Simulation Foundations, Methods and Applications

Levent Yilmaz *Editor*

Concepts and Methodologies for Modeling and Simulation

A Tribute to Tuncer Ören

Springer

Simulation Foundations, Methods and Applications

More information about this series at http://www.springer.com/series/10128

Levent Yilmaz
Editor

Concepts and Methodologies for Modeling and Simulation

A Tribute to Tuncer Ören

Springer

Editor
Levent Yilmaz
Department of Computer Science and
Software Engineering
Department of Industrial and Systems Engineering
Auburn University
Auburn, AL, USA

ISSN 2195-2817 ISSN 2195-2825 (electronic)
Simulation Foundations, Methods and Applications
ISBN 978-3-319-15095-6 ISBN 978-3-319-15096-3 (eBook)
DOI 10.1007/978-3-319-15096-3

Library of Congress Control Number: 2015936148

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Foreword

The work of Tuncer Ören, the archetypal Renaissance man of the modeling and simulation community, embodies the vision of M&S as the essential enabler of future science and engineering. Ören's vision stretches widely over the whole M&S domain encompassing its fundamental body of knowledge, its methodology, its practice, and its ethics. It also includes the quality of M&S products and specific application domains such as cognitive and emotive social simulation. The authors of this tribute book pursue a few—essential—threads of the many emanating from Ören's core vision. As a consequence of the underlying unity of conception, the book is more than a collection of disparate state-of-the art articles. This integration is enhanced by the fact that every chapter is explicitly connected to pertinent features of Ören's thought. Within this perspective, the book covers cutting-edge topics in simulation methodologies; modeling methodologies; quality assurance and reliability of simulation studies; and cognitive, emotive, and social simulation. Notably, it touches on Ören's intense interest in the body of knowledge of modeling and simulation, with a review of existing M&S literature through the newly emerging techniques of journal profiling and co-citation analysis. This book beckons you, whether theorist or practitioner, generalist or domain application professional, to partake in and contribute to Ören's powerful vision.

Potomac, MD, USA
December 29, 2014

Bernard P. Zeigler

Preface

Concepts and Methodologies for Modeling and Simulation aims to present recent advances in the theory and methodology of Modeling and Simulation (M&S). By connecting these developments to the conceptual, theoretical, and methodological foundations developed by Professor Tuncer Ören, this volume serves as a testimonial that honors Dr. Ören's long-lasting and fundamental contributions to the M&S discipline for over 50 years.

Since 2003, I have had the privilege to collaborate with Dr. Ören, whom I see as my mentor and a titan in our field. The articles in this book are a testament to the diversity and innovativeness of his thoughts. As evidenced by this volume, his influences in the philosophy, theory, methodology, ethics, and the body of knowledge of M&S have numerous connections to recent advancements in our field and continue to provide directions for its further development. This book is largely due to the efforts and contributions of the authors, who shared their recent research in the context of Dr. Ören's seminal contributions to the M&S discipline. I am indebted to them for their contributions to this tribute volume. They are not only authorities in their field but also colleagues of Dr. Ören. Hence, they are qualified and entitled to trace recent advancements in their fields to the most influential concepts and methods introduced by him.

In the area of simulation methodologies, Dr. Ören's earlier work on model-based M&S and detailed categorizations and taxonomies of M&S has been highly influential. In particular, normative views for the advancement of M&S, including synergies of artificial intelligence and systems theories, and his comprehensive and integrative views have provided a sound and thorough framework for the development of advanced simulation methodologies. To explain these contributions, the book starts with my reflections on the recent developments in agent-directed simulation (ADS), providing a framework that explores synergies between simulation and agent technologies. The readers can trace the provenance of various ideas explored under ADS to concepts introduced by Dr. Ören when he first examined and demonstrated how artificial intelligence methods can assist simulation. The second chapter in this section presents how model engineering and

service technologies can be leveraged to contribute to System-of-Systems (SoS) Engineering. The third chapter overviews emerging trends and drivers in high-performance simulation systems, while the fourth chapter examines the role of data in the context of dynamic data-driven simulations that connect real-time sensor data to online simulations.

The second section of the book focuses on advanced modeling methodologies. The first chapter in this area focuses on the philosophical fields of ontology and epistemology to delineate the role and use of simulations in relation to the taxonomies and categories of M&S developed by Dr. Ören as part of his contributions to the M&S body of knowledge. The second chapter demonstrates how innovations in modeling formalisms can help manage the challenges in hybrid model composition, especially in the context of agent-based, human, social, and environment models. Specifically, the authors describe the use of a polyformalism model composition approach and highlight its relation to multimodeling strategies that Dr. Ören and I have developed during the early 2000s. The third chapter of this section underlines the importance of a model-based approach to M&S and underlines model building, model-based management, and model processing activities advocated by Dr. Ören. The authors then present a formal, declarative, and visual transformation (model processing) methodology to translate a domain conceptual model to a distributed simulation architecture model.

