Radboud University

Dr. Daniel Strüber
Dr. Carlos Diego Damasceno

Radboud University
Institute for Computing
and Information Sciences

Autumn term 2022/23
Published: 12.09.2022
To be submitted:
19.09.2022 10:30

Software Product Lines

Assignment 2

Tasks 1-3 are designed as live exercises for the tutorial session – for this, we will split up in breakup groups. It's not required to submit anything for these tasks.

Tasks 4 and 5 are to be completed offline and to be submitted until the deadline.

Task 1: FeatureIDE tutorial

Familiarize yourself with feature modeling in FeatureIDE: Follow the tutorial from https://github.com/FeatureIDE/FeatureIDE/wiki/Tutorial

⚠ Focus on the sections "Introduction" and "Create Feature Model" -- that is, don't start the part about "Implementing features" / AHEAD ⚠

Task 2: Chat product line

As part of assignment 1, you developed a simple chat application in Java. Now imagine that you're leading a company that develops chat systems for company intranets and marketing websites (and imagine that there actually exists a market for such a company 😉).

a) Perform a domain analysis: What kind of features could the customer companies desire? For which of these features might a market exist? Which features might set your company apart from other companies that develop chats?

b) What benefits would your company gain from using product line technology? What alternative solutions may exist to using product line technology?

c) Create a feature model for this domain using FeatureIDE. First, create the feature hierarchy. Are there any further dependencies? If yes, model them.

d) Represent the feature model as propositional formula.

e) Specify a few of valid configurations and a few invalid configurations of the feature model.

f) How many valid configurations exist?

Task 3: Feature models (Discussion)

a) Discuss: Why are feature models usually represented as tree, and not as a list/graph/expression/prolog program?

b) The same product line can be represented with different feature models, which, however, lead to exactly the same feature selections and products. Give an example and discuss possibilities for normalization.

Task 4: Runtime parameter implementation

Your chat implementation (assignment 1), so far, should not contain any variability: all features are always active.

Enrich your chat implementation with runtime parameters, so that users can select features. Implement variability only as far as you find it useful -- for some features, there might be a point to have them mandatory. Decide between implementing variability with global parameters or parameter passing.

Submit an archive file, containing your source code and a readme document (txt or pdf) with the following contents:

- Explain your design decisions. In particular, explain your decision regarding global parameters vs. parameter passing.
- Explain how the feature selection works from the user perspective. Is there a risk of invalid feature selections and if yes, how do you address it?
- If you feel strongly about not implementing variability for one or several features, explain why.

Task 5: Design pattern implementation

Enrich your chat implementation from assignment 1 to include two or more design patterns.*

Submit an archive file, containing your source code and a readme document (txt or pdf) with the following contents:

- Explain your design decisions. In particular, explain which design pattern(s) you selected and why.
- Explain how the feature selection works from the user perspective.

* If you’re a two-person group, you only need to implement one design pattern.