SCA2AMALTHEA
LLVM-CLANG SETUP

LLVM-CLANG Setup

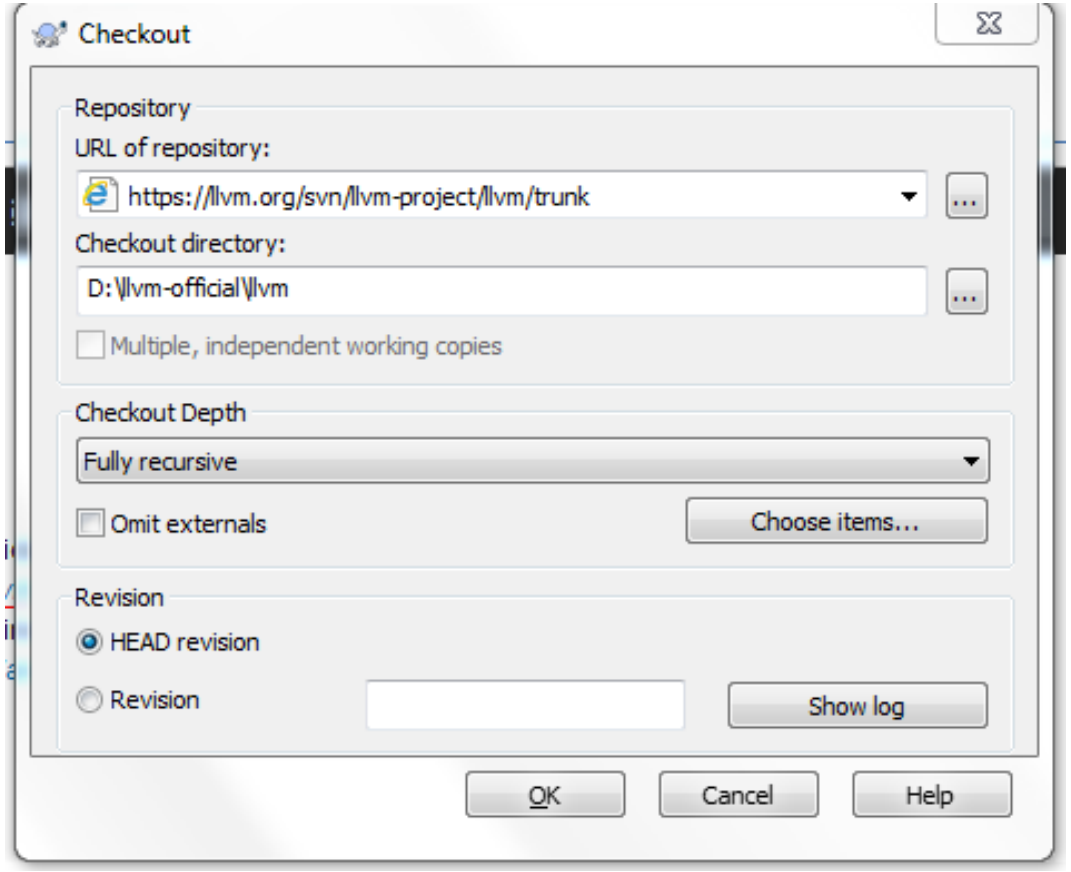
Softwares Required

- Microsoft Visual Studio 2015
- cmake 3.3.2
- SVN Checkout

LLVM-CLANG Setup

Checkout clang-llvm sources

- Right click inside the folder where you intend to download and say SVN Checkout



LLVM-CLANG Setup

Checkout clang-llvm sources

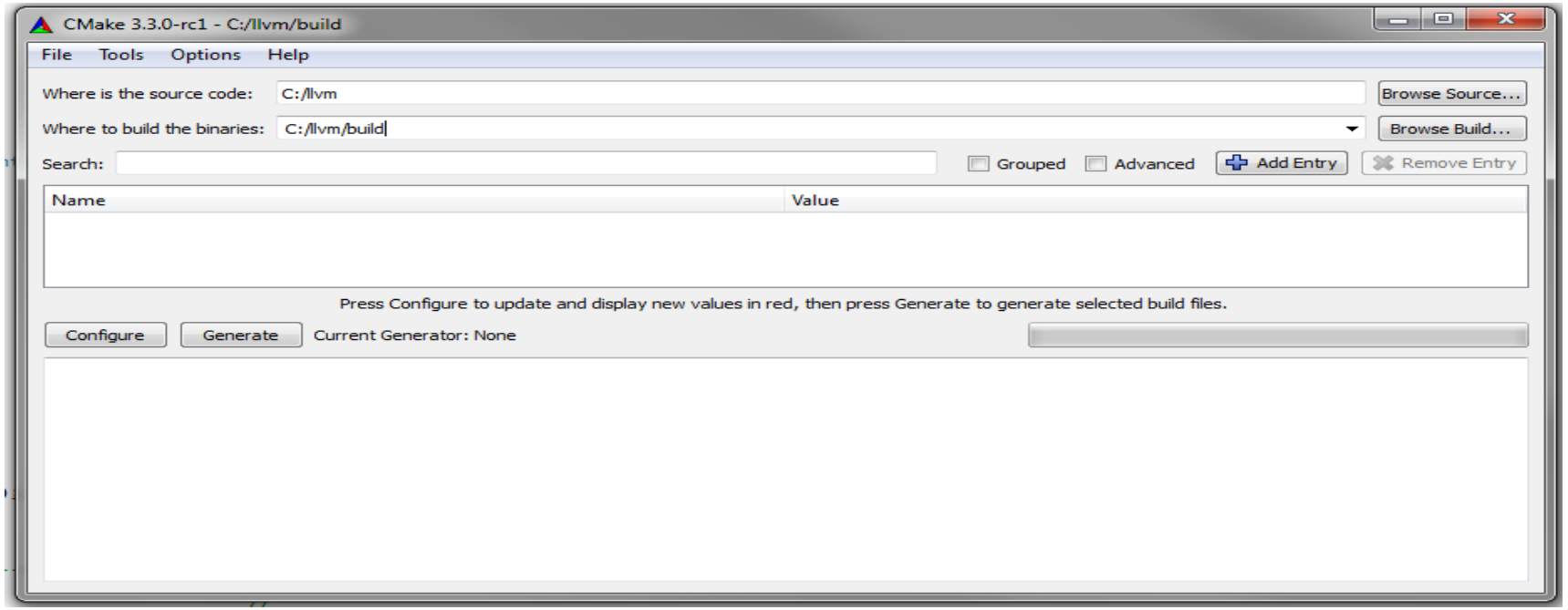
- Url for clang 4.0.1 official sources : http://llvm.org/svn/llvm-project/cfe/tags/RELEASE_401/final
- Url for LLVM 4.0.1 official sources:http://llvm.org/svn/llvm-project/llvm/tags/RELEASE_401/final
- Enter these urls in the text box in the screenshot(previous slide) each for clang and llvm and then provide the check-out directory where the clang and llvm sources have to be checked out.
- Copy the clang source folder in the path (“LLVM source folder”/tools)
- Copy the sca source folder inside ("LLVM source folder"/tools/clang/tools)
- Open the CMakeLists.txt present in the path ("LLVM source folder"/tools/clang/tools) and add the below line at the top of the file (as we copied “sca” folder in previous step)

