Business Process Modelling

Music Store Online

Sandro Cantasano Martino, 660874
Mohamed Arafaath Sathik Basha, 659588

Contents

1 Introduction

The scenario considered in the present final report is afferent to the process of execution of a purchase in an online Music store. The service should be available over the Internet and provide options to choose and remove goods from a shopping cart, as well as to check out and log out at every different moment of the purchase. The customer starts the process by logging in the platform and browsing the catalog offered by the store. After that, it can decide to select a song and play an excerpt in streaming. The customer, as we already defined, can add and remove items from the shopping cart.

When the cart is not empty, it can decide to check out, asking for an invoice. The checkout requires the selection of the credit card number, which is debited at the bank. Afterward, the order is saved, the songs are made available for download and the customer can log out or fill another shopping cart. One week after the transaction, an e-mail is sent to the customer to remind him/her to rate the songs.

We decided to investigate the case in two different ways, the first one afferent to a much simpler way of designing this process and the second one, which tried to put much difficulty and effort, in order to better reproduce the reality. We will show these two in the following sections.

2 The model

2.1 Model with two pools

For the graphic representation of the model, we decided to work with BPMN notation(acronym of Business Process Model Notation); this representation is realized with **bpmn.io** editor available online. The figure below shows the general composition of the first process proposed. As it is, the process is composed of two distinct pools connected with message flow between them. The main purpose of this communication is the collaboration and total compatibility between the two distinct sub processes.

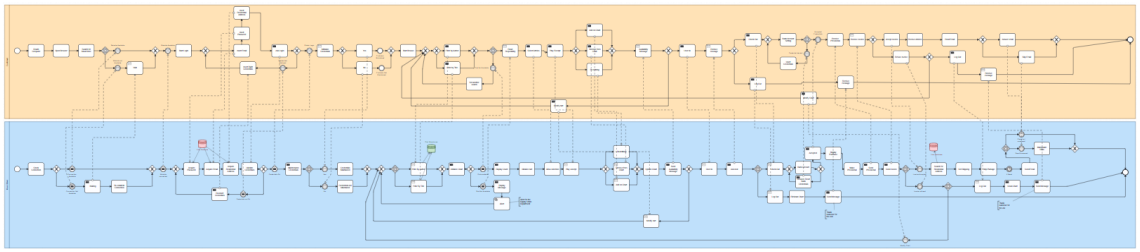

Figure 1: BPMN - Main Vision

The pools are the way we have for design the process in which the **customer** and the **Music store** work. In this way, we entail the customer as the first "input creator" of the process; in fact, the main message flow between the two pool start from the "Customer"'s pool. Talking with high granularity, the only communication arches we have between MusicStore and Customer is afferent to the check that the system performs for example on the credentials for the login, the check if the disk is available on the store or not, the check of the bank credentials. The other arches between the two pools start from the customer's pool. The process of the customer starts obviously choosing to reach a computer and search for the MusicStore.com site. At this time, the customer needs to be able to start browsing if the connection to the site is available. So, in this stage, the MusicStore's pool is aimed to check this connection which, if it is available, sends a message flow to the customer in order to inform it that the site is available and ready to be used.

In this stage we have taken a decision, the site is already online so the customer's pool doesn't start the process. We entail both processes as independent one from another. Another possible way of imagining this process might be to entail it as the customer starts searching on the internet for the site.
The second task performed in the process is the identification of the user, which asks the system to insert its own credentials(email, password). The system store the information from the customer in order to be able to validate or not the credentials. In fact, in this stage, an *Exclusive Gateway* informs the reader that in that position in the music store's pool, the system should take a decision(to validate or not the credentials). The meanwhile, for the customer is mandatory to attend the response of the system through an *Event-based gateway* which allows to pursuit a specific branch in the customer's pool only if the music store's pool takes a decision.

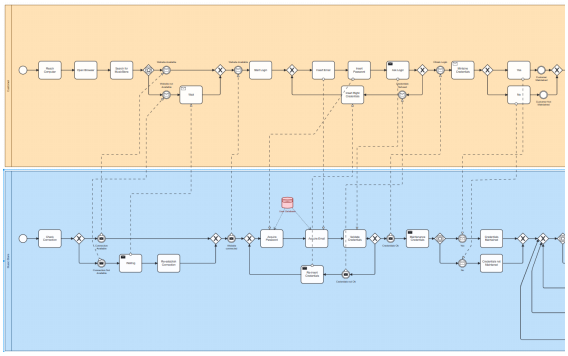

Figure 2: BPMN, first part

Helped from messages flow, the two pools aim to reach a status of loop if the credentials are repetitively wrong, or a status *Login* if the system recognizes the credentials.

