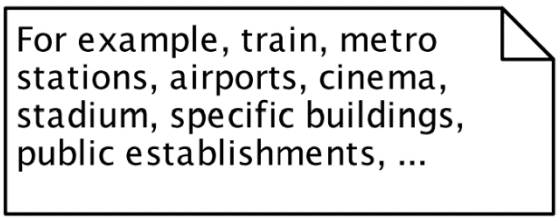
Pizza UML 🍕👷♂

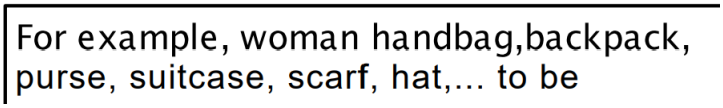
- Hijazi Hussein
- Pezzano Enrico
- Torabi Mohamad

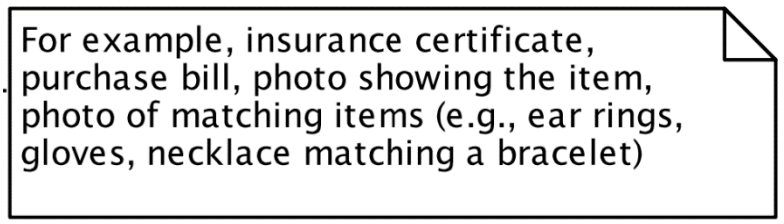
First diagram (class) - domain model

detected mistakes, ambiguous and problematic points (also alphabetic)

- There are some grammatical mistakes such as “phots” instead of “photos” or “birth date” instead of “birthDate“ and ..., which we tried to correct. Following are our founds.

1. The phots The correct grammar is photos.
2. pink roses phone cover The correct grammar is the pink rose phone covers.
3. clover leaf The correct grammar is cloverleaf.
4. red ribbon tied The correct grammar is a/the red ribbon tied.
5. obfuscation replace a part of the item The correct grammar is replaces.
6. whereas picture are always The correct grammar is pictures.
7. 

All examples have to be singular or plural. It wouldn’t be possible to use some singular and some plural.
8. 

Here we have another improper usage of formatting. Between all examples, there has to be a white space.
9. 

The correct spell is earrings. Another misspelling, necklace matching a bracelet has to be a necklace matching bracelet.

10. owner data mach, The correct spelling is match.

11. exact info are consistent It should use is instead of are.

12. birth date : Date [0..1] To define an attribute in UML, we cannot use white space between them. The correct definition is birthDate.

13. <<participant>> SecondHand Shop Classes should be named without white space.