add_subdirectory(sca)

LLVM-CLANG Setup

Building using cmake

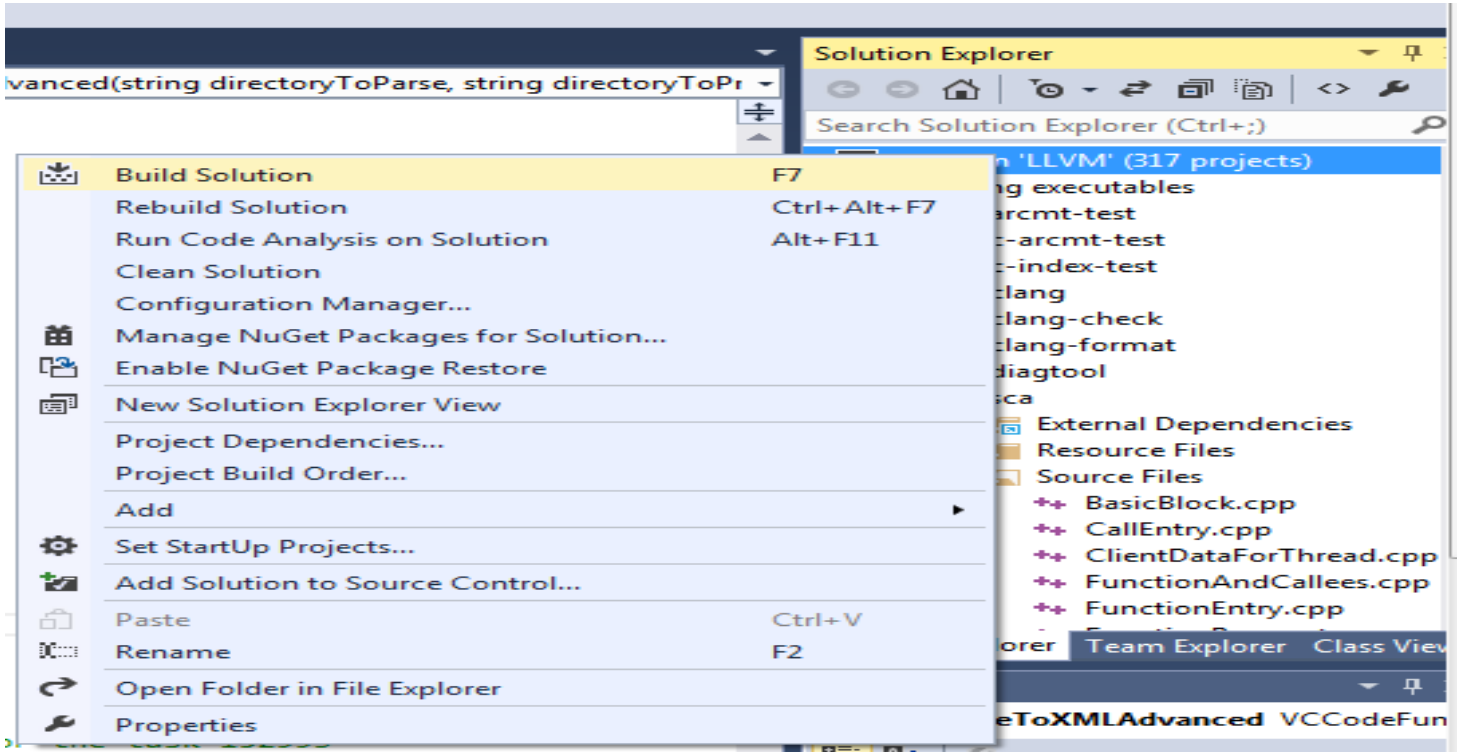
- Create a build directory where the sources of the code will be built (e.g. c:\llvm\build)
- Open cmake-gui for to generate the MS Visual Studio project files to be built:
- From there, select the sources and the build directories, then click on ***configure*** and finish by ***generate*** when the console advise you to do so:



LLVM-CLANG Setup

Building using Visual Studio

- After this step, the necessary files for the build have been generated inside the build directory
- Go inside the build directory and start the file ALL_BUILD.vcxproj with MS Visual Studio
- From Visual Studio, build the project by right-clicking on the “Solution LLVM”->Build Solution



