model it seems to be a mistake since there is no start in the General Hospital of Ciudad real pool
- In the BPMN model, many task names refer to some activity and the behaviour of the tasks represents different from the activity named
- The English language used in the process is not clearly understood. For example, "get presentation" in the New employee pool states that receiving something but this task is sending a presentation sheet to the specific superior. We have changed the name of this task as "given(presentation).
- The task "conducting interviews" was named as a mistake in BPMN. this is the classical problem in interviews/meetings in the business process modelling and the names work fine when using abstract tasks, but they are not suitable for communications. This task also corresponds to communication with the new employee but it was defined as private action in the BPMN model. Thus, we have given correct naming to this task "conduct interview" in our model.
- The layout in the BPMN model is complicated and flows are overlapping. This gives the method user confusion while reading the process. For example, in the BPMN layout style all the beginning element in all the pool are at the same level, and this make things still more complicate. Flows in our methods are clear and represented precisely.
- The task "organize welcome day" in the specific superior pool is modelled as a private action but it corresponds to communication with the new employee. In our method, we modelled it as communication.
- The task "managementOfAdquestionOfWorkUnit" is not clearly defined. We assume that the modellers were Spanish and the level of the English language used in the process of modelling was limited. We also changed the name of this task to "newUnitHiringManagment"
- Notice that the presentation sheet was produced in the task "managementOfAdquestionOfWorkUnit". We do not know if the presentation sheet is produced for all the employees. We modelled this task assuming a presentation was produced for each employee.
- Notice also that in the new employee pool there is a task named "reportDateOfOwnerIncoporation". After this task, the flow continues to receive an employee guide which is a mistake in the BPMN model. Based on the domain knowledge we have explicitly added the task "receive (WORK_UNIT)" since the candidate also has to receive workunit after reporting the date.
- The task "reportStandardsAndSpecificServiceAgreement" in the Immediate superior pool has to communicate with the New employee but was modelled as private action.
- There is a problem in detecting the order of activities of the Specific superior and the Immediate Superior and we have solved this issue in our model. We may produce a better model trying to discover parallelism if it is not inconsistent.
- The task name "takeWorkerToLingerie" is not understandable. We assume the worker is taken to the tailoring of clothes and we changed the name of the tasks to "takeworkerToTailoring". This task is modelled as sub-activity in the BPMN model and it looks unusual.

- In the BPMN model, we also have a problem identifying of the number of process participants. This process was modelled with seven participants but we can also model just having five participants and the participants represented by lanes but in this case, they should never communicate with each other. We may also use swimlanes to split the behaviour into three unrelated parallel flows

- In BPMN model the need of Uniform is made in two pools (i.e., one located in the New employer pool named as "NEED UNIFORM" and the other located in the specific superior pool named as "UNIFORM"). This is done due to the fact that using the pools, the model describes the behaviour of each participant represented by a pool. Whereas we do not need to this in our model. The decision about the need of uniform can be made in a unique point. We can also recover the activities of the single participant by slicing the workflow.
- The task "workLocationUnitReport" in the immediate superior pool, creates confusion in understanding the functionality of the task. We have changed the name of this task to "reportAtLocation(W_UNIT)" since this task states that the person has to report from the location (in general we might think it as doctor or employee reporting to someone from the work location).

5 Conclusions

Throughout the thesis work, we have explored some of the important features of UML models in the field of business process modelling. Our main interest was in understanding and validating UML holistic models (UM1, UM2…UM5) over BPMN case studies. We have applied both hard and loose methods to model the BPMN case studies, and the results are clearly explained in the lessons learned section. While modelling the business process with UML models, we have discovered the ambiguities and problematic issues of BPMN models. The inconsistencies and incomplete parts of BPMN model allowed us to take a deeper inspection of the business process and using our models we have precisely solved issues of BPMN. When Comparing the understandability and readability of BPMN model over UML models, our models have higher precision level. Our UML models are modeled in a such a way that, people from both the technical and non-technical can easily understand business process regardless the type of business process. The views of each model (UM1, UM2…UM5) will allow the user to see the business process with different dimensionality while increasing the comprehensibility. The results of thesis work were produced in the table 2 (BPMN issues summary) comparing the BPMN modelling aspects with UML models giving a brief explanation why one might consider holistic models for the business process.

6 Bibliography

[1] Object Management Group. (2011). "Business Process Model and Notation (BPMN)" (version 2.0) (https://www.omg.org/spec/BPMN/2.0)

[2] Gianna Reggio. "Holistic Business Process Modelling. With UML."

[4] (https://www.businessprocessincubator.com/content/amazon-fulfillment/)

[3] Object Management Group. (2010). "BPMN 2.0 by Example" (version 2.0)

(**http://www.omg.org/spec/BPMN/2.0/examples/ZIP**)

[5] Laura Sánchez-González, Francisco Ruiz, Félix García, Mario Piattini. "BUSINESS PROCESS MODEL IMPROVEMENT BASED ON MEASUREMENT ACTIVITIES"

[6] https://app.genmymodel.com/api/projects/_LElPoNL1EeSyKtHqJeGVMw/diagrams/_LElPp9L1EeSyKtHqJeGVMw/jpeg

---