The second important task consists of asking the user if he wants to maintain the credentials saved or not on the site in order to be ready for a future visit. Practically, the syntax of the Business process model notation is the same as the previous task explained, except that in this case the customer has to respond if he wants or not to be "maintained". The figure below depicts the second part of this primary process. The tasks explained in the picture are afferent to the *filtering* of the search by author or by title that the customer can perform. The system checks if the search of the disk is available on the store and informs the customer about this disponibility; if the availability check is not satisfied, the Music Store allows the customer to come back in research of the disk requested filtering again by title or author. Instead, if the disk is available the customer is allowed to add, remove o do nothing into the shopping cart.

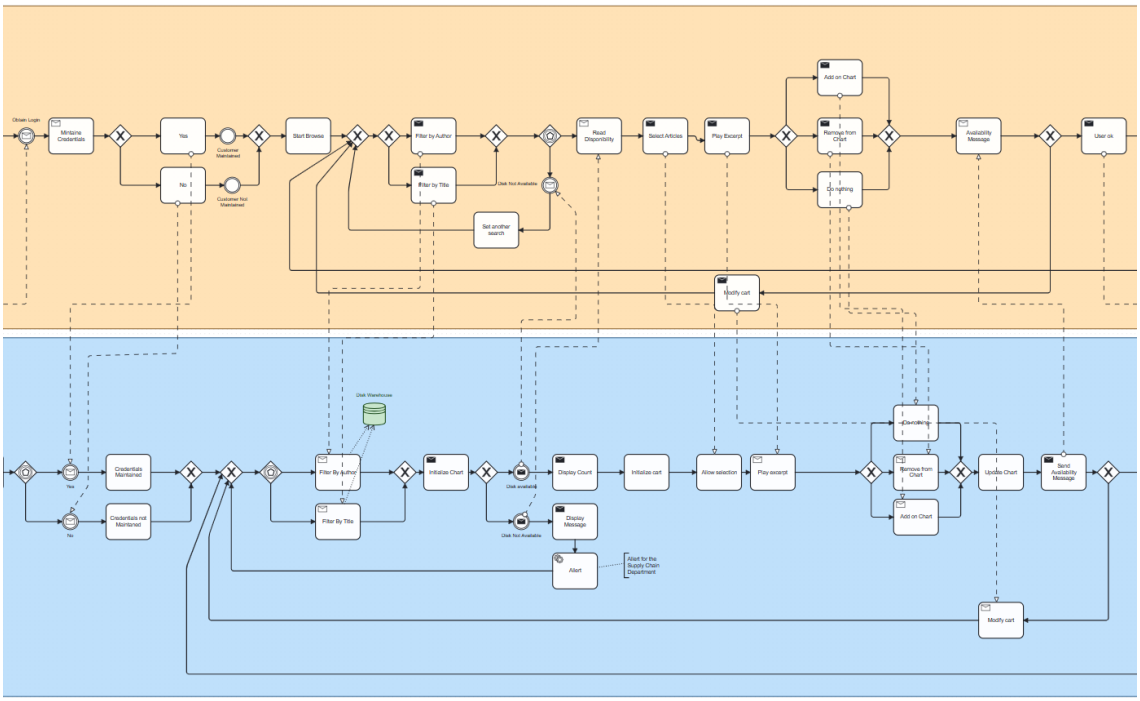

Figure 3: BPMN, second part

Another Exclusive-based gateway allows the customer to come back to the filtering task in order to modify the cart or, if the customer is sure about its selection he can go through the next task that consists of ask for a checkout or logout. Since the characteristic that this process must have is to give the possibility to the customer to modify the cart at every moment, here is possible to see the first attempt. The exclusive-based gateway allows the customer, as we already said, to check out and finalize the purchases or to log out and clean the cart. The first choice leads the user to the **Bank credentials setting**, the second choice, instead, leads to the end of the process. Coming back to the first case, the customer has to wait for the input of the Music store after the insertion of the bank credentials, this is because the Music store has to check the correctness of them and allow the customer to go on until the end of the process or not. If the credentials are wrong, the customer has to re-insert the credentials, instead, if the credentials are good the customer can go through another choice like *accept* or *refuse* the invoice the system sends to him.

