The tool firstly check if we can find a correspondent product in *target*, for each *source* product configuration (line 1). We use the Alloy Analyzer for generating the product configurations from the *source* Feature Model. Then, we construct *source* and *target* sets of assets using its respectively Feature Model and Configuration Knowledge. Next, we bypass asset and product refinement checking and just evaluate the Configuration Knowledge for every possible configuration present in the Feature Model, checking if all existing evaluations of CK with the configurations of FM are still present in the evaluations of the resulting CK and FM. Doing that, we ensure the FM and CK from the evolved SPL jointly refine the original ones[x]. Otherwise, we interrupt the execution and report that is impossible to apply this tool.

Secondly, without generating products the tool checks whether the *target* SPL is well-formed (line 3), which means that it generates well-formed products, that correspond to valid products in the underlying languages used to describe assets[x]. If it finds a problem, it stops the process, reports all the invalid product configurations found and indicates that the product line is not refined and the evolution is not safe.

Then it identifies the assets modified by the evolution, using syntactic analysis (forth line). If these classes belongs to the Graphical User Interface component, we again stop and report that our toolset do not support test generation for GUI. Otherwise, we find the backward impacted classes and for each class, we use soot to find the necessary dependences to compile the sub product.

If this condition is not satisfied, we cannot use this technique.

We can only use … All product pairs could be applied. Cite app and succinctly describe it!

If they do not find changes, they assume that the line is refined;





The Pseudo-Algorithm 2 shows how to access the upper hierarchy levels closer to the User Interface. Firstly, it receives as a parameter the modified classes during the SPL maintenance. Thereafter, it saves these classes in an auxiliary variable (*line 2*), since one level below might already be the GUI. If it happens, we return the same received classes, otherwise we recursively call the method to walk up one level till it find GUI elements.