The third section of the book is devoted to the reliability and quality assurance of models. The section starts with an overview of quality indicators that can be used to support a structured and quality-centered approach to simulation development throughout the entire M&S life cycle. The second paper reviews, summarizes, and describes the influence of important M&S quality assurance papers developed by Dr. Ören. The paper also promotes strategies for the replicability and reproducibility of simulation studies to instill confidence in simulation experiments. The third paper in this section refers to challenges involved in qualitative and quantitative comparisons of agent-based models to calibrated statistical models for the purpose of validation and reproducibility. The last chapter in this section introduces the Generalized Discrete Event System Specification to build more accurate discrete-event models of dynamic systems. This work highlights the need for engineering quality into models to improve their accuracy.

The fourth section of the book focuses on the specification and simulation of human and social behavior, acknowledging Dr. Ören's contributions to model specification language development as well as his recent research in cognitive and emotive simulation modeling including the specification of models of personality, emotions, conflict management, perception, and anticipation. In this section, the first chapter presents work on social science models that benefits from the principles based on Dr. Ören's influences of model specification languages, goal-directed agents, anticipatory simulation, agent perceptions, and multifacetted models. Similarly, the second chapter in this area refers to the multisimulation methodology as a basis to examine bridging human decision processes and computer simulation while also referring to multisimulation as an enabling technology

for backtracking and replaying situated simulation histories with altered conditions as well as futures generated before exploring alternative realities in social sciences.

The last section of the book is devoted to M&S body of knowledge work. The chapter presented in this section was inspired by Dr. Ören's work and shares common ground by profiling and classifying M&S publications in terms of techniques, application areas, and their context in a relevant way with the second and third parts of the body of knowledge, which defines the M&S core areas and supporting domains.

Auburn, AL, USA Levent Yilmaz
December 19, 2014

Contents

Contributors

S.M. Niaz Arifin Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA

Jang Won Bae Department of Industrial and Systems Engineering, KAIST, Daejeon, South Korea

Osman Balci Mobile/Cloud Software Engineering Lab, Department of Computer Science, Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, VA, USA

C. Michael Barton Center for Social Dynamics and Complexity, Arizona State University, Tempe, AZ, USA

Xudong Chai Beijing Simulation Center, Beijing, China

Paul K. Davis Department of Engineering and Applied Sciences, RAND Corporation, Santa Monica, CA, USA

Paul Fishwick Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Norbert Giambiasi Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, Marseille, 13397, France

Maamar El Amine Hamri Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, Marseille, 13397, France

Aimin Hao School of Automation and Computer Science, Beihang University, Beijing, China

Baocun Hou Beijing Simulation Center, Beijing, China

Xiaolin Hu Department of Computer Science, Georgia State University, Atlanta, GA, USA

Korina Katsaliaki School of Economics, Business Administration & Legal Studies, International Hellenic University, Thessaloniki, Greece

Bo Hu Li School of Automation and Computer Science, Beihang University, Beijing, China

Jun Li Sugon Corp., Beijing, China

Tan Li Beijing Simulation Center, Beijing, China

Gregory R. Madey Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA

Gary R. Mayer Department of Computer Science, Southern Illinois University, Edwardsville, IL, USA

Il-Chul Moon Department of Industrial and Systems Engineering, KAIST, Daejeon, South Korea

Navonil Mustafee Centre for Innovation and Service Research, University of Exeter Business School, Exeter, UK

Aziz Naamane Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, Marseille, 13397, France

Halit Oguztüzün Department of Computer Engineering, Middle East Technical University, Ankara, Turkey

Gürkan Özhan Department of Computer Engineering, Middle East Technical University, Ankara, Turkey

Duzheng Qin Beijing Simulation Center, Beijing, China

Hessam S. Sarjoughian Department of Computer Science and Engineering, Arizona State University, Tempe, AZ, USA

Andreas Tolk SimIS Inc., Portsmouth, VA, USA

Isaac I. Ullah School of Human Evolution and Social Change, Arizona State University, Tempe, AZ, USA

Gabriel Wainer Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada

Ming Yang Control and Simulation Center, Harbin Institute of Technology, Harbin, Heilongjiang, China

Levent Yilmaz Department of Computer Science and Software Engineering, Department of Industrial and Systems Engineering, Auburn University, Auburn, AL, USA

Bernard P. Zeigler RTSync Corp., Arizona Center for Integrative Modeling and Simulation, Sierra Vista, AZ, USA

Lin Zhang School of Automation and Computer Science, Beihang University, Beijing, China

Yu Zhang Department of Computer Science, College of St. Benedict, St. John's University, Collegeville, MN, USA

Qinping Zhao School of Automation and Computer Science, Beihang University, Beijing, China

Part I
Simulation Methodologies

Chapter 1
Toward Agent-Supported and Agent-Monitored Model-Driven Simulation Engineering

Levent Yilmaz

1.1 Introduction

The concepts that expound on the synergy of artificial intelligence and simulation date back to late 1970s when Professor Ören (1977) has introduced strategies that delineate the use of computer assistance in model-based activities and later when he promoted in (Ören 1978) the principled use of cybernetics for developing simulation software. In 1980s, Ören and his colleagues, Elzas and Zeigler, edited two volumes (Elzas et al. 1986, 1989) that explicated various facets of the synergy of modeling and simulation, artificial intelligence, and knowledge-based systems. These two volumes became highly influential in putting forward a pathway toward convergence of modeling and simulation (M&S) and artificial intelligence methodologies.