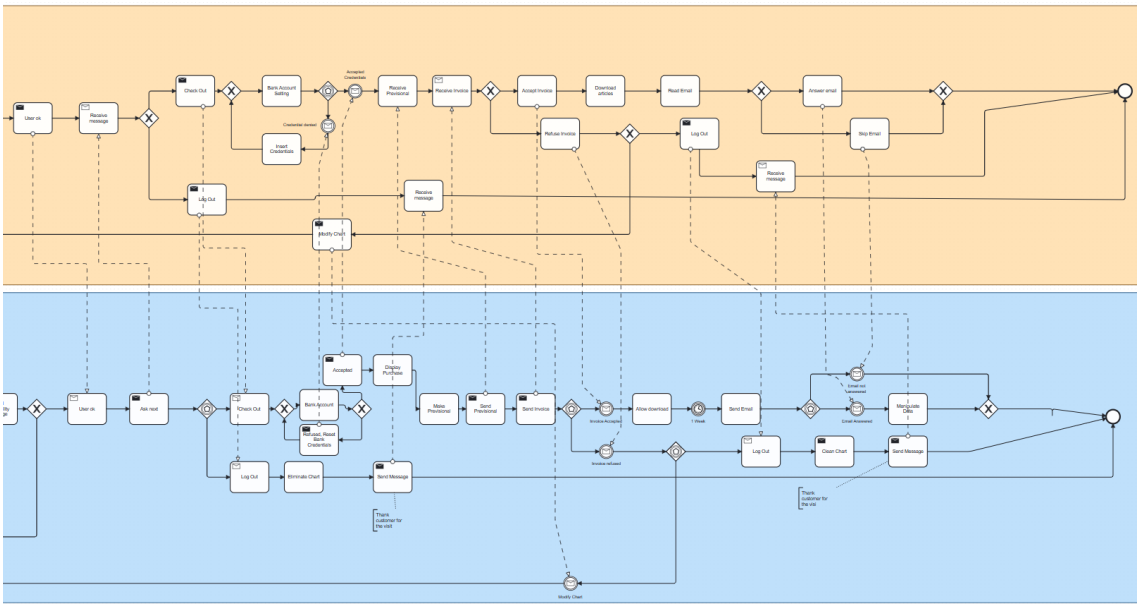

Figure 4: BPMN, second part

In the case of refusing, the second attempt to modify the cart appears in the choice of the customer otherwise he can go through a logout and finish the process. In the case in which he accepts the invoice, he can receive the download and, after one week, receive an email asking to rate the songs downloaded. In this stage, we decide to allow the user to answer or not to this email and if this first choice becomes true, the system can decide to manipulate the data obtained. The end of the process, after this activity, arises.

Instead, in the music store pool, what happens is that the pool makes a sort of provisional document of the purchase, sends this to the customer who can check it and, in concomitance with the *exclusive-based gateway*, in the music store's pool appears and *Event-based gateway* which allows the flow to go through a branch rather than another depending on the condition that will be faster activated by the customer input(condition of *Accept* or *Refuse* the invoice). These two branches create two distinct flows, that on their part, will be activable depending on the customer decisions(in fact, in both patterns there will find *Event-based gateway*, remarking the fact that this gateway appears when a pool has to wait for a decision that passively will condition him).

2.2 Variant with three pool

In this variant of the process, we evaluate the possibility of creating a pool for a shopping cart in order to create a new entity that can help the process to perform some clever task. Adding a new entity, we have tried to model the case in which the customer is on the first attempt of purchasing, so he selects an article and in this way, he might be able just to *add on cart* (obviously you can't remove articles from the cart if it is empty). In fact, the main activity that this new entity performs is to first inform the user of the state of the cart(Empty or not Empty) and after this situation, this pool communicates mainly with the central Music Store's pool.

