Int j simul model 21 (2022) 1, 17-28
Original scientific paper

# HYPOTHESIS-DRIVEN SIMULATION EXPERIMENTS WITH AN EXTENSION TO SED-ML

Cam, S.[*]; Oguztuzun, H.[*] & Yilmaz, L.[**]
[*] Computer Engineering Department, Middle East Technical University,
Universiteler Mahallesi Dumlupinar Bulvari No:1, 06800, Cankaya Ankara, Turkey
[**] Computer Science and Software Engineering Department, Auburn University,
1161 West Samford Avenue, AL 36849, Alabama, USA
E-Mail: semacam_87@yahoo.com, oguztuzn@metu.edu.tr, yilmale@auburn.edu

**Abstract**
Most of the frameworks or assistance systems for experiment specification do not provide a process explicitly based on formally specified hypotheses. This deficiency leads to inaccurate or insufficient record of an experiment, decreasing the trustworthiness and reproducibility of the experiment. Moreover, the wide variety of models, metamodels, tools, and data for experimentation requires Global Model Management (GMM) that is utilizing Model-Driven Engineering techniques, facilitates documentation, sharing, reusability, and replicability of simulation experiments. In this study, we strive to illustrate how to support simulation experimentation with hypotheses as a scientific workflow through GMM with an extension to the Simulation Experiment Description Mark-up Language (SED-ML). In particular, we present a megamodel built to serve as a repository to manage the artefacts employed in a simulation experiment. Based on the SED-ML, and enriched with hypothesis handling, our megamodel attempts to address all the phases of a simulation experiment, including specification, input data generation, execution, and output data analysis.
(Received in August 2021, accepted in December 2021. This paper was with the authors 1 month for 1 revision.)

**Key Words:** Design of Experiments, Simulation Experiment Description Markup Language, Global Model Management, Signal Temporal Logic

## 1. INTRODUCTION

In many sciences and engineering problems, theories and hypotheses about what makes a system work, or explains some phenomena in terms of cause and effect relationships, are put forth. Then, research scientists conduct experiments to test hypotheses with a set of goals and questions in mind. The values of the input variables of the system are modified deliberately, and the resulting output values are evaluated and measured. Experiments produce evidence about whether proposed theories are supported or not [1]. In this sense, a scientific process involves the following steps:

1. Formulate well-structured research questions about a problem and origin.
2. Acquire statistical hypotheses from the questions to answer the questions.
3. Generate some logical consequences of hypotheses in the form of expected behaviour.
4. Design experiments to test the underlying assumptions about the phenomena.
5. Validate the experiment for relevance and reliability.
6. If necessary, design computer simulations (methods for generating behaviour from a model), execute it, and interpret results.
7. Evaluate the correctness of hypotheses, and if necessary, revise the model, experiments, or expected behaviour.

With current developments in computational science and engineering, the main practical problem is a better understanding of the real-world processes becomes time-consuming, error-prone due to limitations of human perception since both simulation experiments and simulations as software are usually created manually. For this reason, evaluating the quality of experiments becomes necessary, and the correctness of the simulators that execute the simulation models is a significant aspect of this evaluation [2]. In this sense, defining simulation models at a germane level of abstraction and accuracy, and designing experiments with ample information would leverage the quality of experimentation. Having a complete record of the experimental conditions increases the credibility and reproducibility of scientific experiments [3, 4].

Nowadays, domain-specific languages (DSL) and mark-up languages have critically influenced academic dialogue on having accurate and sufficient records of simulation experiments. Frequently implemented, DSLs are valuable to develop the experiment specification and design efficiently with the focus on a particular domain for development, and enabling recording and reusing of the experiments for reproducibility purposes [5]. Mark-up languages are also coming to the fore to provide a standard for simulation experiments. Simulation Experiment Description Mark-up Language (SED-ML) [6] is an XML-based format, which uses MIASE standards for encoding, exchanging, and documenting simulation experiments [7] to improve communication among the scientists. It facilitates sharing experiment descriptions, validation, and reuse of simulation experiments. SED-ML provides standardization of the entire process that enhances the reproducibility and credibility of simulations. Even though SED-ML has several useful features for simulation experiments, hypothesis specification is not covered, and we believe that adding the missing piece in this study will enrich the experimentation process significantly.

Keeping the complete record of simulation experiments becomes even more arduous when it comes to scalability. Because there is a growing number of simulation experiment projects, and these models need a supportive environment that is easy to be used by non-programmers for sustainability and manageability of the related models, or performance analysis [8]. The experiment models, a specification for an experiment defined by a DSL or mark-up language in the environment should be accessible and compliant with the other models, such as represented system or data. In particular, the environment must support loading, saving, editing, deleting, archiving, searching, and executing models. Moreover, when numerous models are involved, users encounter burdensome system management issues, including the need to maintain consistency. We recognize that simulation experimentation brings up the following issues:

- logical complexity at multiple levels, such as domain modelling (conceptual modelling) issues,
- model complexity,
- simulation complexity,
- heterogeneity of the operational environments for simulation execution,
- challenges in experiment design.

In that respect, Global Model Management (GMM) [9] concept was already proposed to solve the model management issues in a study that supports simulation experiments with megamodelling [10], i.e., essentially managing models and their relations. GMM aims to manage a large and varied set of artefacts produced in modelling-in-the-large projects that adopt Model-Driven Engineering (MDE) methodology.