LLVM-CLANG Setup

Building using Visual Studio Continued …

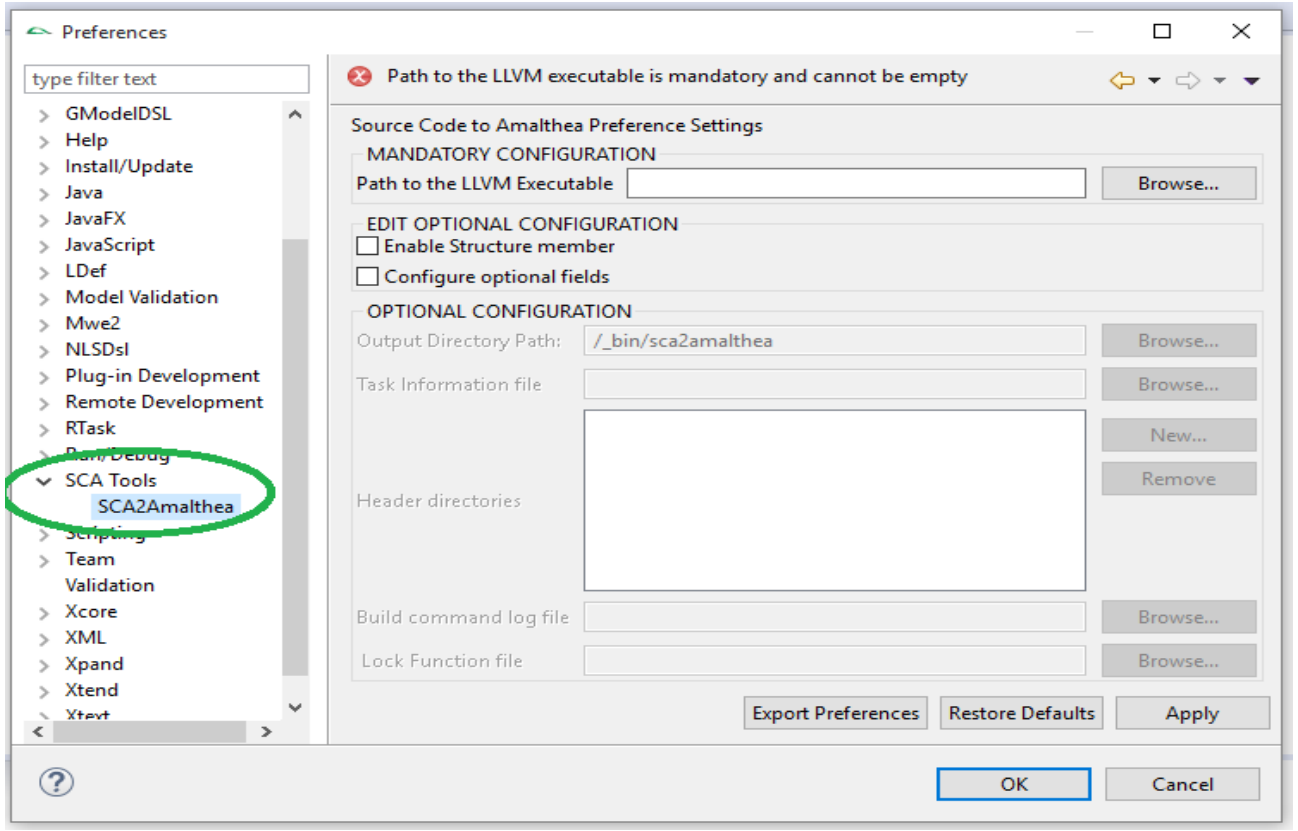
- This should take some time but at the end, it should have built all the necessary parts of LLVM and clang(follow the output on the console).
- sca.exe can be found in (“Build Directory”/Release/bin or “Build Directory”/Debug/bin) depending on the solution configuration given in the visual studio.
- Copy **sca.exe** and **libclang.dll** to a separate location on your hard disk for later use in SCA2AMALTHEA