Concomitantly, the emergence of the software agent concepts (Shoham 1993) that embody and encapsulate knowledge-based deliberation, reasoning, autonomy, interaction, learning, and adaptation mechanisms led to the adoption of agents as model design metaphors in simulation modeling. However, this limited view of the use of agents in M&S was critiqued in Ören (2000a, b). In his panel statement (Ören 2000b) at the 2000 Winter Simulation Conference, Dr. Ören has pointed out a broader vision for the use of agents in M&S. Besides using M&S for modeling agent systems in the form of agent-based models, by which agent concepts are leveraged to create abstractions of the system of interest, Dr. Ören has suggested that agents can also be used to assist or support model behavior generation as part of the simulators or provide cognitive support as front-end or back-end interface to a simulation system. He called this expanded view as *agent-directed simulation* (ADS).

L. Yilmaz (✉)
Department of Computer Science and Software Engineering, Department of Industrial and Systems Engineering, Auburn University, Auburn, AL, USA
e-mail: yilmaz@auburn.edu

L. Yilmaz (ed.), *Concepts and Methodologies for Modeling and Simulation*,
Simulation Foundations, Methods and Applications,
DOI 10.1007/978-3-319-15096-3_1

My collaboration with Dr. Ören on ADS started a few weeks before I joined to Auburn University as a tenure-track assistant professor in 2003. We met at the 2003 Summer Computer Simulation Conference that was held in Montreal, Canada. Following our initial discussions on the emerging trends and critical drivers of the use of software agent technologies in M&S, we developed a framework to structure and delineate the role of agents throughout the whole life cycle of M&S. A year later, this framework served as the basis for the first track of sessions on ADS that Dr. Ören and I organized as part of the 2004 Summer Computer Simulation Conference. Since the organization of this inaugural event, the ADS track of sessions has been and, as of 2014, continues to be one of the key features of the Summer Computer Simulation Conference series. The success of the ADS track in 2004 has led to the first Annual ADS Symposium that was held as part of the 2005 Spring Simulation Multiconference, formerly known as the Advanced Simulation Technologies Conference. Since 2005, the ADS Symposium continues to be organized annually by a growing organization committee. In 2007, Dr. Greg Madey and Dr. Maarten Sierhuis have joined the organization committee; this is followed by Dr. Yu Zhang joining in 2009. Since then, ADS has also been organized as a track of sessions as part of the European Modeling and Simulation Symposium and the SimulTech Conference series.

Our technical collaboration with Dr. Ören, along with other invited contributions, has led to the publication of the first book (Yilmaz and Ören 2009) on ADS. The Agent-Directed Simulation and Systems Engineering book is now viewed as the only book to present the synergy between modeling and simulation, systems engineering, and agent technologies while also expanding the notion of agent simulation to also deal with agent-monitored simulation and agent-supported simulation. This journey in ADS has resulted in numerous publications that range from advanced modeling and simulation methodologies such as multisimulation to concepts and demonstrations of agents with personality, emotions, anticipations, and understanding capabilities with application areas in engineering, defense, and human and social dynamics.

The objective of this chapter is to illustrate how agents can be used to facilitate development of next-generation simulators and assist in conducting simulation experiments. First, we introduce software agents and then characterize the three dimensions of the ADS framework that help explore the use of agents for simulation and the use of simulation for agents. This is followed by the introduction multisimulation and clarification of how agent-monitored and agent-assisted mechanisms facilitate its design and implementation. To emphasize the role of agents besides in model development and simulator (i.e., model behavior generator) design, we highlight the issues and challenges in goal-directed experimentation and present an agent-assisted and model-driven experiment management strategy to effectively address these challenges.

1.2 Agent-Directed Simulation

Agents are autonomous software modules with perception and social ability to perform goal-directed knowledge processing over time, on behalf of humans or other agents in software and physical environments. When agents operate in physical environments, they can be used in the implementation of intelligent (smart) machines and intelligent (smart) systems and they can interact with their environment by sensors and effectors.

The core knowledge-processing abilities of agents include goal-processing, goal-directed knowledge processing, reasoning, motivation, planning, and decision-making. The factors that may affect decision-making abilities of agents (in simulating human behavior) are personality, emotions, and cultural backgrounds. Abilities to make agents intelligent include anticipation (pro-activeness), understanding (avoiding misunderstanding), learning, and communication in natural and body language. Abilities to make agents trustworthy as well as assuring the sustainability of agent societies include being rational, responsible, and accountable. These characteristics lead to rationality, skillfulness, and morality (e.g., ethical agent, moral agent).

Synergy of simulation and agents is called agent-directed simulation (ADS). As shown in Fig. 1.1, it consists of contributions of simulation for agents and contributions of agents for simulation.