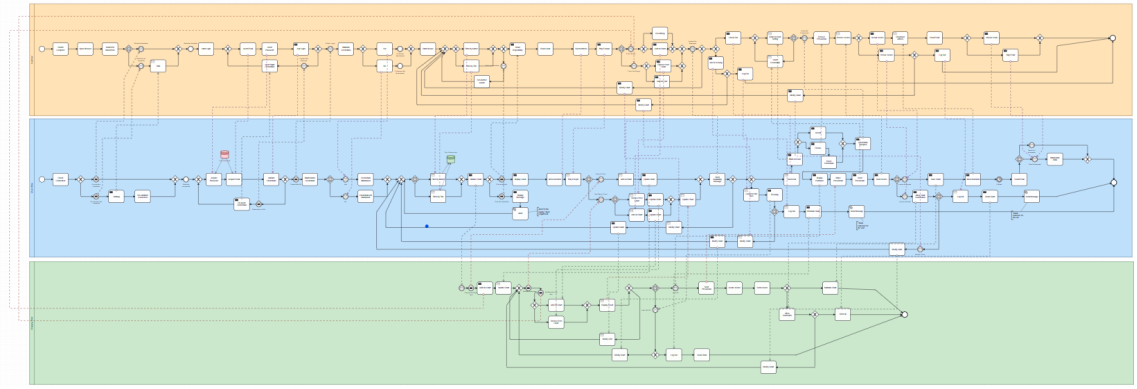

Figure 5: BPMN, variant

Going deeper in detail, the shopping cart pool is initialized from the system of the music store that launches a communication before the customer is able to filter his research on the system, this allows the pool to start. In this stage, the cart is surely empty so it can be able to inform the customer that the only activity it can perform is "adding", launching a communication flow from the cart pool to the system.

Analyzing the Music Store's pool, instead, the situation changed a little bit compared to the case exposed before. We entail a Music store pool as a step size between the customer and the cart, so it is deputed to administrate the cart according to the command proposed by the customer. For this reason, the activity in this pool becomes in much higher number compared to the precedent because the input from the customer has to be continued with an input from the Music Store to the Cart. These two are the main differences between the two models proposed until now because the remaining tasks (maintenance credentials, checking the availability of the product requested, asking for a modification of the cart, a logout or a checkout, didn't change between the past). In terms of "syntactic notation", the possibility of adding only or add and removing items from the cart if it is empty or not empty requires an *Event-based gateway* in the customer pool and in the music store pool since, inserting a new entity a *Shopping cart*, we want to entail it as a main manager of that area.

Speaking about the Shopping cart pool, we were obliged to insert some Events in order to inform both Customer and Music store about the current situation of the cart(Empty or not Empty). we followed an *Event-based gateway*, which is able to properly accommodate the decision of the Customer that, passing through the Music store, launches an information to the Shopping cart about his action. The shopping cart will display the count and in this stage will find another important decision from the customer which will ask to go through the end of the process, asking for the invoice or, in another branch, may ask to modify the cart.

If the customer decides on the second opportunity in this model the *shopping cart* will send the *Provisional* to the music store pool that, on his turn, will prepare the invoice to send to the customer; the customer,

instead, will save the order. Instead, if the customer asks to log out without any purchase(log out), the cart will perform the cleaning of the articles selected in the step before and the process will end up.

3 Petri net

The processes designed through a more abstract model like BPMN were transformed in a Petri net to verify the characteristics and the correctness. The simplest method involves the transformation of each event in a transition, each activity in a transition and each arch in a place. The gateway was transformed transforming the operators in a place with transactions connected according to how many conditions that operator was allowing, specifically in a **XOR-join** if the gateway was presenting multiple incoming arrows and one outgoing arrow it was translated with multiple transactions connecting to a place with just one outgoing arc; in a **XOR-split** if the gateway was presenting multiple outgoing arrows and one incoming arrow, the gateway was translated in one place connected to multiple transactions. There was also the necessity to add a new place from the start(a place without incoming arrow, connected to the first activity) and until the end of the network(a place without any outgoing arrow and with one incoming arrow connected to the last activity). We underline that, even talking about the translation from the BPMN to the Petri net, there are no differences between *Event-based gateway* and *Exclusive-based gateway*(seen as a split). Both were translated as discussed before. Another important thing is connected to the message flow between the different pools. Connect transition from one pool to another consists of creating a new place between the two workflow modules, connecting the transition from one pool to the transition of another one. For this purpose, the software used to transform the BPMN was WoPed.