The aim of this work is to attempt to keep a full record of the experimentation process by incorporating formally specified hypotheses into the experimentation with the support of MDE methodology. The benefit of having formally specified hypotheses is to enrich the simulation experiments with more parametrization, improving the evaluation of the experiments and testing of the hypotheses. At the same time, we want to hold the standardization for sharing and documenting the models and experiments among the research scientists. Within the framework of these criteria, it is of interest to investigate if a widely accepted representation format for simulation experiments can be extended with a formal hypothesis specification. Therefore, we propose a simulation experimentation process based on a partially extended SED-ML with the hypothesis by getting user-defined hypotheses with models as an input, designing experiments to validate these hypotheses, executing and analysing them. We essentially establish explicit links between hypotheses and experiments. For illustration purposes, we briefly introduced a study about the predictive analysis of hospital bed availability. This particular study is also a good opportunity to reveal the versatility and sufficiency of SED-ML extended with a formally defined hypothesis in other areas than computational biology experiments.

The remainder of the paper is structured as follows. In Section 2 (Related Works), we present an overview of the existing works on the specification of simulation experiments and a foundational background for the study. Section 3 (Methodologies) presents Model-Driven Engineering methodology with the Global Model Management concept, Signal Temporal Logic for hypothesis specification and SED-ML for Simulation Experiments. Further, in Section 4, we explain our megamodel specification for Hypothesis-Driven Experiment Design and carry out a small study for predictive analysis of hospital bed availability in Section 5. Finally, Section 6 gives conclusions and future work directions.

## 2. RELATED WORKS

Automating and recording experiments are significant for the reusability and reproducibility of scientific research [11]. In this sense, combining experimental design, model, and hypotheses in one process [12] with an appropriate connection improves experimentation practices. There are studies to standardize simulation experiments and provide accurate and sufficient records of simulation experiments [13]. This demand brought on the experimentation workflows as a response. For example, recently, in [14] the authors proposed an artefact-based workflow to examine the specified requirements and develop methods to accommodate goal-directed guidance to the user. Even though the reproducibility issue is well-covered by the current assistance systems for the entire life-cycle of an experimentation process, still, no experiment generation procedure or framework, to our knowledge, has considered integrating a broadly embraced standard computer-readable exchange format such as SED-ML to improve the usability of simulation experiments among experimenters and software tools in a globally managed megamodel environment. As well as promoting a standard with SED-ML, our method of megamodelling also contributes to the reusability concept by keeping the integrity of the simulation artifacts, i.e., any kinds of by-products produced during the development of simulation, in a fast-growing environment.

The use of formalisms for experiment design is salient to address the experiment specification and design, and reproducibility of the experiments [15]. In this sense, simulation experiment description languages such as Simulation Experiment Specification via a Scala Layer (SESSL) [16], have been designed. SESSL is an internal DSL embedded in Scala, helping to define and generating experiments. An experiment designed in SESSL consists of model specification, the description of replications, the stop state for the simulation run, the objective, and the range and optimization method. Furthermore, dedicated mark-up languages, e.g., SED-ML [6] based on XML, pure functional programming languages [17], and ontologies have proven to be competent in specifying and managing simulation experiments. For this study, it is of interest to achieve the replicability of the experiments developed in diverse DSLs, ontologies [18], or other formal standards and yet continue being compatible with the scientific community.

The literature review has proven that numerous studies subsist as frameworks or assistance systems, and languages or formal standards to formulate and execute a simulation experiment. However, scarce authors have recognized the fact that current formalisms do not derive experiments incorporating hypotheses to enhance the reliability of the research [19]. This paper initially fills this gap in research, so far lacking in the scientific literature. Specifically, we will provide a hypothesis-based experiment design process that takes system specifications, data sets, and the hypothesis concerning the system as inputs, and automatically generates an executable and validatable experiment as an output. With this experiment design process, we aim to solve the aforementioned simulation experimentation issues in the Introduction section by easing the burden of experimentation complexities at multiple levels. Additionally, the hypothesis specification is always carried along with the process so that no information is lost or wasted. Then, with the help of GMM, variety of DSLs, formalisms are aimed to supported within the same environment.

## 3. METHODOLOGIES

While designing and realizing our megamodel for the Hypothesis-driven experiment design process with a SED-ML extension, we relied upon from Global Model Management and megamodel concepts to establish a customized hypothesis-based process design for simulation experiments. For realizing the idea, we utilized and integrated different instruments such as a formalism for hypothesis specification (Signal temporal logic [20]) and SED-ML. In this section, we will first explain hypotheses and statistical hypotheses testing, and each of the methodologies with their purpose and necessity in our work. Later in Section 4, we will reveal how they contribute to our megamodel realization and how they support the integration of the phases of the experiment design process.

### 3.1 Hypotheses and statistical hypothesis testing

A hypothesis refers to a formal claim about a state of a natural population that assumes relations between variables that belongs to an object of investigation [21, 22]. In statistics, hypotheses are tested assuming that the probabilities are distributed over the values of the variables from the observed data models [23] and the hypothesis testing involves two statistical hypotheses: *null* ($H_0$) and *alternative* ($H_1$) hypotheses, which is being called the *test of significance* [24]. Basically, a null hypothesis represents the hypothesis to be tested, whereas an alternative hypothesis is an alternative to the null hypothesis, and the alternative hypothesis is promoted if the null hypothesis is not viable. After defining the hypotheses, a formal hypothesis testing procedure to determine whether to reject a null hypothesis is followed.

There are different types of statistical tests and many factors plays a role for a selection, such as the quantity and level of data, or the statistics preferred in the study [25]. However, for the sake of simplicity, we opted to extend the SED-ML with null and alternative hypotheses specifications only. We acknowledge that further work needs to be performed to enrich our hypothesis specification with more features of statistical testing, such as significance level.

