Using Traditional and Agent Based Toolsets for System Dynamics: Present Tradeoffs and Future Evolution

Nathaniel Osgood

Department of Computer Science
University of Saskatchewan
Email: osgood@cs.usask.ca

1 Introduction

1.1 Background

System Dynamics practitioners have most commonly modeled a systems using a set of state equations (coupled ordinary differential equations). A rich modeling tradition has grown up around this representation, including diagrammatic representations (e.g. stock and flow diagrams), formal analysis techniques (e.g. eigenvalue elasticity analysis), calibration and equilibration techniques and system archetypes and molecules. A number of feature-rich software packages have also been created to support most stages of the state equation modeling process.

Within the past half-decade, a different set of software tools has increasingly been applied by System Dynamics modelers: Agent Based modeling packages. Models created using such software allow for the representation of populations as collections of agents, reified as objects with state (attributes) and parameters (properties). Such packages (such as Repast [1-3], SWARM [4], NetLogo [5], SDML [6], Jade [7], MASON [8], and AnyLogic [9, 10]) are quite extensive and have been well used by the Agent Based modeling community for years.

The encounter of System Dynamics modelers with another set of rich modeling tools has opened considerable opportunities, but has also been accompanied by significant confusion. The features and conventions of traditional System Dynamics modeling packages[1] differ significantly from those of traditional Agent Based packages. These differences raise important and common questions: How

[1] A word on terminology: This paper examines differences and tradeoffs in toolsets historically most closely associated with the System Dynamics and Agent Based communities. While we believe that both of these toolsets are useful for understanding and managing feedback-rich systems and are capable of giving excellent results when employed in accordance with good System Dynamics practice, it is convenient to use brief labels for of these toolsets in order to discuss their many important differences. Choosing short labels is difficult, as it risks presupposing what constitutes the "essential" differences – and tends to ignore other differences that have significant practical consequences for modelers. For example, labeling traditional System Dynamics tools as "ODE-based tools" focuses just one component (mathematical framework) that unites these tools, at the cost of downplaying other important shared features of these tools, such as the support for unit metadata, declarative representation, longitudinal visualization, etc. To keep the reader's attention on the fact that we are primarily focused on tradeoffs between toolsets, we have chosen to use the phrases "traditional System Dynamics tools" and "traditional Agent Based modeling tools".

should one best identify the most appropriate tool set for a given problem? What (if anything) are the essential differences between these modeling tool sets? How fundamental are these differences?

The discussion has been muddied by the fact that there are numerous kinds of differences between traditional System Dynamics and Agent Based modeling packages: Differences in the level of granularity of a model, in how the model behavior is specified, in the state abstractions and nature of the rules employed, etc. The diversity of these differences has led to many different viewpoints on the "essential" character of the differences between traditional System Dynamics modeling approaches and those of Agent Based modeling. Some have characterized the traditions as methods in competition for the same sort of models. Others have argued that the critical difference is that between techniques that specify behavior with *Equations* vs. algorithmic rules [11, 12]. Other approaches have pointed to the primary difference being that of the level of model granularity [13]. Looking forward, it is not obvious to what degree the differences separating traditional System Dynamics and traditional Agent Based packages are necessarily linked, and to what degree are they can evolve independently.

1.2 Paper Organization

This paper seeks to shed some light on the varied short-term tradeoffs between traditional System Dynamics tools and Agent Based modeling tools, and some educated guesses as to how these differences may evolve over the longer-term.

While we view both traditional Agent Based and traditional System Dynamics tools as highly useful vehicles for System Dynamics modeling, it is undeniable that these tools presently impose very different constraints on the nature, expression and analysis of these models. Essential or not, these differences currently have substantial practical implications and that must be weighed by modelers. The next two sections of the paper examine the current state of these tradeoffs. Specifically, the next section begins with a systematic consideration of many characteristics by which these tools differ. These factors relate not only to the mathematical frameworks employed but also to the way in which the packages support model specification and analysis. For each such characteristic, we note some of the tradeoffs that apply to choices with respect to choices of these characteristics. Section 3 then summarizes these tradeoffs.

In the penultimate section of the paper, we take a broader look at the landscape of modeling choices and the tradeoffs involved, and argue about possible futures for the toolsets of traditional System Dynamics and those of traditional Agent Based modeling. We submit that because of the modeling tradeoffs involved, the toolsets over time will converge on support for many common features of, including (mostly) declarative, mathematically precise model specification (including both discrete and continuous components), support for units and dimensions, longitudinal analysis of model results, and modular models. Nonetheless, we believe that both aggregate and individual-based approaches will coexist due to compelling long-term benefits offered to certain modeling problems. Some of these tradeoffs derive directly from model representation, while others reflect the indirect effect of the model approach on the modeling process, such as the opportunity costs associated with time needed to construct, simulate and analyze models.

2 Taxonomies

Within this section we classify systems models according to several characteristics. This taxonomy is not meant to be exhaustive, and is focused specifically on differences separating traditional System Dynamics and traditional Agent Based modeling packages.

2.1 Granularity: Aggregate vs. Individual-Level

2.1.1 Core Concepts

The first distinction we wish to make concerns the granularity at which a model represents one or more populations. The distinction here is similar to the classic distinction between lumped and distributed state space models [14].

Aggregate (lumped) population models characterize a population as being divided ("stratified") into one or more equivalence classes, typically according to their attributes. For example, a model may distinguish between Men and Women, or according to age, gender, and smoking status [15, 16]. While individuals will typically flow between equivalence classes, the stocks are treated as "well mixed" in that no attempt is made to distinguish individuals within a given equivalence class. Equivalence classes are generally stratified both by (dynamic) state (e.g. age, disease stage, or smoking status) as well as by (static) parameters (e.g. gender, ethnicity).

By contrast, individual-based (distributed) population models maintain distinct information on every individual in the population. Every individual is associated with distinct parameters and states.

