A. Borshchev and A. Filippov. From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools. The 22nd International Conference of the System Dynamics Society, July 25 - 29, 2004, Oxford, England

From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools

Andrei Borshchev & Alexei Filippov
XJ Technologies www.xjtek.com and St.Petersburg Technical University
andrei@xjtek.com

Abstract This paper may be considered as a practical reference for those who wish to add (now sufficiently matured) Agent Based modeling to their analysis toolkit and may or may not have some System Dynamics or Discrete Event modeling background. We focus on systems that contain large numbers of active objects (people, business units, animals, vehicles, or even things like projects, stocks, products, etc. that have timing, event ordering or other kind of individual behavior associated with them). We compare the three major paradigms in simulation modeling: System Dynamics, Discrete Event and Agent Based Modeling with respect to how they approach such systems. We show in detail how an Agent Based model can be built from an existing System Dynamics or a Discrete Event model and then show how easily it can be further enhanced to capture much more complicated behavior, dependencies and interactions thus providing for deeper insight in the system being modeled. Commonly understood examples are used throughout the paper; all models are specified in the visual language supported by AnyLogic™ tool. We view and present Agent Based modeling not as a substitution to older modeling paradigms but as a useful add-on that can be efficiently combined with System Dynamics and Discrete Event modeling. Several multi-paradigm model architectures are suggested.

Keywords: multi-paradigm modeling, agent based modeling, system dynamics, AnyLogic

1. Simulation Modeling: Abstraction Levels, Major Paradigms

To make sure we all agree on terms we use please take a look at Figure 1. Modeling is a way of solving problems that occur in the real world. It is applied when prototyping or experimenting with the real system is expensive or impossible. Modeling allows you to optimize systems prior to implementation. Modeling includes the process of mapping the problem from the real world to its model in the world of models, – the process of *abstraction*, – model analysis and optimization, and mapping the solution back to the real system. We can distinguish between analytical and simulation models. In *analytical*, or static, model the result functionally depends on the input (a number of parameters); it is possible to implement such model in a spreadsheet. However, analytical solution does not always exist, or may be very hard to find. Then simulation, or dynamic, modeling may be applied. A *simulation* model may be considered as a set of rules (e.g. equations, flowcharts, state machines, cellular automata) that define how the system being modeled will change in the future, given its present state. Simulation is the process of model "execution" that takes the model through (discrete or continuous) state changes over time. In general, for complex problems where time dynamics is important, simulation modeling is a better answer.