### 3.2 Simulation Experiment Description Mark-up Language

The Simulation Experiment Description Mark-up Language (SED-ML) [6] is an XML-based format, developed for the encoding of simulation descriptions on computational models of biological systems. Its purpose is to store information about the simulation experiment performed on one or more models with a given set of inputs. As the number of computational models, their size, and complexity are increasing at an ever-increasing pace, the need for building on existing studies by reusing models becomes more and more evident. Remarkably, various efforts to standardize the representation of computational models in various areas of biology increases the need for exchange and reuse of models. SED-ML aims to fill the gap by offering reusable and exchangeable models. The SED-ML format consists of six major blocks:

1. **DataDescription** entity specifies datasets for a simulation experiment.

2. **Model** entity is a reference to the models applied in the simulation experiment and to define procedures on these models before simulation (e.g., changing the value of an observable). Each instance of the Model class has a unique and mandatory id.
3. **Simulation** entity assists in the execution of the defined algorithm(s).
4. **Task** entity serves for a single simulation at a time. An experiment description can have as many tasks as required. The tasks do not require a specification for ordering.
5. **DataGenerator** entity produces post-processing procedures and applies those procedures to the simulation result before achieving the output. The post-processing steps can include mathematical manipulations such as normalization of data or mean-value calculation.
6. **Output** entity specifies the simulation output.

## 3.3 Hypothesis extension to SED-ML

Toward a proposal to extend SED-ML to support hypothesis-driven simulation experiments, we added a new block, called *Hypothesis*, to specify hypotheses to the SED-ML specification. A *Hypothesis* entity specifies single or composite hypotheses with a relation among them to test the statistical hypotheses. In this study, SED-ML with hypotheses extension represents the intermediate experiment model for our model transformations, particularly for the SED-ML to Xperimenter (a DSL for experiment design [26]) model transformation. The advantage of having an intermediate experiment model such as Xperimenter is that enhancing the reliability of the generated models by enabling the replication and validation of the intermediate model, itself.

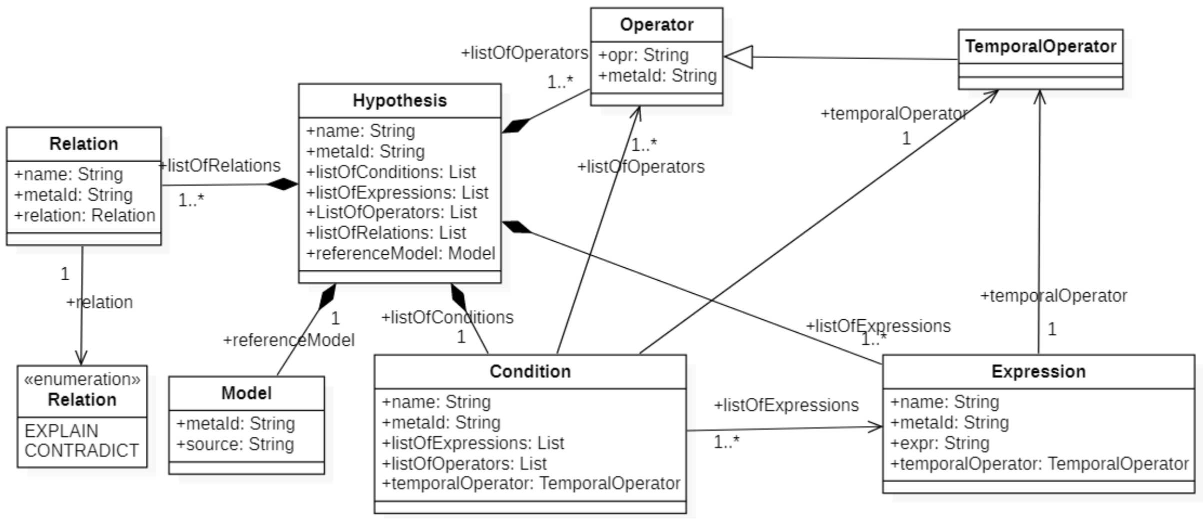

Figure 1: Hypothesis extension for SED-ML metamodel.

Fig. 1 depicts a simplified metamodel definition of the statistical hypothesis extension for the SED-ML. The extended hypothesis block in SED-ML consists of one or more conditions with many expressions representing the STL equations and the expression relations are shown with logical operators, e.g., *and*, *or*. Notably, each *listOfExpressions* block can have multiple *expression*s with multiple *operator*s and a *temporalOperator*. And each *condition* can have a single *temporalOperator*. A *temporalOperator* describes *T*, *P*, *S* and *A* abbreviations within a time interval, e.g., $P_{[1, 1]}$. Also, the *listOfRelations* explains the relations among the hypotheses, or if it leads to the hypothesis expression in the case of a null hypothesis. The relations between the hypotheses are represented with the *relation*. A *relation* can be *EXPLAIN*, if there is a single null hypothesis with multiple conditions that lead to the hypothesis expression. Or a *relation* can be *CONTRADICT* if a null hypothesis is going to be tested with an alternate hypothesis. Finally, a *modelReference* associates the *expression*s of the hypothesis to an actual model.

