TOWARDS A UNIFIED FRAMEWORK FOR HOLISTIC STUDY AND ANALYSIS OF DISCRETE EVENTS SYSTEMS

Hamzat Olanrewaju Aliyu[1,2]

[1]School of Info. and Comm. Tech.
Federal University of Technology
Minna, Nigeria
hamzat.aliyu@futmina.edu.ng

Mamadou Kaba Traoré[2]

[2]LIMOS CNRS UMR 6158
Université Blaise Pascal
Clermont-Ferrand, France
traore@isima.fr

ABSTRACT

Simulation-based and Formal Methods (FM) approaches are often used to provide solutions to complex problems through analysis of system properties. Significant theoretical advancements have been made over the years in the area of developing formalisms and solution algorithms for system-based computational science; however, most existing system development environments are dedicated to certain analysis methodologies thereby making it (almost) a necessity to always build separate models of the same system for different analyses. Therefore, exhaustive analysis of systems often require more modeling efforts and skills since no single method is sufficient to investigate all aspects of a system. In this research, we propose a blueprint for developing an extensible multi-objective framework to harness the synergy of disparate approaches for holistic analysis of Discrete Events Systems (DES). The proposed methodology relies on model-driven development (MDD) techniques to derive models for disparate analysis objectives from a unified reference model at the kernel of the framework.

1. INTRODUCTION

System-based approach to problem solving involves the modeling of complex problems as systems consisting of interacting components. Such system models are evaluated and analyzed using techniques such as simulation, FM and enactment to explore the properties of interest. For example, simulation techniques may be used to investigate system's behavioral (dynamic) properties under the constraints and scenarios defined by the experimental frame while FM provide the rigorous logical tools to investigate the fulfillment of certain system invariants and properties such as safety, liveness, completeness and other specific requirements that must always be met. Enactment refers to the execution of a software prototype of the system to verify its behavior in real time. i.e., the scheduling and execution of events are done using real physical time. It usually allows the execution of activities (actions that do not lead to state transitions) associated with states and run-time interaction with the physical environment (e.g., human-in-the-loop). Since no one method is suitable to exhaustively study all aspects of a system, complementary use of disparate approaches is often required for a holistic analysis of a complex system.

The last few decades have witnessed considerable advancements in the development of formalisms and solvers for the study and analysis of system properties at different levels of abstraction; their practical adoptions are however not commensurate with their existence. This shortfall has been linked to the need for prerequisite mathematical skills for dealing with most formalisms and solution algorithms, limited collaboration between tools leading to limited model reuse, herculean task of creating and managing several models, etc.[1]. Another possible cause is that most of existing computational analysis environments allow to use only one analysis methodology (i.e., simulation or FM or enactment) and are usually not very easy to extend to accommodate other approaches. Thus, one must learn all the formalisms underlining the various methods to create models and go round to update all of them whenever certain system variables must change.

The possibility of realizing disparate analysis methodologies within one environment with minimal modeling skills and efforts and effective reuse of specifications underscores our motivation for the idea

proposed in this paper. A similar proposal was made in [2,3] to aggregate disparate Modeling and Simulation (M&S) resources in one framework and make them easily available to domain experts through a DEVS (Discrete Event System Specification) - based formalism as the front end for model specification. DEVS [4] was chosen as the principal formalism because it is considered to be a universal formalism for M&S [5]. We have explored the chances of extending this proposal to provide support for other analysis methodologies; therefore we consider to provide a more expressive formalism at the kernel of the framework by integrating DEVS concepts with concepts from Z language[6] and Object Orientation (OO)[7]. The same way DEVS is universal for simulation modeling, Z and OO are also considerably universal to model DES for logical analysis and enactment respectively. In order to meet its objectives, we expect the framework to satisfy the following essential requirements:

- Provide a unified high-level formalism at the front-end which is expressive enough to capture the system information required by the disparate analysis techniques.
- From a reference model specified with the unified formalism, automated generation of equivalent models for simulation, logical analysis and enactment should be possible.
- The unified formalisms should be able to serve as extension points to integrate the framework with legacy modeling tools and solvers
- The framework should be able to provide computational support for domain-specific formalisms and frameworks.

