# Process Description

This section describes the process used to identify the classes and responsibilities of the VisKo system. It also describes the techniques, heuristics, assumptions and constraints that were made during the process.

## Candidate Classes Process

This section describes the process Leaf Development used to identify candidate classes.

The process used to identify classes in our system is as follows:

* The team assigned a group in charge of identifying the objects/classes
* The group consisted of three members: Analyst, Designer and V&V
* As a group they divided the software requirements in three equal parts, in order to go through all the requirements: each individual was in charge of revising 97 requirements
* Each member read their assigned requirements, and looked for noun phrases
* Each noun phrase was listed in the section **Noun Phrase** of the Table 1
* Once all nouns were identified, these were separate into obvious classes, uncertain candidates, and discarded candidates
* After two days of revision, the three members combined all the noun phrases together
* The noun phrases were ordered in alphabetical order, to delete any noun phrase that was repeated
* Once all the noun phrases were put together in alphabetical order, the three group members refined a list of candidate classes by following a Guideline:
* Model physical objects – e.g., disks, printers, sensors.
* Model conceptual objects – e.g., windows, files, shots, picks.
* Choose one word for one concept – what does it mean within the domain?
* Be wary of adjectives – does it really signal a separate class
* The members justified each candidate class:
* Why it was obvious?
* Why it was unsure?
* Why it was discarded?
* Finally the members named all the obvious classes, and ordered them in alphabetical order

## Candidate Responsibilities Process

This section describes the process Leaf Development used to identify candidate responsibilities.

Mario…

## CRC Cards

This section describes the process Leaf Development used to create the CRC cards.

The process to create the CRC cards is as follows:

* One team member (V&V) wrote each class into its own individual card
* All the cards were placed into a Google Document, in order to facilitate the access of all members to the cards
* All the members gave descriptions to each class
* In order to assign responsibilities to the cards, the responsibilities were divided into six equal parts
* Each member had to assign their assigned responsibilities to the most appropriate class, by using the heuristics described in section 2.4.2
* Once all team members had assigned all their responsibilities to the appropriate class, the team checked if all classes were described, and if all had responsibilities
* If a class did not have a responsibility, the team discussed why this happen, and assigned a responsibility that fitted the class
* If there were some responsibilities that did not fit to any of the classes, a new class was created for those responsibilities
* The class was given a name that best described the responsibilities it had
* Finally, the Designer reviewed the cards and approved them

## Techniques, Heuristics, Assumptions and Constraints

This section describes the techniques, heuristics, assumptions and constrains that were used/made during the process of identifying classes, responsibilities, and creating the CRC cards.

### Techn­­iques

The techniques used in the processes to identify responsibilities, classes, and create CRC cards have been explained in the above sections, as well as the sections below.

### Heuristics

Assigning responsibilities to classes is not an easy process; therefore our team used the following heuristics discussed in class:

*H2: State responsibilities as generally as possible.*

* The concept of finding common responsibilities that can be shared, to make responsibilities reusable.

*H3: Keep behavior with related information.*

* If information changes, no update messages must be sent between classes [1].
* The class that needs to know that something has changed will know it**.**
* In sum, this makes responsibilities efficient.

*H4: Keep information about one thing in one place.*

* The concept of keeping information about one class in one place (i.e. class User, responsibility, createUser)
* Distribution of information often leads to a duplication, and thus inconsistency, it makes responsibilities safe [1].

*H5: Share responsibilities.*

* The concept of a certain responsibility or a compound responsibility can often be best divided or shared among a few closely related classes, this make responsibilities simple [1].

### Assumptions and Constraints

Mario, Hector or Manuel…