### 3.4 Past time Signal Temporal Logic

Signal Temporal Logic (STL) is a formalism that extends the linear temporal logic [20], and it is a system of rules and symbolism for representing real-valued signals [27]. An STL formula is composed of the Boolean and temporal operators and predicates in the form of linear inequalities. There are several practical usages such as runtime verification [28], and analysis of time series data [29]. In this study, for now, we focus on the past time fragment of STL called ptSTL to specify our hypotheses with a formal method. In this sense, ptSTL only allows the past time temporal operators. In our opinion, ptSTL has the potential to represent the dynamic simulation models [30], where the state variable changes with respect to time (e.g., a car moving through a road). Thus, we made use of ptSTL formulas to correspond to the time course simulation experiments, defined in SED-ML for the continuous time systems. A ptSTL formula is defined with the following grammar in Eq. (1) [31]:

$$\varphi = \mathbf{T} | \mathrm{xi} \sim \mathrm{c} | \neg\varphi | \varphi_1 \wedge \varphi_2 \, | \varphi_1 \vee \varphi_2 | \varphi_1 \mathbf{S}_{[a,b]} \varphi_2 | \mathbf{P}_{[a,b]} \varphi | \mathbf{A}_{[a,b]} \varphi \qquad (1)$$

According to Ergurtuna and Gol, in the grammar, a signal variable is represented by $x_i$, where $\sim \in \{<, >\}$ and $c$ is a constant. The letter $T$ represents the Boolean constant *true*. The standard Boolean operators are represented by $\neg$, $\wedge$ and $\vee$. Also, the temporal operators with time interval $[a, b]$ are represented with $S_{[a, b]}$ *(since)*, $P_{[a, b]}$ *(previously)*, and $A_{[a, b]}$ *(always)*. Finally, the semantics of a ptSTL formula is defined over a signal for a given time interval.

### 3.5 Global Model Management

Global Model Management (GMM) aims to provide support for *modelling in the large*. The objective is to handle models, metamodels, and their properties and relations in a model engineering environment. GMM is a technique to create, store, view, access, modify, and remain the information associated with all these modelling elements. Initially, Bézivin et al. [10], then, other authors have presented several modelling repositories such as REMODD [32] and MDEForge [33]. The foremost objective of the studies is to provide platforms that are making the artefacts stored and acquired readily with the dependencies among the artefacts.

Being a Model-Driven Engineering environment, GMM has the following characteristics:

- Large number of heterogeneous artefacts such as models, metamodels, model transformations, and source code.
- Modelling artefact relationships such as predefined (e.g., conformsTo) and ad hoc (e.g., weaving models).
- Tools for different purposes such as Model to Model (M2M) transformation engines, compilers, and workflow engines for simulation experiments.

## 4. A MEGAMODEL FOR HYPOTHESIS-DRIVEN EXPERIMENT DESIGN

Our ultimate vision is to attempt to provide a design process to guide the experimental scientists by creating an MDE-based ecosystem. Toward this vision, the GMM concept leads us to the management of such a complex system. Accordingly, we launched a megamodel that embodies the modelling artefacts of the scientific experimentation methods, and the required groundwork necessitates building a metamodel for simulation experiment megamodels. The convenience of using the megamodel is that the conforming megamodels can be defined and expanded in time by the experimental scientists for their particular experiments. We call the megamodel GMM4SE [9], short for Global Model Management for Simulation Experiments. GMM4SE identifies the supported types of modelling artefacts as metamodels, transformations, and the relationships between metamodels and model transformations. The user works with them as a cohesive unit, where the artefacts represent data, transformation, or model relationship for a particular megamodel. Fig. 2 straightforwardly depicts the megamodel conforming to the GMM4SE metamodel.

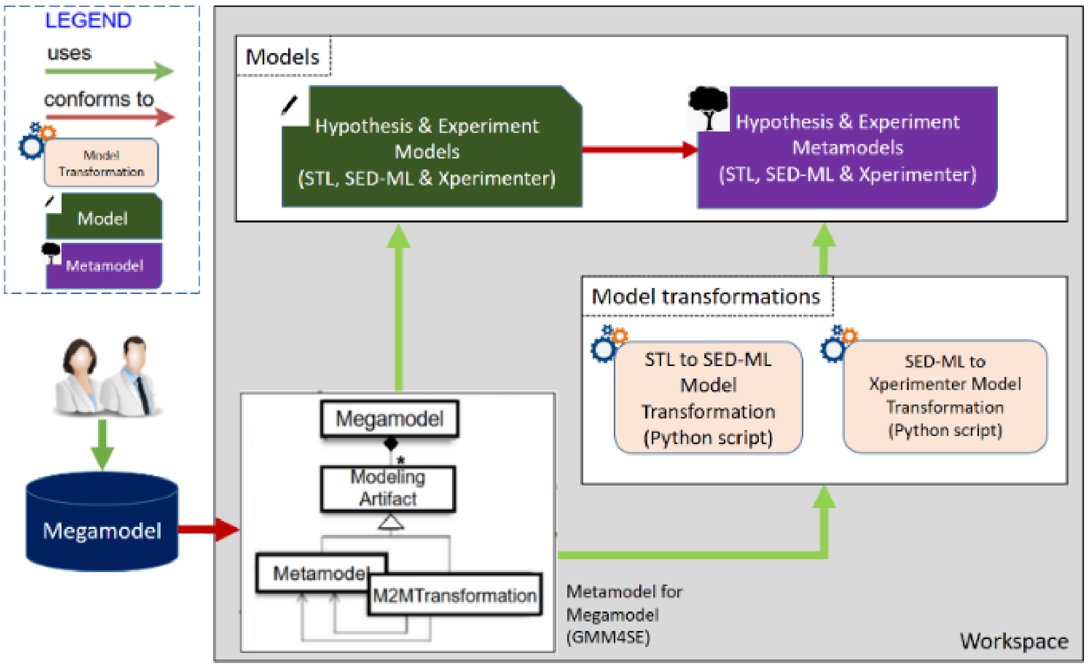
LEGEND
uses
conforms to
Model Transformation
Model
Metamodel
Megamodel
Models
Hypothesis & Experiment Models (STL, SED-ML & Xperimenter)
Hypothesis & Experiment Metamodels (STL, SED-ML & Xperimenter)
Model transformations
STL to SED-ML Model Transformation (Python script)
SED-ML to Xperimenter Model Transformation (Python script)
Megamodel
Modeling Artifact
Metamodel
M2MTransformation
Metamodel for Megamodel (GMM4SE)
Workspace

Figure 2: Overview of megamodel for simulation experiments [9].

