An Assistance Tool to Design Interoperable Components for Co-simulation

Yassine Motie, Alexandre Nketsa, Philippe Truillet

An assistance tool to design interoperable components for co-simulation

Yassine Motie and Alexandre Nketsa and Philippe Truillet

LAAS-IRIT, University of Toulouse III, Toulouse, France
firstName.lastName@irit.fr and alex@laas.fr

Abstract. The high number of electronic devices used and their interactions lead us to the transition from a vision of multi-functions systems, used independently, to systems that are actually distributed and scattered in the environment. The heterogeneity of the components constituting some of these systems ultimately leads to call them "complex". When a complex system [1] requires the use of different components specified by different designers working on different domains, this greatly increases the number of virtual prototypes. Unfortunately, these components tend to remain too independent of one another, thus preventing both the designers from collaborating and their system from being interconnected in order to full one or more tasks that could not be performed by one of these elements only.
The need for communication and co-operation is necessary and encourages the designer (s) to inter-operate them for the implementation of a co-simulation [2] encouraging dialogue between disciplines and reducing errors, costs and Development time.
In this article, we describe an assistance tool in order to generate black-box components, facilitating this design task for novices.

Keywords: Complex systems; models; FMI; co-simulation; component generation

1 Introduction

Designing an interactive system is a difficult task. Designing and evaluating a so-called "complex" system is even more so. From the software point of view, this system can be seen as an integrated set of elements interconnected with each other, in order to satisfy in a given environment one or more pre-defined objectives.
In general, the components of this system include both the facilities, the hardware and software equipment, the data, the services, the qualified personnel and the techniques necessary to achieve, provide and maintain its efficiency. A complex system also has many characteristics such as the heterogeneity of its components, their evolution at different time scales and their geographical distribution integrating both digital systems, physical operators and/or human. These complex systems are usually broken down into subsystems, following either a top-down approach (also known as Stepwise Refinement and functional decomposition), or a bottom-up approach (from existing components that need to be reused). This often leads to a failure for these subsystems to speak the same language to each other and to share information effectively and

correctly - to be interoperable - which is one of the major problems that complex systems frequently face.

In order to succeed this collaboration at a global level, it is important to opt for an open environment that allows for a continuous dialogue between the different parties. One way to do this is by applying co-simulation which is defined as the combination of various simulation models, including other tools, for different components of a complex system. Co-simulation provides the abstraction needed for designers to work on their business expertise. The interest regarding co-simulation, is not only triggered by the coupling of the environments but also by the potential efficiency gain of decoupling a large system model. This is exemplified in [20] where a model of an engine is split into subsystems, leading to a decrease of the simulation time by an order of magnitude.

FMI [21] (Functional Mock-up Interface) standard and data mediation were used in a previous work [5] for the structural and semantic interoperability levels respectively, in order to build an interoperable co-simulation framework applied in neOCampus which is a research project supported by the University of XXX [4]. The idea is that tools generate and exchange models that conform to the FMI specification. Such models are called FMUs (Functional Mock-up Units).
The problem is that the transition from the subsystem studied to an FMU component (black box) requires knowledge of the FMI standard, thus constituting an obstacle for our designers, inhibiting thereby their collaborations.

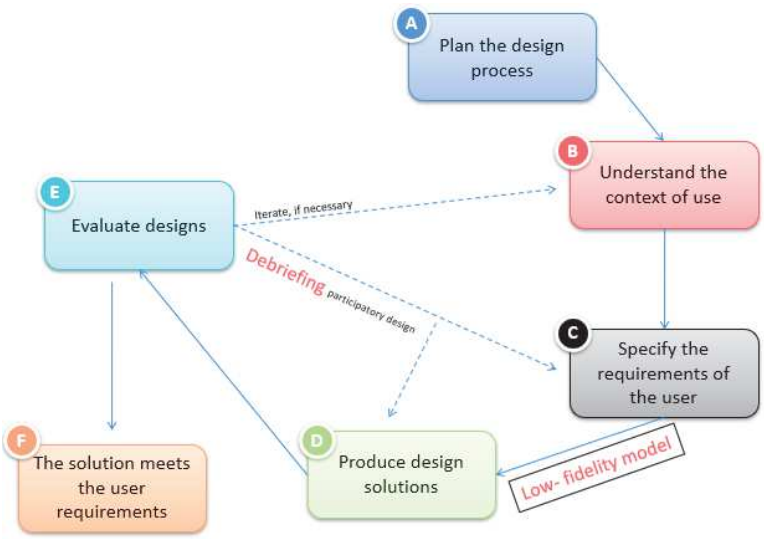

Fig. 1. Our user-centered design approach

Rather than forcing the users to change their behavior to accommodate our framework, we tried, in this work, to optimize the framework's structural interoperability

level around how users can, want, or need to use it. Inspired by the user-centered design approach [7], we studied components generation methods and proposed a prototype-based on the generation tasks to be performed - for partial automation. The idea is to allow the designers to preserve their tools, their favorite languages and their expertise in order to guide them for the co-simulation first step with other heterogeneous simulators.