2.1.2 Benefits to Precision

The fine-grained nature of individual-level models allows them to represent certain types of processes, relationships and interventions with greater precision and flexibility than is possible with aggregate-level models. Such differences can lead to significantly – and sometimes qualitatively –different simulation results ([17-19], [20] cited in [17]). Examples of areas where individual-level models can offer improved precision include the following:

- **Linkages**. Individuals in real-world populations exhibit persistent preferential interactions with particular sets of individuals. Such persistent linkages can exert strong influence on health issues, due the likelihood of transmitting infection (such as among sexual partners or co-located intravenous drug users), social determinants of health (e.g. social support networks [21, 22]), and transmission of risk attitude among peers or family members, and environmental factors (e.g. second-hand smoke). Some of these linkages between individuals are long-term – for example, those linking an individual to other family members, or among particular cohorts within a school. Other linkages are more ephemeral – such as those among casual friends or transient co-workers. Individual-based models can readily accommodate the representation of persistent or dynamic linkages between individuals, either as sparse network structures or with connectivity matrices. By contrast, aggregate models can only approximate such situations with aggregate statistics. Such statistics can be challenging to formulate (given that one often starts with understanding of individual-level network structures). Because they are often based upon equilibrium assumptions (e.g. fixed aggregate patterns of heterogeneity or likelihood densities of behavior change), such

statistics can be misleading in non-equilibrium situations, such as might be triggered by an intervention.

- **Memoryful Processes.** In memoryful processes, the likelihood of a given individual's transition to a new state is dependent in some on their residence time in that state. Because they assume well-mixed populations within each population equivalence class, aggregate models typically represent transitions processes using (memoryless) exponential transition processes. While modeling tools often permit representation of other special transition processes (e.g. those associated with fixed transition times), aggregate models cannot directly represent arbitrary memoryful processes. An awkward compromise is often secured by adapting model structure to explicitly approximate process transitions via disaggregation. While this can yield a close approximation, it can be an indirect and somewhat burdensome structural change force-fitted to accommodate details of process statistics.
- **Heterogeneity.** As we seek to represent additional types (dimensions) of heterogeneity for a population, the number of equivalence classes and computational resource demand rise geometrically with the count of dimensions, and attribute-based disaggregation becomes unwieldy [23]. By contrast, the cost imposed by individually disaggregated models only rises linearly, as each simulated individual need only represent an additional piece of data.
  Other heterogeneity-related motivations for the use of individual-models reflect the desire to represent individuals sparsely distributed over a large geographic area, to analyze the effectiveness of individually-targeted interventions (such as are used in understanding and combating sexually transmitted diseases by targeting key elements of an infection spread networks), and the need to capture the differential impacts of a given intervention across different segments of a population (once again, a common concern in public health).

2.1.3 Tradeoffs and Opportunity Costs

While individual-based models have many virtues to recommend them, these benefits must be balanced with a set of costs. These costs are often in the form of reduced agility in one or more stages of modeling.

Frequently time is a key limiting resource in modeling, and there are typically pronounced opportunity costs involved in modeling: Effort spent in one task limits the attention that can be devoted to others. A systems perspective reminds us of the importance of consider both direct costs (e.g. higher construction time) and indirect costs (such as the opportunity costs of any extra time required for model construction, simulation and analysis). We briefly comment below on impact of individual-level modeling on model construction, parameterization, calibration, validation, simulation and analysis.

2.1.3.1 Model Construction Time: Precision vs. Accuracy

Regardless of framework, building individual-based models within often requires more effort than building aggregate systems. In light of the limited modeler (and, all too frequently, calendar) time available, one result is that there may be less effort spent in refining the model after it is constructed. As noted by Sterman [24], the effort expended in deepening a model may inhibit critical broadening of a model – for example, to include behavioral components, or feedbacks involving factors that are otherwise assumed to remain constant. While the resulting model may more precisely simulate the narrow equilibrium dynamics of a particular fixed set of behaviors, the failure to include critical feedbacks may more important inaccuracy than would an aggregate approximation to low-level population dynamics [24].

The detail complexity of individual-based models models risks securing undue user confidence in the precision of model results, resulting in a model that is "precise but not accurate."

There are important caveats here. Firstly, the author has experience with modeling contexts in which it is faster and more reliable to build a disaggregated model to simulate a low-level phenomenon of known importance than it is to derive the high-level statistical relationships that would be required to describe that phenomenon at an aggregate level. An individual-based model of even phenomena can often given rise to concrete representation of emergent behavior that cannot be found in existing data. While reflexive use of individual-based modeling frequently risks "gilding the lily" of model detail without considering broader behavioral factors, the fact remains that because it is at the individual level that individual choice (behavioral feedbacks) based on limited rationality can often most easily and transparently be represented. As successful individual-based traffic models can attest [25-27], individual-based models can be very naturally coupled with individual-based behavioral models, such as those derived from discrete choice theory.

There are also cases where details of individual-level phenomena are absolutely critical for high-level dynamics, or for effective intervention planning. While it is often possible to construct an aggregate model of the emergent phenomenon after it has been simulated at a low-level [13], flexibility considerations (for example, the ability to plan interventions at a lower level, or the ability to capture the results of interventions that shift the patterns of heterogeneity in a population [28]) will frequently make desirable an model that includes endogenous simulation of those individual-level phenomena.

2.1.3.2 Model Parameterization Time

Because of the additional detail they include, individual-based models frequently require a larger variety and volume of parameter estimates than do aggregate models. Most notably, such models will typically allow for separate parameter values for every individual within the population.

While there are certainly cases in which individual-based data allows for more rapid parameterization of agent-based models than it does corresponding aggregate models of similar phenomena (due to the need to "crunch" this data into aggregate measures), the greater volumes and varieties of data required by agent-based models typically make parameterization of individual-based models a lengthier process than what is required for aggregate models.

2.1.3.3 Model Calibration and Validation Time

An additional consequence of the large number of parameters associated with individual-based models is the likelihood that a larger set of parameters will not be fully known. Such parameters generally require estimation during calibration. Unfortunately, the much greater number of parameter values required for parameter calibration in individual-based models often means that calibration is an underdetermined problem, with many possible assignments of values to model parameters all yielding similar model output. While similar situations are common for aggregate models, the likelihood and degree of the problem is much higher for individual-based models.

Model calibration is often closely tied with model validation (or non-falsification). When we use calibration to estimate model quantities about which we lack data, it is important to confirm the results both against known data series for model outputs or intermediate quantities (including relevant reference modes). Important recent contributions in the individual-based modeling area have emphasized the

importance of confirming the consistency of model results with many patterns of real-world phenomenology [12].

Because calibration is underdetermined for most individual-based models, individual-based models will often require careful cross-checking of more possible calibrations results than would be required for more aggregate models. We note below that running an individual-based model is generally more expensive than running an aggregate counterpart. Both because of the larger count of iterations involved and because of the larger calendar time required for each such iteration, arriving at an acceptably calibrated and validated individual-based model can therefore require substantially more time than would be required for aggregate models.