Since our goal is to corroborate a Hypothesis-Driven Experiment Design process, we integrated frameworks, DSLs, and a workflow engine that are serving for the experimentation domain into the megamodel, including model transformations. Based on SED-ML, our experiment model transformations are always performed from SED-ML to target experiment model. Our proof of concept megamodel specification contains a model-based solution comprised of three kinds of modelling artefacts:

1. A metamodel for the megamodel, capturing the modelling artefact's details and their dependencies,
2. Metamodels, models and model transformations of the languages for experiment specification (e.g., STL, Xperimenter for Simulation Experiments, SED-ML),
3. Metamodels and models of the workflow management engines for experiment design, capturing the steps for creating, executing, and analysing the experiments (e.g., Kepler Workflow Management System).

## 4.1 A workflow for Hypothesis-Driven Experiment Design process

We will give an example to illustrate how to carry out our Hypothesis-Driven Experiment Design process with the megamodel and artefacts. Fig. 3 shows the user operations for the model and hypothesis creation, respectively. The following items detail out the steps.

1. The user creates a system model for the system under study and collects/generates data sets for the defined system. This system model consists of input and output variables, limits and intervals for these variables, and a tracing formula to mark the time traces of the model for the given hypothesis. The data sets should contain values for the input variables.
2. The user specifies a hypothesis for a system under study (e.g., hospital bed availability prediction), in a formally defined hypothesis language (e.g., STL).

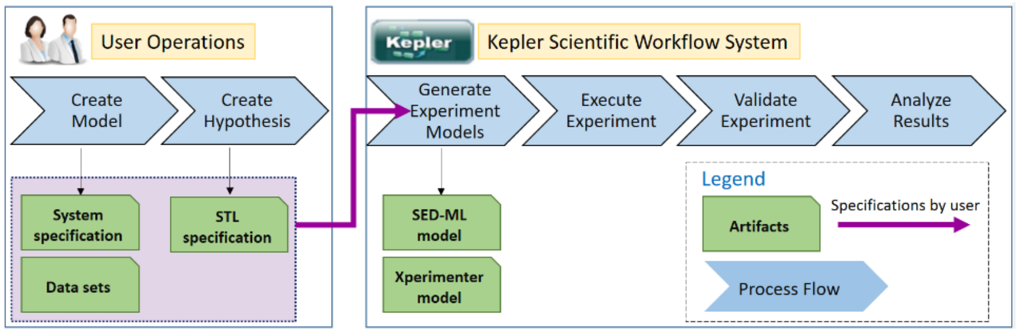

Figure 3: The workflow for Hypothesis-Driven Experiment Design process.

When the user completes the system and hypothesis specification, our workflow for Hypothesis-Driven Experiment Design takes action. The workflow handles the model transformation to create an experiment from the specified hypothesis, also manages the execution, validation, and analyses of the generated experiment.

Our workflow for Hypothesis-Driven Experiment Design, defined as a workflow, is given in Fig. 4.

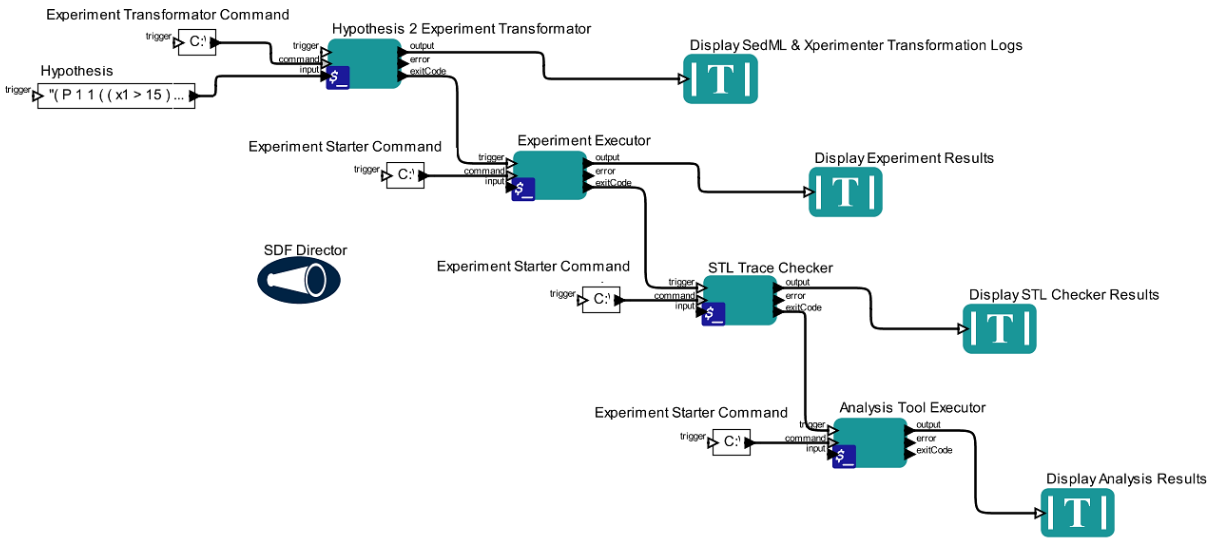
Experiment Transformator Command
Hypothesis
"( P 1 1 ( ( x1 > 15 ) ...
Hypothesis 2 Experiment Transformator
Display SedML & Xperimenter Transformation Logs
Experiment Starter Command
Experiment Executor
Display Experiment Results
SDF Director
Experiment Starter Command
STL Trace Checker
Display STL Checker Results
Experiment Starter Command
Analysis Tool Executor
Display Analysis Results

Figure 4: The Kepler workflow for hypothesis-based experiment design.

