Parser Implementation:

The task for implementing the parser was handled by the Software Architect branch. We decided to go with ANTLR for creating a parser. ANTLR is an exceptionally powerful and flexible tool for parsing formal languages. The grammars are clean and concise, and the generated code is efficient and stable.   
The parser code that was written was reversed engineered as we didn’t have any exact definition to progress with. The design approach used was top to bottom, since we already knew the language that we were going to design the grammar for – Java.

The parser tree looked like the diagram below:

The inputs from the file had to be of the following format:

e3(0,1,0){

--> e3

-->+ e2

},

e4<B>(0,0,1){

e1 -->

-->% (e4,e2)

-->+ e1

}

Where “e3(0,1,0)” represents the event name along with its markings taken in boolean form (executed, included, pending) in the order. Inside the curly braces, the relationships are mentioned.

This code was later merged with the visualization code (explained in detail in the next section), and we did tests on it to check its implementation.