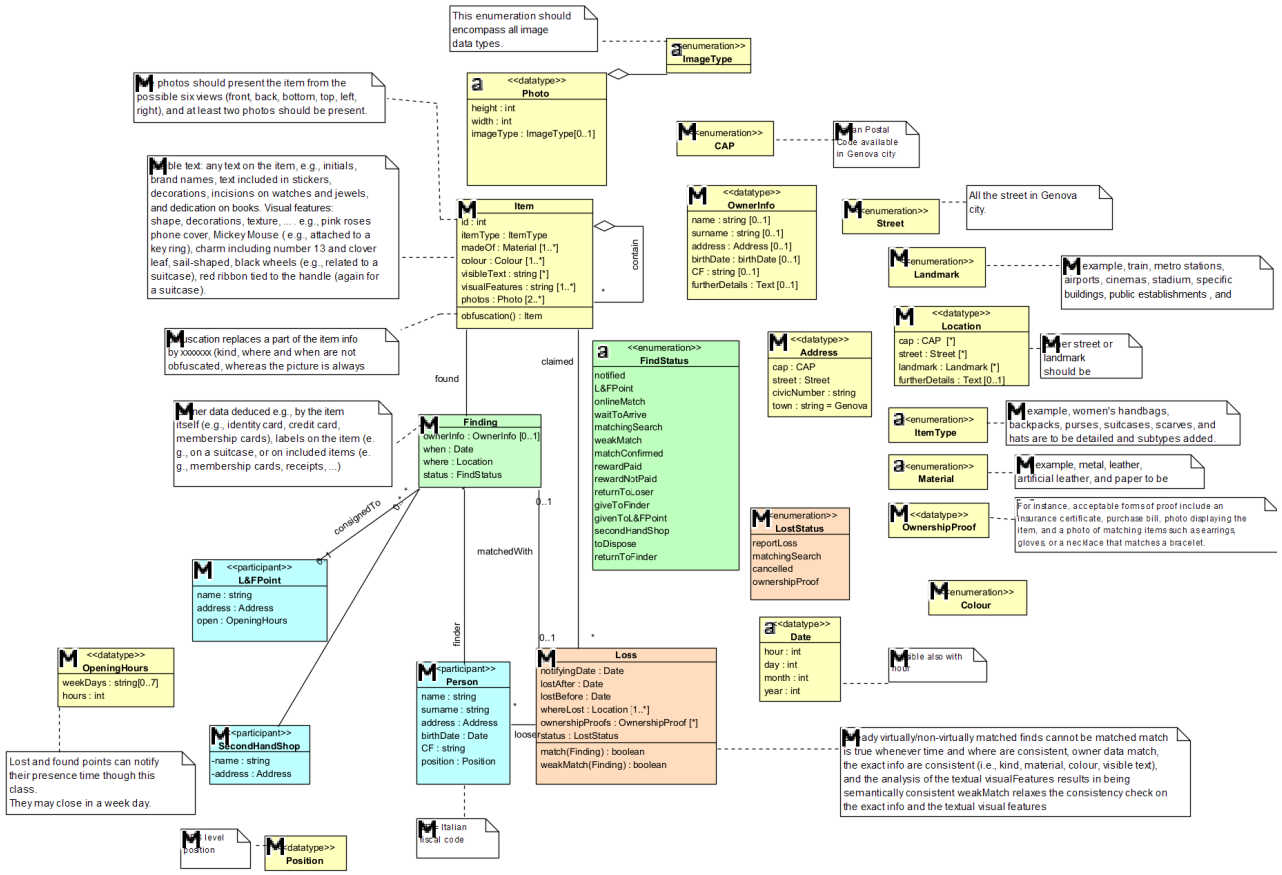
14. <<datatype>> OwenrshipProof The correct spelling is OwnershipProof.

- Ipsum
- …

suggested improvements

- Datatype related to the “Photo” object should include some attributes such as Height, Width, ImageType and BinaryContent (encoding of the image, like base64 etc); +imagetype enumeration
- Clearer explanation of cap code
- For each enum. Without a proper explanation with a note, we should add one
- Owner data object and Person class share the same attributes, so they’re redundant and we can make a User class (abstract) containing name, surname, address, and birthdate attributes for both objects, using inheritance(WE HAVE TO DISCUSS ABOUT IT!!!)
- Item
  value : int
  kind : Kind
  value and Kind could not be a proper name and may cause misunderstanding. Id instead of value and type instead of kind.
- colors : Colour It is better to use one spelling color or colour because it may cause misunderstanding.
- Photo class doesn’t have any attribute. It has to at least have width and height.
- We are in Genova city, so we don’t need Town class.
- Date doesn’t day, month and year. To know the exact date we need these parameters.
- In the Person class, the birth date is defined in an incorrect way. An attribute cannot consist a white space between.

proposal for making more precise parts of the model either unmodelled or modelled just by using natural language



- We propose renaming certain attributes in the Item class. Change value to id, because value may cause misunderstanding. The id is the exact meaning of counting items. Also, kind attribute was changed to itemType.
- Our goal is to improve the quality of the notes and fix any grammar mistakes. We found many grammatical mistakes and ambiguous notes which pointed out some of them in the mistake section. We improved all the notes
- In the Item class, we do not need a separate class for photo and contentPhoto. We changed the number of possible collecting photos to infinite to save all photos in the photo attribute.
- In the OpeningHours class diagram, we added weekDays and hours to the OpenningHours class and clarified using a note field that L&FPoints can close on weekdays. Also made it clear using a note field.
- SecondHandShop's attribute list now includes name and address fields. In previous diagram it didn’t contain any special information.
- We create an association between Item class and secondHandShop because second hand shops contribute to buying or selling items.
- We removed the Town class and changed the multiplicity of the town attribute to 1 because there is only one Town, Genova.
- Add a note for Street to describe how it works.
- We changed the name of the attribute ownerDate in the Finding class to ownerInfo because it is more clear and meaningful.