2.1.3.4 Model Simulation Time

An additional critical effect relates to the time required to simulate a model. The time needed to simulate individual-level models is often greater than that for aggregate models. These costs reflect the high cost of simulating a large population. Simply simulating the behavior of each individual in the population typically imposes a heavy cost; interactions between individuals can yield much higher costs yet: For models with dense ($\omega(1)$) networks, performance costs rise superlinearly with population size. This computational burden is significantly worsened by the frequent need to run Monte-Carlo analyses in order to capture the pronounced stochastic effects that obtain at the individual level. While this disparity can be reduced through clever tool design, optimization and parallelization, the fact remains that simulating systems with very large state spaces is likely to be considerably more expensive than simulating models with small state spaces.

In addition to being a direct burden, the higher simulation time of individual-based models imposes indirect costs by slowing and impeding model analysis. The next section examines this effect in greater detail.

2.1.3.5 Model Analysis Time

The larger simulation burdens of individual-based models have a first-order impact on model analysis – and sometimes all but prevent interactive exploration of model dynamics. There is a significant reduction in insight from asking "what if" questions if the answer only arrives after the question is no longer in short-term memory. Tools can aid this process: Techniques for batch simulation of exploratory scenarios ahead of time and for fostering a user's memory of the purpose of a scenario when examining scenario results can help reduce – but not eliminate – the insight gap.

Both aggregate and individual-based models can give rise to counter-intuitive emergent behavior. Understanding how the model produces such behavior is naturally easier when there are fewer "moving parts" (e.g. state variables) within a model. A common result is that users of individual-based models often spend considerably more time analyzing their model results than would be required for aggregate models. This extra analysis time often takes time away from other tasks, such as model refinement (including broadening), improving model calibration, data collection, etc.

As a result of both slower simulations and the need to examine effects across a larger population, individual-based models with larger populations necessitate slower and more laborious analysis.

2.1.3.6 Summary

The indirect effects of model granularity discussed in this section can have significant impact on the modeling process, particularly in light of gaps in existing tool support. Even as tool convergence proceeds, we expect that the larger time generally required to build, parameterize, calibrate/validate, execute and analyze individual-based models will maintain the popularity of aggregate and multi-scale modeling approaches.

2.2 Continuous vs. Discrete Rules

The section above has focused on model granularity and its implication for the modeling process. Models may be further characterized as to whether they are continuous and/or discrete in rules (and, by implication, in time). A closely related question pertains to whether such systems make use of discrete or continuous rules (e.g. to represent policies).

Discrete time systems proceed in a punctuated, discontinuous fashion at particular points in time. Continuous time systems are treated as evolving in a smooth fashion over time. In terms of time-behavior, real-world systems will often include both continuous and discontinuous processes. Deriving solutions to continuous-time systems on digital computers requires discrete approximations (such as are used in numerical integration algorithms). Failure to take into account the presence of discrete rules within a model during simulation can lead to significant (and occasionally dramatic) inaccuracies between the discrete approximation and the continuous-time solution. For example, strict lower bounds on state variables (the level of a reservoir, or size of a population) will often impose discrete constraints and other discontinuities on otherwise continuous processes. Traditional System Dynamics packages generally assume a purely continuous time set of processes. Failure of numerical integration algorithms (such as those typically used in traditional System Dynamics tools) to explicitly account for such behavior can lead to highly visible artifacts, such as unphysical reports of negative volumes of the reservoir or negative populations. While selection of smaller timesteps will often reduce the extent of such discrepancies, a fully general solution requires explicit handling of discrete rules by the framework.

2.4 Specification Characteristics

The sections above have focused on the mathematical structure represented by a model – regardless of how that model is specified. Another important difference separating traditional Agent Based toolkits and those used in traditional system dynamics concerns model specification. A given model may be specified (implemented) in many different ways – using a series of equations, as code in a general purpose imperative programming language, in domain specific languages, etc. This section discusses several ways in which the specification mechanism used for a model can vary, and the consequences of these differences on the modeling process.

2.4.1 How vs. What

A critical practical distinction regarding specifications concerns the degree to which a modeler is focused on the *what* vs the *how* of simulation.

In accordance with standard Computer Science terminology, we use the term *declarative* to describe specification mechanisms that permit the user to focus (almost entirely) on describing *what* model they want to be simulated. Declarative frameworks permit the modeler to specify the relationships they wish

to obtain between model variables over time. The software framework is then responsible for performing the bookkeeping and calculations necessary to maintain these relationships [29].

We contrast such systems with non-declarative (algorithmic) frameworks mechanisms that require a user to specify both the model and a (typically stateful[2]) *how* that model is to be simulated. Such frameworks traditionally require a user to specify (or closely coordinate) implementation details, such as the declaration and initialization of model state variables and intermediate quantities, timestepping mechanism, explicitly order and perform state variables updates, the handle the mechanics of input/output mechanisms, etc.

The distinction is rarely absolute – even declarative frameworks will typically require some measure of user reasoning about the *how* of a simulation. But there may be an order of magnitude or more in the level of time investment required of the user in this reasoning.

Declarative frameworks have a number of notable advantages:

- **Problem-Centric Focus.** Perhaps the most important advantage of declarative frameworks is the problem-centered focus they foster. In line with Djikstra and Parnas' principle of the "Separation of Concerns" [30, 31], declarative frameworks reduce the burden on the modeler of concern regarding the implementation details of a system, thereby permit a clearer consideration of the model formulation.
- **Transparency**. Implementation details are distractions for anyone trying to reason about the structure of a model. Declarative frameworks allow observers to more quickly understand the essentials of a given model.
- **Reduced Likelihood of Errors.** By simplifying and clarifying modeling presentation, declarative frameworks lower the risk that a programmer will overlook model formulation errors. In modifying the model, the modeler is also less likely to make a mistake that will jeopardize the soundness of model results (for example, inadvertently violating the implicit ordering constraints among updates to model state variables, or accidentally overwriting an earlier assignment to state variables).
- **Improved Performance.** One of the advantages of delegating the bookkeeping and choreography of model simulation to a software engine is that the engine can perform more aggressive performance optimizations than would generally be possible with code created in general-purpose programming languages (e.g. C++, Java). The greater structure typical in declarative systems affords considerably greater precision reasoning about the correctness of model equation transformation and parallelization than would be possible with general-purpose code [29].
- **Tool Support.** The use of declarative formulations simplifies software support for verification mechanisms (e.g. unit and dimensionality heterogeneity tests), domain-specific error messages, and mechanisms for debugging, analysis and visualization.