3.0.1 Customer

The Petri net represent the process of the customer in his main process and in the variant are shown in the figure below.

Figure 6: Petri net, Customer-Main model

Figure 7: Petri net, Customer-Main model

- The net is composed of 50 places, 62 transitions and 124 arches in the main model, the second one is composed of 52 places, 65 transitions and 130 arches;
- both the net are a **workflow net** (N) and **sound** the three properties(**no dead tasks**, **option to complete** and **proper completion**) are respected;
- the nets are **S-system** because the pre-set and the post-set of every transition contains exactly one place. This allow us to find an *S-invariant* for the net, which will be of the form **I**=[k k ... k];

- the workflow nets are not a **T-system** because a lot of places have much more than one transition in their pre or post-set;
- the nets are **free-choice net** because for every single pair of transition, their pre-set is identical or disjoint;
- the nets are **well-structured** because they don't have any *PT-handles* neither *TP-handles*;
- the net is **bounded** since the net is S-system so, as a consequence of the Fundamental property of S-system the number of tokens is an invariant;in addition, the net is **live** therefore it is also **dead-lock free**;
- the nets are both **S-coverable** because it is crossed by an *S-component* composite of the entire place of the net. This was also explicable from the fact that the system is *live* and *bounded*, so for the *S-coverability theorem* it is also S-coverable;
- the nets are also **connected**(**strongly** in the N^*).

Since both the nets are *bounded*, the **reachability graph** is finite and matches with the **coverability graph**. The reachability graph is composed of 52 *vertices* and 62 *edges* for the variant model, instead for the main one we have 50 *vertices* and 58 *edges*.

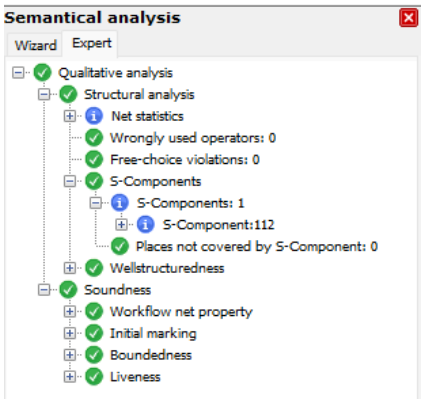

Figure 8: Semantic analysis - Customer, main model

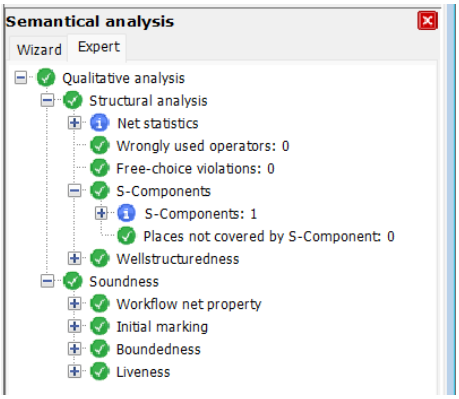

Figure 9: Semantic analysis - Customer, variant model

3.0.2 Music Store

The Petri net representing the process of the customer in his main representation is shown in the figure below.

Figure 10: Petri net, Music Store-main model

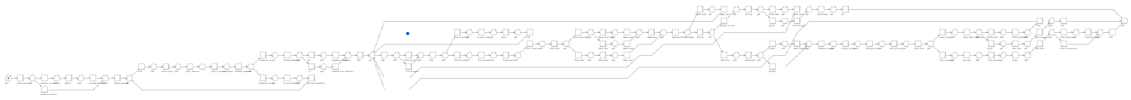

Figure 11: Petri net, Music Store-variant model

- The net is composed of 57 places, 69 transactions and 138 arches in the main model, the variant one is composed of 70 places, 83 transactions and 166 arches;
- both the net are a **workflow net** (N) and **sound** since the three properties(**no dead tasks**, **option to complete** and **proper completion**) are respected;
- the net is also **connected**(**strongly** in the N^*).
- the nets are **S-system** because the pre-set and the post-set of every transaction contain exactly one place. This allow us to find an *S-invariant* for the net, which will be of the form **I**=[k k ... k];
- the workflow nets are not a **T-system** because a lot of places have much more than one transition in their pre or post-set;
- the nets is **free-choice net** because for every single pair of transactions, their pre-set is identical or disjoint;
- the nets are **well-structured** because they don't have any *PT-handles* neither *TP-handles*;
- the net is **bounded** since the net is S-system so, as a consequence of the Fundamental property of S-system the number of tokens is an invariant; in addition, the net is **live** therefore it is also **dead-lock free**;
- the nets are **S-coverable** because it is crossed by an *S-component* composite of the entire place of the net. This was also explicable from the fact that the system is *live* and *bounded*, so for the *S-coverability theorem* it is also S-coverable;