Agent simulation involves the use of simulation conceptual frameworks and technologies to simulate the behavioral dynamics of agent systems by specifying and implementing the behavior of autonomous agents that function in parallel to achieve objectives via goal-directed behavior. In agent-based model specifications, agents possess high-level interaction mechanisms independent of the problem being solved. Communication protocols and mechanisms for interaction via task

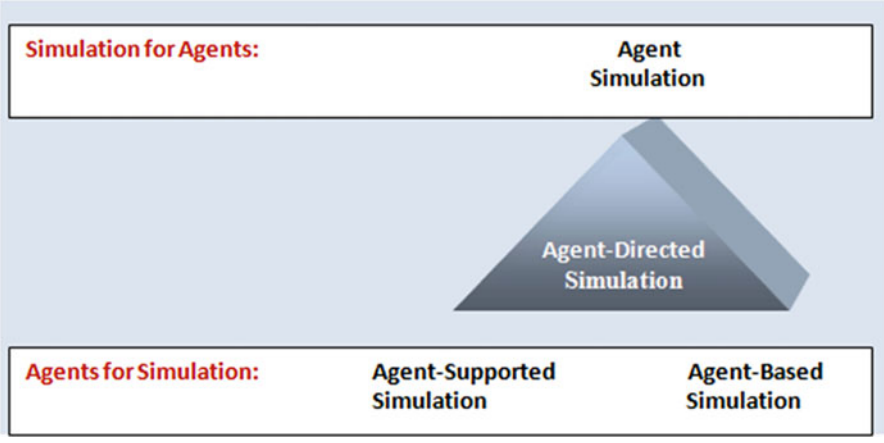

Fig. 1.1 Agent-directed simulation

allocation, coordination of actions, and conflict resolution at varying levels of sophistication are primary elements of agent simulations. Simulating agent systems requires understanding the basic principles, organizational mechanisms, and technologies underlying such systems.

The principal aspects underlying such systems include the issues of action, cognitive aspects in decision-making, interaction, and adaptation. Organizational mechanisms for agent systems include means for interaction. That is, communication, collaboration, and coordination of tasks within an agent system require flexible protocols to facilitate realization of cooperative or competitive behavior in agent societies Agent-based modeling in which agents are used as design metaphors to conceptualize and specify agent systems is becoming the most common methodology in agent simulation. On the other hand, simulation(s) can be used to realize deliberation architecture of agents in an agent system by simulating the cognitive processes in decision-making. While embedding simulations within the deliberation architecture of agents is not necessarily a novel concept on its own, it is still an unexplored territory in MAS design and implementation.

Agent-based simulation (or *agent-monitored simulation*) is the use of agent technology to monitor and generate model behavior. This is similar to the use of AI techniques for the generation of model behavior (e.g., qualitative simulation and knowledge-based simulation). Development of novel and advanced simulation methodologies such as *multisimulation* suggests the use of intelligent agents as simulator coordinators, where run-time decisions for model staging and updating take place to facilitate dynamic composability. The perception feature of agents makes them pertinent for monitoring tasks. Also, agent-based simulation is useful for having complex experiments and deliberative knowledge processing such as planning, deciding, and reasoning. Agents are also promoted and demonstrated as critical enablers to improve composability and interoperability of simulation models and software, in general.

Agent-supported simulation deals with the use of agents as a support facility to augment simulations and enable computer assistance by enhancing cognitive capabilities in problem specification and solving. Hence, agent-supported simulation involves the use of intelligent agents to improve simulation and gaming infrastructures or environments. Agent-supported simulation is used for the following purposes: (1) to provide computer assistance for front-end and/or back-end interface functions, (2) to process elements of a simulation study symbolically (e.g., for consistency checks and built-in reliability), and (3) to provide cognitive abilities to the elements of a simulation study, such as learning or understanding abilities.

1.3 Agent-Monitored Simulator for Exploratory Multisimulation

We define *multisimulation* as simulation of several aspects of reality in a study. It includes simulation with multimodels, simulation with multi-aspect models, and simulation with multistage models. Simulation with multimodels allows computational experimentation with several aspects of reality; however, each aspect and the transition from one aspect to another one are considered separately.

Simulation with multi-aspect models (or multi-aspect simulation) allows computational experimentation with more than one aspect of reality simultaneously. This type of multisimulation is a novel way to perceive and experiment with several aspects of reality as well as exploring conditions affecting transitions. While exploring the transitions, one can also analyze the effects of encouraging and hindering transition conditions. Simulation with multistage models allows branching of a simulation study into several simulation studies, each branch allowing to experiment with a new model under similar or novel scenarios. Each different strategy component characterizes a distinct aspect.

Multisimulation can be used to branch out multiple simulations, where each simulation uses a specific component configured with an exclusively selected strategy component. Similarly, multiple distinct stages of the problem can be qualified at a given point in time during the simulation by virtue of the evaluation of an updating constraint. In such a case, multisimulation enables branching multiple distinct simulations each one, which generates the behavior of distinct plausible stage within the problem domain.

Multisimulation with multimodels, multi-aspect models, or multistage models needs mechanisms to decide when and under what conditions to replace existing models with a successor or alternative. Staging considers branching to other simulation studies in response to a scenario or a phase change during experimentation. Graphs of model families facilitate derivation of feasible sequence of models that can be invoked or staged. More specifically, a graph of model families is used to specify alternative staging decisions. Each node in the graph depicts a model, whereas edges denote transition or switching from one model to another. Figure 1.2 depicts the components of the abstract architecture of a possible multisimulation engine.

