Modeling, reverse-engineering, and testing variability :

The Jhipster case study

Jhipster is a tool for generating a complete and modern Web stack, at the back and front-end level (server side with Spring Boot, client-side with AngularJS and Bootstrap, and Yeoman, Bower, Grunt and Maven for builiding the application).

Jhipster is quite popular with more than 5000 stars in Github. It is available online : https://jhipster.github.io/

The exercises below can be seen as preliminary steps before comprehensively conducting an empirical study of a variability-intensive system. It can contribute to the field of product lines and thus some questions are open and non trivial[1].

1. We want to model the configurations authorized by Jhipster. Explain why playing with the configurator and tries every possible combination of options is not a viable solution.
2. Write a feature model that characterizes the configurations of the Jhipster configurator. (We will use FAMILIAR for this purpose.) A valid configuration of the resulting feature model should be authorized by the configurator. Due to the limits of a manual strategy based on playing the configurator (see point 1), it is highly recommended to have a look at some artefacts of Jhipster for specifying the feature model. (~prompt.js)

https://github.com/jhipster/generator-jhipster/blob/master/generators/server/prompts.js
https://github.com/jhipster/generator-jhipster/blob/master/generators/client/prompts.js
https://github.com/jhipster/generator-jhipster/blob/master/generators/app/prompts.js

3. Compute the number of valid configurations of Jhipster
4. Let's compare our solutions!
5. Explain how we could re-find existing bugs of Jhipster using a feature model

[1] Some researchers in DiverSE (http://diverse.irisa.fr) are currently working on this subject