[2] Most non-declarative specification mechanisms make use of *imperative* algorithmic programming languages, in which a program operates by manipulating program state. While popular, reasoning about such state-based systems is tricky. An alternative algorithmic specification approach – functional programming – is much closer to declarative mechanisms in that it describes the "how" of a program algorithmically, but in a side-effect-free fashion that fosters transparency and formal mathematical insight by permitting equational reasoning.

The above advantages do not come without tradeoffs. Declarative frameworks are typically more difficult to build than corresponding software for imperative frameworks. This reflects the fact that the software to support the language must typically include mechanisms traditionally relegated to programmer control. A recurring challenge in designing declarative specification techniques (and, specifically, modeling languages) is the need to balance simplicity in the specification of common models with desires for a more general framework [32].

2.4.2 Explicit vs. Implicit Mathematical Semantics

2.4.2.1 Background

An additional – and closely related – dimension by which specifications differ is the degree to which they explicitly adhere to some consistent mathematical approach.

All computational specifications ultimately have mathematical meaning grounded in formal language semantics. The question here is whether the process being specified (which presumably represents some approximation to real-world external processes) is explicitly stated within a consistent mathematical framework, or whether the consistency of the mathematical structure depends upon the happenstance of modeler discipline.

2.4.2.2 Consequences

Whatever its mathematical basis[3], having a clear mathematical framework underlying a model offers several distinct benefits. Firstly, it allows formal checking of properties that are not automatically true of arbitrary programming language code or equations, but which know should hold true for models of the real world. Dimensional homogeneity is a prime example of such a property [33-38]. Just as processes in the real world operate identically regardless of the unit systems maintained by observers, we should be able to formulate a model of the real world with similar properties. At a more operational level, this means that the model should be "dimensionally homogeneous" [37, 38], in the sense that we should expect the units (and dimensions) on the left hand side of an equation to match those on the right hand side, identical dimensions for quantities that are being added, only dimensionless quantities as parameters for transcendental functions, etc.

Having a clear mathematical basis for a model specification (whether algorithmic or algebraic) is the first step towards verifying dimensional homogeneity. By contrast, general purpose programming language code can easily mistakenly include expressions that are numerically feasible but lack clear mathematical meaning – for example, code that adds together quantities drawn from different semantic types ($ + people), units (miles + km) or is otherwise dimensionally inhomogeneous, code that inverts improperly structured dimensional matrices [34], code that updates different state variables using incompatible integration schemes, or that inadvertently updates a state variable based on the value of other state variables in the current – rather than the previous – timestep.

Having an explicit and formal mathematical grounding for a model offers many other key benefits as well. Such a grounding often permits rigorous model analysis and permits the formal reasoning needed for

[3] Appropriate mathematical frameworks for systems models vary widely, from continuous-time differential equations, difference equations, to hybrid discrete/continuous frameworks (such as are used in hybrid automata) and discrete state transition functions characteristic of cellular automata models.

model generalization. Such reasoning may assist the modeler in their tasks, such as in closed-form model equilibration, calibration, and understanding of emergent properties.

Supported analyses varies with the mathematical framework. For example, use of ordinary differential equations permits reasoning about linearized behavior, and eigenvalue-based studies of such linearized systems – for example, with loop and parameter elasticity estimates. It also allows for formal study of how different pieces of the model behave in isolation. Hybrid automata [32] would permit not only ODE-based analyses but additional analyses as well, such as reachability tests, μ-Calculus model checking [32].

Lack of an explicit, consistent mathematical framework hinders formal understanding of the causes of emergent behavior, generalization of models, and model calibration.

2.4.2.3 Linkage to Declarative vs. Non-Declarative Tradeoffs

Generally speaking, fully or mostly declarative specifications facilitate mathematical reasoning. For example, declarative representation of a system as a set of differential equations, a state transition diagram, an attributed network, or a hybrid automaton would allow for straightforward application of the appropriate varieties of mathematical reasoning.

Algorithmic specifications of a model may also be more or less mathematically explicit. For example, functional (side-effect-free) programming languages also enable mathematical reasoning, by virtue of their support for equational reasoning, such as the ability to substitute an argument for a formal parameter, the ability to evaluate expressions without regards to ordering constraints, etc. [39]

2.4.3 Metadata Support

The important role of dimensional homogeneity tests was noted above. Dimensional information is a particularly useful example of *metadata* that can be maintained for simulation models. The techniques of dimensional analysis allow great value to be derived from dimensional information during multiple stages of the modeling process [40]. Other examples of helpful metadata include information on a datum's source (data repository or calculation of origin, date at which the data was obtained), measurement regime (e.g. sampling frequency), degree of confidence (e.g. specified by a probability distribution or qualitative statement), etc.

2.4.4 Modularity Mechanisms

Models are often composed of pieces. An important consideration for a particular framework is the abstraction mechanisms (if any) that will allow for a system to be built up from pieces. Object-based abstraction (including separating model from implementation) is a common technique for individual-level modeling, but is far from the only option for modelers. Procedural abstractions (and associated vector-based data structures) are more traditional technique, and can be used to good effect, for example, in conjunction with individual-based models [41].

2.5 Analysis/Visualization Support

The main purpose of many models is to gain insight into the phenomena in the real world, and the impact of actions on those phenomena. In order to better understand a complex system in the real world, we build a model of that system that is simpler, but still often quite complex. We hope that this model will capture important aspects of behavior in the real-world, and seek to understand the causes for the overall behavior of this model.

Tools to facilitate nimble and insightful analysis facilitate both direct delivery of greater value from the model (in terms of improved understanding) and indirectly (by permitting the modeler to put additional effort into other modeling tasks, such as refining the model or more completely exploring the policy space.)

3 Impact of Model Characteristics on Quality Attributes

The discussion above both laid out some general characteristics with respect to which we can classify modeling tools and noted some of the impacts related choices impose upon aspects of model quality. We roughly summarize those tradeoffs in the table below.

	Transparency	Performance	Ease of Creation	Generality	Analyzability/ Understanding	Ease of Parameterization	Ease of Calibration& Validation	Accuracy	Model Breadth	Scalability (Population)	Scalability (Heterogeneity)	Modifiability
Aggregate	+-	++	+	-	++	+	++	--	++	++	-	+
Discrete & Continuous	+		+	++	+			+				
Declarative	++	+	++		+				++			++
Equational	++		+		+		+		++			+
Explicit Math	++		+		++		++					++
Modular	++		+						+			++
Longitudinal reporting/ visualization					++							
Metadata Support	++		-		+					++		+

Table 1: Direct and Indirect Impacts of Tool Support (Rows) on Quality Attributes (Columns)

4 Traditional System Dynamics and Agent Based Tools: The Short-Term

System Dynamics practitioners presently have access to both traditional System Dynamics tools and traditional Agent Based tools. These toolsets presently differ significantly in terms of many characteristics above. Within the next subsection we briefly summarize the implication of each of these toolsets for the classification dimensions introduced in Section 2. The following subsection then summarizes the implication of this support on model quality attributes.