A meta-simulator is an agent that generates staged composition of models by traversing the model stage graph and coordinates their simulation and staging within distinct simulation frames. Each frame simulates a distinct subset of models derived from the model stage graph. Note, however, that not all staged compositions are feasible or useful. Hence, the meta-simulator needs to consult with the model recommender before model staging to determine if emergent trigger or transition condition in the simulation is consistent with the precondition of the model to be staged. More than one model in a family can qualify for staging; in such cases, separate simulation frames need to be instantiated to accommodate and explore plausible scenarios.

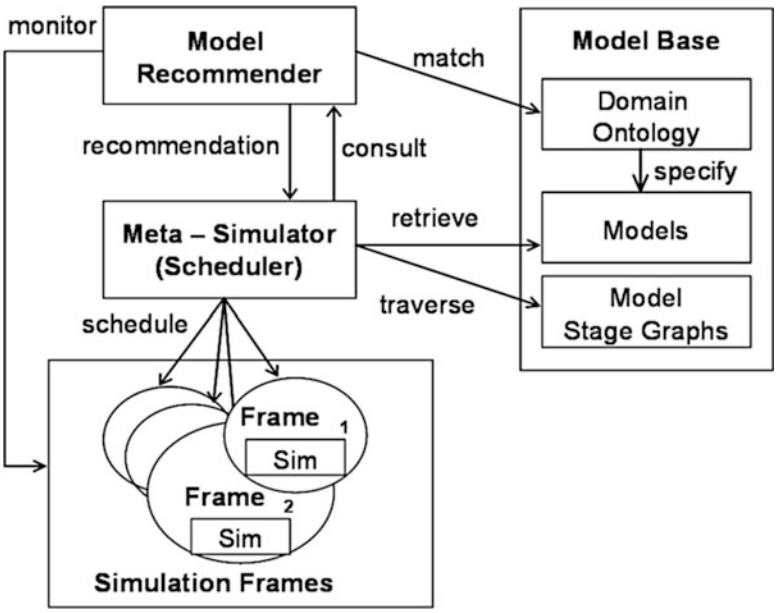

Fig. 1.2 Components of a multisimulation engine

Given a collection of models (or more generally, a family of models), a stage graph can be generated automatically by an optimistic approach that connects every available node (model) to every other node within the domain of problem. The edges in a model stage graph denote plausible transitions between models as the problem shifts from one stage to another. One can consider each model as a separate conflict management protocol (i.e., compromise over actions, compromise over outcomes, negotiation, and mediation) or a phase in the conflict process (i.e., escalation, resolution), where a phase (i.e., resolution) can constitute alternative models (i.e., mediation, negotiation, third-party intervention).

The subsets of staged models can be identified by traversing and enumerating the graph in some order (i.e., depth first). Infeasible paths may be due to an unreachable node, or it may result due to conflicts between the transition condition and precondition of the target model. Infeasible paths due to incompatible sequences of models are common. Each edge indicates that there is some legitimate solution; yet, it does not imply that every solution containing the edge is legitimate. As argued above, each model in a family of models is associated with a precondition. A precondition denotes the conditions required for a model to be instantiated. Hence, the feasibility of staging a successor model depends on the satisfiability of its precondition (relevance) by the condition of the transition and the post-condition of the predecessor model. As a result, not all enumerated staged sequences of model components are feasible.

Model recommendation in multisimulation can simply be considered as agent-supported exploration of the model-staging space that can be computed by a

reachability analysis of the graph. There are two modes for the usage: (1) offline enumeration of paths using the graph and performing a staged simulation of each model in sequence one after the other, unless a model-staging operation becomes infeasible due to conflict between the transition condition and the precondition of the successor model, and (2) run-time generation of potential feasible paths as the simulation unfolds. In both cases, an online model recommender plays a key role to qualify a successor model.

The first case requires derivation of sequence of models using a traversal algorithm. The edges relate families of models. Therefore, the actual concrete models, the preconditions of which satisfy the transition condition, need to be qualified, since transition to some of these model components may be infeasible due to conflict between a candidate model and inferred situation. Identifying such infeasible sequences is computationally intractable; otherwise, it would have been possible to determine if the conjunction of two predicates is a tautology by using a polynomial time algorithm.

Experience in the component-based simulation paradigm, however, indicates that for most model components, preconditions are simple. Hence, it is possible to eliminate some models that violate the transition condition. For the remaining possible transitions, it is possible to select one of the three strategies: (1) omit all difficult qualification conditions, (2) decide on an edge-by-edge basis which specific models of a model family to include, and (3) include all difficult edges. Omitting all difficult associations between transitions and model preconditions is conservative. This strategy excludes all infeasible models. The cost is the exclusion of some feasible edges. Hand-selecting those associations between transition conditions and models facilitates inclusion of feasible models. Nonetheless, the costs involved with this level of accuracy are the potential human error and effort needed to filter out infeasible models. Choosing to include all difficult associations is liberal, in that it ensures inclusion of all feasible models. The cost is the inclusion of some infeasible models, hence the inclusion of some undesirable staged compositions that enforce models to be simulated even when their qualification conditions are violated. Nevertheless, it is possible to screen out such models using an online model recommender.