In the next section, we propose a blueprint to realize a framework that satisfies these requirements.

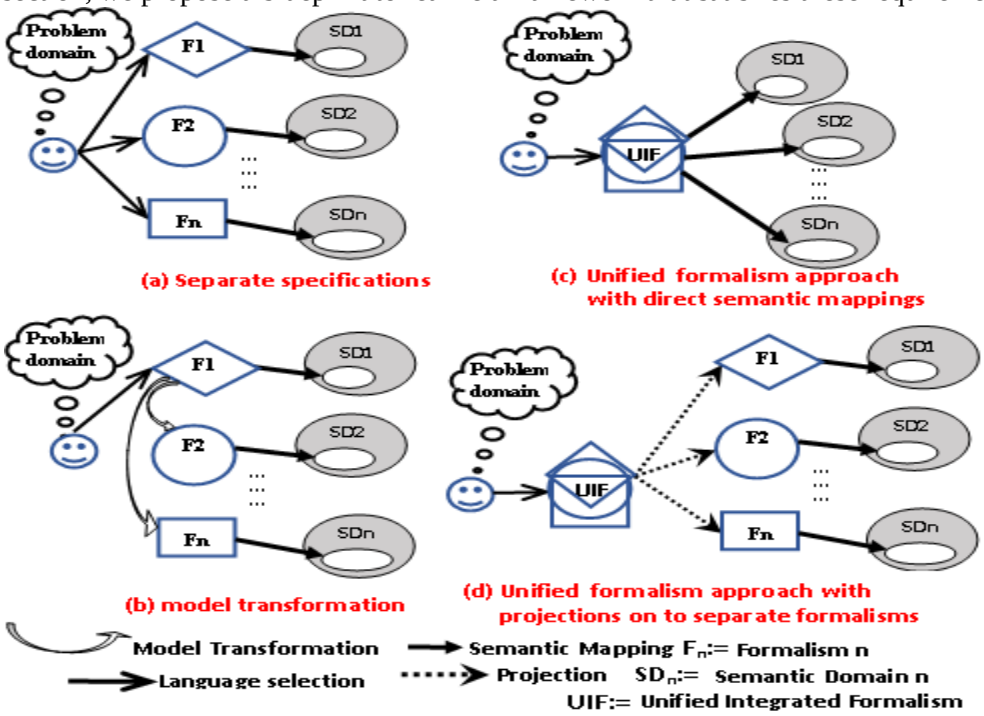

Figure 1 Model-based system analysis practices

2. PROPOSAL

The underlying principle of the approach we propose relies on model-driven development techniques to realize its objectives. We compare the approach with the common practices in model-based system analysis methodologies in Figure 1. Figure 1(a) describes the practice of manually creating separate models for the different analysis techniques which is characterized by a burdensome task of creating and updating several models of the same system.

The approach in Figure 1(b) tries to solve the problems of (a) by creating only one model based on one of the different formalisms and with the help of model transformation techniques, automate the

generation of models based on the other languages. Unfortunately, this approach is often constraint by the difference in the relative expressiveness of the F_1, F_2, .., F_n. Thus, there are often a few constructs in each of the target formalisms of the transformation that are not captured in the source language which must be manually provided. Hence, the task of creating and updating other models is reduced but not eliminated. Moreover, the user needs to be skilled in all the formalisms to perform the complementary modeling task. We propose a unified formalism approach as described in (c) and (d). The UIF is obtained by consistent integration of the concepts in the different formalism. Interestingly, though they have different objectives and expressiveness, DES formalisms have significant concepts that are common to their vocabularies. For example, they all express concepts such as states, transitions, input, output albeit at different levels of refinement.

