Benchmark Engineering tool integrated with SmartMDSD

BenchmarkEngineeringTool provides an interface to be used with SmartMDSD. The interface simply loads the ".standardized_problem" files created by the tool into C++ structures in the SmartMDSD. The loaded structures can be then used by the developers to ease developing the BenchmarkExperiment based on the developer's created model yet leveraging the structure offered by the BenchmarkEngineeringTool. The following guideline states the steps on how to load the ".standardized_problem" files into the SmartMDSD.

One-time install

1. EMF4Cpp using `bash setupemf4cpp.sh`. This script install EMF4CPP and its dependencies (e.g. Qt5).

Create Benchmark Project

1. Import **BenchmarkExperimentLib** into the SmartMDSD workspace using `File > Import > General > Existing Projects into Workspace > select archive > BenchmarkExperimentLib`
2. Build the **BenchmarkExperimentLib**

```
cd BenchmarkExperimentLib/src-gen
cmake .
make
```

3. Create **ComponentBenchmark...**
4. Amend **ComponentBenchmark...**/smartsoft/CMAKELists.txt to include

```
# Begining of File
## Location of BenchmarkExperimentLib/src-gen
add_subdirectory("../../BenchmarkExperimentLib/src-gen/"
"${CMAKE_CURRENT_BINARY_DIR}/build")
## Set location of installed emf4cpp
set(EMF4CPP_PATH="/home/smartsoft/emf4cpp")
## Location of BenchmarkExperimentLib/src-gen and Qt5
include_directories(../../BenchmarkExperimentLib/src-gen/ /usr/local/include/emf4cpp
${EMF4CPP_PATH}/emf4cpp.tests/resource ${EMF4CPP_PATH}/emf4cpp /usr/include/x86_64-
linux-gnu/qt5)
find_package(Qt5Widgets)
find_package(Qt5Declarative)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS}
    ${Qt5Widgets_EXECUTABLE_COMPILE_FLAGS}")

set(CMAKE_CXX_FLAGS "-Wall -std=c++11 -fPIC")
set(CMAKE_CXX_FLAGS_DEBUG "-g -DDEBUG")
set(CMAKE_CXX_FLAGS_RELEASE "-O3 -funroll-loops")
set(CMAKE_LD_FLAGS "-L.. -L/opt/emf4cpp/lib")
```

```
# End file
target_link_libraries(test-create emf4cpp-ecorecpp emf4cpp-ecore emf4cpp-ResourceTests
emf4cpp-benchmark_experiment emf4cpp-standardized_problem emf4cpp-basicAttributes
emf4cpp-communicationObject emf4cpp-componentDefinition emf4cpp-serviceDefinition
emf4cpp-communicationPattern emf4cpp-coordinationPattern emf4cpp-componentMode
emf4cpp-stateMachine emf4cpp-parameterDefinition Qt5::Widgets)
```

5. Create activity in the **ComponentBenchmark...**
6. in the created activity.cc, add the following dependencies

```
#include <cassert>
#include <unordered_map>
#include <string>

#include <ecorecpp/resource/ResourceSet.hpp>
#include <ecorecpp/resource/URIConverter.hpp>
#include <ecorecpp/resource/XMLResource.hpp>
#include <ecorecpp/MetaModelRepository.hpp>

#include <ecore.hpp>
#include <ResourceTests.hpp>

#include <ecorecpp.hpp>
#include <standardized_problem.hpp>
#include <basicAttributes.hpp>
#include <fstream>
#include <memory> // for std::auto_ptr
#include <assert.h>

#include <cassert>
#include <unordered_map>
#include <string>

#include <ecore.hpp>
#include <standardized_problem/Standardized_problemPackage.hpp>
#include <basicAttributes/BasicAttributesPackage.hpp>
#include <typeinfo>
```

7. Import the benchmark_model generated from the BenchmarkEngineeringTool by calling the following lines:

```
::ecorecpp::MetaModelRepository::_instance()-
>load(::basicAttributes::BasicAttributesPackage::_instance());

::ecorecpp::MetaModelRepository::_instance()-
>load(::standardized_problem::Standardized_problemPackage::_instance());
// Location of the file to be loaded:
const QUrl file =
QUrl::fromLocalFile("/home/smartsoft/Desktop/Database/sp.standardized_problem");
    ::ecorecpp::resource::ResourceSet rSet;
```

```
    rSet.getResourceFactoryRegistry()->getProtocolToFactoryMap()
        [file.scheme().toStdString()].reset(new
::ecorecpp::resource::XMLResourceFactory());

auto resource = rSet.createResource(file);
resource->load();

// StandarizedProblem object
auto standardizedProblem = ::ecore::as<::standardized_problem::StandardizedProblem>
(resource->getContents()->get(0));
std::cout << "Loaded standardized problem:" << standardizedProblem->getLabel();
```

6. Accordingly, the standardized_problem structures can be used in the created activity

Issue that can be faced

Issue 1: SMART_MACROS cannot be found

> Solution 1: Don't run Eclipse with Admin rights
>
> Solution 2: Get another fresh Smartsoft Virtual Machine

Issue 2: CMake error: missing necessary privelages

> Solution 1: run `sudo chown -R .` in the workspace folder containing the Component project

Note

Always clean remove the `build` directory of any imported project or `CMakeFiles, CMakeCache, Make`, and rebuilt it using step 2