Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

TOWARD A MODEL-DRIVEN ENGINEERING FRAMEWORK FOR REPRODUCIBLE SIMULATION EXPERIMENT LIFECYCLE MANAGEMENT

Alejandro Teran-Somohano

Industrial and Systems Engineering
Auburn University
Auburn, AL 36849, USA

Orçun Dayıbaş

Department of Computer Engineering
Middle East Technical University
Ankara, TURKEY

Levent Yilmaz

Computer Science and Software Engineering
Auburn University
Auburn, AL 36849, USA

Alice Smith

Industrial and Systems Engineering
Auburn University
Auburn, AL 36849, USA

ABSTRACT

Goal-directed reproducible experimentation with simulation models is still a significant challenge. The underutilization of design of experiments, limited transparency in the collection and analysis of results, and ad-hoc adaptation of experiments as learning takes place continue to hamper reproducibility and hence cause a credibility gap. In this study, we propose a strategy that leverages the synergies between model-driven engineering, intelligent agent technology, and variability modeling to support the management of the lifecycle of a simulation experiment. Experiment design and workflow models are introduced for configurable experiment synthesis and execution. Feature-based variability modeling is used to design a family of experiments, which can be leveraged by ontology-driven software agents to configure, execute, and reproduce experiments. Online experiment adaptation is proposed as a strategy to facilitate dynamic experiment model updating as objectives shift from validation to variable screening, understanding, and optimization.

1 INTRODUCTION

Simulation involves goal-directed experimentation with dynamic models for a variety of purposes, including scientific discovery, training, education, and entertainment. Experiment design for simulation is a well-developed area with classical references such as (Law and Kelton, 2000) and (Kleijnen, 2005). The significance of methodological support for experimentation gave rise to the field of statistical Design of Experiments (DoE) to produce objective and unbiased results. DoE allows practitioners to construct well-defined procedures for sampling experiment outcomes while offering a common framework that can be used to repeat and validate those outcomes.

Earlier studies in computer assistance in experiment management include support for addressing statistical issues in simulation-based problem solving (Tao and Nelson 1994) as well as proposals for using expert systems in experiment design (Ören 2001; Wilson et al. 2000). More recently, scientific workflow systems have emerged as infrastructures to provide means for specifying, deploying, and executing scientific data generation and analysis activities in the form of workflow activities. For instance, Taverna (Oinn et al. 2008), myExperiment (De Roure et al. 2009), and the open-source Kepler project (Altintas et al. 2004) are major systems that allow users to define computations in the form of activities, which are then composed and orchestrated to support computational experiments. However, general-

purpose workflow systems are not intended to embody either statistical knowledge (e.g., input data modeling and output analysis) or experiment design and adaptation expertise needed for continuous management of stochastic simulation experiments. The strategies and tactics used in simulation experiments are often left implicit.

Our proposed approach defines simulation experiments in terms of a lifecycle with distinct phases, each with specific requirements and opportunities for computational assistance. The features of experiments and the governing processes are defined in terms of explicit experiment models, which can be made available at run-time and be interpreted by intelligent agents. By closing the loop via an experiment life-cycle, we propose online adaptation of experiment models based on feedback received from previous iterations. As such, experiment models are used as explicit run-time introspective models to assist the experimentation process. Furthermore, experiments are defined at multiple levels of abstraction, starting with a feature model that uses the vocabulary of the experiment domain expert. Agent-assisted mapping of the feature model onto a more specific experiment domain ontology, followed by model transformation, results in executable scripts for batch-mode experimentation.

The proposed approach is founded on three critical pillars: (1) model-driven engineering, (2) agent-assisted workflow systems, and (3) design of reproducible experiments. The synergistic interactions between these elements are leveraged to improve computational assistance for each phase of the life-cycle. Ontology modeling is used to specify the structure of the experiment model and to allow the use of model transformers to shift the focus in experiment design to high-level features that are then compiled into executable experiment scripts. Agents are used to prune the experiment ontology space to relevant DoE concepts and attributes that are related to features selected by the experiment designer. Feature-oriented domain analysis is used to support experiment variability modeling to represent standard features in simulation experiments as well as the relationships and constraints among those features.

The rest of the paper is organized as follows. Section 2 presents the background on the aforementioned areas that converge in our framework. These topics include experiment reproducibility and replicability, agent-assisted scientific workflows for standardizing experimental procedures, and model-driven engineering for simulation experiment management. Section 3 introduces our proposed framework and its major components. Section 4 delineates future directions in research, and section 5 concludes with an overview of the lessons learned.

2 BACKGROUND

The proposed simulation experiment management framework harnesses the synergies between model driven engineering, intelligent agents, scientific workflows, and statistical design of experiments. The premise of this work is based on increasing awareness of the emergent credibility crisis in computational research in general, and simulation research in particular (Yilmaz and Ören 2013). Knowledge that cannot be reproduced and replicated is not considered as scientific knowledge. Therefore, if Modeling & Simulation (M&S) is to sustain its credibility as a scientific discipline, it needs to improve reproducibility and replicability.