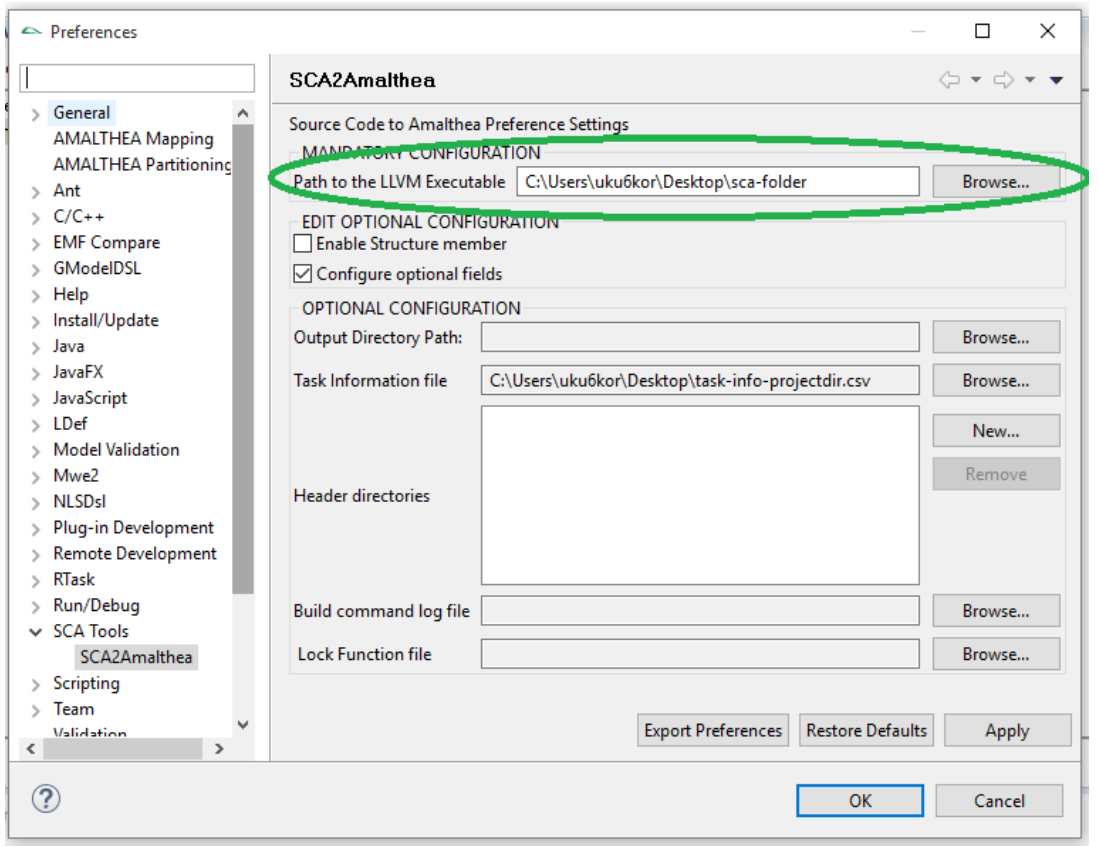
SCA2AMALTHEA
AMALTHEA Generation

SCA2AMALTHEA

Using Built SCA Executable

- Go to eclipse Windows->Preferences->SCA2AMALTHEA (see Figure 1)
- In “Path to LLVM Executable” option provide the path where your sca.exe is placed on your hard drive(see Figure 2)