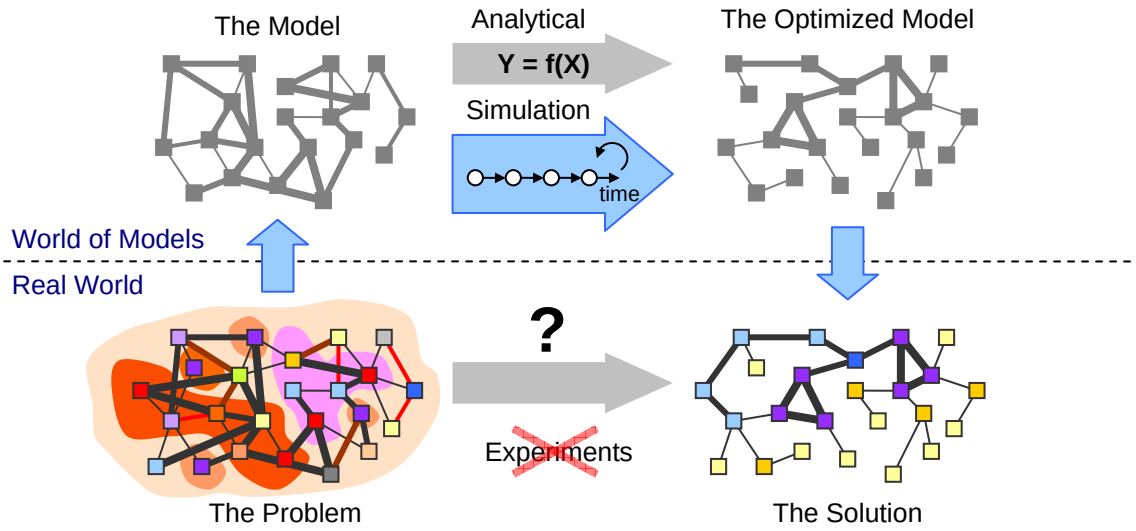

Figure 1: Analytical (Static) and Simulation (Dynamic) Modeling

Levels of Abstraction in Simulation Modeling Figure 2 shows the range of problems that are efficiently addressed with simulation modeling. The problems are arranged on the scale with respect to the typical level of abstraction of the corresponding models.

At the detailed level we have so-to-say "physical" modeling where individual objects with exact sizes, distances, velocities and timings matter. Mechatronic and control systems, micro-level traffic simulations are located at the very bottom of this chart. Factory floor models with conveyors and stations are located a bit higher because there one typically abstracts away from exact physical trajectories and uses average timings. Same applies to warehouse logistics models with storages, transporters, loading and unloading operations.

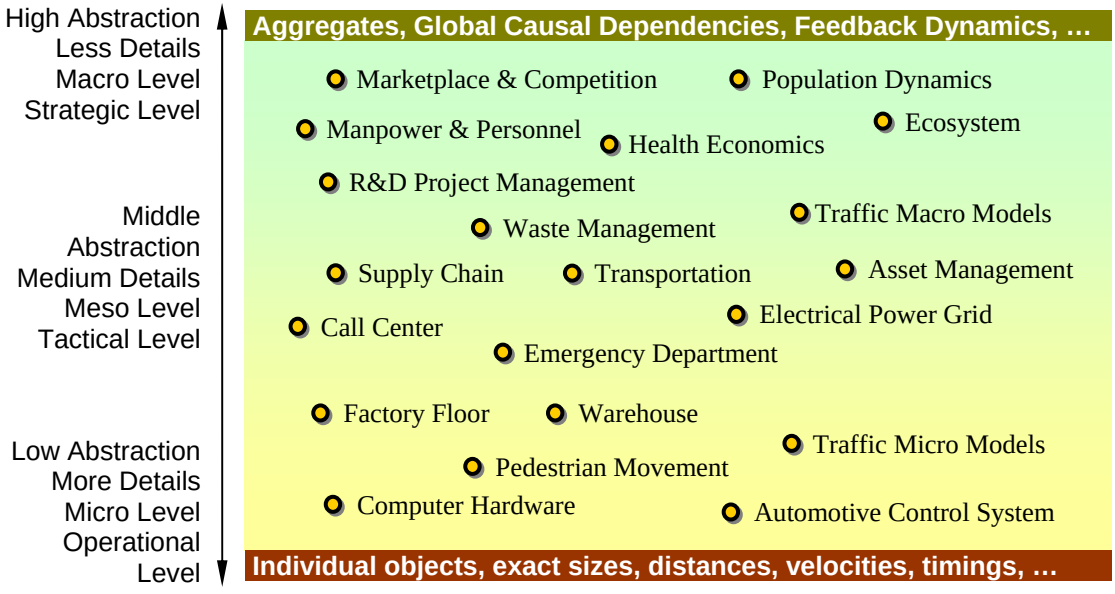

Figure 2: Applications of Simulation Modeling on Abstraction Level Scale

Business process and service systems modeling typically operate with schedules and timings, although physical movement is present sometimes, e.g. emergency department room layout would be taken into account.

Network and transportation simulation deals with schedules, latencies, capacities and loading/unloading/processing times. Macro level traffic and transportation models may not consider individual vehicles or packets, they use their volumes instead. Supply chains are being modeled at very different abstraction levels, so this item could be placed anywhere in Middle to High abstraction range.