Reproducibility refers to the ability to repeat simulation experiments under controlled conditions to regenerate results using existing simulation code and data. On the other hand, replicability involves independent derivation of results by creating a new implementation. However, the implementation of the replicated model differs in some way (e.g., platform, modeling formalism, language) from the original model, but should be an executable representation to facilitate conducting the same experiments. Similar to other fields within computational science, reproducibility and replicability of simulation experiments must involve variation and sweeping of the parameter space, activity management, and the use of a software platform. Besides, in computer simulation, there are additional challenges involved in the management of simulators and simulation algorithms, the use of different simulation paradigms, and poorly documented assumptions about experiments.

Scientific workflow systems have emerged as problem solving environments that allow the formalization and execution of processes that a scientist follows to derive publishable results from raw data (Altintas et al. 2004). Such systems allow scientists to define analysis tasks in terms of an executable activity-flow. To improve reproducibility and replicability, scientific workflows can be reused and shared with others, who can then deploy and repeat workflows in their own platforms. Among the scientific workflow platforms commonly used by scientists include Taverna, Kepler, and myExperiment. worMS is a workflow system prototype that exclusively targets M&S software (Rybacki et al. 2011). It aims to integrate workflow support to M&S software to facilitate simulation set-up, execution. and analysis.

While workflow systems can be effective in both streamlining the M&S activities and improving their reuse, they have the following limitations: (1) workflows are tightly coupled to a concrete implementation over a specific platform, (2) the process and the data are also coupled, making it difficult to vary them independently, and (3) workflows do not embody expert knowledge to manage and update experiment specifications to enforce experiment design constraints as learning takes place. Furthermore, workflow designers need to be aware of algorithmic and computational details pertaining to specification and execution of workflows. That is, workflow systems focus on the definition of imperative activity specifications, instead of synthesizing them from higher-level computation-independent declarative experiment feature and domain models. However, the adoption, reuse, reproducibility, and independent replication and extension of existing workflows will be improved if users are shielded from platform and computation-specific aspects of workflow management. Instead, end-users should focus on the strategic and tactical aspects of simulation experiments.

As shown in Figure 1, the Model-Driven Engineering (MDE) methodology (Gaševic, Djuric, and Devedžic 2009) provides a framework and strategy to move from the platform-independent experiment domain space to the technical space involving platform-specific executable simulation experiment scripts. The experiment domain is often defined in terms of the terminology of the DoE, whereas the Executable Script Language (ESL) such as NIMROD (Peachey et al., 2008) deploys the experiment to batch-run the simulation on a specific platform.

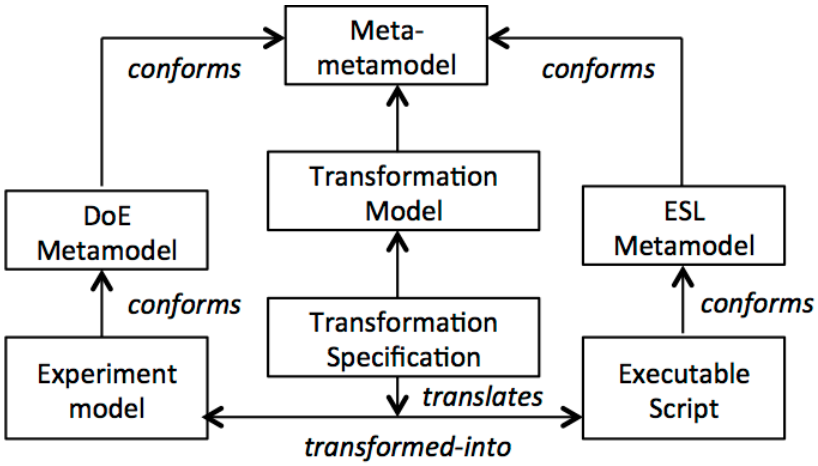

Figure 1: MDE for Experiment Model Transformation.

The MDE perspective focuses on the specification of the experiment modeling language as well as the transformation rules for mapping experiment conceptualization space to the realization (implementation) space. Transformation rules can also be defined to map concepts in one technical space (e.g., DoE) to another (e.g., NIMROD). Similar to the way we define simulation models as abstractions of a phenomena in the real world, meta-models are used to conceptualize and define the abstract syntax of modeling languages. That is, a meta- model is yet another abstraction that specifies the properties of the

model itself. A model conforms to its meta-model in a way that a program conforms to the grammar of the programming language in which it is developed.

An experiment has both structural and dynamic aspects. Dynamic perspective involves systematic, iterative processes seeking answers to a set of questions about the system being simulated. The questions to be answered determine the objective (or objectives) of the experiment. The unfolding of this process can be described in terms of a life-cycle. Hence, as shown in Table 1 we can define an experiment life-cycle in terms of the steps required to reach the experiment's objectives. Though it is stated that experiments require multiple iterations to reach their objectives, we believe that there has not been enough emphasis on this point. Our approach encompasses these phases and stages, while explicitly putting them in an experimental cycle, where the experiment itself is computationally modified as it progresses towards its objective.