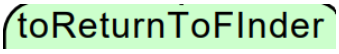
The format of the various sections is free; you may use text, and commented fragments of the provided documents, and you may generate new images using the provided Visual Paradigm file

Secondo diagram (state) - finding and loss life

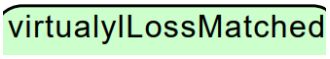
DETECTED MISTAKES, AMBIGUOUS AND PROBLEMATIC POINTS (ALSO ALPHABETIC)

- There are some grammatical mistakes such as “atAL&Fpoint” instead of “atL&Fpoint”, “toReturnFinder” instead of “toReturnFinder“ and ..., which we tried to correct. We attached the findings here.

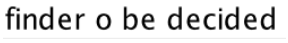
The correct spelling is atL&Fpoint.



The correct spelling is toReturnToFinder.



The correct spelling is virtuallyLossMatched.

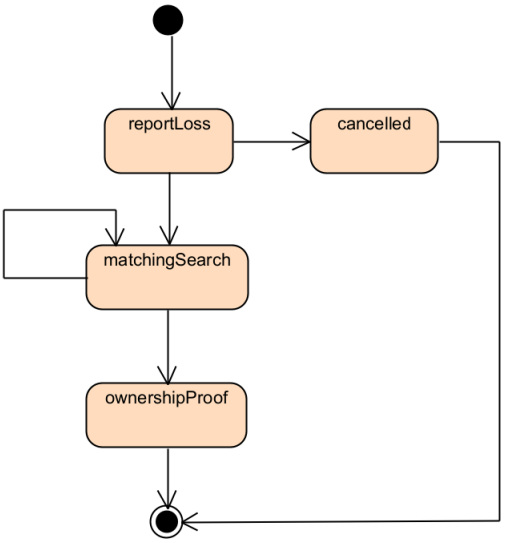


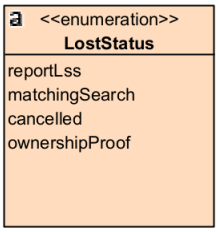
The correct spell is finder or/to be decided.

- The chosen names of some states are not proper or may be too long sometimes e.g. virtuallyLossMatched and lossMatchConfrim so improved it and made a better collocation of names.
- The two mains notes were not clear so we made it somehow more clear.
- In the matched state, we suppose it doesn’t need to start from the submitted step because we report our loss in the submitted step one time.

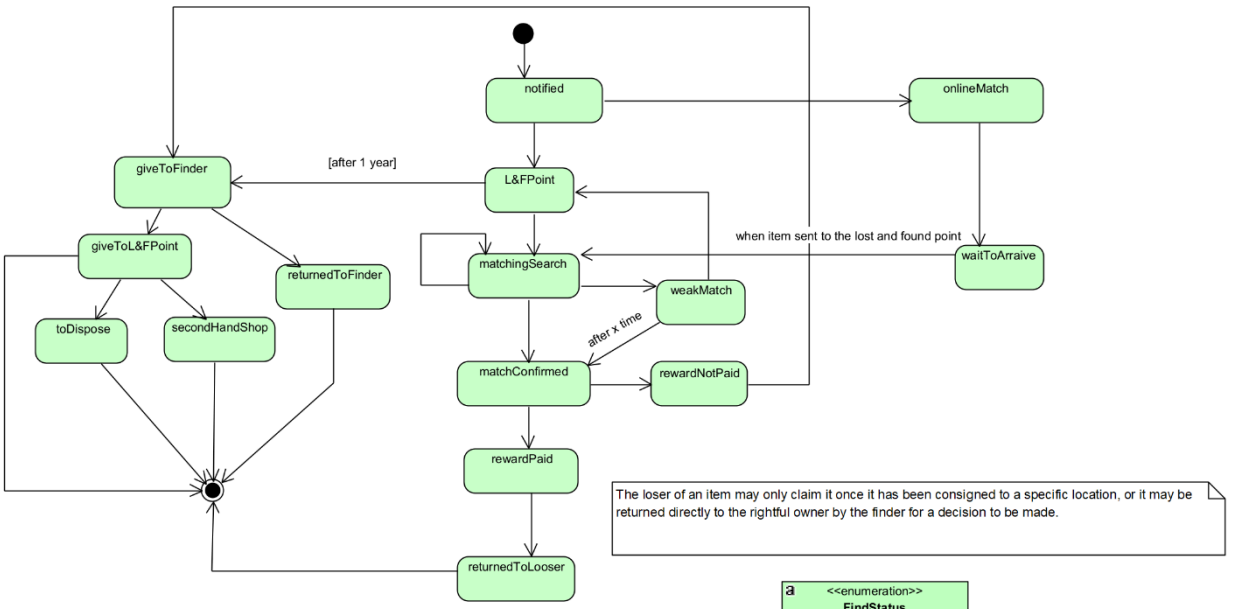
suggested improvements

- It would be possible to choose more proper names for states. We tried to choose appropriate names for states in both diagrams. In the following picture, you can observe the name change.

