B. Approach and Tool Chain

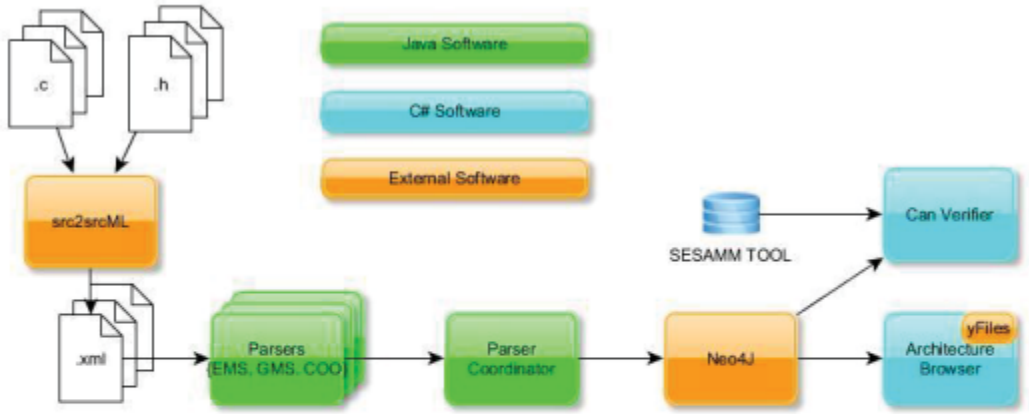


Fig. 3. Overview of the developed tool chain

The developed tool-chain, as illustrated in Fig. 3, consists of four main parts: A back-end with parsers and a parser coordinator, a Neo4J [18] database, and two front-end applications: *Architecture Browser* and *CAN Verifier*.

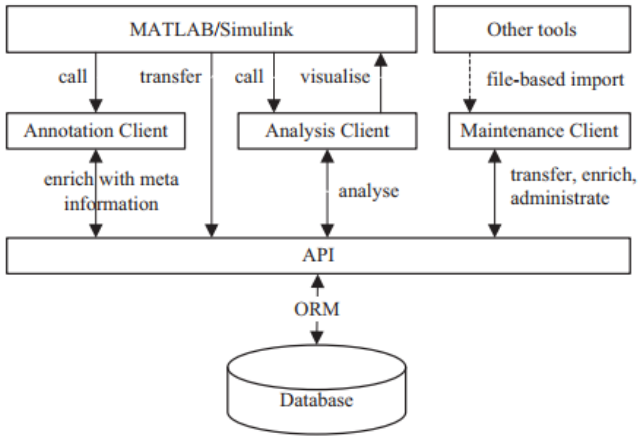


Fig. 4. Architecture of the analysis framework based on a central database

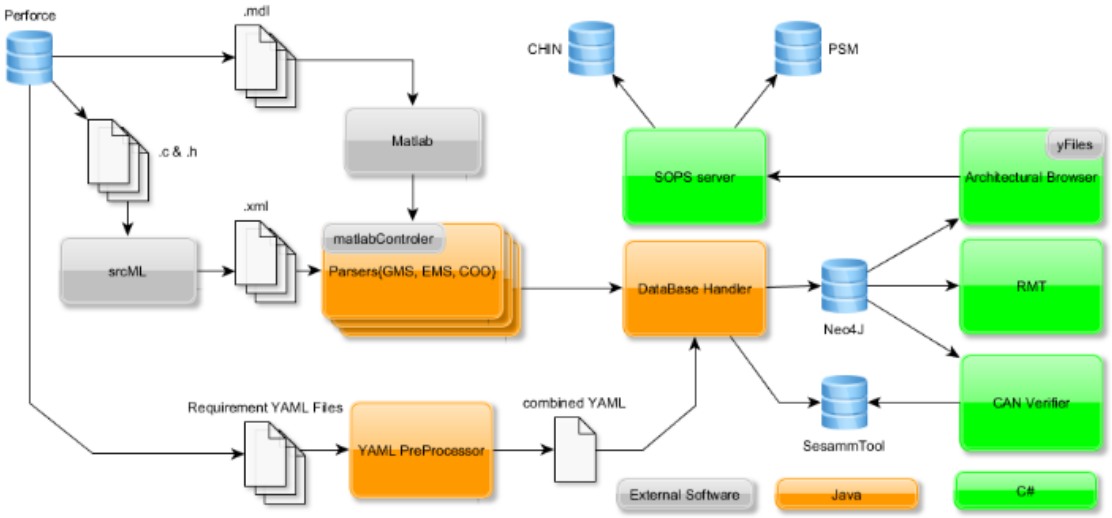
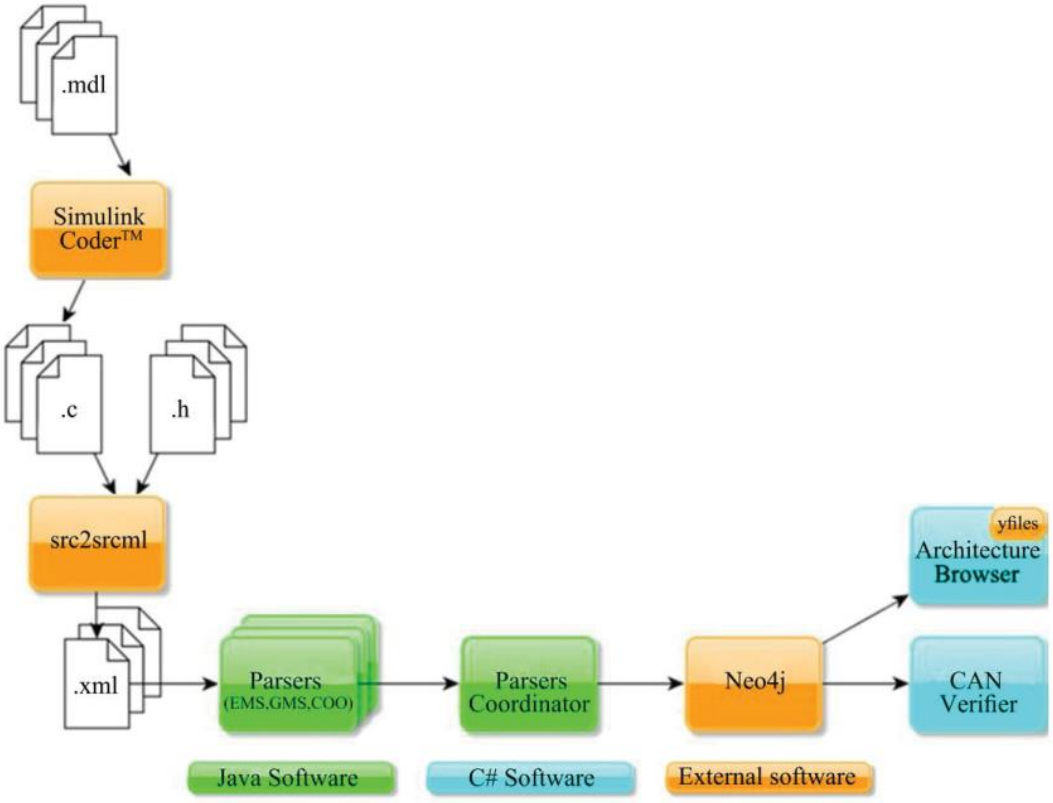