Problems at the top of the chart are typically approached in terms of aggregate values, global feedbacks, trends, etc. Individual elements such as people, parts, products, vehicles, animals, houses are never considered there. The model of system dynamics at this level is typically based on reasoning like "if the number of jobs increases then we will have increase of in-migration".

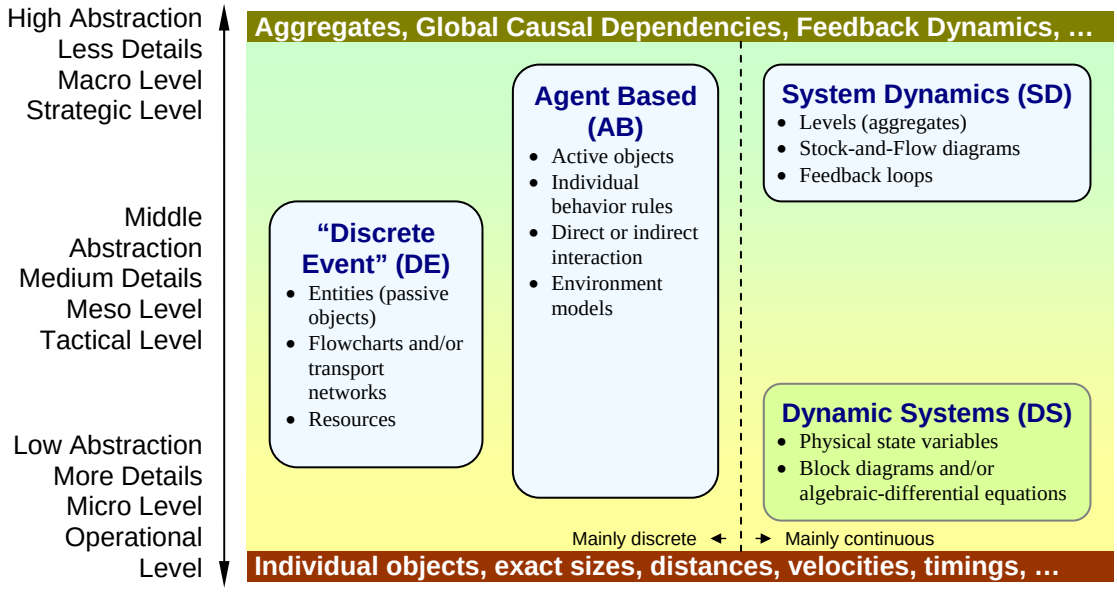

Figure 3: Approaches (Paradigms) in Simulation Modeling on Abstraction Level Scale

Major Paradigms in Simulation Modeling The major approaches (paradigms) in simulation modeling are shown in the same scale in Figure 3: System Dynamics (SD), "Discrete Event" (DE) and Agent Based (AB). SD and DE are traditional, AB is relatively new. There is also Dynamic Systems (DS) field, but it stays a bit aside as it is used to model and design "physical" systems. Technically, SD and DS deal mostly with continuous processes whereas "DE" (by which we mean all numerous descendants of GPSS) and AB work mostly in discrete time, i.e. jump from one event to another.

Consider how approaches correspond to abstraction. Dynamic Systems or "physical" modeling is at the bottom of the chart. System Dynamics dealing with aggregates is located at the highest abstraction level. Discrete event modeling is used at low to middle abstraction. As for Agent Based modeling, this technology is being used across all

abstraction levels. Agents may model objects of very diverse nature and scale: at the "physical" level agents may be e.g. pedestrians or cars or robots, at the middle level – customers, at the highest level – competing companies.

SD, DS and DE historically have been taught at universities to very distinct groups of students such as management, control engineers and industrial engineers. As a result, there currently exist three separate practitioners' communities (three different worlds) that never talk to each other. AB until recently has been almost purely academic topic. However, the increasing demand for global business optimization have caused leading modelers to look at AB and combined approaches to get deeper insight into complex interdependent processes having very different natures. Therefore, there is a request for platforms that would allow for integration and efficient cooperation between different modeling paradigms.