Table 1: Phases and stages of the experiment life-cycle (Lorscheid, Heine, and Meyer 2012).

Phase	Stage	Output
I: Experiment Preparation	1. Experiment objective formulation	List of experimental objectives
	2. Variable classification	Variable classification table
II: Experiment Execution	3. Definition of response variables and factors	Response list, Factor table (including levels and values)
	4. Design selection	Experiment design matrix
	5. Estimation of experimental error variance	Number of required replications for statistical reliability
	6. Experiment execution	Final effect matrix
	7. Analysis of effects	ANOVA table, Effect strength and direction table, Factor and interaction significance table
III: Analysis of Experiment	8. Outcome analysis with respect to experiment objective	Updated response and factor list

The importance of strategic issues on proper design of experiments, as well as the tactical issues in collecting and analyzing data are highlighted in (Donohue 1994; Himmelspach et al. 2008; Kleijnen 2005; Kleijnen et al. 2005; Sanchez and Wan 2009; Kelton 2000; Lorscheid, Heine, and Meyer 2012). There also exist tools that support experiment management (Ioannidis et al. 1997; Perrone, Main, and Ward 2012; Leye and Uhrmacher 2012), but their focus is not the entire life-cycle of an experiment. Furthermore, prior work does not take full advantage of the synergies between MDE, agent technology, and workflow management systems to advance the state of the art and practice of experiment management.

3 CONCEPTUAL FRAMEWORK

Simulation experiments often consist of multiple iterations. At each iteration, to account for the observations, the experiment's structure and goals could change. To embody iterative nature of experimentation, we divide the experiment life-cycle into three steps: design of the experiment, execution of the experiment and adaptation of the experiment as learning takes place. The first step corresponds to stages 1 through 5 as shown in Table 1. The second step corresponds to stages 6 and 7. In the third step (corresponding, in part, to stage 8), the results of the previous step are used to determine whether the experiment objectives are met; if not, adjustments to the experiment design are recommended and the cycle is restarted with the design step. This section describes this framework in greater detail, starting with a high-level, conceptual view of the framework.

3.1 Components of the Simulation Experiment Management Framework

Our framework is based on the MDE principles. This is reflected in an abstraction layer where multiple abstract domain models define the operation of the framework along with the structure of the data elements. This additional layer facilitates the inclusion of intelligent agents that guide the process and assist the user, while ensuring the integrity of the data being generated. The framework architecture is shown in Figure 2. The three basic steps of an experiment life-cycle are shown enveloping the system components. These are ontology-assisted experiment design, experiment synthesis and execution, and experiment model adaptation.

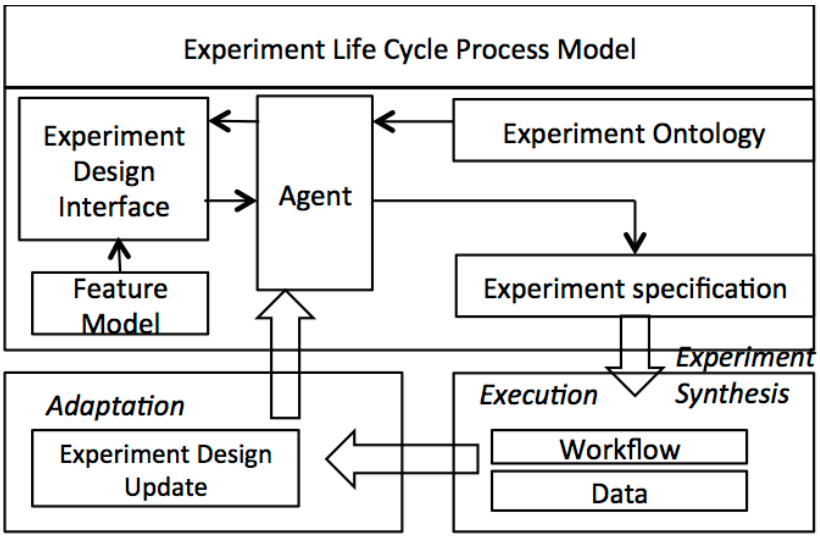

Figure 2: Simulation Experiment Management Framework Architecture.

3.2 The Experiment Life-cycle Process Model

The experiment life-cycle process model orchestrates the elements of our framework. It drives the design, execution, and adaptation of an experiment. Figure 3 depicts the UML-based activity-flow specification of the process model, including the produced and consumed data in each step. Though most of the activities in the diagram correspond to the phases and stages found in Table 1, the loop makes the iterative nature of the process explicit. The process model presents the activities that take place throughout the entire life-cycle of an experiment. Next, we examine what takes place in each step of the proposed design-execute-adapt cycle.

