Software Product Line Engineering

Summer 2019, Assignment 3

Bauhaus-Universität Weimar

Prof. Dr.-Ing. Norbert Siegmund
Stefan Mühlbauer, M. Sc.

Assignment issued: Friday, 10 May, 2019
Submission due: Thursday, 23 May, 2019, 23:59 CEST
Presentantion: Friday, 24 April, 2019

Task 1: Comparison of Implementation Techniques (12 marks)

(a) Explain the differences between *versions* and *variants* using a self-chosen sample software system. (2 marks)

(b) How can the following techniques be used to implement software product lines? (3 marks)

(i) version control systems (VCS), such as *git* or *Mercurial*,

(ii) build tools, such as *make* or *maven*, and

(iii) pre-processors, such as *Antenna* or *Munge*

(c) What are their respective advantages and disadvantages? Provide three examples or use cases, where each of these techniques can be reasonably used? (3 marks)

(d) If variability in a SPL is developed and maintained using either a VCS, pre-processor, or build tool, how do these approaches affect the following aspects, respectively? (4 marks)

(i) Performance (e.g., execution time or resource utilization)

(ii) Collaboration among different developers

(iii) Acquisition of third-party software components

(iv) Fine-grained extensions of the code base

Task 2: SPL Implementation with Pre-processors (10 marks)

In the last assignments you implemented a chat application with a number of features (cf. assignment 1, task 4) and created a feature model for the application (cf. assignment 2, task 4e). In this task, we will merge the existing implementation and the feature model to make the chat application configurable using a pre-processor.

Re-implement the application from previous assignment as a product line in Java and the pre-processor *Antenna*. The application should have the following features (2 marks each), which, again, can have a simplistic realization. In the first assignment not all implemented features needed to be accessible, but now, each feature can be part of a product variant. Therefore, features should interact smoothly and *all* valid configurations must compile and run without errors (test cases are not required).

a) Color (can be shown as text, e.g., `<blue>text</blue>`)

b) Authentication (e.g., hard-coded password in the source code)

c) Two encryption modes (encryption via ROT13 cipher and via exchanging of first two letters of a message)

d) Logging (e.g., use a simple log file)

e) **New:** Both GUI and CLI interfaces as alternative implementations.

Submit your complete answers (PDF) and implementation (Eclipse/FeatureIDE project) as an archive with name and matriculation number until 23 May, 23:59 CEST to `stefan.muehlbauer@uni-weimar.de`.