2 Related work

The participatory design cycle begins with the analysis of user needs and activities. The ISO 16982 standard offers, for example, methods of user observation, questionnaires, interviews, or the study of available documents.
To achieve this phase with designers like those of the neOCampus operation, one should understand their needs to co-simulate (need for data, need to validate their simulation,etc.), their working environments, the tools they use, and their levels of expertise with computer (software, FMIs, GUIs,...). \\
Often in a phase following this first, it is useful to put into practice methods of creativity, such as brainstorming [13], to produce ideas for solutions. There are variants and more specialized methods like the "Group Elicitation Method" [14] which proposes "brainwriting", a written variant of brainstorming. In our case where it's difficult to bring together the different researchers working in neOCampus, we assumed that this phase isn't that relevant at that moment while we collected the main needs of our users and make sure of the importance of the implementation of a black-box components generation tool, and especially to save the learning effort of the FMI standard. We went then directly to the design phase, then replace this brainstorming by adding a debriefing step as shown in (cf. Fig. 1).
For the creation of solutions, the C and D phases of the process, there are many possibilities. The most common is to use low-fidelity prototypes. These are produced as in our case by us designers from the ideas generated collectively. They are used to present to users solutions to evaluate, validate or refute concepts or interactions, and to choose or propose new ideas. For the realization of these prototypes the designer has the choice between several methods. These are often based on the use of visual content.
Rettig [15] and Snyder [16] show, for example, the use of paper prototyping, in which manipulative and discussion interfaces are prepared in the form of drawings or collages. The "Wizard of Oz" experiment [17] proposes to simulate the interactive functioning of the final prototype. This methodology is often based on such a visual paper mock-up. Serrano [18] proposes with "Open Wizard" a software solution of magician of Oz for multimodal systems. It allows to simulate input modalities but does not allow the simulation of the output modalities.
An alternative to the Wizard of Oz is to code low-fidelity prototypes. According to Sefelin [19], the results achieved with these prototypes are equivalent to those obtained with paper models. In addition, the interviews conducted at the end of the tests comparing paper models and software prototypes reveal that 22 of the 24 subjects say

they prefer working with software prototypes. New technologies like Adobe Flash or MS Silverlight make it easy to create low-fidelity prototypes.

We coded a low-fidelity prototype which addressed two very interesting points:

1. Adequacy between proposed functionalities and user needs
2. Adequacy between the interface and the users

Designers working in the neOCampus project mostly use COTS (commercial off-the-shelf) simulation software [3] to build and test their simulation models. Integrating these models to form a single meta-model is a major issue especially when distributed simulation technologies are not anchored in these software [6]. We have seen that because of the lack of communication and collaboration between these different designers, their models were built completely in a disconnected way and do not benefit from the exchange of information that can simplify and accelerate their work.

Beyond these practices, another problem identified concerns the difficulty of being able to generate the co-simulation components (essentially components implementing the FMI) from the different simulators used. Indeed, the majority of designers are experts in their field but 1/ are not necessarily computer experts and 2/ if they are, not often expert in co-simulation nor in FMI. This time of taking over the FMI technology (time and practice) has led us to propose a mechanism to support the generation of FMU components based on user practices. It's an interface that is intended to be easy to use for novice users, adapted and allows to accompany the process of generation of components ready to connect to our platform (cf. Fig 2).