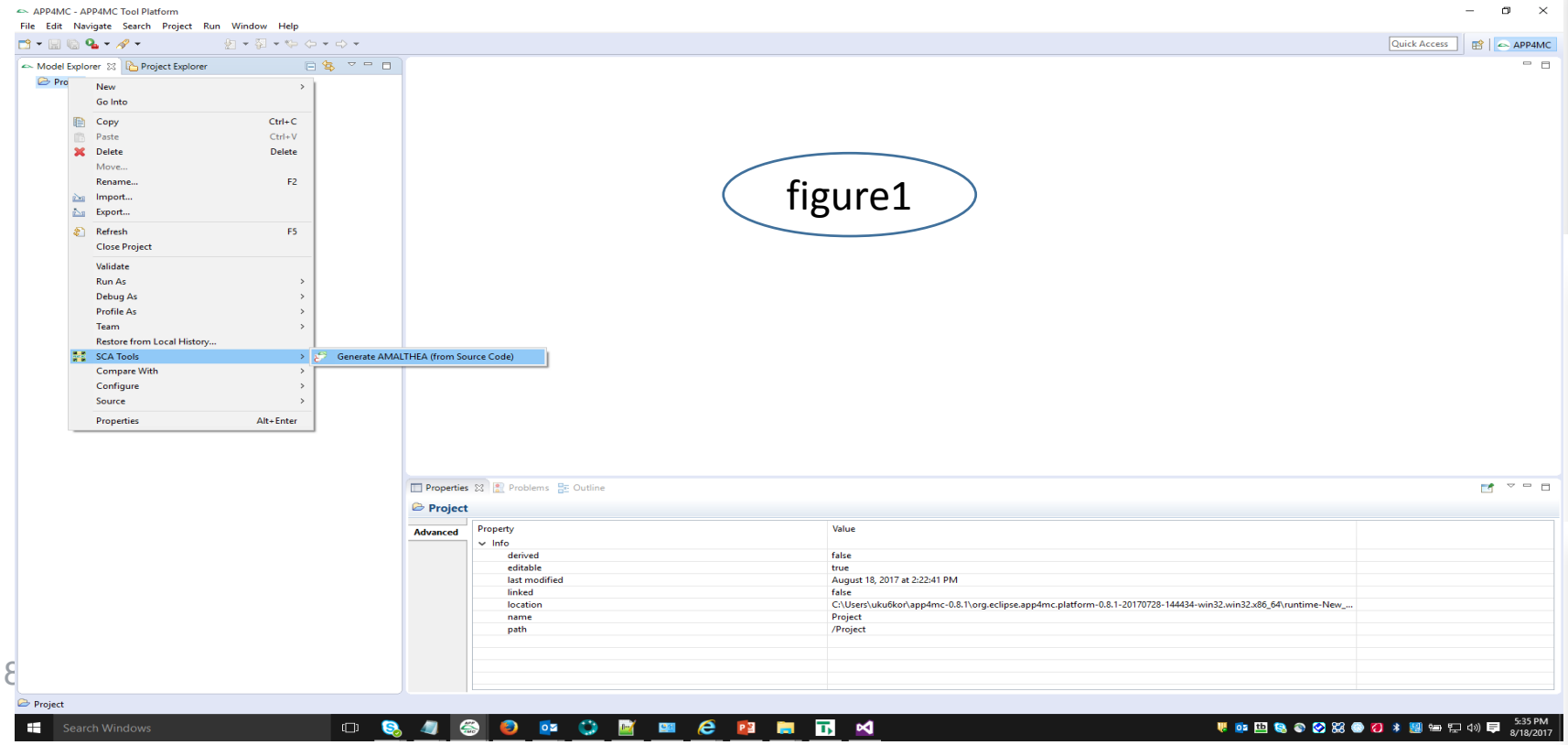
SCA2AMALTHEA
Configuration Options

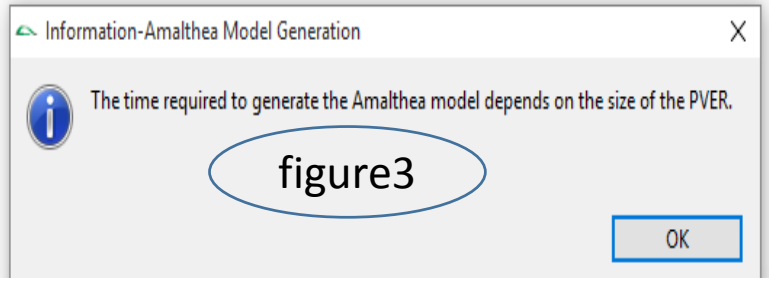
- **Enable Structure Member** – When selected shall generate the complete access to the member level of a structure, otherwise it shall just report access to any member of structure as an access to structure. By this option is false
- **Output Directory Path –** Can be configured to place the output to other directory then default ones
- **Task Information file –** A comma separated file which contains names of the **Tasks/ISRs** to categorize C functions to their respective AMALTHEA software model element. If file is not provided all functions shall be generated as **Runnables** in AMALTHEA Model. For a sample configuration please refer to ***sample_task_isr_info_file.csv*** file.
- **Header directories:** This is a list of header directory paths to be included. If not configured then the entire project directory is considered as header directory.
- **Build command log file:** This field is for providing absolute path of the build_cmd.log file which contains the c file names to be parsed. If not provided then the c files from the entire project directory are parsed.
- **Lock Function file –** A comma separated file which contains names of the C functions which should be considered as lock acquire/release C functions**.** The name shall appear as semaphore in AMALTHEA OS Model**.** The **lock acquire function call** shall be modelled as **acquire <semaphore-name>** and **lock release function call** will be modelled as release **<semaphore-name>.** For a sample configuration please refer to ***sample_lock_function_info.csv*** file.

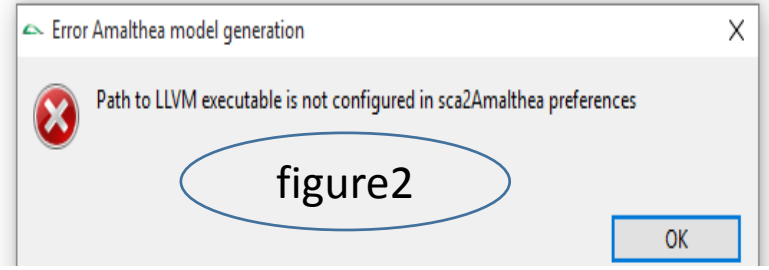
SCA2AMALTHEA

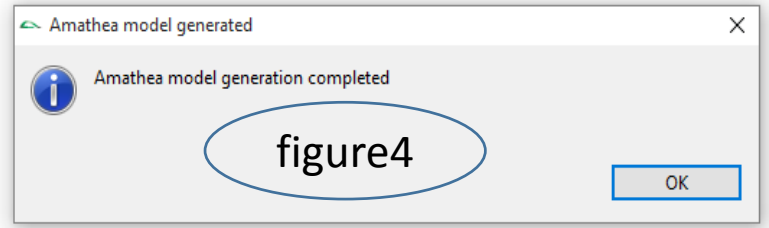
Launching from Menu

- Right click on the project and click on SCA Tools ->Generate AMALTHEA(from source code) as shown in figure 1.
- If the Path to LLVM executable is not configured in sca2Amalthea preferences then a popup as shown is figure 2 comes up.
- If the Path to LLVM executable is configured in sca2Amalthea preferences then the Amalthea generation job starts and a popup shown in figure 3 pops up.
- Once the Amalthea model generation job completes then the popup shown in figure 4 comes up.









SCA2AMALTHEA

SCA2Amalthea Preference Page

- Sca2Amalthea preference page consists of mandatory and optional configuration.
- Optional configuration can be enabled and disabled using the “Configure Optional fields” check box as shown in figure 1(disabled) and figure 2(enabled).