The second more ambitious yet flexible approach is to delay the enumeration process until a model is qualified at run time. Run-time generation of feasible staging using the graph of model families requires monitoring and evaluation of transition conditions as the simulation unfolds. An agent-planning layer connected to simulator would be capable of identifying, qualifying, and, if necessary, selecting and instantiating a model based on the specified preferences and options. Furthermore, in the case of an impasse or lack of knowledge on preferences among qualifying model switch strategies, a planning layer can guide exploring alternative contexts (games) in some order. The scheduler agent follows the recommendations made by the planner agent to instantiate distinct simulation frames.

Focus points maintain candidate models and associated simulations. A focus point manages branch points in the simulation frame stack. Suppose that a goal instance (i.e., stage transition condition) is at the top of the stack. If only a single

model qualifies for exploration, then it is pushed onto the stack. Yet, if more than one model matches the condition, a simulation focus point is generated to manage newly created simulation branching (discontinuity) points. Each one of these simulation focus points has its own context. When a path is exhausted, the closest focus point selects the next available model to instantiate the simulation frame or return to the context that generated the focus point. As simulation games are explored, a network of focus points is generated. Determining which focus point should be active at any given time is the responsibility of the meta-scheduler. When more than one model is qualified, then scheduler needs to decide which one to instantiate. Control rules can inform its decision. Three steps involve in deploying a new simulation frame in such cases: matching, activation, and preference. The matching steps should both syntactically and semantically satisfy the request. The activation step involves running a dynamic set of rules that further test the applicability of models with respect to contextual constraints. Finally, the preference steps involve running a different set of rules to impose an activation ordering among the active frames.

1.4 Agent-Supported and Model-Driven Simulation Experiment Management

Model-driven engineering (MDE) has emerged as a practical and unified methodology to manage complex simulation systems development by bringing model-centric thinking to the fore. The use of platform-independent domain models along with explicit transformation models facilitates deployment of simulations across a variety of platforms. While the utility of MDE principles in simulation development is now well recognized, its benefits for experimentation have not yet received attention. However, simulation is the act of using simulators and models to perform goal-directed experimentation, and hence, model-driven experimentation needs to be an integral part of the overall MDE-based simulation development process. The provision of machine-interpretable experiment models grounded on the statistical Design of Experiments (DOE) methodology will not only enable computer assistance for selecting proper experiment designs but also facilitate reliable evaluation of results. On the other hand, ad hoc scenario configuration files that are often used for conducting experiments are not only difficult to maintain but also lack the capability to define the concepts, relations, and constraints associated with the DOE methodology.

We present our strategy, which brings together MDE (Stahl and Volter 2006), DOE, optimization, and Intelligent Agent Technology (Yilmaz and Ören 2009) to systematically design experiments and execute optimization algorithms by using abstract, but formal, experiment models defined in terms of a Domain-Specific Language (DSL). Model transformation (Rashid et al. 2011) methods facilitate synthesizing experiments and optimization meta-heuristics from high-level

specifications and then allow their orchestration using software agents that interpret the design and conduct the experiments and optimization algorithms.

1.4.1 Domain-Specific Experiment Specification Based on the DOE Metamodel

In the context of MDE, it is mandatory to clearly specify the structure of the problem domain. In our case, it is the experiment design domain. The metamodel provides a domain vocabulary and grammar in the form of an abstract syntax, along with static semantics serving as constraints over the design. One of the first things an analyst must do to design an experiment is to identify the factors. An experiment may have many factors, each of which might be assigned a variety or range of values, called the levels of the factor in DOE. Factors are classified into types, including quantitative/qualitative, discrete/continuous, and controllable/uncontrollable. Simulations come in many flavors. There are deterministic, stochastic, and dynamic (terminating or nonterminating) simulations. A design is a matrix where every column refers to a factor and each row describes a particular combination of factor levels.

Each unique combination of factor levels is called a design point. We extend this characterization of DOE to identify key concepts, relationships, and attributes. Then, we develop a metamodel in the form of experiment domain ontology and encode it as a grammar for defining the Domain-Specific Language. The DOE ontology (Somohano-Teran et al. 2014) serves as the base model specifying the commonalities across DOE experiments. The differences and optional/alternative designs will be captured using a feature model that will be used to extend and configure the base model with selected experiment design options (e.g., factorial design vs. fractional factorial design).

To operationalize the metamodel, we use the *Xtext* environment, which is a platform for developing Domain-Specific Languages. The *Xtext* platform facilitates generation of the parser and a specific editor with syntax coloring and highlights for the developed DSL. The metamodel serves as the grammar of the language allowing users to define an experiment instance in terms of the vocabulary of the experiment domain ontology (Somohano-Teran et al. 2014).

1.4.2 Feature Models for Experiment Variability Modeling

An experiment design can have various mandatory, alternative, and optional features. Features are prominent attributes that facilitate modeling variants of experiments to support different objectives. For instance, the type of the experiment design (e.g., factorial, fractional factorial), the optimization strategy (e.g.,

evolutionary strategy vs. simulated annealing), and the analysis method (e.g., ANOVA vs. MANOVA) are potential features that collectively define plausible configurations of an experiment. A feature model is a model that defines the features and their dependencies in the form of a feature diagram. In our approach, features are interpreted as views into the DOE ontology, and selected features with their associated concept structures are woven into the base experiment model to derive the experiment specification. Each feature is defined in terms of a set of ontological constructs (e.g., concepts, associations, attributes) and is integrated into the experiment specification to configure the conceptual (i.e., data) and experiment workflow components (e.g., batch-run specification, variance reduction method, warm-up period, run length in terminating simulations, number of replications, pseudorandom number generator) of an experiment design.