The followings explain the main modules that carry out the workflow:

1. **Hypothesis 2 Experiment Transformator**: is responsible for translating a formal language to an experiment specification (specifically, from ptSTL to Xperimenter model transformation, and SED-ML).
2. **Experiment Executor**: translates experiment specification into an executable, runs the executable and returns the throughputs (specifically, from Xperimenter to Kepler Workflow Management System).
3. **STL Trace Checker**: validates if the result of the experiment fits to the user-defined hypothesis, and finds the non-fitting time traces.
4. **Analysis Tool Executor**: enables analysis options for the user.

## 5. A PREDICTIVE ANALYSIS OF HOSPITAL BED AVAILABILITY DURING COVID-19 PANDEMIC

In order to gain a preliminary demonstration of a megamodel and the process for the hypothesis-driven experiment design, we opted for a current and simple study domain to degrade the complexity of the proposed system. The study investigates how to design and execute an experiment from a hypothesis with a system model and data as the user input in order to have a complete record of an experimentation process. At the same time, achieving a variety of experiment models is another purpose to create a GMM eco-system for experimentation process. We consider that a hospital bed availability prediction system serves excellent for this purpose based on its importance, especially during COVID-19 pandemic. A rise in the number of COVID-19 patients burden hospitals and it is also a valid indicator of the necessity of taking further measures against the pandemic.

## 5.1 Hypotheses about hospital bed availability in Ankara during COVID-19

The Hospital bed availability study is modelled based on the hospitals in the capital city of Turkey, Ankara dedicated to serving the COVID-19 patients in Table I. There exist six major hospitals and these hospitals are intentionally selected as they serve most of the patients in Ankara. Authorities state that depending on the daily situation and their capacity, the hospitals transfer patients to the closest hospitals. For example, if the *Bilkent Hospital* gets filled up, the closest hospitals *Ankara Gazi Hospital* and *Sehit Sait Erturk Hospital* will start to admit more patients than average, depending on the increase of daily COVID-19 patients. Thus, keeping the accurate number of occupancy and predicting the possible increase in the number of patients becomes quite important for the healthcare professionals.

Table I: Selected hospitals with COVID-19 services in Ankara and their capacities [34].

| Code | Hospital name | Bed capacity |
|---|---|---|
| $h_0$ | Ankara Gazi Hospital | 117 |
| $h_1$ | Bilkent Hospital | 3810 |
| $h_2$ | Diskapi Yildirim Beyazit Education and Research Hospital | 300 |
| $h_3$ | Gulhane Education and Research Hospital | 1150 |
| $h_4$ | Sehit Sait Erturk Hospital | 115 |
| $h_5$ | Yeni Sincan Hospital | 480 |

Hospital bed availability became one of the major concerns in many countries during the COVID-19 pandemic. Thus, we formulated our concern based on the previously established steps of the scientific process in the Introduction section, with an appropriate question addressing the problem and hypotheses targeting to solve the problem. Specifically, the formalized questions as ptSTL formulas describe the hypotheses where *j* represents the number of daily COVID-19 patients, and *k* represents the daily number of admitted COVID-19 patients from the neighbour cities at a certain state.

1. **Question**: What are the conditions that lead to fullness on hospital $h_1$ on the next day?
2. **Conditions**: The following conditions in Eq. (2), defined according to the formal specification in Section 2.4 originate fullness on hospital $h_1$ on the next day:

$$\begin{gathered}\varphi = \varphi_1 \lor \varphi_2 \lor \varphi_3 \qquad (2)\\ \varphi_1 = \mathrm{P}_{[1,1]}((h_1 > 1500\,) \land (j > 3000) \land (k > 50))\\ \varphi_2 = \mathrm{P}_{[1,1]}((h_1 > 2500) \land (j > 3000))\\ \varphi_3 = \mathrm{P}_{[1,1]}((h_3 > 900) \land (j > 3000) \land (k > 50))\end{gathered}$$

Each sub-formula $\varphi_1, \varphi_2$ and $\varphi_3$ states a condition that leads to fullness on hospital $h_1$ on the next day.

- $\boldsymbol{\varphi_1}$: on the occasion of more than 1500 patients at hospital $h_1$, the number of hospitalized COVID-19 patients (*j*) is more than 3000, and more than 50 patients get transferred to Ankara (*k*),
- $\boldsymbol{\varphi_2}$: on the occasion of more than 2500 patients at hospital $h_1$ and the number of hospitalized COVID-19 patients is more than 3000 (*j*),

- $\boldsymbol{\varphi_3}$: on the occasion of more than 900 patients at hospital $h_3$, the number of hospitalized COVID-19 patients (*j*) is more than 3000, and more than 50 patients get transferred to Ankara (*k*).

3. **Null hypothesis ($H_0$):** If one of the condition occurs, then the hospital $h_1$ observes fullness by growing 80 % over its capacity where the condition is $h_1 > 3048$.
4. **Alternative hypothesis ($H_1$):** If one of the condition occurs, then the hospital $h_1$ does not observe fullness by growing 80 % over its capacity where the condition is $h_1 \leq 3048$.

We assign our individual hypothesis-based experiment design workflow in Fig. 4 for the remainder of the steps (4, 5, 6, and 7) of the scientific process. However, for reasons of space, only the experiment result of the study is given in this paper, and the complete study with the extended SED-ML file can be found in the GitHub repository [35].