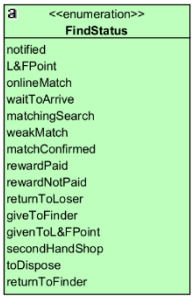




We have two options for dealing with losses. The first option is for the person who lost something to search for it. If they find a match, they get their item back. If they don't find a match, they won't get their item back. The second option is for the person who lost something to register their search and be notified if a matching item is found. Depending on the option chosen, the Lost item may or may not be returned to its owner.



The loser of an item may only claim it once it has been consigned to a specific location, or it may be returned directly to the rightful owner by the finder for a decision to be made.



- In Lstatus, we could apply a self-transaction on the matched state instead of the submitted transaction to rematch items which are weak matches or don't confirm.

- In the Fstatus diagram, lossMatched couldn’t transact to the final point directly so we supposed it was a wrong transaction and deleted it.
- In toReturnToFinder state, we considered writing a state twice may cause misdirection. So we removed one of the duplicate items.
- We considered it is up to the finder to keep the item or give it to the L&FPoint. But Point should decide to dispose or give it to the second-hand shop.
- Search for matching happens in the matchingSearch state until an item finds a match/weak match so we fixed it.
- We add a state for weak matches. If a weak match happens, the loser should wait for X time because someone else proves it belongs to their.
- We add a position in which if the item does not arrive in L&FPoint, wait until it arrives.

Third diagram (activity) - dispose unclaimed item

DETECTED MISTAKES, AMBIGUOUS AND PROBLEMATIC POINTS (ALSO ALPHABETIC)

- More than one flow from the initial node ...
- The else branch mustn't be there, there's not decision node
- "SECOND_HANDMARKE" instead of "SECOND_HAND_MARKET" found in two actions, as a grammar error
- "SECOND_HAND_MARKET" … "Market" does not appear at all as a partecipant in the class diagram, so it's ambiguos
- "SECOND_HAND_SHOP" should be written "SecondHand Shop" just like in the class diagram
- "FIND.finder" is not coherent with the class diagram
- "FIND.item" is not coherent with the class diagram
- The decision node after the proposal can't have more than two flows