Following the determination of experiment objectives, goals, and questions of interest driven by a feature-model, the system categorizes the parameters into dependent, control, and independent variables. This classification facilitates specification of an experiment design. Following the synthesis and execution of the experiment, results are aggregated to a dependent variable specification defined in MathML. After the analysis of effect and determination of significant factors, experiment adaptation continues by re-classifying variables by categorizing insignificant factors as control variables. Furthermore, for optimization experiments, the adaptation agent can utilize a heuristic search or optimization method such as Genetic Algorithms to determine optimal values of dependent variables. Adaptation of the experiment model facilitates shifting the focus from variable screening to full factorial design, followed by optimization, or sensitivity and robustness analysis.

3.3 Design of the Experiment

The experiment life-cycle starts with the design of the experiment. In this step, the user is guided by an intelligent agent that interprets the experiment ontology to determine what inputs are required from the user to complete the selected design feature. At the end of this step, an experiment description—including experiment meta-data such as experimenter's name, experiment date, experiment design structure, data structures for the experimental results and the type of desired graphical output—is passed on to the next step and can be saved for future use.

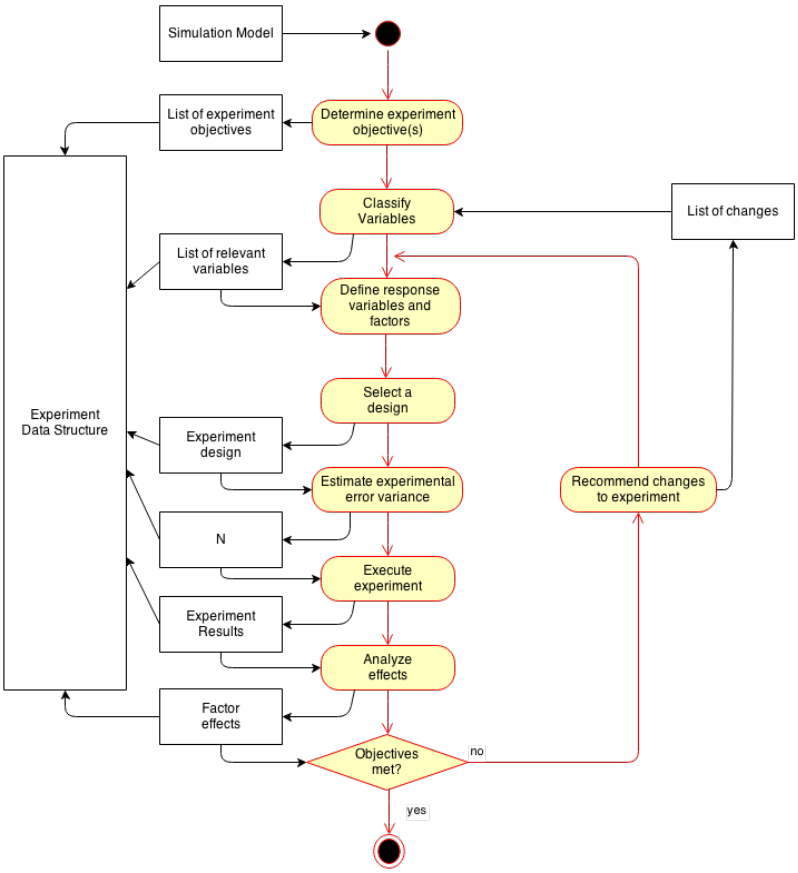

Figure 3: Top-level Experiment Activity Model.

3.3.1 Feature-Oriented Experiment Modeling

In designing the experiment, we distinguish between the goals, strategies, and tactics. The goals (questions) and the high-level of features of an experiment are driven by Feature-Oriented Domain Analysis. A feature model (Batory 2005) represents a family of experiments by allowing explicit specification of variability in the configuration of experiments. From the feature model, experiment designers can select configurable requirements to specify the features of the experiment. In effect, features serve as views into the experiment ontology, which specify the strategies and tactics necessary to implement the features. Features are categorized into two groups; simulation-based features (related to aspects such as model type, random number generation method, etc.) and DoE-based features (objective, number of factors, analysis method, etc.). In this paper we will focus exclusively on the DOE-based features:

- *Objectiv*e refers to the goal of the simulation experiment and influences the execution and analysis of the experiment. In general, there are three types of objectives. Comparative designs seek to compare the factor effects and are used for choosing between alternatives. Screening designs are used for monitoring the effects of one or more factors. Response surface design aims to reduce variation of results in a specific value range.
- *Number of factors* – given that computational resources are limited, the number of factors are key to designing efficient experiments.
- *Sampling Method* decides the values of factors for all design points.
- *Analysis Method* – the standard analysis method in DoE is analysis of variance (ANOVA). The design of the experiment, however, influences the specific application of that method. For instance, if there exists more than one response variable, MANOVA (Multivariate Analysis of Variance) is used. One-way ANOVA is used if there exists only one factor. Factorial ANOVA is used to study more than one factor with a single response variable.