Figure 11: Image over the tool-chain for the ESPRESSO demonstrator (see chapter 5 related work). It explains how data is retrieved from the source (perforce) and is used in different tools. The information about the requirements are collected from source and parsed to the neo4j graph database as pre-processed yaml-files. Information about the system architecture is also retrieved from perforce (c and h-files) and parsed into the database. The prototype (RMT) collects the information from the neo4j graph database and visualise it.

Second proposed Tool chain:

Second proposed tool chain shown in Figure 8 designed based on our study to improving tool chain for .c and .h with this small revision that we add one more ring to this tool chain which it is "Simulink Coder".

"Simulink Coder" generates and executes C and C++ code from Simulink diagrams, State flow charts, and MATLAB functions. So in this case we only have to use "Simulink Coder" to generate C codes from Simulink model and then deliver .c file to src2srcML and the rest of our tool chain would be the same .Still there are two issues remained, the XML codes generated by "src2srcml" needs new parsers (1), and the extra fee that we need to pay for "Simulink Coder", make it costly in compare with other solutions (2).



Third proposed Tool chain:

According to mentioned above our tool chain which have required specification and functionality will be:

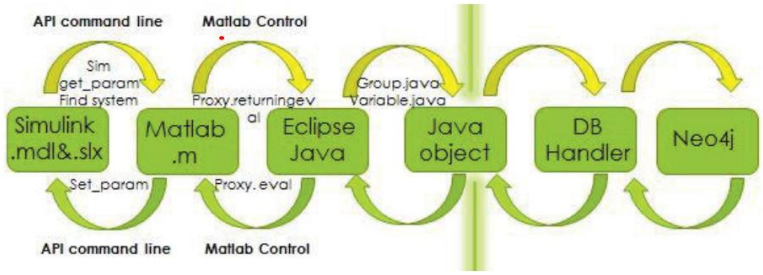


Figure 9: Third tool chain (implemented one)

As you can see in our final tool chain Simulink models (.mdl or SLX files) are feed of our tool chain, then you can see MATLAB environment .which we can communicate by using API command lines as below:

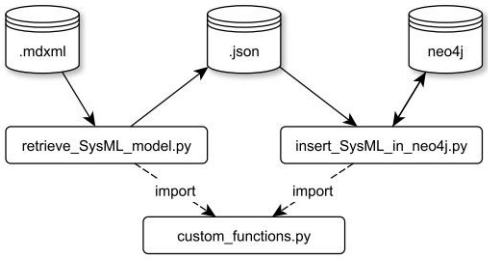


Figure 23. *Setup of the extract transfer load software to import MagicDraw SysML data to Neo4j.*

Data Extraction Transfer and Load Process of Simulink Model to Neo4j