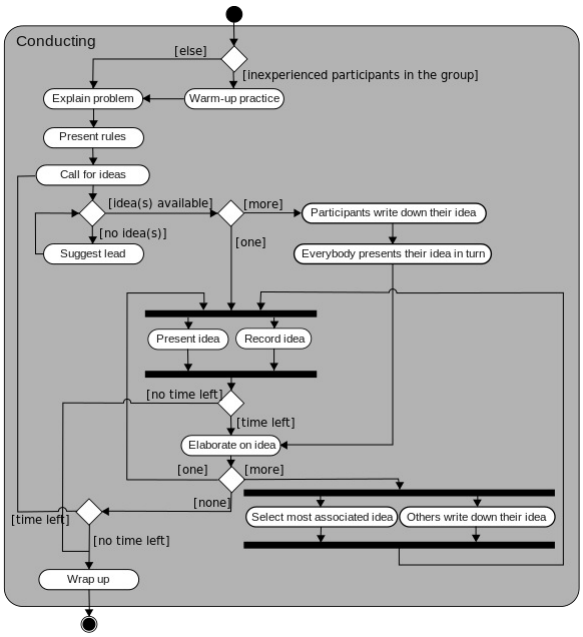

Restaurant Activity Diagram.

Sequence Diagram

Once we are done drawing our Activity diagrams, the next step of refinement is the Sequence diagram. In this diagram we list the actors or objects horizontally and then we depict the messages going back and forth between the objects by horizontal lines. Time is always progressing downwards in this diagram.

File:Restaurant Sequence Diagram.png Restaurant Sequence Diagram.

The Sequence diagram is a very important step in what is called the process of object-oriented analysis and design. This diagram is so important, because on the one hand it identifies our objects/classes and on the other hand it also gives us the methods for each of those classes, because each message turns into a method. Sequence diagrams can become very large, since they basically describe the whole program. Make sure, you cover every path in your Sequence diagrams, but try to avoid unnecessary repetition. Managers will most likely not understand Sequence diagrams.

Collaboration Diagram

The Collaboration diagram is an intermediate step to get us from the Sequence diagram to the Class diagram. It is similar to the Sequence diagram, but it has a different layout. Instead of worrying about the timeline, we worry about the interactions between the objects. Each object is represented by a box, and interactions between the objects are shown by arrows.

File:Restaurant Collaboration Diagram.png Restaurant Collaboration Diagram.

This diagram shows the responsibility of objects. If an object has too much responsibility, meaning there are too many lines going in and out of a box, probably something is wrong in your design. Usually you would want to split the box into two or more smaller boxes. At this stage in your design, this can still be done easily. Try to do that once you started coding, or even later, it will become a nightmare.

Class Diagram

For us as software engineers, at least the object-oriented kind, the Class diagram is the most important one. A Class diagram consists of *classes* and lines between them. The classes themselves are drawn as boxes, having two compartments, one for *methods* and one for *attributes*.

File:Restaurant Class Diagram.png Restaurant Class Diagram.

You start with the Collaboration diagram, and the first thing you do is take all the boxes and call them classes now. Next, instead of having many lines going between the objects, you replace them by one line. But for every line you remove, you must add a method entry to the class's method compartment. So at this stage the Class diagram looks quite similar to the Collaboration diagram.

The things that make a Class diagram different are the attributes and the fact that there is not only one type of line, but several different kinds. As for the attributes, you must look at each class carefully and decide which variables are needed for this class to function. If the class is merely a data container this is easy, if the class does some more complicated things, this may not be so easy.

As for the lines, we call them relationships between the objects, and basically there are three major kinds:

- the association (*has a*): a static relationship, usually one class is attribute of another class, or one class uses another class
- the aggregation (*consists of*): for instance an order consists of order details