SUGGESTED IMPROVEMENTS

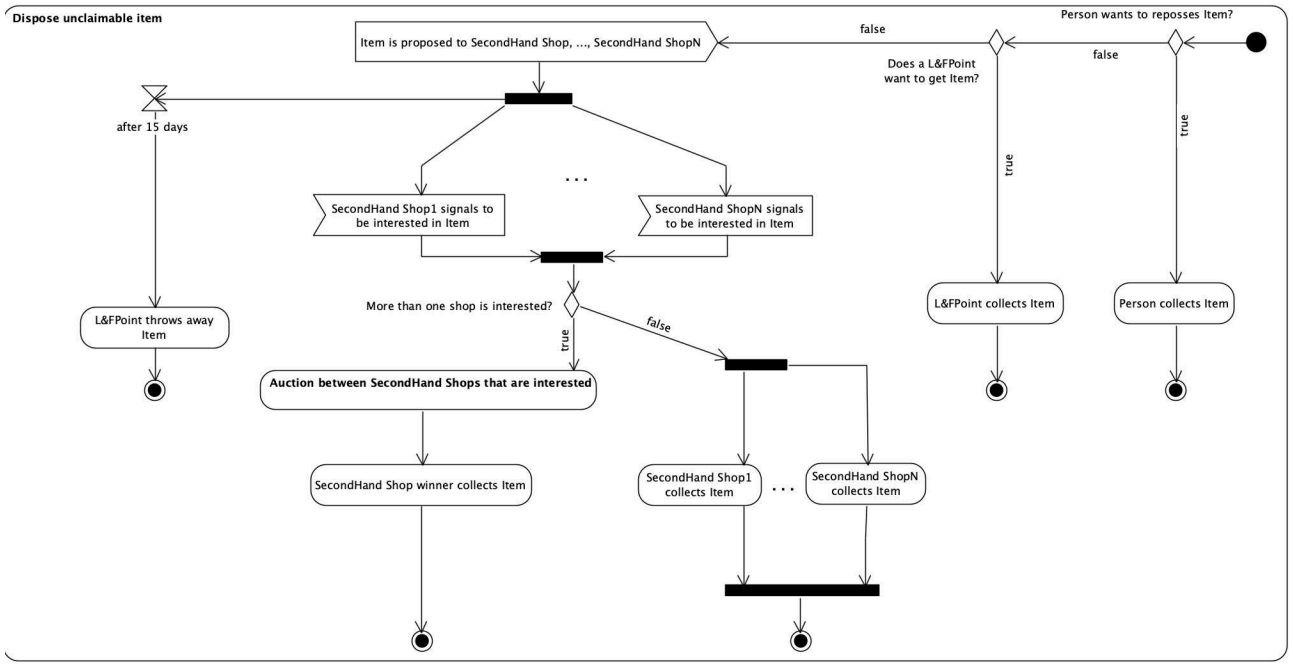
- Obviously grammar corrections
- The proposal action to the second hand market could be a "Send Signal" one
- Changing all the "MARKET" name to just "Shop" to be coherent with the class diagram partecipants and not ambiguous
- The interest signalling action (by the second hand shop) could be a "Accept Event" one
- "Person" instead of "FIND.finder" so that it's coherent with the class diagram
- "Item" instead of "FIND.item"so that it's coherent with the class diagram
- Changing all actors names to the ones of the partecipants we have on the class diagram (FIND.finder becomes just Person, Finding.Item becomes just Item)
- We could add some activity and action to represents an auction between all the shops that show interest in the item, so that only one of them (the winner) will recover the item
- A decision node after the initial node, to see if the Person wants the Item back with a relative activity final node
- If not, another decision node to see if a LFPoint wants to get the Item with a relative activity final node
- If not, a fork node (after the send signal one) for N+1 shops
- An activity final node for just the waiting
- A merge node for the N flows of shop's interest, with a relative decision node to check if there's more than one interested
- Relative activity node and relative activity final node
- If only one's interested, fork node for the N flows
- Than relative merge one with its activity final node

PROPOSAL FOR MAKING MORE PRECISE PARTS OF THE MODEL EITHER UNMODELLED OR MODELLED JUST BY USING NATURAL LANGUAGE

First of all we should correct every grammar error and improve the name of the action: so SECOND_HAND_MARKET instead of SECOND_HANDMARKE and change every name so that it's coherent with the class diagram (SecondHand Shop instead of SECOND_HAND_SHOP1 and SECOND_HAND_MARKET1). Secondly we have to remove the "else" label from the top-right flow after the initial node because without a decision node it doesn't make sense. Another grammar or ambiguos mistake is FIND.finder and FIND.item so we have to make a coherent change in just a "Person" and "Item" label.

After we can change the “proposal” and the “interest signaling” actions Send Signal Action and Accept Event Action, respectively, so that the diagram is more dynamic.

Now we can manage the multiple flows by adding a fork node after the “proposal” action and after the accept event action we have to add the relative merge node for the N flows.
At this point we have to know if more than one shop is interested, so that we would be able to manage it with a proper activity node; if not, we add another fork node (again for the N flows, one for each shop), see which one of them wants to collect the Item and after, merge them.
Of course there will be an activity final node for each main flow, so that the hypothetical application won’t deadlock itself (with only one end-node, it has to wait for every flow to finish, but in our case, obviously, only one will finish).

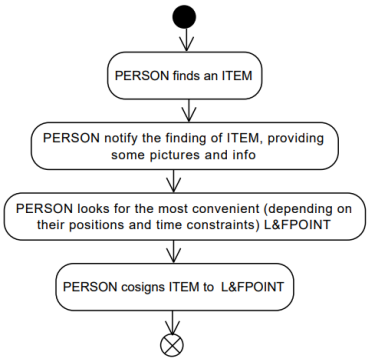


FINAL RESULTS

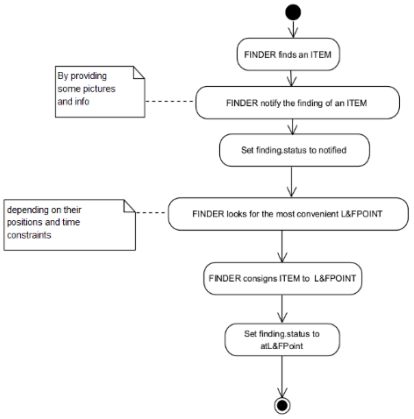
Fourth diagram (activity) - finding an item