4.1 Tool Support

- **Granularity: Aggregate vs. Individual-Level.** Existing traditional System Dynamics tools provide basic support for both aggregate and individual-level models, but the typical restriction to static equations and fixed numbers of state variables (subscripted or otherwise) complicate the construction of individual-level models. While a fixed-size individual-level population can be easily modeled using arrayed indices, fluctuating populations are considerably more awkward to capture. Similarly, while it is simple to construct matrices to represent networks of individuals, the representation is needlessly expensive (in both space and time) for sparse graphs, and somewhat awkward for dynamically shifting networks. By contrast, traditional Agent Based tools tend to be crafted for convenient representation of individual level models, but offer only limited support for aggregate representations. The capacity to mix and match aggregate structure and individual-level structure in multi-scale models makes AnyLogic [10] a particularly attractive system.
- **Continuous vs. Discrete Time & Rules.** The numerical integration schemes used by traditional System Dynamics tools generally employ uniformly spaced timesteps that make no attempt to support hybrid discrete/continuous rules. As noted in the Section 2, such rules are quite common in managerial contexts and for enforcing capacity constraints. The presence of such rules can lead to problems with numerical integration (and often lead to recourse to extremely fine timesteps as an unsatisfactory and expensive workaround). Many traditional Agent Based tools have used an equally rigid but fixed discrete regimen, with no attempt to provide support for a continuous time abstraction; the modeler is thus responsible for imposing any explicit timestep framework. AnyLogic [10] is once again notable for its support for hybrid continuous/discrete logic, yielding a more flexible and accurate framework.
- **How vs. What.** The Agent Based community has long made extensive use of imperative languages object-oriented such as Objective C, C++ and Java. Unfortunately, the traditional reliance of Agent Based approaches upon code created in general-purpose programming languages imposes a stiff penalty on model transparency. The need for variable and data structure declarations, bookkeeping duties, the syntactic rules and complex data flow of general purpose programming languages hinder modeler focus on the underlying model (and the problem that the user is trying to solve).

  These negative effects on transparency are not limited to manual analysis – use of general purpose programming languages hinders automated analysis as well. For example, Unit homogeneity checks are considerably complicated by the presence of general-purpose data structures and iterative and recursive branch and looping constructs.

  On a hopeful note, it is worth observing that within recent years, there has been considerable progress in enriching the software support for declarative Agent Based methods in some frameworks, most notably the AnyLogic [9, 10] toolset and the upcoming RePast Simphony [2, 3] framework.

  By contrast, traditional System Dynamics modeling software has for years supported almost purely declarative model specification. Within such packages, a modeler must describe little more than the integral equations that must obtain during model execution and initial conditions. As noted above, such declarative specifications offers substantial and multi-faceted benefits to the modeler.

- **Analysis & Visualization Tools.** Traditional System Dynamics software offers considerable functionality aimed at improving insight into model execution. Such software records longitudinal transcripts of model execution that specify the value held by each model variable at many (if not all) timepoints and allows the user to quickly and easily create longitudinal graphs of such behavior. Additional functionality is often provided to create custom graphs (without the need to re-run the model), explore subscripted variables, graphically view empirical fractiles and mean values of Monte Carlo ensembles, and report descriptive statistics. These features are particularly valuable when used with aggregate models. Because of the fairly rapid simulation time of most such models, providing tools for nimble analysis permits truly interactive exploration of the dynamics of many models. Analysis tools in System Dynamics packages could be made considerably more nimble and convenient yet – for example, by permitting the use of ad-hoc queries involving functions of existing variables, and for better support for visualizing individual-level models, where individual information is spread across several arrayed stocks. But it is clear that modelers benefit greatly from the accessibility of the analysis and visualization features of System Dynamics tools.

  By contrast, Agent Based packages traditionally offer very limited analysis support. Library frameworks such as Repast and Swarm require users to shoulder much of the work of providing visualizations of model behavior. While such packages provide users with class libraries that support creating graphs, exporting data, etc., the user must add to the model definition code to create, initialize and invoke these instances of these classes. Because the process of creating a new (and previously unanticipated) visualization involves modifying model code, recompiling and reexecuting a model, and because Agent Based model simulation is often highly expensive, such traditional systems raise significant barriers for exploration of model behavior. Newer packages, such as Simbuilder, NetLogo and AnyLogic provide additional facilities. The most advanced of these (AnyLogic) is particularly notable in that it provides built-in mechanisms to interactively run a simulation, visually explore the current state of a simulation (at the current timestep), and save away transcripts of the simulation. While a great advance over the tools offered by traditional Agent Based modeling packages, the user seeking to understand longitudinal behavior of a system over time must either anticipate the need to create a longitudinal view of a state variable prior to a simulation or apply external tools to exported data.
- **Explicit vs. Implicit Mathematical Semantics.** Traditional System Dynamics tools have long worked with models formulated in a specific, uniform mathematical framework – ordinary differential equations. The use of this framework has been critical to developing insights into the behavior of System Dynamics models. The approach has also been the basis for diagramming, analyzing, and generalizing models, for reasoning about dimensional homogeneity, etc.

  By contrast, traditional Agent Based tools have for the most part been mathematically agnostic, delegating responsibility for the mathematical structure of a model to the user. While models built in these techniques typically exhibit rich emergent behavior, the lack of mathematical transparency hinders reasoning about the specific causes of this emergence, generalization, formal analysis, etc. The lack of an explicit mathematical statement of model structure also greatly complicates checks of basic model soundness (e.g. dimensionally homogeneity). The AnyLogic toolset provides a great advance for this tradition in that it permitting modelers to use of ordinary differential equations for individual-level models, in addition to supporting mathematically precise representation of processes such as probabilistic state transitions.