### 5.2 Experiment result

Trace analysis is a useful technique for verifying formal proofs. A trace checker analyses the traces and outlines any violations of the proffered formula. Due to its frugality and practicality of the method, employing a trace checker for STL specifications appears to be reasonable in terms of experiment result validation in this study. Taking that into consideration, we employed the STL Trace Checker [31] to validate the experiment output that we conducted. The trace checker takes the previously stated conditions for the hospital bed availability analysis for the hospital $h_1$ alongside the generated datasets and returns the proving and disproving data traces. The output of the STL Trace Checker appears to tally with our expectations for the number of the traces providing the conditions, i.e., 62, and the number of the traces the disproving the conditions, i.e., 4.

Based on the quantitative comparison of the throughputs from the STL Trace Checker and the experiment run we conducted, we also observed that the throughputs are consistent with the analysis reported by us. Therefore, we can humbly claim that our workflow is a working environment that keeps the full record to design and execute an experiment from a hypothesis with a system model and data from the user. Additionally, the workflow is capable of achieving different experiment models by employing SED-ML to target model transformation (e.g., SED-ML to Xperimenter).

## 6. CONCLUSIONS AND FUTURE WORK

Our study fundamentally confirms that we applied the initial step toward a reference implementation of a Global Model Management environment for Hypothesis-Driven Experiment Design. During this attempt, we benefited from the megamodelling techniques that expedite the management of models with a variety of operations. Moreover, we conducted a study that fills the gap between a hypothesis and the relevant experiment by accomplishing an experiment derivation from the hypothesis. Altogether, our study for hypothesis-based experiment design connects experimental design, model, and hypotheses in a single process with a widely acknowledged simulation experiment description language, i.e., SED-ML. It enhances the replicability of scientific experiments and the reproducibility of scientific results with a scalable experimentation medium. We believe that our research will serve as a base for future studies on the experiment design, such as expanding our experiment design with Type I and II Errors for complete statistical hypothesis testing.

As we proceed to expand the megamodel with new DSLs and formalisms for experiment design and management systems, we realize that future investigations are necessary to enhance the precision and reliability of the specification mechanisms and introduce new model transformation algorithms that map hypotheses to experiment designs. This insight leads us to blend agent technology with our megamodel. Coupling cognitive computing to simulation experiments in Model-Driven Engineering (MDE) environment [5] is an idea to address various activities in scientific discovery such as defining problems, building mechanisms to model a system, answering the questions.

## REFERENCES

[1] Montgomery, D. C. (2004). *Design and Analysis of Experiments*, 6th edition, John Wiley & Sons, New York

[2] Onofrejova, D.; Janekova, J.; Grincova, A.; Soltysova, Z. (2020). Simulation and evaluation of production factors in manufacturing of fireplaces, *International Journal of Simulation Modelling*, Vol. 19, No. 1, 77-88, doi:10.2507/IJSIMM19-1-504

[3] Merali, Z. (2010). Computational science: … Error, … why scientific programming does not compute, *Nature*, Vol. 467, No. 7317, 775-777, doi:10.1038/467775a

[4] Joppa, L. N.; McInerny, G.; Harper, R.; Salido, R.; Takeda, K.; O'Hara, K.; Gavaghan, D.; Emmott, S. (2013). Troubling trends in scientific software use, *Science*, Vol. 340, No. 6134, 814-815, doi:10.1126/science.1231535

[5] Yilmaz, L.; Chakladar, S.; Doud, K. (2016). The goal-hypothesis-experiment framework: a generative cognitive domain architecture for simulation experiment management, *Proceedings of the 2016 Winter Simulation Conference*, 1001-1012, doi:10.1109/WSC.2016.7822160

[6] Kohn, D.; Nicolas, L. V. (2008). SED-ML – An XML format for the implementation of the MIASE guidelines, Heiner, M.; Uhrmacher, A. M. (Eds.), *Computational Methods in Systems Biology*, Springer, Berlin, 176-190, doi:10.1007/978-3-540-88562-7_15

[7] Waltemath, D.; Adams, R.; Beard, D. A.; Bergmann, F. T.; Bhalla, U. S.; Britten, R.; Chelliah, V.; Chelliah, V.; Cooling, M. T.; Cooper, J.; Crampin, E. J.; Garny, A.; Hoops, S.; Hucka, M.; Hunter, P.; Klipp, E.; Laibe, C.; Miller, A. K.; Moraru, I.; Nickerson, D.; Nielsen, P.; Nikolski, M.; Sahle, S.; Sauro, H. M.; Schmidt, H.; Snoep, J. L.; Tolle, D.; Wolkenhauer, O.; Novere, L. N. (2011). Minimum information about a simulation experiment (MIASE), *PLoS Computational Biology*, Vol. 7, No. 4, Paper e1001122, 4 pages, doi:10.1371/journal.pcbi.1001122

[8] Kim, B. S.; Kim, T. G. (2019). Cooperation of simulation and data model for performance analysis of complex systems, *International Journal of Simulation Modelling*, Vol. 18, No. 4, 608-619, doi:10.2507/IJSIMM18(4)491

[9] Cam, S.; Dayibas, O.; Gorur, B. K.; Oguztuzun, H.; Yilmaz, L.; Chakladar, S.; Doud, K.; Smith, A. E.; Teran-Somohano, A. (2018). Supporting simulation experiments with megamodeling, *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, 372-378, doi:10.5220/0006586703720378

[10] Bézivin, J.; Jouault, F.; Valduriez, P. (2004). On the need for megamodels, *Proceedings of the OOPSLA/GPCE: Best Practices for Model-Driven Software Development Workshop, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 9 pages

[11] King, R. D.; Rowland, J.; Oliver, S. G.; Young, M.; Aubrey, W.; Byrne, E.; Liakata, M.; Markham, M.; Pir, P.; Soldatova, L. N.; Sparkes, A.; Whelan, K. E.; Clare, A. (2009). The automation of science, *Science*, Vol. 324, No. 5923, 85-89, doi:10.1126/science.1165620