Since both the nets are *bounded*, the **reachability graph** is finite and matches with the **coverability graph**. The reachability graph is composed of 52 *vertices* and 62 *edges* for the variant and 57 *vertices* and 66 *edges* for the main model.

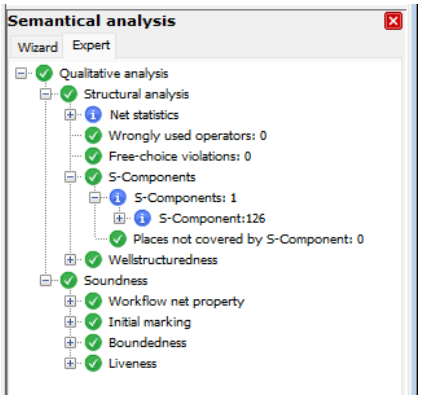

Figure 12: Semantic analysis - Music store, main model

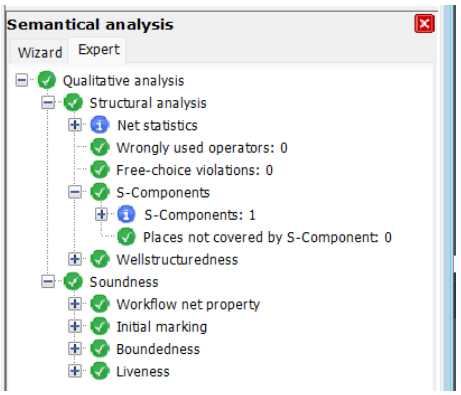

Figure 13: Semantic analysis - Music store, variant model

3.0.3 Shopping cart

The Petri net representing the process of the customer in his main representation is shown in the figure below.

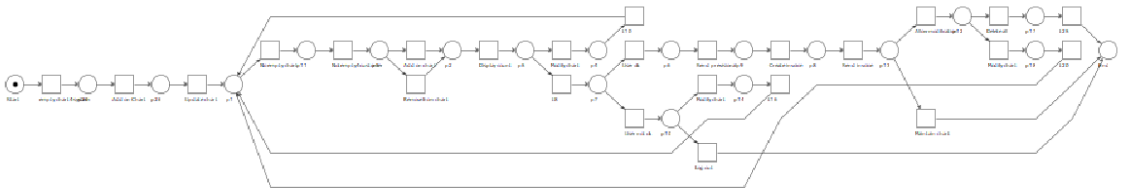

Figure 14: Petri net, Shopping cart model

- The net is composed of 20 places, 25 transactions and 50 arches;
- both the net are a **workflow net** (N) and **sound** since the three properties(**no dead tasks**, **option to complete** and **proper completion**) are respected;
- the net is also **connected**(**strongly** in the N^*).
- the net is **S-system** because the pre-set and the post-set of every transaction contain exactly one place. This allow us to find an *S-invariant* for the net, which will be of the form **I**=[k k ... k];
- the net is not a **T-system** because a lot of places have much more than one transition in their pre or post-set;
- the net is **free-choice net** because for every single pair of transactions, their pre-set is identical or disjoint;
- the net is **well-structured** because they don't have any *PT-handles* neither *TP-handles*;
- the net is **bounded** since the net is S-system so, as a consequence of the Fundamental property of S-system the number of tokens is an invariant;in addition, the net is **live** therefore it is also **dead-lock free**;

- the net is both **S-coverable** because it is crossed by an *S-component* composite of the entire place of the net. This was also explicable from the fact that the system is *live* and *bounded*, so for the *S-coverability theorem* it is also S-coverable;

Since both the nets are *bounded*, the **reachability graph** is finite and matches with the **coverability graph**. The reachability graph is composed of 20 *vertices* and 24 *edges*.