- **Metadata Support.** Traditional System Dynamics tools have long supported annotation of models with metadata. The sole type of semantic metadata presently supported is structured algebraic reasoning are unit annotations, but most packages also allow for a means of incorporating additional comments on model variables. Traditional Agent Based tools still lack support for the rudiments of unit metadata. This limitation reflects the difficulty of performing unit homogeneity checks on general purpose programming language code even were such information available.
- **Modularity Mechanisms.** Traditional Agent Based systems models have made excellent use of the class-based abstraction mechanisms provided by object-oriented programming languages. Use of class-based abstraction can help facilitate reuse of code between frameworks and certainly fosters a separation of concerns on the part of the modeler. There has also been considerable success implementing individual-level simulations within procedural paradigms that lack any object-based abstraction (e.g. MATLAB [41]).
  By contrast, despite articulation of common building blocks of models such as molecules and some innovation by individual researchers, traditional System Dynamics packages have generally made use of monolithic models with little opportunity for structured reuse of model components.

4.2 Implications for Model Quality Attributes

In the short term, we argue that existing tools of traditional System Dynamics and Agent Based approaches have much to offer – and much room for growth. The table below summarizes some of the tradeoffs implied by the above analysis. The contents of each entry in the table take into account the relevant discussions within the previous 3 sections. While the ratings are somewhat subjective, the table does clearly bear out the fact that both traditional System Dynamics tools and traditional Agent Based Modeling tools offer strong benefits, but in different areas.

Existing System Dynamics tools are mature, and offer many virtues. The declarative, explicit mathematical frameworks go a long distance in simplifying understanding, analysis, and calibration of models. Unfortunately, the generality and accuracy of the frameworks is somewhat limited by incomplete support for individual-based modeling, which is important for representing details of some processes and interventions. The modifiability of such tools is compromised by lack of modularity effective mechanisms; modelers seeking to re-use model structure often resort to cut-and-paste approaches, which are fragile given model evolution.

At this point in their evolution, the biggest shortcomings of Agent Based modeling tools is that they risk forcing the user to muddle through too much of the "how" (forcing the user to focus on the problem-irrelevant minutia of choreography and bookkeeping of a simulation), and, improperly used, provide too little of the "why" (the critical understanding of the critical *reasons* behind observed emergent behavior). Consistent use of a formal mathematical structure would aid in both of these areas. Performance is also a major concern due to the opportunity costs imposed, particularly on model analysis.

It is also worth emphasizing that the table imperfectly captures the textured nature of the tradeoffs; for example, the transparency of traditional Agent Based models can exceed that of traditional System Dynamics models when representing fluctuating highly heterogeneous populations. As another example, we judge the "modifiability" of both traditional System Dynamics and Agent Based models as being

fairly good for different reasons; the former enhances modifiability through transparency and mathematical clarity, while the latter through the generality achieved by fine-grained representation.

	Transparency	Performance	Ease of Creation	Generality	Analyzability/ Understanding	Ease of Parameterization	Ease of Calibration	Accuracy	Model Breadth	Scalability (Population)	Scalability (Heterogeneity)	Modifiability
TSD	++	++	++	+	++	++	++	+	++	++	-	+
ABM	+		+	++		+	+	++			++	+

Table 2: Current Toolset Implications for Model Quality Attributes

5 Looking Forward

The previous section has examined some of the tradeoffs that presently obtain between traditional System Dynamics tools and those of traditional Agent Based modeling. Within this section, we take a longer-term – and more speculative – view on the evolution of the approaches.

While there are presently good reasons for choosing different sets of modeling tools for different modeling problems, the above analysis must be taken with caution. Far from being logically linked and inseparable, we would argue that many characteristics of the toolsets are superficial historic artifacts that reflect the happenstance of tool development rather than inherent tradeoffs between two different modeling paradigms. For example, there is no inherent reason why traditional Agent Based modeling tools could not evolve to make use of a fully explicit, declarative mathematical specification, or to support unit heterogeneity checks. On a somewhat more incremental basis, the use of functional programming languages would make code in Agent Based models significantly more transparent – for example, all updates to a particular state variable of a given agent could occur in one point, and mathematical reasoning about the behavior of Agent Based systems would be simplified by virtue of the equational reasoning permitted. Similarly, traditional System Dynamics tools could be broadened to better support hybrid discrete/continuous rules, or dynamic populations of agents realized via a more flexible subscripting mechanism.

We believe it is likely that that both sets of tools will evolve to reflect the shared concerns and needs of System Dynamics modelers. Within Section 3 we have described some of the relationships between particular characteristics of model architecture and specification mechanism and quality attributes. We expect that modeling packages will in general evolve so as to deliver increasing value to modelers, as expressed through those quality attributes.

Based on our brief survey of these characteristics above, it would appear that modelers would significantly benefit from tools with certain characteristics, including the following:

- **Declarative specification.** Specification mechanisms that permit a user to focus on the "what" rather than the "how" have manifest benefits to the modeler, to communication, and to performance. While there are domains in which achieving declarative specification is likely to be highly challenging, we believe that there are compelling existing declarative mechanisms for both aggregate and individual-based models (for an example of each, consider the declarative dialects used by traditional System Dynamics modeling tools, and the declarative specifications for hybrid automata widely in use in hybrid analog-digital control theory).
- **Support of both discrete and continuous time transitions.** As traditional System Dynamics has shown, continuous-time frameworks are extremely attractive for modeling both natural and artificial systems. Discrete constraints are also common in such processes (e.g. reflecting capacity limits) and in rules used by managers to regulate them. Tools that support both continuous and discrete time reasoning are therefore of clear desirability, and would eliminate some common inaccuracies seen with traditional System Dynamics models.
- **Support for full spectrum of granularity.** System Dynamics models have common needs for both aggregate and individual-based structures. Traditional System Dynamics tools would offer valuable by supporting mechanisms that will support more flexible individual-based models, such as dynamically-resized subscript dimensions or subscript sets. AnyLogic's support for both individual-level and aggregate formulations is especially valuable in this regard.
- **Support for rich longitudinal and cross-sectional analysis tools.** Nimble and rich analysis tools directly contribute to the value offered by models, and permit more efficient modeler focus. Both traditional System Dynamics and (particularly) Agent Based modeling tools have significant opportunities for growth in this area. An important challenge will be to facilitate insight into the behavior of large-scale individual-based models.
- **Explicit mathematical foundation.** Whatever its basis (e.g. set theory, ordinary differential equations, hybrid automata, etc.), a clear, consistent mathematical foundation fosters model analysis, communication, generalization, helps avoid grosser lapses in model formulation and soundness. General purpose computer code is by itself inadequate for these purposes (although shortcomings can be reduced by coupling it with a parallel statement of the mathematical structure of the model being realized in code). Agent Based modeling tools will benefit strongly from the ability to express models in consistent mathematical terms.
- **Modular composition.** Creation of and reasoning about simulation models is simplified by modularity. Modularity opens the door for reuse of model components, simplifies model modification, eases multi-person development of models, and invites opportunities for third-party support. Object-based encapsulation provides an attractive approach, as do hybrid automata for individual-based models.
- **Metadata support.** Unit annotations have long accompanied System Dynamics models; such annotations have for many years enjoyed limited support from traditional System Dynamics tools. System Dynamics tools would do well to deepen and broaden their support for unit and dimensionality checks [40]. Modelers would also benefit from both informal and formal support for additional types of metadata, such as information on the origin (source name and time) of model parameters, the level of confidence that obtains about such data, the stratified nature of the original source, and other types of information.