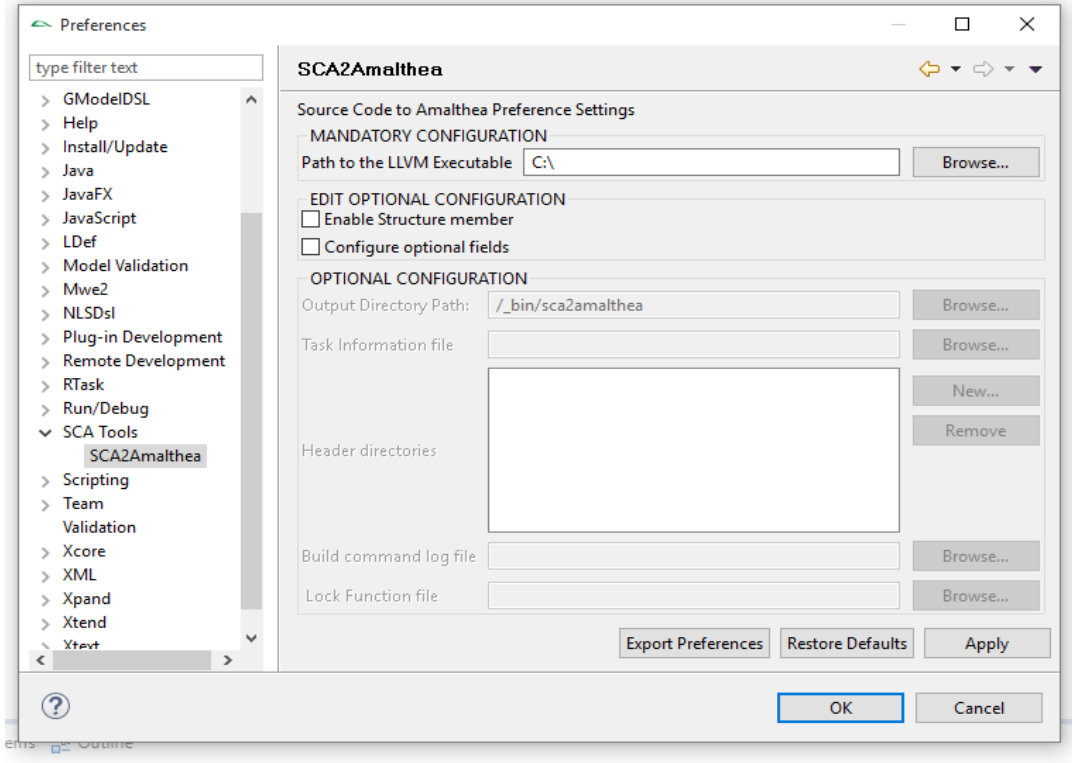


Figure 1

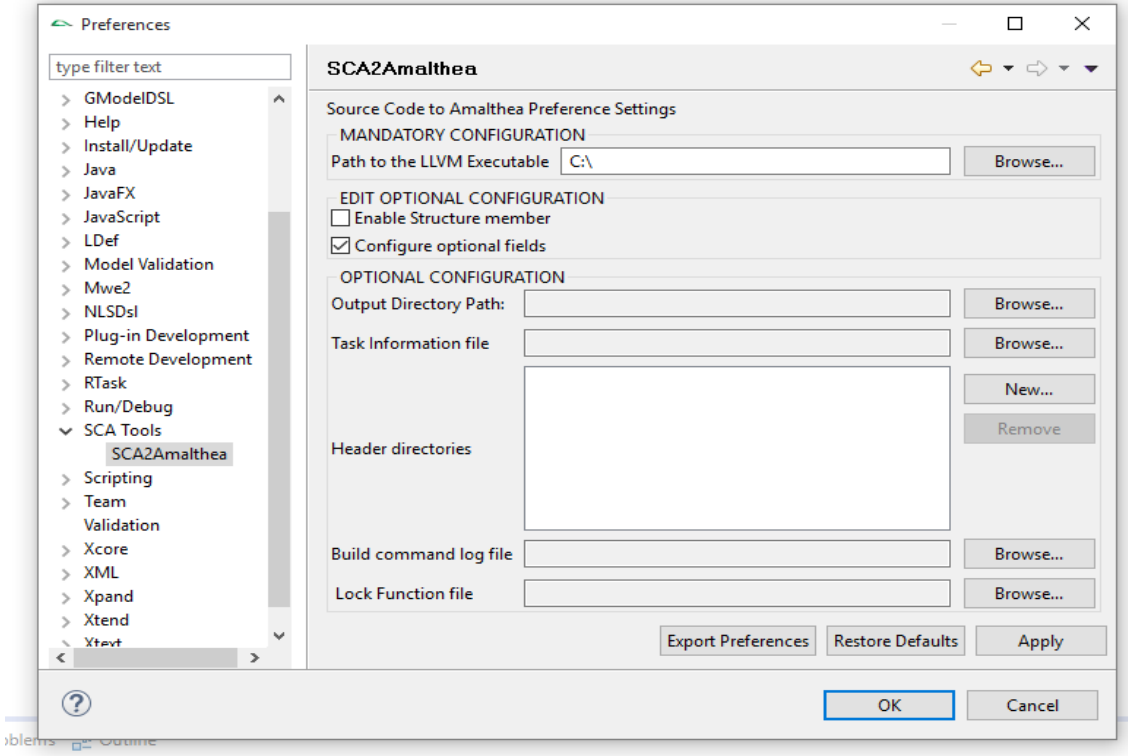


Figure 2

SCA2AMALTHEA

SCA2Amalthea Preference Page –Mandatory Configuration

- Mandatory Configuration consists of only one field “Path to the LLVM Executable”.
- When the sca2Amalthea preference page is loaded for the first time and Ok button is clicked without configuring the “Path to the LLVM Executable” field then the popup as shown in figure 1 is shown.
- For subsequent loads of the sca2Amalthea preference page ,error as shown in figure 2 is shown if the “Path to the LLVM Executable” is not configured.