DETECTED MISTAKES, AMBIGUOUS AND PROBLEMATIC POINTS (ALSO ALPHABETIC)

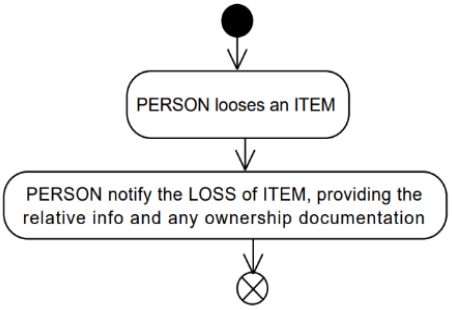
First section:



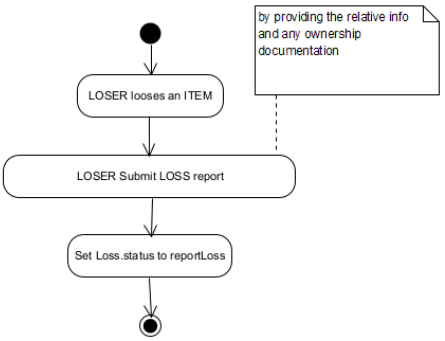
(Figure 4.1) – Original version



(Figure 4.2) – Improved version



(Figure 4.3) – original version



(Figure 4.4) – improved version

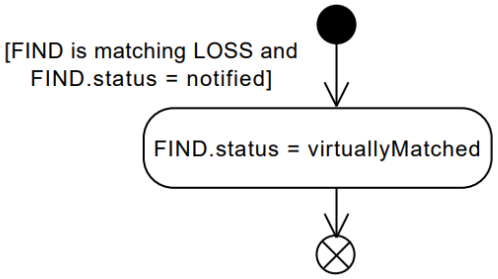
MISTAKES AND AMBIGUOUS POINTS:

- PERSON (Figure 4.1) should be changed to something more precise and for that we used the word FINDER ex. FINDER finds an ITEM figure(4.2).
- Person should be changed to more precise word, we chose the word LOSER (Figure 4.4).
- Flow Final node at the end should be changed to Activity Final Node since we are ending the entire activity not a specific flow figure(4.2) figure(4.4).
- Cosgins word in figure 4.1 is incorrect it should be changed to consigns as shown in figure 4.2

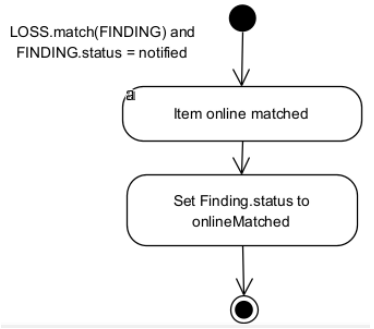
SUGGESTED IMPROVEMENTS:

- Making actions text shorter, and attached a note for extra information related to each step.
- Adding the changing of lost item status after any related step (Figure 4.2) (Figure 4.4).
- Making actions text shorter, and attached a note for extra information related to each step.

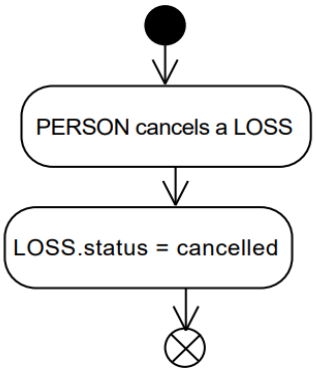
Second section:



