

## Module feedback on Formal Software specification

Stefania Nicolosi  
Esra Günlü  
Jeroen Berk  
Ruud-Peter Lemmens

Feedback on the specification of the other team.

The system itself, with the underlying components seemed logical, A pet store, a pet shop to buy food and toys for the pet, the pet itself, and a personal inventory, to which one had access by means of a personal login.

However as soon as we started acting out, we found two main problems.

- What is the exact task of a specific component?
- What component communicates to the user, and what component communicates with another component?

Especially in the shop situation, it wasn't always obvious what happened. The login was clearly dealt with by the system, and the system provides a few options, such as feed the pet (playing with the pet was defined as another possibility, not clearly defined that it can be accessed from the main menu).

The login procedure on itself was well defined, however we wonder were this component like "pet net" is defined in relation to the zoo. Is it a sub-system, a logon/ validation component and how is my data stored. We weren't sure if the system, or the inventory system would check and manage our buy.

Furthermore, there were some problems with the consistency in the system, we tried to stick to one of the diagrams, to act it out, and else it would have become a conflicting system, which we couldn't act out. Main problem here was the definition of "Interface" and "Main menu", were we optioned that "main menu" actually is a snap shot state of the interface menu.

Feedback on the acting out team of our personal system.

This proved to be a very useful completion. However the results can be rather ambiguous. We found very specific flaws in our system, resulting in a parent animal "transforming" into a parent human, because of an error in naming. It also made us aware that some components in the system remain active, although the "state" would be dead. Here we made an error in our system, resulting in an animal running away, after its dead

## Conclusion on the module

The module was useful in the sense we learned beyond the fact of "just another language for your portfolio". Of course, we made errors in our system, but the module also dealt with interpretation. This isn't only a key concern during the definition of a system, but in a broader field of the designer: Is my design so well-defined that it's not ambiguous or open for misinterpretation? Does it convey the message I want my design to tell?

Some practical remarks:

Perhaps it would be worth to work a little bit on the presentation. We sometimes were overwhelmed by the dimensions of the schematics and the "jumps" towards the next slide

were rather big. Perhaps it would have been a nice setup to start out with a simple case and start building up the UML specification. A very limited system, which grows in complexity over the days