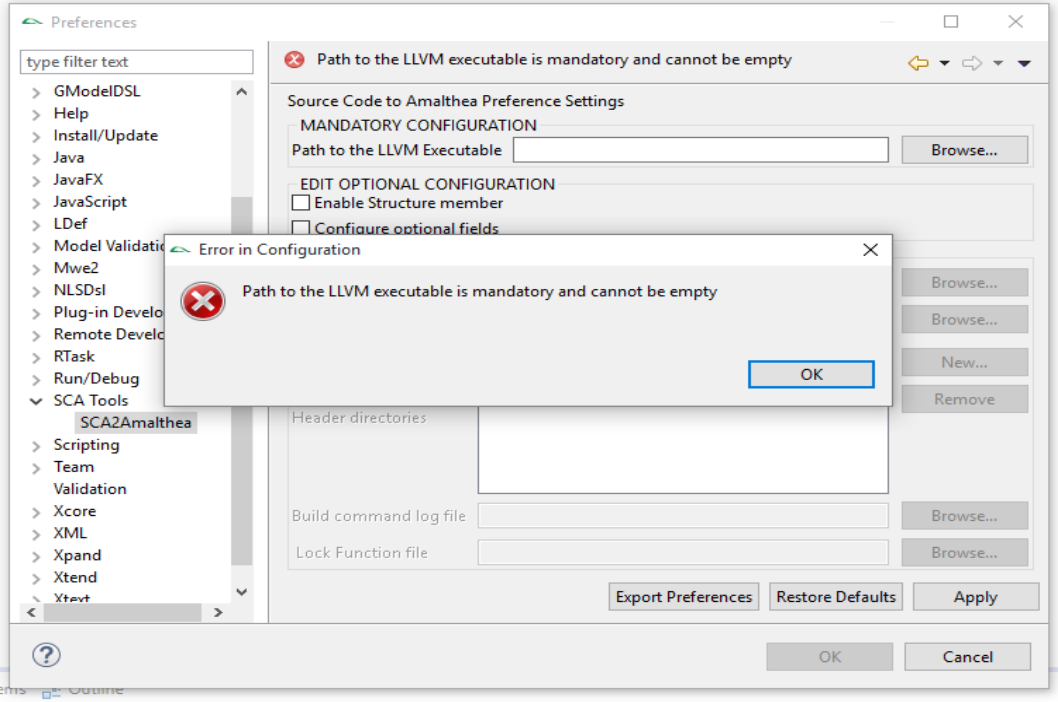


Figure 1

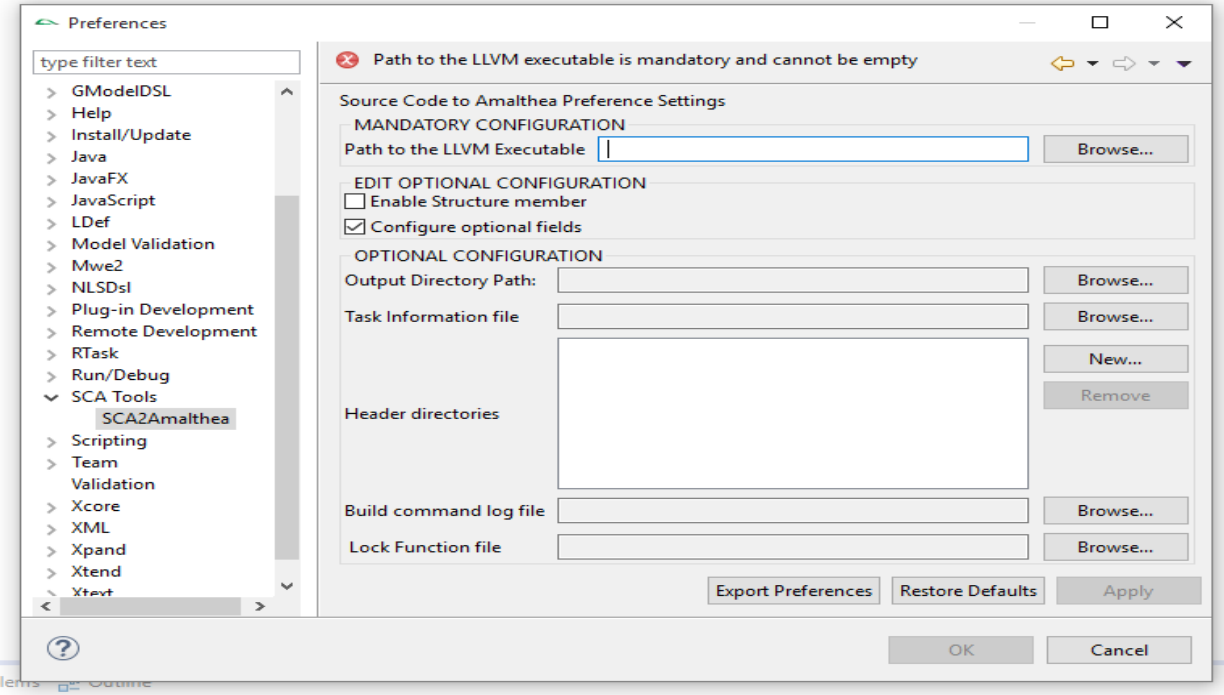
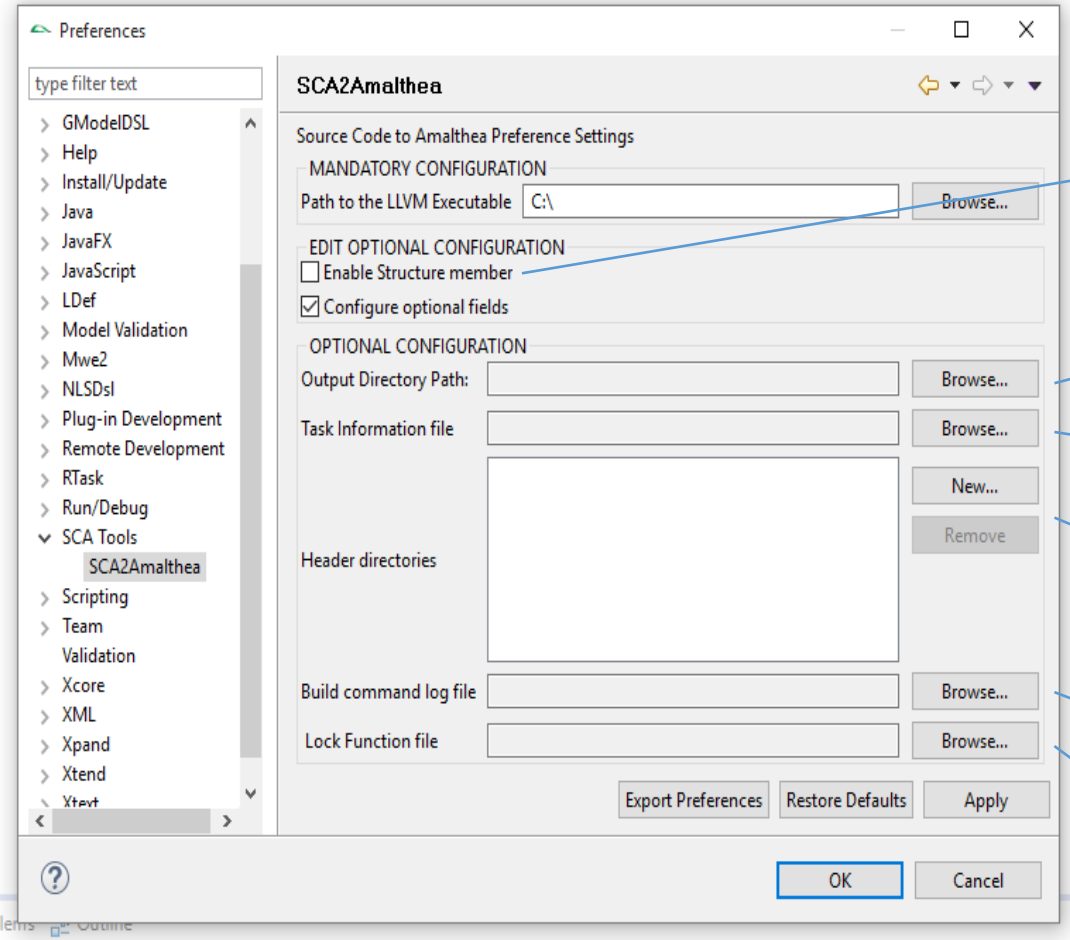