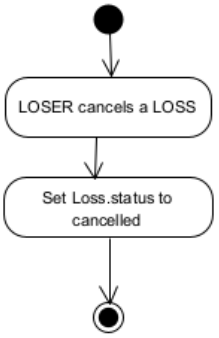
(Figure 4.5) - original diagram



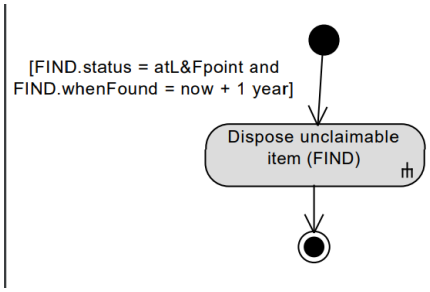
(Figure 4.6) – improved version



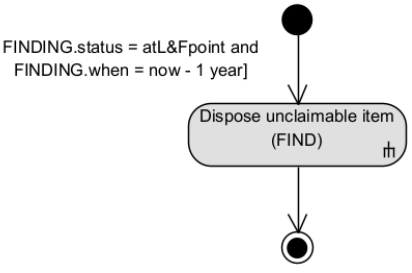
(Figure 4.7) - original diagram



(Figure 4.8) – improved version



(Figure 4.9) - original diagram



(Figure 4.10) – improved version

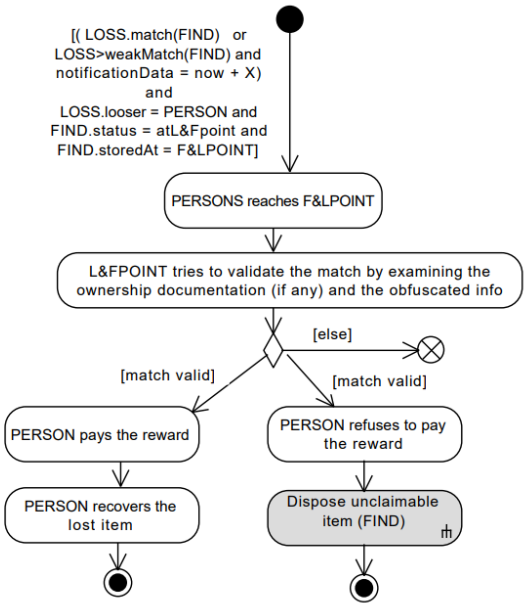
Mistakes and Ambiguous points:

- Person should be changed to more precise word, we chose the word LOSER for the person who lost an item (Figure 4.8).
- Flow Final node at the end should be changed to Activity Final Node since we are ending the entire activity not a specific flow look at figures from 4.5 to 4.8.
- In both figures 4.5 and 4.9, 'FIND' is ambiguous because we don't have any class with that name. Therefore, it should be renamed as 'FINDING' to match the one we have in the class diagram.
- The condition for starting the activity should be more precise figure (4.5) “FIND is matching LOSS” instead we write it in this way “LOSS.match(FINDING)” (Figure 4.6).
- The condition “FIND.whenFound = now + 1 year“ (figure 4.9) is mostly wrong:
    1. FIND should be changed to FINDING.
    2. ‘whenFound’ is not an attribute for ‘FINDING’, Therefore, it should be changed to ‘when’ to match the one we have in the class diagram.
    3. To check if an item has been found for more than one year, we should subtract 1 year, not add.

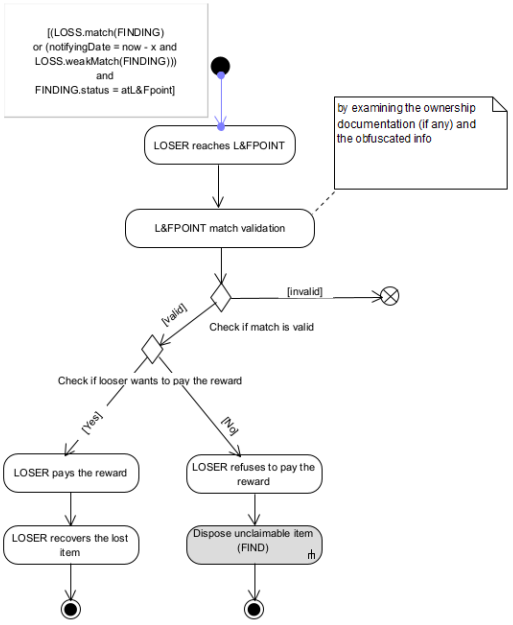
SUGGESTED IMPROVEMENTS:

- Adding the changing of lost item status after any related step (Figure 4.5) to (Figure 4.8).