[12] Waltz, D.; Buchanan, B. G. (2009). Automating science, *Science*, Vol. 324, No. 5923, 43-44, doi:10.1126/science.1172781

[13] Rahmandad, H.; Sterman, J. D. (2012). Reporting guidelines for simulation-based research in social sciences, *System Dynamics Review*, Vol. 28, No. 4, 396-411, doi:10.1002/sdr.1481

[14] Ruscheinski, A.; Warnke, T.; Uhrmacher, A. M. (2020). Artifact-based workflows for supporting simulation studies, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 32, No. 6, 1064-1078, doi:10.1109/TKDE.2019.2899840

[15] Krueger, C. W. (1992). Software reuse, *ACM Computing Surveys*, Vol. 24, No. 2, 131-183, doi:10.1145/130844.130856

[16] Ewald, R.; Uhrmacher, A. M. (2014). SESSL: a domain-specific language for simulation experiments, *ACM Transactions on Modeling and Computer Simulation*, Vol. 24, No. 2, Paper 11, 25 pages, doi:10.1145/2567895

[17] Warnke, T.; Uhrmacher, A. M. (2019). Reproducible parallel simulation experiments via pure functional programming, *Proceedings of 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications*, 8 pages, doi:10.1109/DS-RT47707.2019.8958655

[18] Blondet, G.; le Duigou, J.; Boudaoud, N. (2016). ODE: an ontology for numerical design of experiments, *Procedia CIRP*, Vol. 50, 496-501, doi:10.1016/j.procir.2016.04.199

[19] Lorig, F.; Lebherz, D. S.; Berndt, J. O.; Timm, I. J. (2017). Hypothesis-driven experiment design in computer simulation studies, *Proceedings of 2017 Winter Simulation Conference*, 1360-1371, doi:10.1109/WSC.2017.8247880

[20] Baier, C.; Katoen, J.-P. (2008). *Principles of Model Checking*, The MIT Press, Cambridge

[21] Kitchin, J. (1994). 6 - Basic statistical inference, Stanford, J. L.; Vardeman, S. B. (Eds.), *Statistical Methods for Physical Science* (Book series: Methods in Experimental Physics), Academic Press, Cambridge, 155-186, doi:10.1016/S0076-695X(08)60256-2

[22] Mertens, W.; Recker, J. (2020). New guidelines for null hypothesis significance testing in hypothetico-deductive IS research, *Journal of the Association for Information Systems*, Vol. 21, No. 4, 1072-1102, doi:10.17705/1jais.00629

[23] Tietjen, G. (1986). *A Topical Dictionary of Statistics*, Springer Science & Business Media, Boston

[24] Freedman, D.; Pisani, R.; Purves, R. (2007). *Statistics*, 4th edition, W.W. Norton & Company, New York

[25] Bajpai, N. (2009). *Business Statistics*, Pearson Education, New Delhi

[26] Dayibas, O.; Oguztuzun, H.; Yilmaz, L. (2019). On the use of model-driven engineering principles for the management of simulation experiments, *Journal of Simulation*, Vol. 13, No. 2, 83-95, doi:10.1080/17477778.2017.1418638

[27] Donze, A. (2013). On signal temporal logic, *Proceedings of the International Conference on Runtime Verification*, 382-383, doi:10.1007/978-3-642-40787-1_27

[28] Bartocci, E.; Deshmukh, J.; Donze, A.; Fainekos, G.; Maler, O.; Nickovic, D.; Sankaranarayanan, S. (2018). Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications, Bartocci, E.; Falcone, Y. (Eds.), *Lectures on Runtime Verification*, Springer, Cham, 135-175, doi:10.1007/978-3-319-75632-5_5

[29] Vazquez-Chanlatte, M.; Deshmukh, J. V.; Jin, X.; Seshia, S. A. (2017). Logical clustering and learning for time-series data, Majumdar, R.; Kunčak, V. (Eds.), *Computer Aided Verification*, Springer, Cham, 305-325, doi:10.1007/978-3-319-63387-9_15

[30] Law, A. M. (2015). *Simulation Modeling and Analysis*, 5th edition, McGraw-Hill, New York

[31] Ergurtuna, M.; Gol, E. A. (2019). An efficient formula synthesis method with past signal temporal logic, *IFAC-PapersOnLine*, 43-48, doi:10.1016/j.ifacol.2019.09.116

[32] France, R. B.; Bieman, J. M.; Mandalaparty, S. P.; Cheng, B. H. C.; Jensen, A. (2012). Repository for model driven development (ReMoDD), *Proceedings of the 34th International Conference on Software Engineering*, 1471-1472, doi:10.1109/ICSE.2012.6227059

[33] Basciani, F.; Di Rocco, J.; Di Ruscio, D.; Di Salle, A.; Iovino, L.; Pierantonio, A. (2014). MDEForge: an extensible Web-based modeling platform, *Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud*, 66-75

[34] TTB Ankara Tabip Odasi. Verilerle Ankara'nin Sagligi (in Eng.: TTB Ankara Medical Chamber. Ankara's Health in Data), from *https://ato.org.tr/files/documents/ATO%20Rapor%20-Verilerle%20Ankara%27n%C4%B1n%20sagl%C4%B1%C4%9F%C4%B1.pdf*, accessed on 05-05-2021

[35] GitHub repository. Hypothesis-driven experiment design with an extension to SED-ML, from *https://github.com/semacammetu/Hypothesis-driven-experiment-design-with-SEDML-extension*, accessed on 23-08-2021