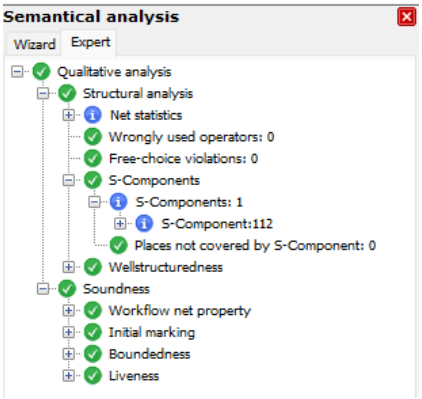

Figure 15: Semantic analysis - Shopping cart

4 Workflow system

In this section the nets of the process of the music store and customer are transformed in *workflow module*: from the music store's pool to the customer's pool, for the main model, were added 19 new incoming arches instead from the customer's pool to the music store's pool were added 21 new incoming arches.

For the variant, 15 incoming arches were added from the music store to the customer(23 incoming arches for the opposite), 2 incoming arches from the cart to the customer(0 for the opposite), 8 incoming arches from the music store to the shopping cart(3 incoming arches for the opposite).

Since the workflow modules are **compatible**, we can connect them in order to involve a *workflow system*, representing the interaction between them. For this purpose, were added a unique starting place and ending place which reconnect every single workflow module in order to represent a workflow system, with the help of an **AND-split** in the starting point and an **AND-join** between the modules and the ending place.

4.0.1 Main model

The main model is composed of 149 places and 132 transactions. Since there was no possibility to declare an **semantical analysis**, we decided to test the workflow system using the *Tokengame*. Running every single branch of our module we discovered that the system doesn't seem to have place where more than one token lies in a single marking, so we can ensure that the module is **safe**. Every transition is live(there exists the possibility of firing a transition at any time) so we can conclude the net is also **dead lock-free**. For the presence of the connections between the two modules, the system isn't **T-system** nor **S-system** so we are not able to infer anything(soundness, safeness)).

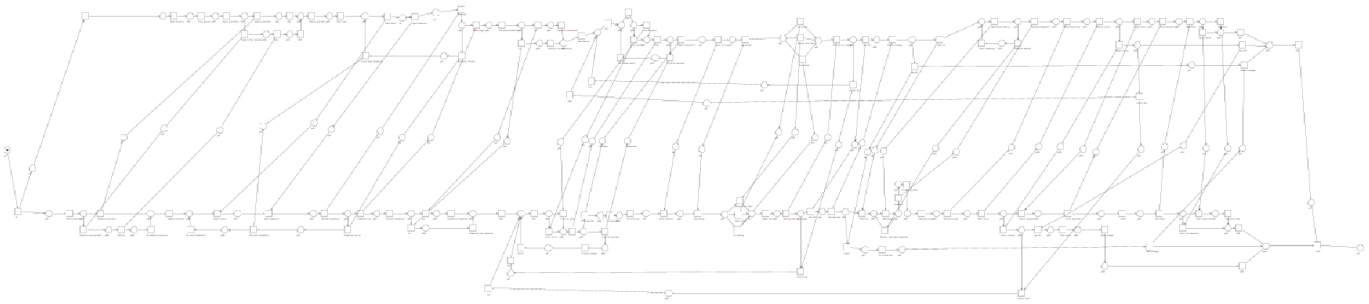

Figure 16: Workflow system, main model

The workflow system is **safe** since any place has not more than one token passing on them, for this reason, it is obviously also **bounded**; this is easily checked from the *coverability graph* which doesn't contains no omega symbol, so every place appear to be marked just one time at marking. Since the interface between the two pools connects the transitions with a place in between, the workflow system doesn't show the property of a **free-choice** because a lot of transitions have many outgoing or incoming arches. Since the property of **proper completion** and **option to complete** are reached, no tokens are left in the process if one token reaches the End and there is every time a path for reaching the end of the process, we can ensure that this workflow module is also **sound**.