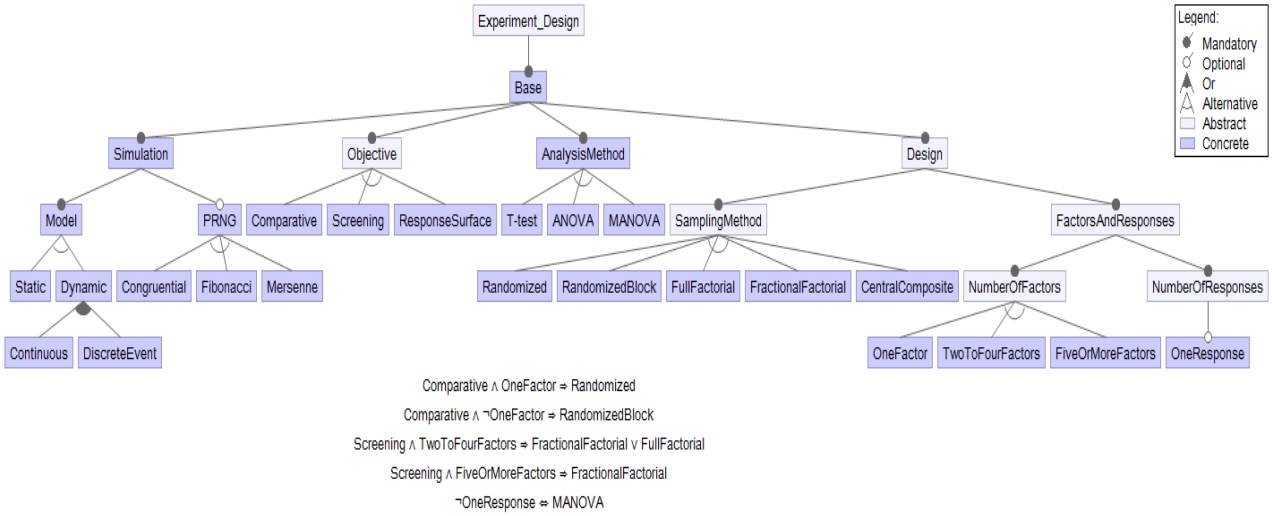

Figure 4: Provisional Feature Model for Simulation Experiment Design.

A provisional feature model, including the simulation-based features and constraints, is shown in Figure 4. In this model, abstract features are used to support the understanding of different concrete features, so they are not mapped to an implementation artifact. Concrete features, on the other hand, have specific mappings onto experiment implementations. For example, "Analysis Method" is mapped to the implementation of typical DoE plots (main effects mean or interaction plots) while the function of the "Objective" feature is simply to group the sub-features.

3.3.2 Experiment Ontology

The experiment description is constructed based on an experiment ontology. A provisional ontology is shown in Figure 5. The ontology encompasses the structural elements of an experiment, including the experiment's evolving objectives through one or more iterations. Each iteration contains an experiment design, its results, and the outputs. The experiment's design is defined in terms of a set of responses, factors, and factor levels. These are defined by the user during the design step and are updated, if needed, during the adaptation step. Based on the design of the experiment, as well as the experiment's objectives and the user's input, multiple data structures are created that are for use in subsequent steps of the experiment life-cycle. These structures contain the outcome of the experiment execution and the results of

the statistical analyses performed on that outcome. These structures include a variety of result tables, statistical analysis results and charts.

Since not every computer simulation practitioner is formally trained in statistical design of experiments, the framework would be enhanced significantly by including an intelligent agent providing the know-how and expertise required for designing experiments. The model-based approach provides us with the necessary infrastructure so that the agent's logic engine can make inferences and decisions based on the user's input and the expert knowledge it possesses. The agent navigates the ontology and recommends relevant concepts applicable for the selected features and objectives.