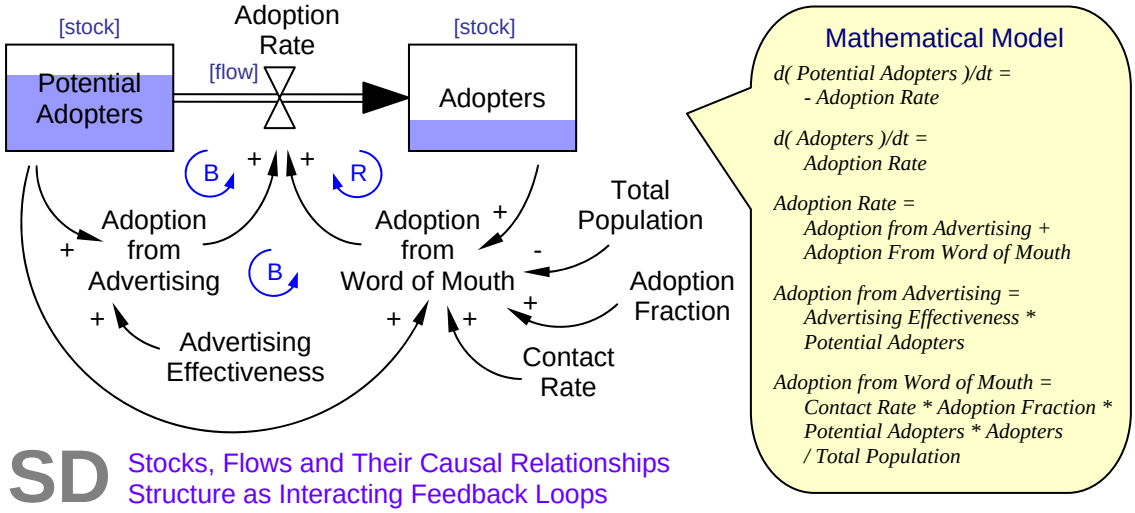

Figure 4: Classic System Dynamics Model: Bass Diffusion in Vensim™

System Dynamics Developed by an electrical engineer Jay W. Forrester in the 1950s, System Dynamics is "the study of information-feedback characteristics of industrial activity to show how organizational structure, amplification (in policies), and time delays (in decisions and actions) interact to influence the success of the enterprise" [Forrester 1958 and 1961]. The range of SD applications includes also urban, social, ecological types of systems. In SD the real-world processes are represented in terms of stocks (e.g. of material, knowledge, people, money), flows between these stocks, and information that determines the values of the flows. SD abstracts from single events and entities and takes an aggregate view concentrating on policies. To approach the problem in SD style one has to describe the system behavior as a number of interacting feedback loops, balancing or reinforcing, just like shown in Figure 4, and delay structures.

> **Bass Diffusion Model** In the classic textbook model of product diffusion (Figure 4, [Sterman, 2000]) *Potential Adopters* become *Adopters* at *Adoption Rate* that depends on advertising and word of mouth promotion. The impact of advertising is modeled as a constant percent of *Potential Adopters* (namely, *Advertising Effectiveness* = 0.011 in this paper) becoming *Adopters* each time unit. Therefore, the corresponding summand of *Adoption Rate* equals *Potential Adopters * Advertising Effectiveness*. For

> word of mouth adoption it is assumed that everybody contacts everybody else in this population group. The number of contacts per person per time unit is *Contact Rate* (100). In case one of the two people in contact is adopter and another one – not yet, the latter one will adopt with the probability *Adoption Fraction* (0.015). Then, during a time unit, each adopter will convert *Adopters * Contact Rate * Adoption Fraction * [Potential Adopters / (Potential Adopters + Adopters)]* people into *Adopters*. The expression in the square brackets is the probability of another person being not already adopter.

