Group 5: Sibrecht Bouwstra, Sander Kouwenberg, Bart-Jan van Putten, Jos Verbeek, Alice Verdonk

Feedback on UML specification of the other team

Use Case Diagram

- pinch animal's eyes: you also need to get the animal first, just like with play and feed, so why not change the name of 'take care of pet' to 'lure pet'?

Class Diagram

- incorrect use of multiple inheritance: pet is crocodile and prawn at the same time
- pet inherits animal from two sides (repeated inheritance = incorrect)
- solution: an animal is a croc or a prawn (inheritance) and an animal *can* be a pet (composition)
- Both male and female animals have a property givebirth() but in real life this is not possible

Activity Diagrams

- written in such a short form that we had to guess the meaning, e.g. intention = not OK does this mean a bad intension or no intension?
- pinch: this diagram is incomplete, the animal starts screaming right away
- the use case diagram, description, and activity diagrams are not consistent

Sequence Diagram

- nice explanations in the sidebar
- it describes only one possible scenario, probably there should be multiple simpler diagrams, thereby more scenarios are possible (in this case mating of crocs can only take place after an owner has killed a prawn

State Machines

- information overload (many states, many transitions, cluttered view)
- modelled as an activity diagram

Feedback on our own specification

We have realized that there were some minor inconsistencies within the activity diagram (play with animal), some activities were perhaps not necessary and some activities didn't have a complete ending. For example the elephant would steal the Rolo but would not be able to eat it.
The state diagram didn't use the proper naming; it was more an activity diagram and it didn't encompass states. For us as a group it was fairly difficult to create the sequence diagram because we were not sure which operations and messages should be included, because in our case the elephant was fairly passive, the person would invoke an operation and the elephant would send back what he would do, but he would not ask another question from the person making this a fairly one-way system.
We have also realized that communication is one of the key aspects for this module, all diagrams have to be consistent, and in order for this to be the case we needed to update the diagrams while we were adding changes. One of the problems of working as a fairly large group was, that it was difficult to divide the tasks because all diagrams were needed in a specific order, we could start with the next diagram without having the first one ready.Our collaboration diagram was pretty obsolete, because the interaction between classes was already explained in our sequence diagram.

Feedback on module

This module provided us with a method to help us structure a program/interactive system, and it also showed us how to use object oriented programming for a project. The module was pretty well balanced, we have received a lot of information and were also given the time to experiment with the UML standard. However, it might have been better if we would have received assignments in between lectures, so we could have practised small parts of for example an activity diagram. If you see the whole activity diagram in the beginning it might seem too complicated while it is actually fairly simple to understand if you get an explanation about the notation. It would be easier to understand if for example the activity diagram would be built up in small steps. The acting out of the other group's specification was a nice way to end the module, and made it easier for us to understand where the flaws were in the system. This acting could also be a good way to remove bugs from the diagrams earlier on in the module.
If we were to do this module again, we would probably require a better planning, because now we didn't have enough time to really read through our diagrams to debug these and make them consistent.