COST
EUROPEAN COOPERATION
IN SCIENCE & TECHNOLOGY

SHORT TERM SCIENTIFIC MISSION (STSM) – SCIENTIFIC REPORT

The STSM applicant submits this report for approval to the STSM coordinator

Action number: IC1404
STSM title: Multi-Paradigm Modelling for Cyber-Physical Systems (MPM4CPS)
STSM start and end date: 22/04/2018 to 27/04/2018
Grantee name: Erwan Bousse

PURPOSE OF THE STSM/

Cyber-physical systems (CPS) are complex systems dealing with a wide range of concerns for very diverse stakeholders. Executable Domain-Specific Languages (DSLs) are crucial for the development of such systems as they enable the early dynamic verification and validation of the behavior of systems. Once different behavioral aspects of a CPS have been defined in the form of heterogenous executable models conforming to different executable DSLs, these models can be integrated with each other using several composition techniques. However, understanding the behavior of a set of integrated executable models remains a difficult analysis task, and requires tools able to deal with an heterogeneous set of models. The GEMOC Studio is an open-source Eclipse package that includes a language workbench to build and compose new executable DSLs, and a modeling workbench to create, execute and coordinate models conforming to executable DSLs. More precisely, executable DSLs can optionally be defined with an explicit model of concurrency (using MocML) which makes it possible to compose their semantics (using B-Cool), which enables the coordination of the execution of conforming models. In addition, the GEMOC Studio provides omniscient debugging facilities that enable the runtime exploration of the behavior of an executable model both backwards and forwards in time. Such facilities greatly help better understanding model executions. Unfortunately, current GEMOC omniscient debugging facilities are adapted for debugging single models, and not sets of coordinated heterogeneous models conforming to composed DSLs. Therefore, in this STSM, we propose to investigate how to make the GEMOC Studio omniscient debugging facilities available for composed executable DSLs. The proposed STSM belongs to the Working Group 1: Foundations of the COST Action MPM4CPS.

DESCRIPTION OF WORK CARRIED OUT DURING THE STSMS

Two main tasks were carried out during this STSM:

First, as the GEMOC Studio only provides facilities to compose languages with explicit models of concurrency, a first task was to enable the use of omniscient debugging facilities for such concurrent executable DSLs. This required making the existing omniscient debugger compatible with concurrent executable DSLs, which itself required refactoring and and adapting different parts of the GEMOC Studio.

Second, once omniscient debugging facilities have been made compatible with concurrent DSLs, they still need to be improved to support the proper debugging of concurrent behaviors. In particular, non-

Funded by the Horizon 2020 Framework Programme
of the European Union

determinism is inevitable, which calls for facilities to explore and understand the different paths that can be taken in an execution, e.g. going back in time and trying different alternative execution steps. One of the main parts of this task is to change the execution trace metamodel used by the omniscient debugger to enable the creation of different execution branches, thereby giving the possibility to explore different paths with the omniscient debugger. Then, all tools that depend on this metamodel must be adapted, and the omniscient debugger can be extended with new backtracking services.

DESCRIPTION OF THE MAIN RESULTS OBTAINED

Regarding the first task, we discussed and established a workplan on what should be done in the current code to make the omniscient debugger compatible with concurrent executable DSLs without adding new exploration services. The resulting plan was (1) to remove all dependencies of the existing omniscient debugger to the part of the framework dedicated to sequential executable DSLs, and (2) to consider the execution of each set of concurrent execution steps as one single atomic execution step, which preserves the semantics of the sequential omniscient debugger while enabling the exploration a sequence of sets of steps. We concretized this plan by making the following changes in the code of the GEMOC Studio (each link is a pull request that was merged or that will be merged in the near future):

- https://github.com/eclipse/gemoc-studio-modeldebugging/pull/31
- https://github.com/eclipse/gemoc-studio-modeldebugging/pull/34
- https://github.com/eclipse/gemoc-studio-modeldebugging/pull/36
- https://github.com/eclipse/gemoc-studio-modeldebugging/pull/39

As a result, the omniscient debugger of the GEMOC Studio can now provide sequential debugging services for concurrent executable DSLs.

Regarding the second task, we discussed and brainstormed on how could the exising trace metamodel be changed to also allow traces with multiple execution paths. While many different alternatives were proposed, the main result of these discussions were the following observations: (1) each time an element of the metamodel refers to some part of the trace “in the future” (eg. the ending execution state of an execution step), it is likely that the cardinality of the feature but be changed to [0..*] to allow multiple possible futures ; (2) all existing tools that only work with one execution path at a time will require an operator to “project” a multipath trace into a singlepath trace. An additional concrete result is the creation a first draft of candidate new trace metamodel in a dedicated branch in the GEMOC Studio source code repository: https://github.com/eclipse/gemoc-studio-modeldebugging/tree/unified-trace-metamodel2

FUTURE COLLABORATIONS (if applicable)

The goal of this STSM was to make the very first steps towards a complete technique to provide meaningful omniscient debugging services for coordinated executable DSLs. Future collaborations are therefore planned both to carry on the tasks that were started in the STSM, and to start the next steps of this work, namely to take the definition of the coordination into account in the debbugging services.