1.4.3 Agent-Assisted Experiment Model Generators

The major elements of our tool suite for experiment synthesis are highlighted in Fig. 1.3. The DOE methodology and extant research in simulation experiment design (Kleijnen 2005; Sanchez et al. 2014) provide the basis for the formulation of experiment domain requirements. The resultant concept statement is used toward developing the DOE metamodel, which is defined in terms of the Ecore meta-metamodel over the Eclipse Modeling Framework.

The DOE ontology defines the vocabulary and grammar, i.e., the abstract syntax for building the experiment domain space model. To support the instantiation of the experiment models conforming to the DOE metamodel, a suitable Domain-Specific Language is needed. This involves the definition of the concrete syntax and the development of an appropriate editor. Tools that support the creation of DSLs are readily available, and we used the popular Eclipse-based Xtext environment. The feature model provides additional variants to bring additive variability to the base model defined by the DSL. The experiment model defined by the DSL is configured with the aspects specified in the feature model. Weaving an aspect element means that all properties of the element, including its associated components, are woven into the base model. Aspect-oriented modeling methods (Rashid et al. 2011) provided the requisite strategies to achieve this objective.

The Experiment Design Agent, shown in Fig. 1.2, evaluates the generated experiment design matrix and improves the effectiveness and efficiency of the design across the parameter space of the simulation application. A trade-off analysis between the number of design points and the number of replicates per design point is carried out in relation to the type of experiment being conducted. Consider, for instance, two options: one with many replicates per design point and another with more design points with fewer replicates.

The first option enables explicit estimation of response variances that can vary across scenarios. If the primary objective is to find robust system designs, then some replication at every design point is essential. If the goal is to understand the system

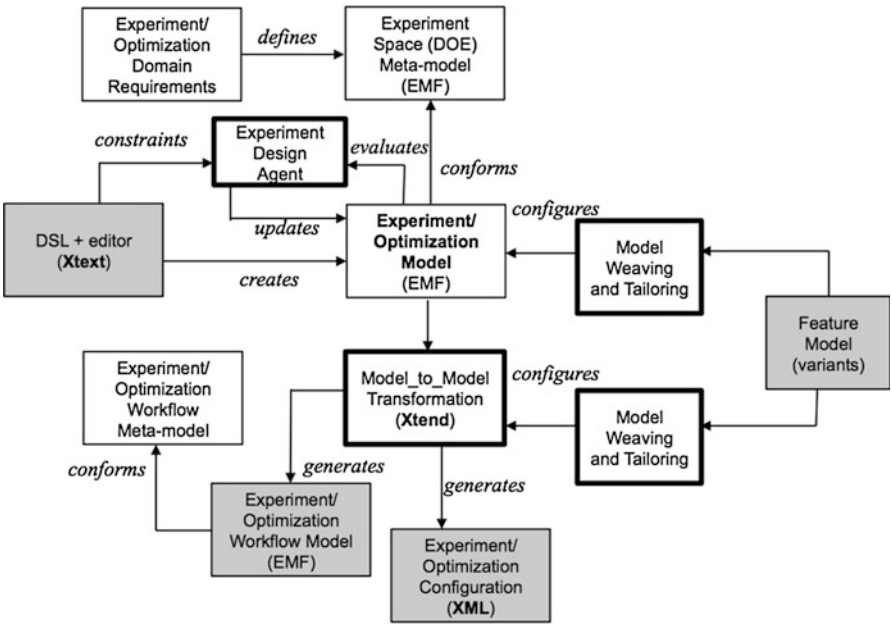

Fig. 1.3 Overview of experiment/optimization model synthesis

behavior, this requires understanding the variance, again mandating replication. However, if the goal is that of comparing systems and a constant variance can be assumed, then this constant can be estimated using classic ordinary least squares regression. Replication is then of less concern, and the second option (exploring more design points/scenarios) can be a better way to spend scarce computer resources. Even beyond the trade-off is the consideration of how many design points and replicates according to the users are needed to conserve or expend computational resources. Is a quick answer needed or are limited computational resources at hand? Or, is time not of the essence and computational resources are not severely constrained? These environments will dictate different appropriate DOE, and the agent interacts with the user to determine the best combination.

The Experiment Design Agent, if necessary, makes recommendations for updating the experiment design by evaluating it across a number of design criteria. A major design attribute is the number of scenarios required to enable estimation of metamodel parameters using saturated designs such as the fractional factorial design. To choose among different designs, *orthogonality* can be used to simplify computations while observing constraints on factor level combinations (e.g., in a queuing system, if arrival rates far exceed service rates, this will result in unacceptable steady-state average waiting times). The experiment design also determines the standard errors for the estimated parameters. The DOE literature uses several criteria to assure that the sum of these standard errors is minimal, and the

design evaluation agent in updating the design prior to execution of the experiment plan can use such criteria.