Figure 2

SCA2AMALTHEA

SCA2Amalthea Preference Page –Optional Configuration



When checked struct members would be present in the Amalthea model.

This is a the path where all the output files are stored. If the output directory path is not configured then the generated Amalthea file is present in the (project dir/_bin/sca2amalthea)

This is a csv file containing task/ISR information. If the task information file is not provided then all the functions present in the Project are treated as runnables.

This is a list of header directory paths to be included. If not configured then the entire project directory is considered as header directory.

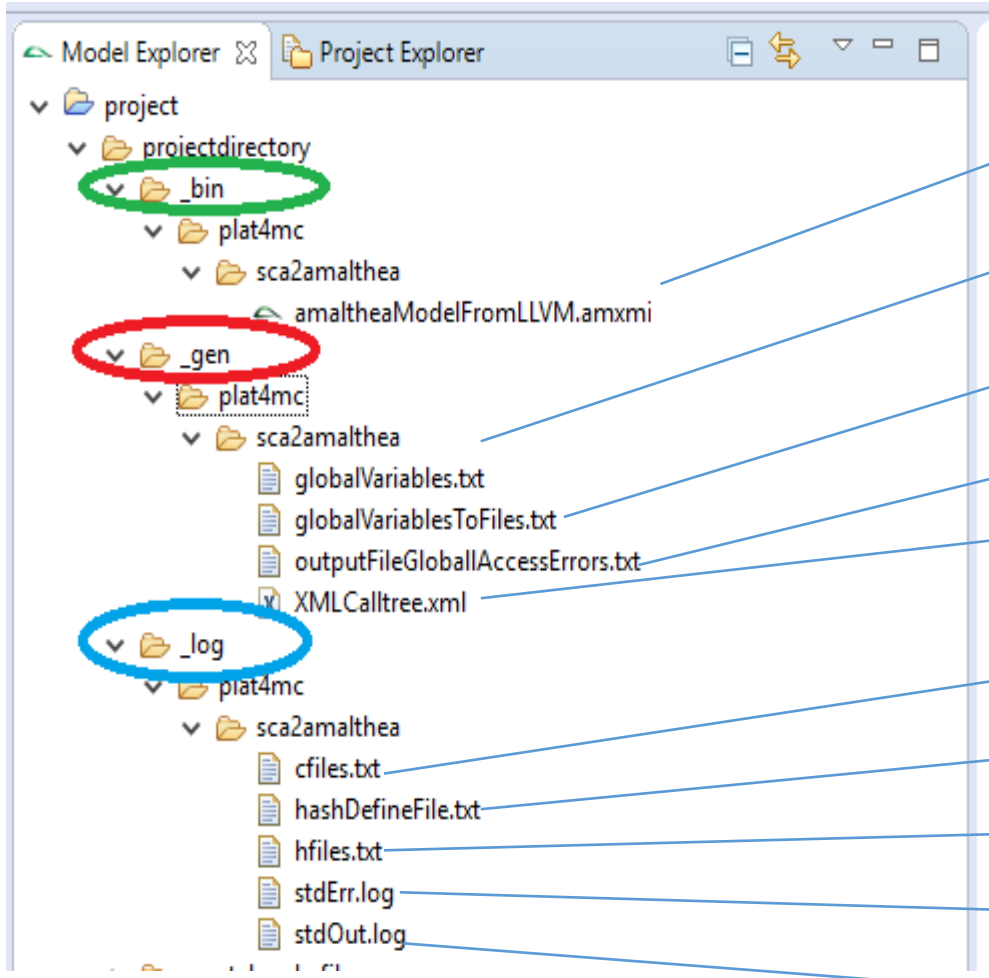
This field is for providing absolute path of the build_cmd.log file which contains the c file names to be parsed. If not provided then the c files from the entire project directory are parsed.

This field takes in a .csv file which contains the names of the C functions which should be considered as locks. If not provided all the functions are considered as runnables.

SCA2AMALTHEA

SCA2Amalthea Preference Page –Output Directory

- If the output directory is configured in the sca2Amalthea preferences then all the below files will present in the directory that is configured else the files will be present in the directories as shown.



This is the generated Amalthea model file.

This contains all the global variable names declared inside the c/h files of the project.

This contains all the global variable names, file name in which they reside and also the line and column where they are declared.

It contains all those label accesses whose names are not present in the labels section of the xmlCallTree.xml

This is an intermediate representation file which is basically the output of sca.exe. It contains the call tree information, label accesses ,type-def info and so on.

It contains all the c file names to be parsed

It contains all the macros and their values that would be used while parsing.

It contains all the h file names to be parsed

It contains all the error messages logged by sca.exe while parsing.

It contains all the output info messages logged by sca.exe while parsing.