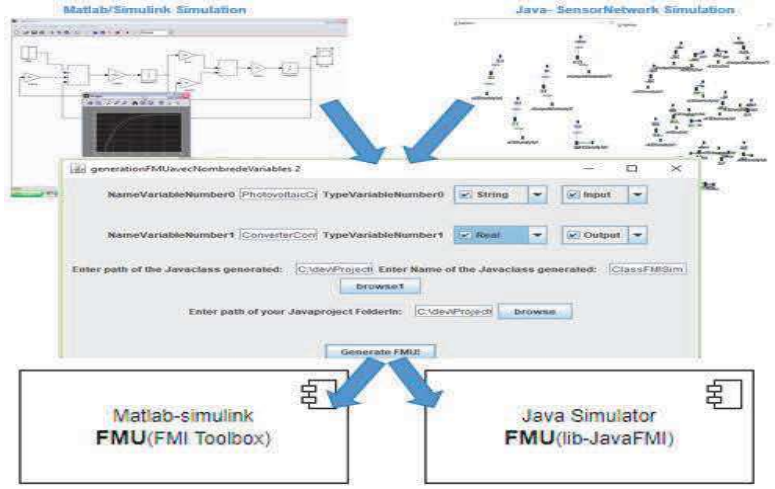

Fig. 2. Overview of the FMI component generation process

Many efforts have been made aiming to implement and test the FMI standard since its release. [8] discusses a generic interface's implementation and technical problems and challenges helping the importation of FMUs into a simulator. [9] describes the implementation of FMI in SimulationX. In [10] an integration strategy for rapid pro-

totyping for Modelica models into the FMI standard is presented. Since the modelling and simulation step is done separately by each designer, where the model has to be set up and tested against its specifications first. Standalone executables may be launched, traced, and debugged using additional tools like an IDE (Integrated Development Environment). The next step in the FMU generation process is to determine the interface of the simulation model, which is later exposed through the FMI. This consists of the definition of input and output quantities, or states, as well as internal timing (accuracy and precision required by the simulation model) and external timing (simulation step size for data exchange considerations. These informations are gathered with the information about the FMUs architecture in a modelDescription.xml file, which is connected to the software code using functions usually provided by the FMU SDK (software development kit) [11]. Some other issues we are facing is filling-up the gap between the semantics of FMI and the semantics of the source formalism of the diverse calculation models (state machines, discrete event, data flow or timed automata) [12].

The parts of an FMU are put together after being compiled. A zip file is shaped and the ".dll file" is put in the binaries folder for the interrelated platform. The modelDescription.xml file is set in the root folder. It is then a deliberate choice to put the model source files in the source folder.

We first analyzed the information about an FMU stored in the modelDescription.xml file. For example, the latter contains elements like ModelVariables defining all the variables of the FMU that are visible / accessible via the FMU functions. This inventory made it possible to extract the minimum necessary and sufficient variables for a simulation in order to generate as easily as possible the FMU component allowing co-simulation between systems.
Then, we built a task model to understand the steps to be performed, in order to build an FMU component, and proposed several models of which (cf. Fig. 3).

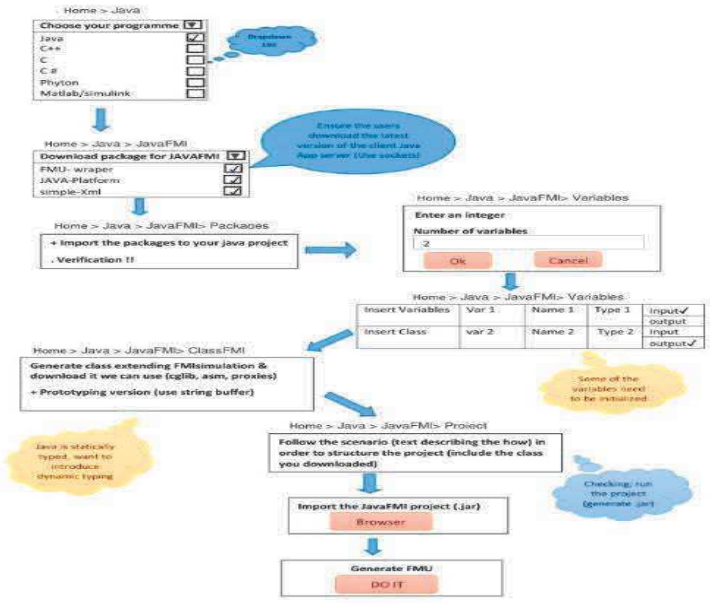

Fig. 3. low-fidelity models implementing the scenario

These steps were essential to identify the automatable steps of those where the actions of the user are essential.
Finally, based on these identified tasks, we proposed a "medium fidelity" interface, functional on an identified scenario for which we conducted a pre-experiment.

3 Preliminary study

As mentioned earlier, we found in surveys that a large majority of users were unfamiliar with FMI technology and had trouble "to jump in". That's why we wanted to check that using an interface, to help with the elaboration of FMU components, made sense for the designers (to understand the process and allow to collaborate) and that it was useful (in terms of time saving in the construction of a component for example, ...).

3.1 Participants and procedure

Participants: We conducted this pre-experiment with 7 novice participants and FMI experts. 6 participants were adult men (mean age = 20 years old, standard deviation = 5) and 1 woman (age = 27 years old). Participants were recruited from xxxx, xxxx and xxxx laboratories and were familiar with computers.