Space-filling design is yet another area that allows sampling design points not only at the edges of the hypercube that defines the experiment space but also in the interior. A space-filling design with good and balanced space-filling properties helps users avoid making many assumptions about the nature of the response surface and hence results in a robust design (Sanchez et al. 2014). In simulation experiments, restricting factor values to realistic combinations may complicate the design process. To this end, we need features in the DSL to allow the user to express such constraints for interpretation and evaluation by the design agent. Furthermore, the agent should embody knowledge about the conditions and constraints under which specific experiment design schema work. The number of factors and the complexity of the response surface are two critical criteria to assess and classify designs. For instance, while coarse grids used for variable screening are effective if the number of factors is few and the complexity is low, sequential bifurcation (Sanchez et al. 2014) is used if the number of factors increases. Moderate levels of complexity and number of factors are often handled by composite or central composite designs (Law and Kelton 2000). On the other hand, if the number of factors and response surface complexity are high, Latin hypercube and frequency designs are often used. The design agent will make recommendations and configure the design matrix to better fit the characteristics of the problem and experiment space.

Following the instantiation of the experiment model under the Eclipse Modeling Framework (EMF), model-to-model transformation is utilized via the *Xtend* platform to synthesize both the experiment design matrix (in machine-interpretable XML format) and the experiment workflow model that includes specification of the computational activities for the selected experiment type. For instance, an evolutionary strategy optimization feature requires components (e.g., to support the variation, interaction, selection processes) that are different than elements of a strategy based on particle swarm optimization that seeks optimal design parameters. In this domain-specific architecture, the abstract experiment and feature models represent the experiment domain, whereas the generated workflow model along with the concrete experiment configuration represents the experiment workflow space. The workflow space model is used toward generating experiment infrastructure modules, which are initialized and coordinated by the Experiment Orchestration Agent to conduct the experiment according to the plan specified in the experiment configuration.

Figure 1.4 highlights the major components needed for the synthesis and execution of the experiment. By using the Eclipse-based *Xpand* Model-to-Code generation facility, we define a template for each workflow type to instantiate an application skeleton comprised of generic software components that implement the workflow. The orchestration agent initializes the experiment infrastructure modules and defines the simulation settings such as run length, warm-up period, variance reduction method setup, batch-run specifications, design point (scenario) initialization, and distribution of scenario executions.

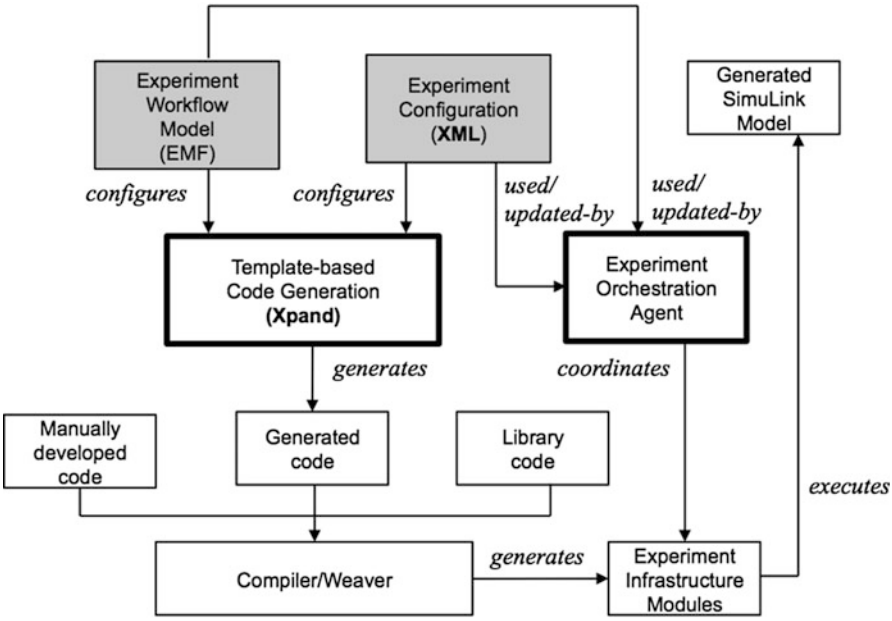

Fig. 1.4 Overview of experiment code synthesis and orchestration

The orchestration agent incorporates experiment design adaptation capabilities so that factors that are not significant in explaining the differences in the dependent variables are reclassified as control variables, and, if necessary, design schema will adapt as experimentation moves from variable screening to factor analysis and then to optimization.

1.4.4 Analysis Model Derivation

The aggregation of results for effective analysis and communication is a critical step. The mapping of the raw simulation data to abstract models that conform to specific configurations that lend themselves to effective communication and analysis benefits from model-driven engineering and transformation principles. For identifying robust solutions, a good analogy is exploratory analysis. Three-dimensional rotatable plots, contour plots, and trellis plots provide features that support exploration. Hence, data models that capture the ontology of these visualization artifacts are effective in conveying the results. Also, regression trees and Bayesian networks are effective ways of communicating which factors are most influential on the performance measures.

The process involved in transforming the raw simulation data to a format that can be consumed by external analysis tools has two main steps: (1) syntactic