We anticipate convergence of systems modeling traditions in terms of support for the above. The future may well belong to tools (such as AnyLogic) that can adequately span a wide variety of levels of analysis and techniques.

6 Conclusions

This paper has provided a brief overview of tradeoffs between tools drawn from two modeling traditions: Agent Based modeling and System Dynamics. We have examined those tradeoffs in the present and have made some arguments for how those tradeoffs are likely to evolve.

In the short-term, we have argued that present toolsets differ in many ways that impact modeling desiderata, such as transparency, performance, scalability, ease of calibration, etc. For example, selection of an Agent Based modeling tool presently not only implies the use of an individual-based approach that offers greater flexibility in representing dynamic processes, but also carries with it implications for specification style, analysis tools, and metadata. Our findings suggest that existing traditional System Dynamics and traditional Agent Based tools presently offer very important advantages in distinct areas, and suffer from significant shortcomings in other areas. We believe there is much for creators of these toolsets to learn from the others.

The primary challenges of modeling using existing Agent Based packages are the opportunity costs they impose on model breadth and risks of barriers to modeler understanding. We discuss each of these issues in turn.

The opportunity costs reflect several factors. Given that almost all Agent Based models are individual-based, one of the most important risks is that attention and time spent on creating deeper and more detailed models will rob attention from critical ways of broadening a model, or from time spent understanding model behavior. A well-known antipattern in object-oriented approaches – whether for Agent Based computational modeling modeling [42] or software engineering [43] – occurs when model creators include problem-irrelevant details, motivated by belief that objects ought to "represent reality" (what might be loosely called "Modeling without Purpose"). Abstraction is key to success in both modeling and software development – not just at the level of encapsulation, but also at the level of omitting factors that are likely to be irrelevant. The pertinence of a given level of detail is frequently not *a priori* clear to the modeler, and often requires explicit experimentation with disaggregation. An important responsibility in System Dynamics modeling is regularly and consciously questioning *all* model boundaries—including model breadth and depth. Because abstraction is the starting point aggregate approaches (more common in traditional System Dynamics), modelers using approaches must frequently consciously weigh the opportunity costs of making a model deeper vs. broader. Judicious consideration of opportunity costs requires more initiative and care when using bottom-up Agent Based methods.

Agent Based tools raise challenges to model understanding caused by several factors. These include the absence of an explicit and consistent mathematical framework on the basis of which one can reason formally about the causes of emergent model behavior, prolonged model simulation times, limited built-in support for analyzing longitudinal model behavior, and a code-based model specification technique that often requires user involvement with (problem-) irrelevant details of model bookkeeping and

choreography. Use of judicious and disciplined modeling process consciously based around a consistent mathematical framework together with newly available tools such as AnyLogic [9, 10] and the new RePast Simphony [2, 3] can considerably reduce these risks. Nonetheless, with existing Agent Based modeling tools, one must be vigilant to avoid letting the "hows" overwhelm insights into the "why" behind model behavior. AnyLogic's incorporation of multiple modeling frameworks within a single package is particularly notable for easing the construction of declarative models and simplifying the creation of multi-scale models.

Traditional System Dynamics tools have reached a greater stage of maturity than those for Agent Based modeling, but still offer many areas for growth. Such tools would benefit from better support for discrete rules/events, incorporation of more convenient specification of fluctuating individual-based populations, improved visualization mechanisms for individual based-models and robust mechanisms for modularly composing existing models. Better support for individual-based models would benefit experimentation with hybrid modeling.

We anticipate a convergent path forward for these tools. Reflecting the importance of certain quality attributes for modelers (e.g. transparency, ease of analysis and calibration) we argue that both toolsets are likely to evolve support for common sets of features, such as primarily declarative model specification, mathematically precise model specification, support for both continuous and discrete time events/rules, and incorporation of metadata such as dimensional and unit annotation for quantities. While we believe that convergence is likely to occur in the large majority of areas, we do believe that there is at least one important area in which existing differences will remain, because of the very mixed nature of the tradeoffs involved: Model granularity. There will remain compelling reasons to make use of everything from individual-based disaggregation to hybrid systems to highly aggregate population representations in models (and sometimes within the same multi-scale model) [42]. Some of these tradeoffs derive directly from model representation and its impact on model expressiveness, while others reflect the indirect effect of the model approach on the modeling process, such as the opportunity costs associated with time needed to construct, parameterize, calibration/validate, simulate and analyze models. As a result of the significant advantages brought by aggregate and individual-level models for different programs, we argue that modelers will benefit from toolsets supporting a wide variety of levels of aggregation, the ability to nimbly disaggregate on an experimental basis, and the integration of multiple levels of modeling abstraction within the same model.

7 Bibliography

[1] M. J. North, N. Collier, T., and J. Vos, R. , "Experiences creating three implementations of the repast agent modeling toolkit," *ACM Trans. Model. Comput. Simul.*, vol. 16, pp. 1-25, January 2006.

[2] M. J. North, T. R. Howe, N. T. Collier, and R. J. Vos, "The Repast Simphony Runtime System," in *Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms* Argonne National Laboratory, Argonne, IL USA 2005.

[3] E. Tatara, M. J. North, T. R. Howe, N. Collier, T., and J. Vos, R. , "An Introduction to Repast Simphony Modeling by Using a Simple Predator-Prey Example," in *Agent 2006 Conference on Social Agents: Results and Prospects,*, Argonne National Laboratory, Argonne, IL USA, 2006.

[4] N. Minar, *The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations*, 1996.

[5] S. Tisue and U. Wilensky, "NetLogo: Design and Implementation of a Multi-Agent Modeling Environment," in *SwarmFest 2004* Ann Arbor, MI: Swarm Development Group, 2004.

[6] S. Moss, "SDML: A Multi-Agent Language for Organizational Modelling," *Computational and mathematical organization theory*, vol. 4, p. 43, 1998.

[7] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE: a FIPA2000 compliant agent development environment," in *Proceedings of the fifth international conference on Autonomous agents* Montreal, Quebec, Canada: ACM Press, 2001.