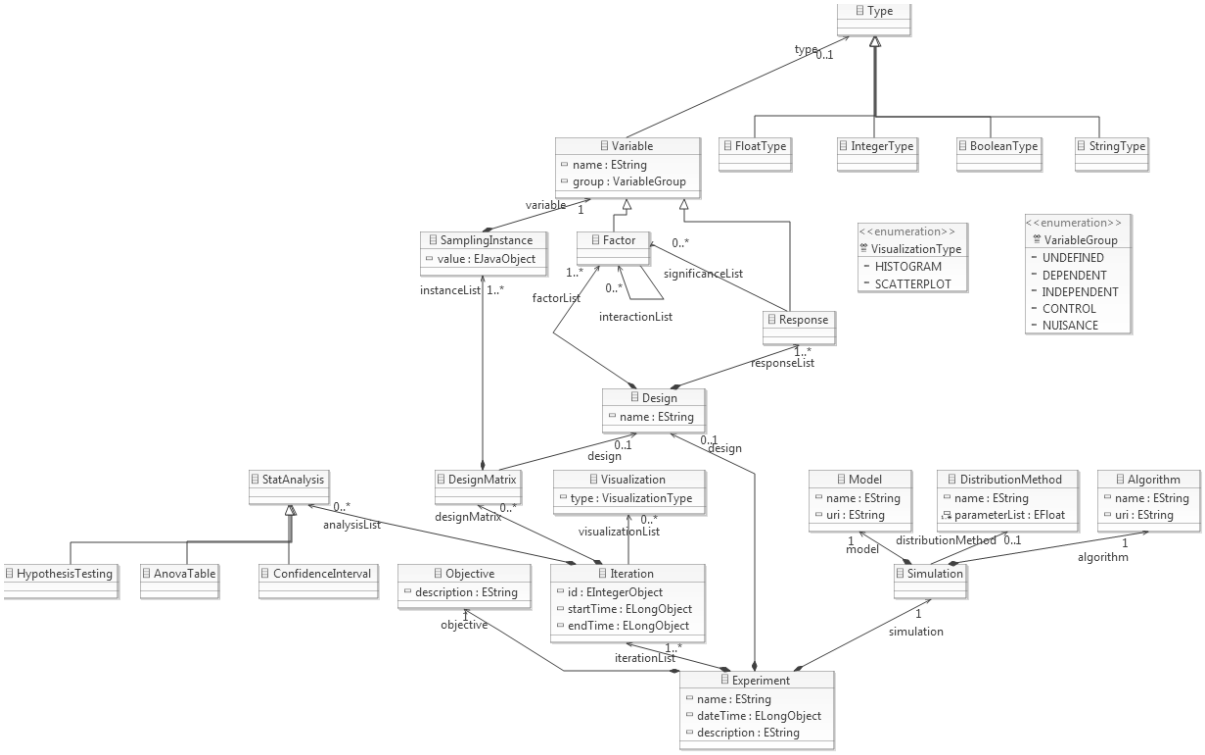

Figure 5: Provisional Experiment Ontology.

Consider the simulation model of a simple hypothetical queuing system shown in Figure 6. It is assumed that a simulation model description already exists and has the following information:

- Configuration parameters: λ, μ and k.
- Response variables: time in system, WIP, utilization for servers 1 through k

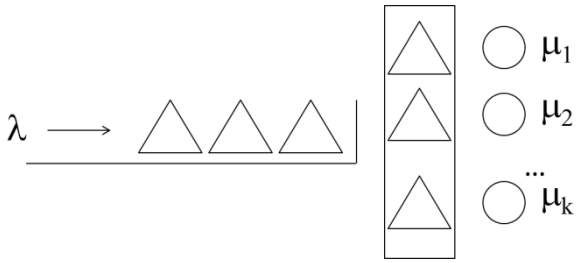

Figure 6: Queuing System

The experiment feature model defines the following features and possible values:

- Types of experiment objectives available: screen factors, optimize, and compare.
- Sampling method: randomized, randomized block, full factorial, fractional factorial, central composite.
- Number of factors: one, two to four, five or more
- Number of responses: one.

When the user creates a new experiment, an experiment skeleton in the form of Extensible Markup Language (XML) document is generated, containing only the most basic information about the experiment:

```
<?xml version="1.0"?>
<experiment description="Comparison between input factors and the time in system" date="01/04/2014" name="Experiment1" id="15">
    <experiment_objective description="" id="1" met="false"/>
  - <experiment_iteration number="1">
        <experiment_design name=""/>
        <experiment_results id="" comments=""/>
    </experiment_iteration>
</experiment>
```

Suppose the user selects the comparison objective, with the aim of comparing the effects of λ and k on the average waiting time in system. This requires the design to account for one response variable and two factors (λ and k), at an unknown number of levels. Once the user has input this information, the agent generates a preliminary list of designs that fit this description.

```
<?xml version="1.0"?>
<experiment description="Comparison between input factors and the time in system" date="01/04/2014" name="Experiment1" id="15">
    <experiment_objective description="Compares the effect of multiple factors on the response" id="1" met="false">Comparison</experiment_objective>
  - <experiment_iteration number="1">
      - <experiment_design name="">
            <response name="" type=""/>
          - <factor name="" type="">
                <factor_level level_number=""/>
            </factor>
        </experiment_design>
        <experiment_results id="" comments=""/>
    </experiment_iteration>
</experiment>
```

The user has two alternatives: either select a design directly from the list generated by the agent, or manually input the number of factors and their values, the factor levels, and any other relevant information, as well as any additional design constraints such as blocks, center points, etc. The agent would then assure that the input is consistent and fits one of the experiment design patterns. In this case, the user selects a 2^k factorial design with two factors, and is then requested to input the value of the two factor levels for each one of the factors.

```
<?xml version="1.0"?>
<experiment description="Comparison between input factors and the time in system" date="01/04/2014" name="Experiment1" id="15">
    <experiment_objective description="Compares the effect of multiple factors on the response" id="1" met="false">Comparison</experiment_objective>
  - <experiment_iteration number="1">
      - <experiment_design name="">
            <response name="Time in System" type="Double">time_in_system</response>
          - <factor name="Arrival rate" type="Integer" source="lambda">
                <factor_level level_number="1">10</factor_level>
                <factor_level level_number="2">15</factor_level>
            </factor>
          - <factor name="Number of servers" type="Integer" source="k">
                <factor_level level_number="1">2</factor_level>
                <factor_level level_number="2">5</factor_level>
            </factor>
        </experiment_design>
        <experiment_results id="" comments=""/>
    </experiment_iteration>
</experiment>
```

After the user selects a specific design, the agent proceeds as follows: (1) The type of statistical analysis is determined. In this case, 2^k factorial designs are analyzed using ANOVA. (2) There is now enough information to generate the experiment structure. This is shown in a results table, which includes a row for each factor level combination to be examined, with a cell corresponding to the response value. This table will be filled with the raw data of experiment outcomes. The order of the factor level combinations will be executed at random.