Equipment: We have developed the prototype under Java/QT on a laptop running Windows 10 OS (Core I7, 32 GB RAM, 17" screen)

Procedure: After signing the consent form, we exposed the two tasks to be performed for the pre-experimentation.

Task (1): an open program in the Eclipse framework was provided to the participants. The program was the same for all participants. They were asked to generate an FMU component using the latest version of the JavaFMI library and following the script available via a tutorial we provided them with (library download, class creation, component generation fmu2).

For the second **task** (2), we asked them to launch and use the graphical interface, helping with the FMU component's generation, that we developed. This interface proposes to guide the user in the generation process from the number of input and output variables, entering names and types of variables, respect of cast and spell-checker, until the download of the generated component, providing a summary of the performed tasks (cf. Fig. 5).

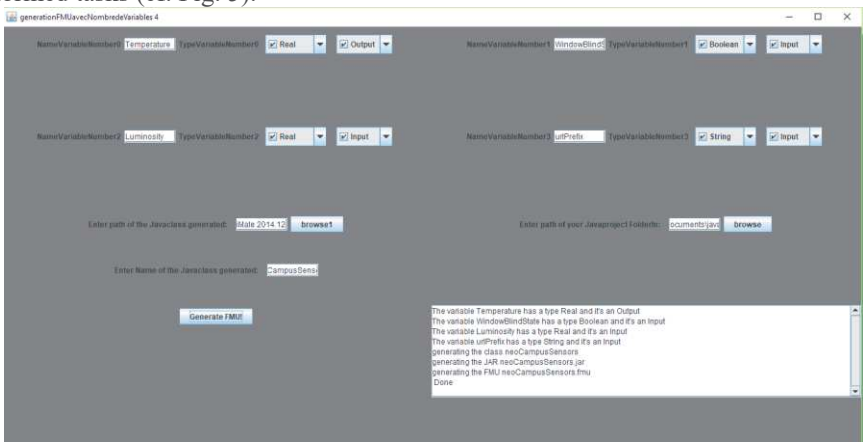

Fig. 4. screenshot of the proposed interface

Five participants (number P1 to P5) performed the task (1) then (2) and the last two ones (number P6 and P7) performed the task (2) before (1) to counterbalance learning effects.

Analysis: We recorded and analyzed the processing time of the different participants either by following the prescribed scenario or by using our interface. The number of actions performed was also recorded and compared to the fact that the participants were FMI experts or not, or if they used to work with integrated development environments (and more specifically Eclipse).

Independent Variables: with interface, without interface.

Dependent Variables: time (continuous variable), number of actions (discrete variable), FMI expertise: expert or not (categorical variable), experience related to frequency of use of IDE (categorical variable).

The aim is both to verify our initial hypotheses (the main objective of the tool is to allow non-experts to generate FMU components for co-simulation) and to iterate on our solution. We completed the pre-experimentation phase by debriefing with the participants.

4 Results and discussion

Users gave very mixed opinions on the experience, but most of them told us that following the script and trying to complete the program, to compile it and run it was a daunting task. In fact, one of these participants decided to leave the experiment after 32 minutes and found that the exercise was difficult.

6 users could reach the end of the task (1) but actually only 3 were able to generate a FMU component (two of which were experts FMI and one who frequently used the IDE and java as the main language in everyday work). The other 3 committed code errors that blocked this generation.
On the other hand, the generation of the FMU component using our interface (task 2) was successful by all the participants.

We were able to realize that the order of the tasks did not matter much regarding the results in terms of time of completion of the task.

Table 1. results – task completion time

Participant	Time (1)	Time (2)	IDEuses	FMI expert
P1	25’	3’	4	oui
P2	32’ -ab.	5’	4	non
P3	42’	7’	3	non
P4	65’	14’	1	non
P5	54’	8’	2	non
P6	34’	10’	3	non
P7	30’	13’	4	oui

(cf. Table 1.) describes the times performed by the 7 participants (participants P6 and P7 first performed task (2) before (1)).

Participants are distinguished according to their level of FMI expertise and their use of IDEs (with scale: 1: never, 2: rarely, 3: regularly, 4: all the time).

There is systematically (cf. Fig. 6) a longer realization time (at least a factor of 3) for the execution of the task (1) than for the task (2) regardless of their expertise. The ANOVA analysis also reveals a significant effect of the experience of an IDE on the time of completion of the task ($F(1,8)=50.02$, $p < 0.001$).

Fig. 5. Task completion time for tasks (1) and (2)

With only two expert users, we cannot say much about the impact of the expertise on the time of completion of the task. Nevertheless, we can note here again a saving of time of realization by using our interface, independently of the degree of expertise of the subject. The number of actions performed (being analyzed) is also sharply down (by a factor of 8).