[8] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "MASON: A multiagent simulation environment," *Simulation*, vol. 81, pp. 517-527, 2005.

[9] A. Borshchev, N. Osgood, G. McDonald, M. Paich, H. Rahmandad, M. Heffernan, S. Metcalf, and C. Johnson, "Agent Based Modeling: Why Bother? ," International Conference on System Dynamics 2005, 2003.

[10] XJ Technologies,Software Program "AnyLogic",ver. 6,St. Petersburg, Russia,2007

[11] H. V. D. Parunak, R. Savit, and R. L. Riolo, "Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide " in *Multi-Agent Systems and Agent-Based Simulation*. vol. Volume 1534/1998 Berlin / Heidelberg: Springer Berlin / Heidelberg, 1998, pp. 10-25.

[12] V. Grimm, "Pattern-Oriented Modeling of Agent-Based Complex Systems: Lessons from Ecology," *Science*, vol. 310, p. 987, 2005.

[13] H. Rahmandad and J. Sterman, "Heterogeneity and Network Structure in the Dynamics of Contagion: Comparing Agent-based and Differential Equation Models," in *22nd Annual International Conference on System Dynamics* Oxford, 2004.

[14] V. Kecman, *State-space models of lumped and distributed systems*. Berlin ; New York: Springer-Verlag, 1988.

[15] T. Tengs, N. Osgood, and L. Chen, "The Cost-Effectiveness of Intensive National School-Based Anti-Tobacco Education: Results from the Tobacco Policy Model," *Preventive Medicine*, vol. 33, pp. 558-70, 2001.

[16] T. Tengs, N. Osgood, and T. Lin, "Public health impact of changes in smoking behavior: results from the Tobacco Policy Model," *Medical Care*, vol. 39, pp. 1131-1141, October 2001.

[17] M. Edwards, "Comparing an individual-based model of behavior diffusion with its mean field aggregate approximation," *Journal of artificial societies and social simulation*, vol. 6, 2003.

[18] M. Edwards, N. Ferrand, F. Goreaud, and S. Huet, "The relevance of aggregating a water consumption model cannot be disconnected from the choice of information available on the resource," *Simulation Modelling Practice and Theory*, vol. 13, pp. 287-307, 2005.

[19] J. Watmough, "The Dynamics and Structure of Groups: Two Case Studies of the Common Honey Bee,"Department of Mathematics and The Institute of Applied Mathematics, University of British Columbia, Vancouver, BC, September 1996. 133 p

[20] N. Picard and A. Franc, "Aggregation of an individual-based space-dependant model of forest dynamics into distribution-based and space-independent models," *Ecological Modelling*, vol. 145, pp. 69-84, 2001.
[21] L. F. Berkman and S. L. Syme, "Social networks, host resistance, and mortality: a nine-year follow-up," *American journal of epidemiology*, vol. 109, pp. 186-204, 1979.
[22] T. E. Seeman and S. L. Syme, "Social networks and coronary artery disease: a comparison of the structure," *Psychosomatic medicine*, vol. 49, pp. 341-54, 1987.
[23] N. Osgood, "Representing Heterogeneity in Complex Feedback System Modeling: Computational Resource and Error Scaling," in *Presented at the 22nd International Conference of the System Dynamics Society* Oxford, UK: International System Dynamics Society, 2004.
[24] J. D. Sterman and K. McLeroy, "Systems Methodologies for Solving Real-World Problems: Applications in Public Health," in *SPRING 2007 SYMPOSIA SERIES ON SYSTEMS SCIENCE AND HEALTH*, Rockville, Maryland, 2007.
[25] M. Ben-Akiva, *Discrete Choice Analysis: Theory and Application to Travel Demand*, 1985.
[26] C. Antoniou, M. Ben-Akiva, M. Bierlaire, and R. Mishalani, "Demand simulation for Dynamic Traffic Assignment," in *8th IFAC/IFIP/IFORS symposium on transportation systems*, Chania, Greece, 1997.
[27] M. Ben-Akiva, M. Bierlaire, H. Koutsopoulos, and R. Mishalani, "DynaMIT: a simulation-based system for traffic prediction," in *DACCORD Short Term Forecasting Workshop*, Delft, The Netherlands, 1998.
[28] D. S. Shepard and R. J. Zeckhauser, "Long-term effects of interventions to improve survival in mixed populations," *Journal of Chronic Diseases*, vol. 33, pp. 413-33, 1980.
[29] K. Czarnecki and U. Eisenecker, *Generative programming : methods, tools, and applications*. Boston: Addison Wesley, 2000.
[30] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, vol. 15, p. 1053, 1972.
[31] E. W. Dijkstra, "On the role of scientific thought," in *Selected Writings onComputing: A Personal Perspective*: Springer-Verlag, 1982, pp. 60-66.
[32] T. A. Henzinger, "The Theory of Hybrid Automata," in *11th Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, 1996, p. 278.
[33] G. I. Barenblatt, *Scaling*. Cambridge: Cambridge University Press, 2003.
[34] G. W. Hart, *Multidimensional analysis : algebras and systems for science and engineering*. New York: Springer-Verlag, 1995.
[35] H. E. Huntley, *Dimensional analysis*. New York: Dover Publications, 1967.
[36] E. d. S. Q. Isaacson and M. d. S. Q. Isaacson, *Dimensional methods in engineering and physics : reference sets and the possibilities of their extension*. New York: Wiley, 1975.
[37] R. C. Pankhurst, *Dimensional analysis and scale factors*. New York: Published on behalf of the Institute of Physics and the Physical Society by Chapman and Hall; Reinhold, 1964.
[38] T. Szirtes and P. Rózsa, *Applied dimensional analysis and modeling*. New York: McGraw Hill, 1998.
[39] D. A. Plaisted, "Equational reasoning and term rewriting systems," in *Handbook of Logic in Articial Intelli-gence and Logic Programming,*. vol. 1, D. M. Gabbay and J. Siekmann, Eds. Oxford: Oxford University Press, 1993.
[40] N. Osgood, "Enriching Dimensionality Analysis in System Dynamics Modeling," in *System Dynamics Winter Conference 2007*, Austin, Texas, 2007.
[41] S. Wijesinghe, "Workshop on Agent-Based Models for Crowd Dynamics," in *International Conference on System Dynamics 2005*, Boston, 2005.
[42] V. Grimm, "Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future?," *Ecological modelling*, vol. 115, p. 129, 1999.
[43] B. Meyer, *Object-oriented software construction*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1997.