Therefore, the expressiveness of UIF is technically the union of the common concepts and those specific to each of F_1, F_2, .., F_n. The semantics of UIF may be defined in some new domains as described in (c) or in F_1, F_2, .., F_n so that their associated semantics and tools are integrated into the framework.

Based on this principle, we propose a framework in Figure 2 for holistic analysis of DES. In Figure 2, the metamodel of UIF (at the centre of the diagram) is defined by integrating concepts from DEVS, Z and OO to specify a formalism that is expressive enough to realize the framework's objectives; the "UIF-based model editing" mechanism provides high-level concrete syntax for even non-experienced users to specify DES models in a generic way that is not completely tied to any of the anticipated analysis methods. Simulation, Logical analysis and Enactment engines may be implementations of Figure 1(c) or (d). With the help of MDD techniques, the user can specify a high-level UIF-based model and automatically generate the artifacts required by each of simulation, logical analysis and enactment engines.

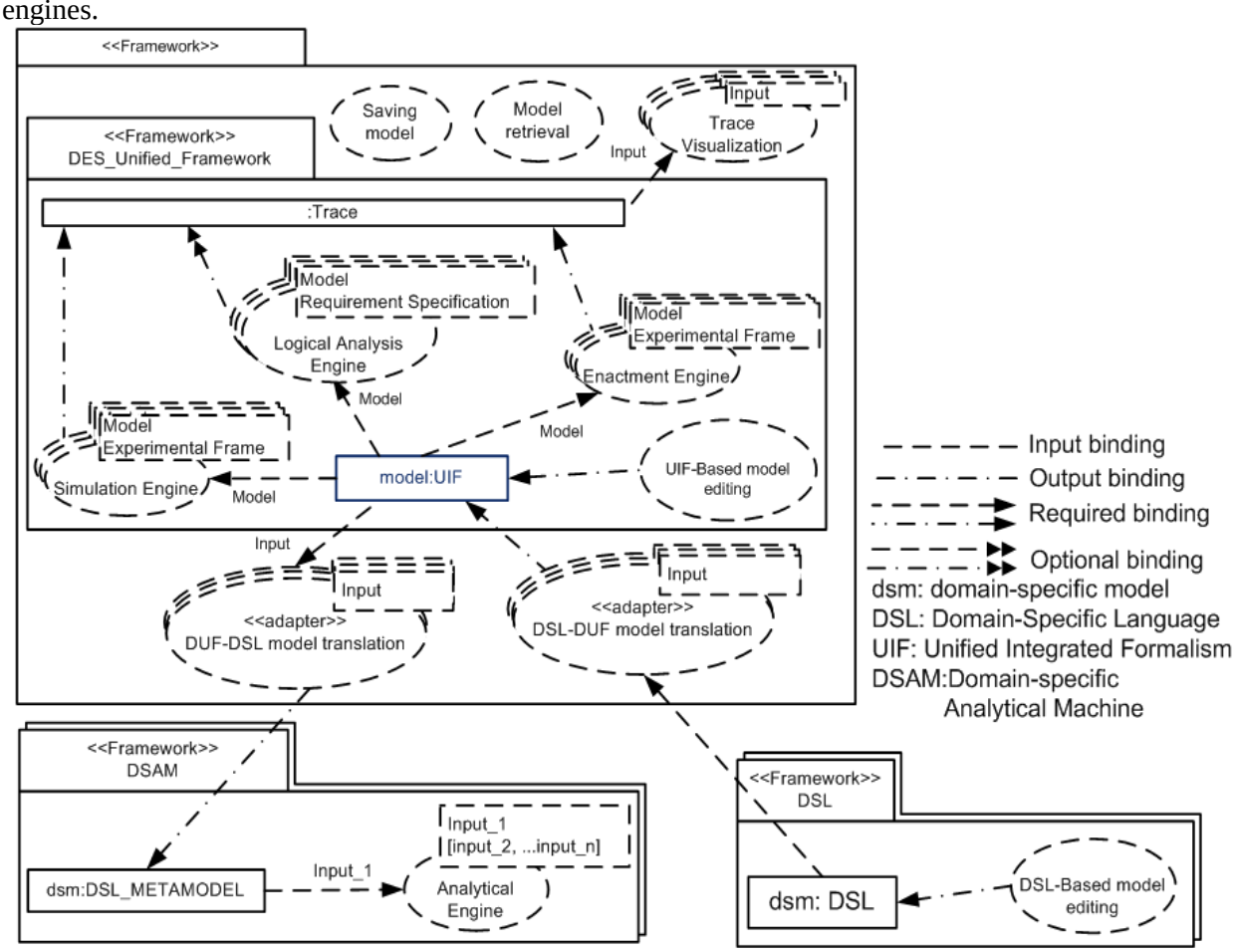

Figure 2 A blueprint for a framework for holistic analysis of DES

In this way, the task of creating and updating models is reduced since the user deals with only the reference model from which others are derived.

UIF serves as the extension point through which various analysis tools can be progressively added to enrich the framework with the capabilities they offer. Such legacy analysis engines (described as DSAM in Figure 2) may be connected with the framework through a DUF-DSL adapter to create the necessary compatibilities between the UIF and the metamodel of DSAM. The framework may also be used to serve as a computational engine for external DSLs through a DSL-DUF adapter. For example, given a DSL that allows a domain expert to concentrate on describing domain concepts, we may generate from the DSL's model, a model that conforms to the UIF so that the DSL can benefit from the different kinds of analysis that may be possible with the framework.

3. RESULTS