Mathematically, an SD model is a system of differential equations. Important things to know about SD modeling: a) as long as the model works only with aggregates, the items in that same stock are indistinguishable, they do not have individuality, and b) the modeler has to think in terms of global structural dependencies and has to provide accurate quantitative data for them. SD modeling is supported by 3-4 tools that are very much alike.

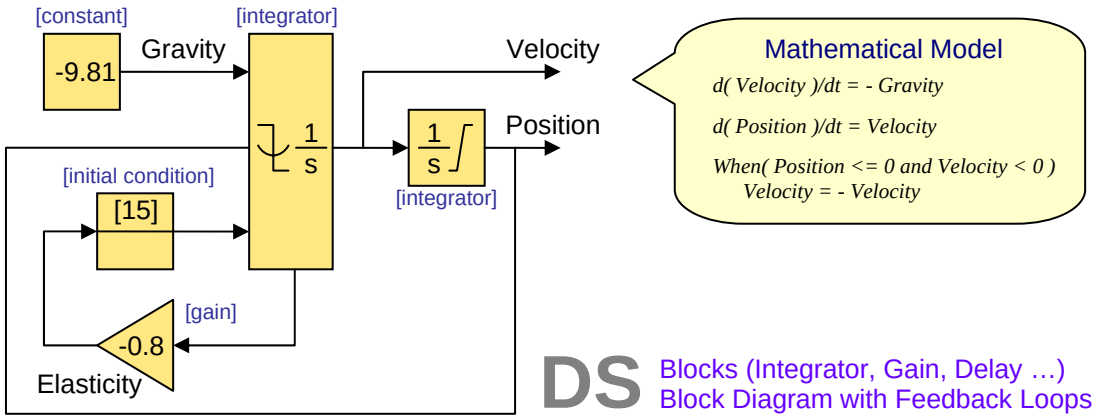

Figure 5: Dynamic System Model: Bouncing Ball in MATLAB™ Simulink™

Dynamic Systems modeling is actually the ancestor of System Dynamics. It is used in mechanical, electrical, chemical, and other technical engineering disciplines as a standard part of the design process. Block diagram like the MATLAB™ Simulink™ one shown in Figure 5 is a typical graphical modeling language that a control engineer would use, there are other graphical as well as textual languages for specific domains. The underlying mathematical model of a dynamic system would consist of a number of state variables and algebraic differential equations of various forms over these variables. In contrast with the SD, integrated variables here have direct "physical" meaning: location, velocity, acceleration, pressure, concentration, etc., they are inherently continuous, and are not aggregates of any entities. The mathematical diversity and complexity in dynamic systems domain can be much higher than in system dynamics, and the tools used for dynamic system simulation could easily solve any SD problem with even much better accuracy than SD tools. However, DS tools have been developed to be embedded in the engineering design cycle and therefore system dynamics modelers would feel uncomfortable if they used them because *DS tools do not support the way SD modelers think*.

In our further investigation below we will not be considering dynamics systems simulation because it has quite distinct application area where other methods can rarely be applied (although there are attempts to model, e.g. molecules of gas as agents)