4.0.2 Variant model

The variant is composed of 194 places, 174 transactions and 450 arches. Since there was no possibility of obtaining a proper response from the *semantical analysis* from WoPed nor the analysis from the plug-in *ProM*, we decided to analyze the workflow system with the help of **Tokengame** purposed by WoPed. It allows us to understand that the workflow module shows a lot of problems starting with the presence of the entity of the Shopping cart. In fact, its presence may allow us to help the customer perform the task of adding and removing of the articles in the way he is in the first or next attempt of purchases but unfortunately, this leads us to the position of not being able to manage the token in the entrance from the shopping cart to the customer since:

- if the customer is in the first attempt of purchase, the Shopping cart informs the user that the only action he can do is "add". Until now, everything appears fine. The Shopping cart's module, however, works for sending a new token which informs the customer that now the cart is "not Empty"(after the biggest *XOR-join* in the Shopping cart module). But if the customer decides to end the process by adding just one single disk and ask to check out, we may discover that a token that informs the customer about the "not Emptyveness" of the cart is still there. So for this reason we can firstly show that this process it's not **sound** since it doesn't respect the properties of **proper completion** and **option to complete**. We were not able to fix the problem properly.
- another important thing to underline is that starting from the second transition of the Shopping cart that informs the system about the fact that the cart is empty, there is no possibility to better organize the choreography between these two pools in order to better organize the possibility that also the system of the music store understand if we are in a situation of Empty cart or not Empty cart. In fact, this is true also for the communication between the Shopping cart pool and the music store pool, in the place "p108" a token remains blocked and the remaining transaction appears to be not fireable, so we discover a **dead tasks**.

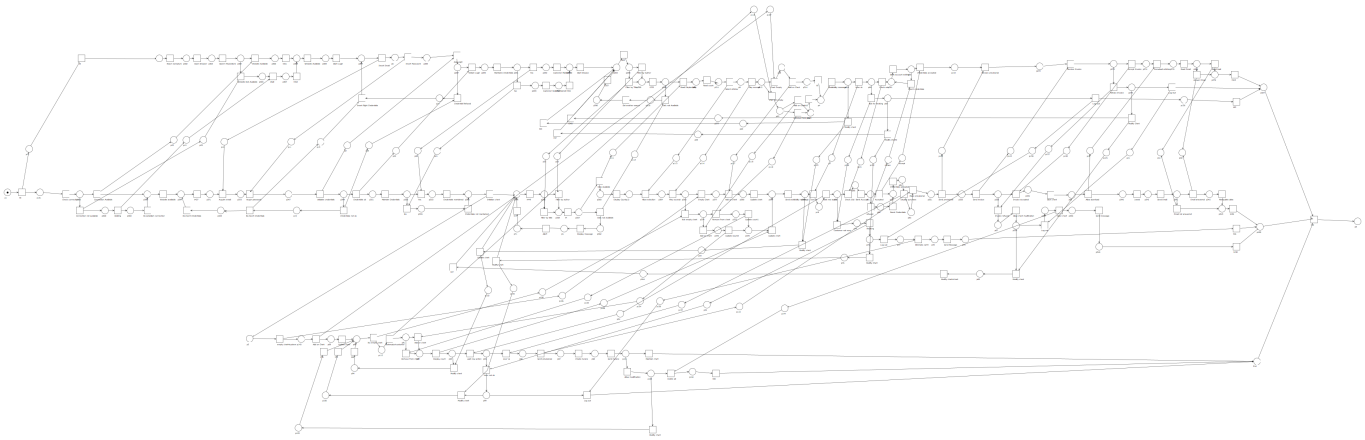

Figure 17: Workflow system, variant model

- with the help of *Tokengame* the workflow system seems to be **safe** and therefore **bounded** ;
- since there is no possibility to go through "p108", in order to fire the remaining transition after that place, those transitions are **not Live** so the entire system is **not Live**;

- since the connection between the three pools is communication between them which connects activities, that transition will have many incoming and outgoing arches from them so the workflow system lost the property of **free-choice net** since the presence of this new connection flow create conflict in the transactions;

So we believe that a better choreography may fix the problem with that workflow system. However, for the presence of the connections between the two modules, the system isn't **T-system** nor **S-system** so we are not able to infer anything(soundness, safeness). The **Coverability graph** appears huge(that is the motivation we don't insert in this final report, but whoever reads will be absolutely able to plot it with the proper function of WoPed) and doesn't appear finite, and this is another clue that the workflow is **not sound**.