Once the design step is concluded and the experiment description has been completed (at least in its structural form), the experiment must be executed. This is done by, first, synthesizing the description of the experiment and transforming it into executable code; second, by executing the script; and, third, by retrieving the simulation outcome. The results of this step are then stored in the experiment description. Experimenting with different factor level combinations requires execution of the simulation multiple times with varying parameter values. To this end, the synthesis process generates a runnable script that can invoke the simulator to generate the behavior of the model with different parameter values. Once all the runs are completed, the data are aggregated and summarized for statistical analysis and chart generation. For instance, an ANOVA analysis can be performed on the raw outcome data. These analyses can also be stored in the experiment description and will serve as the basis for the adaptation step.

As was mentioned above, simulation experiments are iterative processes. Hence, each iteration requires the experiment structure to be modified according to previous results. However, the changes to the experiment design should not be ad-hoc. Rather, they are influenced by the experiment objectives and the results obtained. Therefore, the adaptation step of the experimental life-cycle requires agent-supported schema update. Those components of the experiment that can be modified as well as the configuration constraints are formally described in the feature model.

4 FUTURE WORK

The challenges for implementing the experiment management framework fall into two major categories: first, there are those challenges arising from translating expert experiment domain knowledge into specific rules and procedures; second, the transformation of abstract platform-independent experiment models into executable scripts for batch-running the experiments. In the first category, we encounter issues related to the development of an intelligent agent that exhibits expert design of experiments knowledge. The following are the pertinent issues: What are the rules that relate experiment objectives to specific designs? How should we change the design as we acquire new experimental data? In the second category, we have to deal with elaborating strategies and techniques for unifying the principles and software tools of MDE, scientific workflows, and feature modeling. As discussed in the introduction section, the experiment management framework is part of a larger effort to support simulation reproducibility and replicability. The proposed strategy offers avenues of opportunity for researchers in computer simulation to study reproducibility of simulation experiments, a critical objective and keystone of any scientific effort and a requirement for credibility.

5 CONCLUSIONS

This paper presents a conceptual framework for managing computer simulation experiments with the purpose of improving reproducibility and replicability. It provides a roadmap for building a software tool that can support and aid the design and execution of simulation experiments, as well as their changes over time. Experiments are iterative and any attempt to manage them should take this into account.

To streamline the reproducibility of simulation experiments, experiments need to be explicitly modeled, reused, managed, and transformed into executable scripts coordinated by a process that is consistent with the standard Design of Experiments methodology (Lorscheid et al. 2012). To this end, the objective of the Simulation Experiment Management System is to allow users to design, execute, store, adapt, and share computer simulation experiments.

Simulation experiments (and experiments in general) are systematic, iterative processes for determining the truth or falsehood of a set of hypotheses. In order for these experiments to gain validity and certainty, it is necessary that they are repeatable, that is, others must be able to reproduce the results and, in consequence, confirm or deny the experimenter's conclusions. This basic principle of all scientific endeavors has been significantly neglected in the computer simulation community, in part due to the difficulties arising from reproducing simulation experiments across different platforms and systems. It is the purpose of our work to remedy the lack of proper experiment management protocols.

REFERENCES

Altintas, I., C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. 2004. "Kepler: An Extensible System for Design and Execution of Scientific Workflows." In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, 423–24. doi:10.1109/SSDM.2004.1311241.

Batory, D. 2005. *Feature Models, Grammars, and Propositional Formulas*. Springer.

De Roure, D., Goble, C., and R. Stevens. 2009. "The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows', Future Generation Computer Systems, 25, 561–567.

Donohue, Joan M. 1994. "Experimental Designs for Simulation." In *Proceedings of the 26th Conference on Winter Simulation*, 200–206. Society for Computer Simulation International.

Gaševic, D., D. Djuric, and V. Devedžic (2009). *Model-Driven Architecture and Ontology Development*. Springer. 2009.

Himmelspach, Jan, R. Ewald, and A. M Uhrmacher. 2008. "A Flexible and Scalable Experimentation Layer." In *Proceedings of the 40th Conference on Winter Simulation*, 827–35. Winter Simulation Conference.

Ioannidis, Yannis E, Miron Livny, Anastassia Ailamaki, Anand Narayanan, and Andrew Therber. 1997. "ZOO: A Desktop Experiment Management Environment." In *ACM SIGMOD Record*, 26:580–83. ACM.