We have defined a language that has the potentials to realize the visions of the UIF as described in the previous sections by combining system constructs from DEVS, Z and OO. Though the language's editor is currently under construction, we have been able to reach a considerable level of confidence using handcrafted models and transformations that the goal of the framework is achievable. When completed, intend to first integrate the language's metamodel with those of existing simulation and analysis tools using appropriate MDD techniques to automate the process of moving from UIF-based high level models to the different analysis methodologies.

4. CONCLUSIONS

We have presented an abstract framework for holistic analysis of DESs. The framework has a unified formalism, UIF, at its kernel to specify generic reference models that is believed to capture all system information required by simulation, logical analysis and enactment methodologies. The objective of the framework is to make theoretical advancements in model-based computational science available to prospective users especially domain experts with limited knowledge of abstract formalisms. This is achieved by providing a high level generic front end to the user so that a general model specified can be fed to the different mechanisms to provide the artifacts required for the corresponding analysis methodologies. For further research, we are currently developing an implementation of a UIF that is built on concepts from DEVS, Z and OO for the realization of such frameworks.

REFERENCES

[1] Vittorini, V., Iacono, M., Mazzocca, N., and Franceschinis, G., 2004. The OsMoSys approach to multi-formalism modeling of systems. *Software and Systems Modeling*, *3*(1), 68-81.

[2] Touraille, L., Traoré, M. K., and Hill, D. R. 2011. "A model-driven software environment for modeling, simulation and analysis of complex systems". In *Proceedings of the 2011 Symposium on Theory of Modeling and Simulation: DEVS Integrative M&S Symposium* (pp. 229-237). SCS International.

[3] Traoré, M. K. 2008. "SimStudio: a next generation modeling and simulation framework". In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems and workshops* (article no. 67). ICST.

[4] Zeigler, B. P., Praehofer, H., and Kim, T. G. 2000. "*Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*". Academic press.

[5] Vangheluwe, H. L. 2000. "DEVS as a common denominator for multi-formalism hybrid systems modelling". In *Computer-Aided Control System Design, 2000. CACSD 2000.* (pp. 129-134), IEEE.

[6] Spivey, J. M. 1988. "*Understanding Z: a specification language and its formal semantics*". Cambridge University Press.

[7] Martin, J., and Odell, J. J. (1994). *Object-oriented methods*. Prentice hall PTR.