Visual inspection of a feature-based context-oriented system

Kim Mens and Benoît Duhoux

December 2020

1 Approach of the lab sessions

Since the beginning of these lab sessions, we have been exploring how to design smart or context-aware software systems using context and feature modelling. We proposed a specific development methodology to help design such systems. Then we introduced our feature-based context-oriented programming language. Nevertheless you have probably observed that developing such systems still remains complex due to the high dynamic nature of such systems and the combinatorial explosion of contexts and features. To help developers visualise which features get activated and how they adapt the system, but also to get an overview on what contexts and features are activated or not, we propose two different visualisation tools: the FEATURE VISUALISER and the CONTEXT FEATURE MODEL VISUALISER.

In this lab session, we will introduce our visualisation tools to help you understand better how your system is adapted according to the contexts. After a demonstration of our visualisation tools, we will ask you to discover how to use the tools to inspect a smart *e-Shop* toy example. Then you will use them again, but now on your own implemented case study. We will also ask you to find and explain two bugs that we had purposefully introduced in an extension of our smart *e-Shop* example. Finally we will ask you your feedback on the usability and usefulness of such visualisation tools from a developer's point of view when developing a feature-based context-oriented program.

2 Using our visualisation tools

We have already explained how we can run a feature-based context-oriented application for which we can interact with a *Context Simulator* command line tool to simulate at run-time changes of the surrounding environment. Now we will explain how we can use our visualisation tools.

First, you have to launch a server with the option '*-a*' by running the command `ruby server.rb -a` in the folder `tools/server`. It launches the server and opens the communication ports of all the existing tools: CONTEXT SIMULATOR, FEATURE VISUALISER and CONTEXT FEATURE MODEL VISUALISER. If you want to use only a specific tool, as previously for the CONTEXT SIMULATOR command line tool, you have to use the specific option '*FeatureVisualiser*' or '*ContextFeatureModel*' to open the communication ports for the FEATURE VISUALISER tool or the CONTEXT FEATURE MODEL VISUALISER tool, respectively.

Then, you have to run the CONTEXT SIMULATOR tool in a terminal in the folder `tools/context_simulator` or open in your browser the `tools/feature_visualiser/feature_visualiser.html` web application or the `tools/feature_model_visualiser/context_feature_model.html` web application to start the FEATURE VISUALISER tool or the CONTEXT FEATURE MODEL VISUALISER tool, respectively.

Finally you must run your application with `ruby main_script.rb -d` in your application folder. Note that you need to specify the option '*-d*' if you want to use our visualisation tools.

If you need to visualise a new running application or you execute again our current system, please refresh the web application.

3 Getting started with our visualisation tools

This exercise will be time-boxed to 40 minutes maximum.

As a first exercise, you have to analyse the provided smart *e-Shop* system, named *smart_e-shop_v0.3.0*, in the *apps* folder of the archive *RubyCOP.zip* provided during the lab session 6. The goal of this exercise is to learn how you can use our visualisation tools.

Before starting to use the visualisation tools, can you draw which classes are adapted by which features and activated by which contexts? For this exercise, focus you only on the *Device* context, i.e. how these contexts trigger which features and how these features adapt the system. Pay attention to the order of the activation.

Once you have drawn how you think the system gets adapted, verify with the visualisation tool if your drawing was accurate. Run the server, start the FEATURE VISUALISER tool and launch the application of the smart *e-Shop* system.[1]. Analyse step-by-step how the visualisation tool displays which features adapt which classes. (Ensure you have set up the *step-by-step mode* in the visualisation tool.)

Now that you have gained a first experience with our FEATURE VISUALISER tool, we will ask you to do the same but with the CONTEXT FEATURE MODEL VISUALISER tool. Start this visualisation tool, run again the application and observe it.

[1]Reminder: to allow the application to communicate with our visualisation tools, you need run the application of your system with an option '*-d*'

After this exercise, refresh the visualisation tools and rerun the application. Analyse with your visualisation tools how the system is adapted by which features, which contexts or features are (de)activated when we change the device on which we run the application. *Activate the `Smartphone` context and deactivate the `Desktop` context at the same time.*

4 Inspecting your own implementation

This exercise will be time-boxed to 20 minutes maximum.

Inspect your implemented smart system, either your smart messenger system or your smart calculator system, using our visualisation tools to examine whether your contexts and features are well activated, but also if your features adapt the correct classes of your system. If you noticed some problems in your implemented smart system, fix it.

5 Finding bugs

We will present you two subtle issues in the implemented smart *e-Shop* application, which you should be able to explain. To be able to understand and explain these subtle problems, you will have to use our visualisation tools to better visualise the execution behaviour of the smart system and how it behaves depending on the contexts we activate.

Issue #1

Upon executing the application, an end-user sees the entire catalogue in a desktop view. Assume the end-user has a limited budget and sets up his preferences to see only products having a low price. *Activate the `LowBudget` context through the Context Simulator command line.* Because he has several running programs that he wants to monitor on his laptop, he wants to use the mobile version to save screen space. *Activate the `Smartphone` context and deactivate the `Desktop` context simultaneously.* What do you observe? Can you explain why you have this result?

Issue #2

The starting point of this question is the result of the previous question. You first need to activate the `LowBudget` context. Then you must activate the `Smartphone` context and deactivate the `Desktop` context at the same time. As you have probably observed, the application shows again the entire catalogue. During the time the end-user verifies if her profile is correctly updated, she receives a mail about a bonus from her manager for her excellent work. Happy, she modifies her profile to show all the products having a price less than 1000. *Activate the `MediumBudget` context.* Can you explain the issue you encounter?

How can you solve the problem (without upgrading the programming language and the implementation of the smart *e-Shop*)?

6 Collecting your feedback

After your experience on our visualisation tools, we will ask you to fill in a form to gather your feedback as developers of our programming approach to assess the usability and usefulness of them. We will give you the form by an announcement on Moodle. This form will have to be filled together with your teammate. Do not forget to identify with your unique anonymous identifier. Be the most accurate. We count on you!