5 Conclusion and future works

This preliminary work enabled us to first make an inventory of the practices of the different actors of the neOCampus project. In order to allow the different experts to communicate and collaborate, we understood that it was preferable that they could first and foremost keep their own practices by allowing them to build or improve their own “business” simulator.

We therefore opted for the interoperability of these heterogeneous simulators based on the FMI co-simulation standard overcoming model semantic gaps and offering them a validation platform for co-simulation. In this process, we have targeted one of the most difficult tasks, which is the generation of pluggable components in our platform, from heterogeneous simulations. We have therefore designed an interface facilitating this generation by focusing on automation with an understanding of the generation process. Hence, we pre-evaluated our interface in order to improve it. Our interface will aim the integration of the total control of the co-simulation with potentially visualization tools adapted to each participant in this co-simulation according to its needs.

The experience was enriching and appreciated. This constantly improving interface will not only be a mechanism to facilitate collaboration and simplify the co-simulation of our different systems, but can also be a plus for the FMI community that in all areas and for different uses is brought to generate components and for some languages and simulation environment for which there is currently no documentation or library to do so.

References

1. G. Bar-Yam, Y. (1997). Dynamics of complex systems (Vol. 213). Reading, MA: Addison-Wesley.
2. Rowson, J. A. (1994, June). Hardware/Software Co-Simulation. In DAC (Vol. 94, pp. 6-10).
3. C. A. Boer and A. Verbraeck, "Distributed simulation and manufacturing: distributed simulation with cots simulation packages," in Proceedings of the 35th conference on Winter simulation: driving innovation. Winter Simulation Conference, 2003, pp. 829–837.
4. M.-P. Gleizes et al., "neocampus: A demonstrator of connected, innovative, intelligent and sustainable campus," in International Conference on Intelligent Interactive Multimedia Systems and Services.Springer,2017, pp. 482–491.
5. Y. Motie et al., "A co-simulation framework interoperability for Neo-campus project (regular paper)," in European Simulation and Modelling Conference (ESM), Lisbon, 25/10/2017-27/10/2017. EUROSIS, 2017
6. S. J. Taylor et al., "Integrating heterogeneous distributed cots discrete-event simulation packages: an emerging standards-based approach, Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, vol. 36, no. 1, pp. 109–122, 2006.
7. Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User- centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications, 37(4), 445-456.*
8. Chen, W., Huhn, M., Fritzson, P.: A generic FMU interface for Modelica. In: 4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, pp. 19–24 (2011)
9. Noll, C., Blochwitz, T.: Implementation of modelisar functional mock-up interfaces in SimulationX. In: 8th International Modelica Conference (2011)
10. Elsheikh, A., Awais, M.U., Widl, E., Palensky, P.: Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface. In:2013 Workshop on Modeling and Simulation of Cyber-Physical Energy SystemsMSCPES 2013, pp. 1–6 (2013).
11. Qtronic: FMU SDK: free development kit (2014)
12. S. Tripakis, "Bridging the semantic gap between heterogeneous modeling formalisms and fmi," in Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS). IEEE, 2015, pp. 60–69
13. ISO/TR 16982. Ergonomics of Human-System Interaction–Usability Methods Supporting Human-Centred Design. 2002
14. Boy, G. A. (1997). The group elicitation method for participatory design and usability testing. interactions, 4(2), 27-33.
15. Rettig, M. (1994). Prototyping for tiny fingers. Communications of the ACM, 37(4), 21-27.
16. Snyder, C. (2003). Paper prototyping: The fast and easy way to design and refine user interfaces. Morgan Kaufmann.
17. Kelley, J. F. (1984). An iterative design methodology for user-friendly natural language office information applications. ACM Transactions on Information Systems (TOIS), 2(1), 26-41.
18. Serrano, M., & Nigay, L. (2009, October). OpenWizard: une approche pour la création et l'évaluation rapide de prototypes multimodaux. In Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine (pp. 101-109). ACM.

19. Sefelin, R., Tscheligi, M., & Giller, V. (2003, April). Paper prototyping-what is it good for?: a comparison of paper-and computer-based low-fidelity prototyping. In CHI'03 extended abstracts on Human factors in computing systems (pp. 778-779). ACM.
20. G. Hippmann, M. Arnold, and M. Schittenhelm. Efficient simulation of bush and roller chain drives. In Proc. Multibody Dyn., ECCOMAS Conf., 2005.
21. T. Blochwitz, M. et al., Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models, in Proceedings of the 9th International MODELICA Conference: September 3-5; 2012, pp. 173-184, DOI 10.3384/ecp12076173.