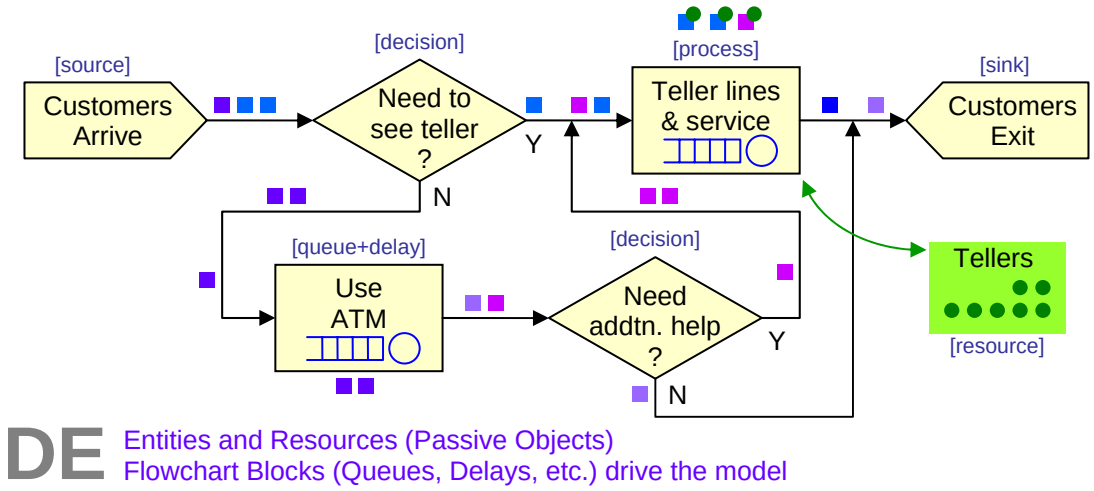

Figure 6: Discrete Event Model: Bank Kiosk in Arena™

Discrete Event We will reserve the term "discrete event modeling" for the modeling approach based on the concept of entities, resources and block charts describing entity flow and resource sharing, see Figure 6. This approach roots to 1960s when Geoffrey Gordon conceived and evolved the idea for GPSS and brought about its IBM implementations [Gordon 1961]. Entities (transactions in GPSS) are passive objects that represent people, parts, documents, tasks, messages, etc. They travel through the blocks of the flowchart where they stay in queues, are delayed, processed, seize and release resources, split, combined, etc. The classic (Arena™-like) view of the discrete event flowchart is shown in Figure 6. There are many (about a hundred) commercial tools that support this modeling style, some are general purpose, some target particular niches: service, manufacturing, logistics, business processes, call centers, etc. Their user interfaces may look very different because of tool specialization, but underneath they all have similar discrete event simulation engines that push entities through blocks. For the purpose of this investigation, we would like to underline that DE modeling may be considered as definition of a *global entity processing algorithm*, typically with stochastic elements.

Agent Based Many different developments have been going on under the slogan of Agent Based modeling in very different disciplines like artificial intelligence, complexity science, game theory, etc. There are no universally accepted definitions in this area, and people still discuss what kind of properties an object should have to "deserve" to be called an "agent": pro- and re-activeness, spatial awareness, ability to learn, social ability, "intellect", etc. [Schieritz and Milling 2003]. We are not going to suggest more definitions or argue here because agents we use in our modeling practice are very diverse, but we would like to stress just one feature of agent based models: they are essentially *decentralized*. Compared to SD or DE models, there is no such place in AB model where the global system behavior (dynamics) would be defined. Instead, the modeler defines behavior at individual level, and the global behavior emerges as a result of many (tens, hundreds, thousands, millions) individuals, each following its own behavior rules, living together in some environment and communicating with each other and with the environment. That is why AB modeling is also called *bottom-up modeling*. You can view the Figure 7 as agent based model of country

population dynamics. In this model a part of the agent behavior is defined as a *statechart* (this construct is explained below), and housing, jobs, transport infrastructure, etc. are represented in the environment model. We will pay more attention to AB modeling in this presentation because we intend to show where and how this new modeling approach can be practically applied and how does it compare to SD and DE.

