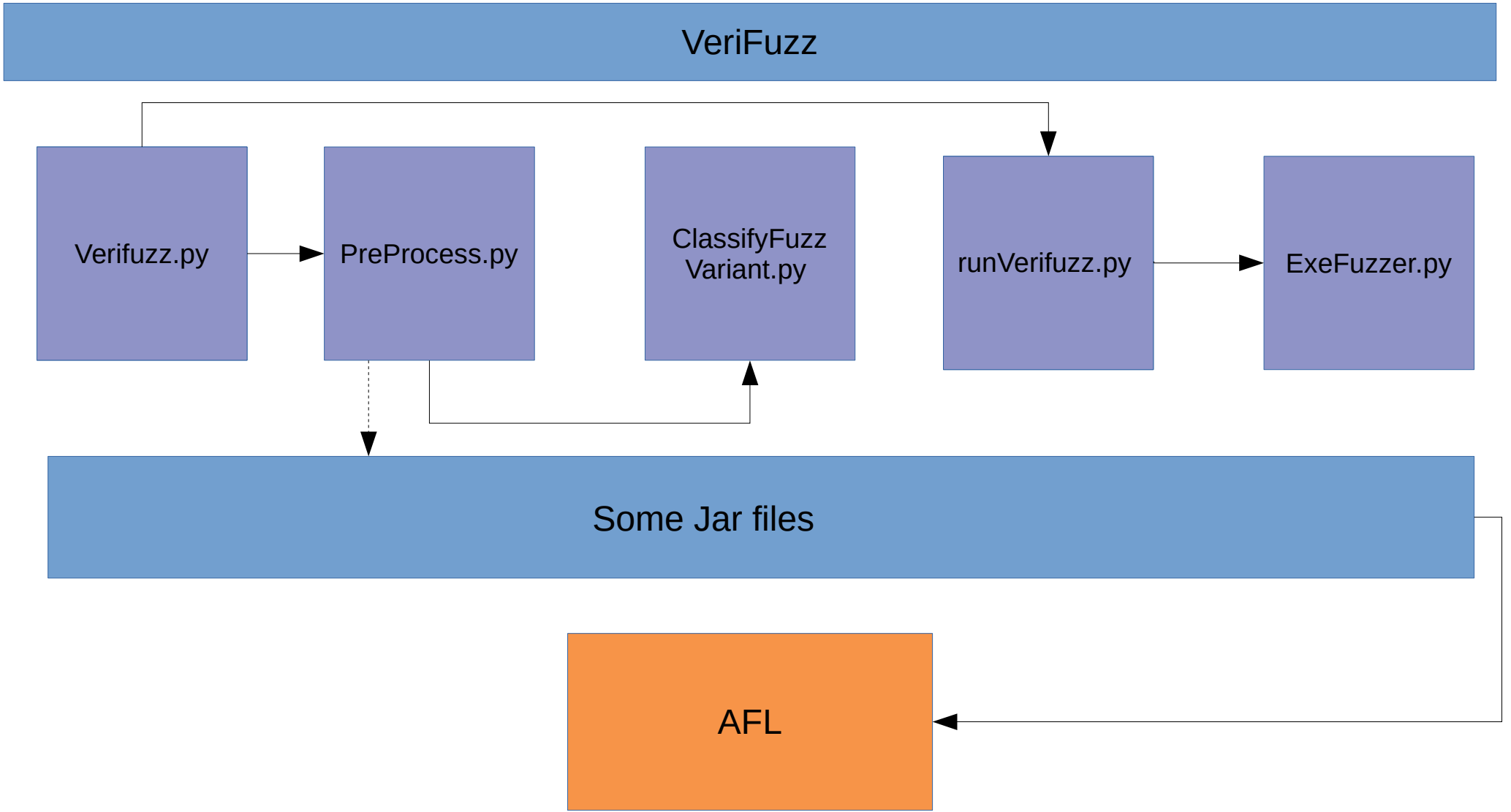
VeriFuzz
Verifuzz.py
PreProcess.py
ClassifyFuzz
Variant.py
runVerifuzz.py
ExeFuzzer.py
Some Jar files
AFL

Verifuzz.py -->

- process cmd line arg
- Set enviornment variables
- Make temp directories
- Go to preProcessInputFile.py via command
- Goes to runVerifuzz.py

preProcessingInputFile.py--->

- Process cmd arg and set env vars
- Creates a .prj file and write some data
- Calls a cppfe command for IR generation
- Runs cmmand to invote fuzzManager.RunFuzzingAnalysis java object file for performing feature extraction
- Uses above output and runs "classifyFuzzVariant.py"
- Run the same fuzzManager.RunFuzzingAnalysis java object file with different input and output directories for performing instrumentation

ClassifyFuzzVariant.py ---->

- Loads and classify the program and dumps result in algoCategory.txt using decision tree classifier

runVerifuzz.py -->

- Sets appropiate env vars
- Runs exeFuzzer.py with different AFL, Flags AlgoCategory, Property (Branch Coverage ,Branch error,etc)

exeFuzzer_veriFuzz.py--->

- Determines type of algo using AlgoCategory.txt
- Transforms the instrumented c code with “transform” binary
- Compiles the transformed code
- Sets some Afl Flags According to Requirement
- randomSeedGenCompile with either vs32.o vs64.o byteTrackerForRangeAnal.c for Random seed generation dumped in file name s1_default
- If failed then try whitbox seedGen
- And then much code about generating seed with different techniques and generating witness files
- Finally run AFL with Above output