Kelton, W David. 2000. "Design of Experiments: Experimental Design for Simulation." In *Proceedings of the 32nd Conference on Winter Simulation*, 32–38. Society for Computer Simulation International.

Kleijnen, Jack P. C. 2005. "An Overview of the Design and Analysis of Simulation Experiments for Sensitivity Analysis." *European Journal of Operational Research* 164 (2): 287–300.

Kleijnen, Jack PC, Susan M Sanchez, Thomas W Lucas, and Thomas M Cioppa. 2005. "State-of-the-Art Review: A User's Guide to the Brave New World of Designing Simulation Experiments." *INFORMS Journal on Computing* 17 (3): 263–89.

Law, A.M. and W.D. Kelton. 2000. *Simulation Modeling and Analysis*. 4th ed. New York: John Wiley.

Leye, S. and A. M Uhrmacher. 2012. "GUISE-a Tool for GUIding Simulation Experiments." In *Proceedings of the Winter Simulation Conference*, 305. Winter Simulation Conference.

Lorscheid, Iris, Bernd-Oliver Heine, and Matthias Meyer. 2012. "Opening the 'Black Box' of Simulations: Increased Transparency and Effective Communication through the Systematic Design of Experiments." *Computational and Mathematical Organization Theory* 18 (1): 22–62. doi:10.1007/s10588-011-9097-3.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., and K. Glover. 2004. "Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows." Bioinformatics, 20, 3045–3054.

Ören, T.I. 2001. Software Agents for Experimental Design in Advanced Simulation Environments. In: S.M. Ermakov, Yu.N. Kashtanov, and V. Melas (eds.) Proc. of the 4th St. Petersburg Workshop on Simulation, June 18-23, 2001, pp. 89-95.

Peachey, T. C., T. N. Diamond, D. A. Abramson, W Sudholt, A. Michailova, S. Amirriazi 2008. "Fractional Factorial Design for Parameter Sweep Experiments using Nimrod/E." Journal of Scientific Programming, Volume 16, Numbers 2,3, 2008.

Perrone, L Felipe, Christopher S Main, and Bryan C Ward. 2012. "Safe: Simulation Automation Framework for Experiments." In *Proceedings of the Winter Simulation Conference*, 249. Winter Simulation Conference.

Rybacki, S., J. Himmelspach, F. Haack, and A M. Uhrmacher. 2011. "Worms: A framework to support workflows in M&S." In *Proceedings of the Winter Simulation Conference*, pp. 716-727. Winter Simulation Conference, 2011.

Sanchez, S. M and H. Wan. 2009. "Better than a Petaflop: The Power of Efficient Experimental Design." In *Winter Simulation Conference*, 60–74. Winter Simulation Conference.

Tao, Yu-Hui and B. L. Nelson. 1997. "Computer-Assisted Simulation Analysis." IIE Transactions, vol. 29, no. 3, pp. 221-231

Wilson, L.F., D. Burroughs, J. Sucharitaves, and A. Kumar. 2000. An Agent-Based Framework for Linking Distributed Simulations. *Proceedings of the 2000 Winter Simulation Conference,* 1713-1721.

Yilmaz, L and T Ören. 2013. "Toward Replicability-Aware Modeling and Simulation: Changing the Conduct of M&S in the Information Age." In *Ontology, Epistemology, and Teleology for Modeling and Simulation*, edited by Andreas Tolk, 44:207–26. Intelligent Systems Reference Library. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-31140-6_11.

AUTHOR BIOGRAPHIES

ALEJANDRO TERAN-SOMOHANO is a PhD Candidate in Industrial and Systems Engineering at Auburn University. He holds a Bachelor's degree in Computer Engineering from the Instituto Tecnologico Autonomo de Mexico (ITAM) and M.S. in Industrial Engineering from Auburn University. His email address is ateran@auburn.edu.

ORÇUN DAYIBAŞ is a PhD Candidate in Computer Engineering at Middle East Technical University (METU). He holds a Bachelor's degree in Computer Engineering from Hacettepe University and M.Sc. in Computer Engineering from METU. His email address is orcun.dayibas@metu.edu.tr.

LEVENT YILMAZ is Professor of Computer Science and Software Engineering at Auburn University with a joint appointment in Industrial and Systems Engineering. He holds M.S. and Ph.D. degrees in Computer Science from Virginia Tech. He is the founding organizer and General Chair of the Agent-Directed Simulation Symposium series and is the Editor-in-Chief of the Simulation: Transactions of the SCS. His email address is yilmaz@auburn.edu.

ALICE E. SMITH is the W. Allen and Martha Reed Professor of Industrial and Systems Engineering at Auburn University with a joint appointment in Computer Science and Software Engineering. She has authored papers with over 2,000 ISI Web of Science citations and has been a principal investigator on projects with funding totaling over $6 million. She is an area editor of INFORMS Journal on Computing and Computers & Operations Research and an associated editor of IEEE Transactions on Evolutionary Computation. Her email address is smithae@auburn.edu.