Third section:



(Figure 4.11) - original diagram



(Figure 4.12) – improved version

MISTAKES AND AMBIGUOUS POINTS:

- Person should be changed to more precise word, we chose the word LOSER for the person who lost an item (Figure 4.12).
- The condition for entering the activity has major mistakes:
    1. ‘FIND’ should be change to FINDING to match the class name.
    2. ‘LOSS>weakMatch’ should be ‘LOSS.weakMatch’.
    3. ‘LOSS.loser = PERSON’ is unecessary condition since the loser is always gonna be a person.
    4. ‘FIND.storedAt’ should be removed since we don’t have any attribute to track the storing status.
    5. ‘notificationData = now – x’ in this condition. We don't have to add 'now' to 'x' because we want to know the past value. Therefore, it should be a subtraction, not an addition.

SUGGESTED IMPROVEMENTS:

- Making actions text shorter, and attached a note for extra information related to each step (Figure 4.12).
- The decision node has a duplicated condition '[match valid]' – one path leads to the scenario where the 'LOSER' refuses to pay the reward, and the other leads to the scenario where the 'LOSER' pays the reward. However, for a clearer diagram, it should contain two decision nodes: one that checks if the match is valid with conditions 'valid' and 'invalid' (Figure 4.12), and the second that checks if the person wants to pay the reward with the conditions 'Yes' and 'No' (Figure 4.12).

Fifth diagram (activity) - DomainArchitecture

DETECTED MISTAKES, AMBIGUOUS AND PROBLEMATIC POINTS (ALSO ALPHABETIC)

- The multiplicity of “Person” could be wrong in some cases

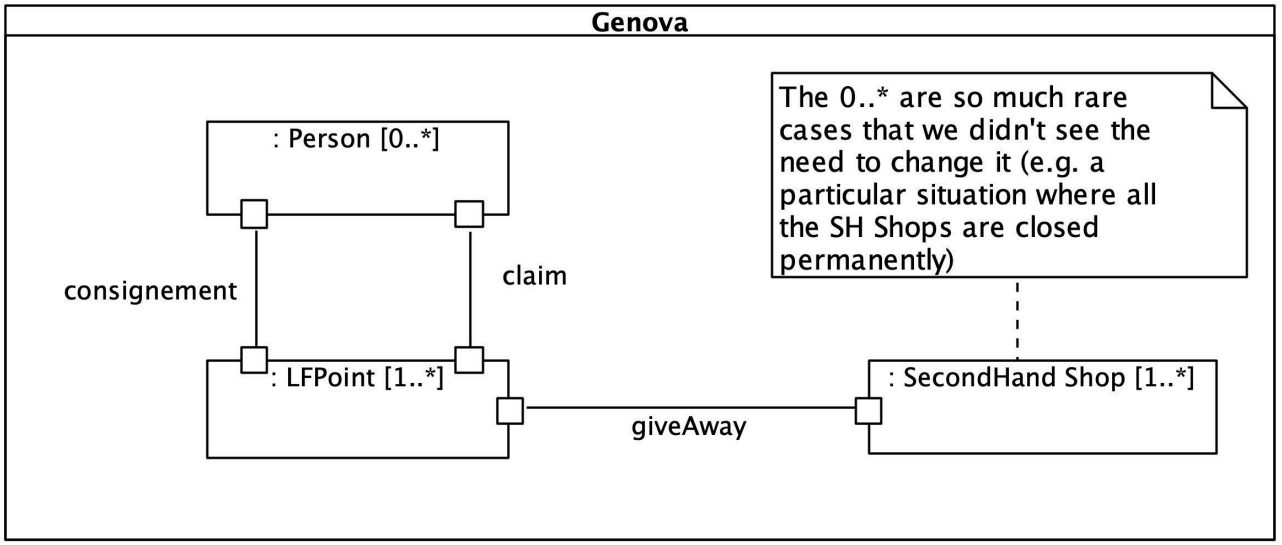
SUGGESTED IMPROVEMENTS

- Change “1 to many” of Person to “0 to many”

PROPOSAL FOR MAKING MORE PRECISE PARTS OF THE MODEL EITHER UNMODELLED OR MODELLED JUST BY USING NATURAL LANGUAGE

In particular in two cases, the multiplicity of Person (1..*) is wrong:

1. If we’re at the beginning of the L&F carrier, we obviously don’t have any lost Item or Person that we think could be the owner of something
2. If we would be able to bring all the lost Items to someone (either the owner, the finder or a shop), we obviously wouldn’t have any object to dispose to anyone.



FINAL RESULTS