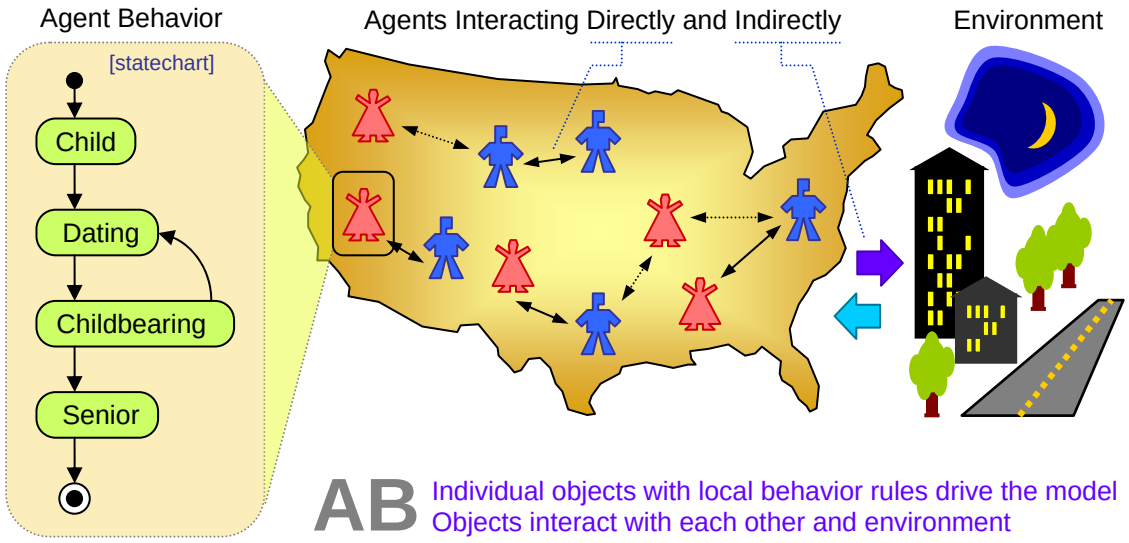

Figure 7: Agent Based Model Generic Architecture. Behavior (Statechart) in AnyLogic™

Correspondences between the approaches We will now restrict the scope of our investigation with the systems that contain large numbers of active objects (people, business units, animals, vehicles, or even things like projects, stocks, products, etc. that have timing, event ordering or other kind of individual behavior associated with them). In the following two sections we are going to illustrate that for that type of systems *Agent Based approach is more general and powerful because it enables to capture more complex structures and dynamics*. The other important advantage is that it *provides for construction of models in the absence of the knowledge about the global interdependencies*: you may know nothing or very little about how things affect each other at the aggregate level, or what is the global sequence of operations, etc., but if you have some perception of how the individual participants of the process behave, you can construct the AB model and then obtain the global behavior. Thus, even if there exists, say, a SD model that answers the question, it might be much easier to build the AB model. Agent based models are also typically *easier to maintain: model refinements normally result in very local, not global changes*.

2. Correspondence between SD and AB Models

In this section we will investigate the relationships between System Dynamics and Agent Based models. We will first show how to "re-conceptualize" an existing SD model into an AB model, and then – how to enhance the AB model to capture more sophisticated dynamics. The following quick reference will help to understand the technique.

Statecharts In our modeling practice we extensively use statecharts to specify behavior of agents. Statechart is actually a state machine with several useful enhancements suggested by David Harel, adopted by worldwide community and included as a part of the standard UML (The Unified Modeling Language, [UML]). Statecharts enable you to graphically capture different states of the agents, transitions between them, events that trigger those transitions, timing, and actions that the agent makes during its lifetime. Such construct as composite states enables to specify modes of agent operation. Agent may have several statecharts working in parallel and interacting: this is useful when one models several aspects of the agent's life, e.g. education and family. As a general remark we would like to say that while the traditional (SD and DE) simulation modeling communities have not really generated any new ideas within the last few decades, the software engineering world has made a huge progress in approaching the complexity of systems. A good part of that experience has been accumulated as the UML. And although the UML "as is" cannot be used as a simulation modeling language (due, for example, to the lack of executable semantics, etc.), such things as separation of structure and behavior, object-orientedness (the concept of classes and instances, encapsulation), etc. can save a lot of simulation modeler's time and efforts if applied within the model design process.

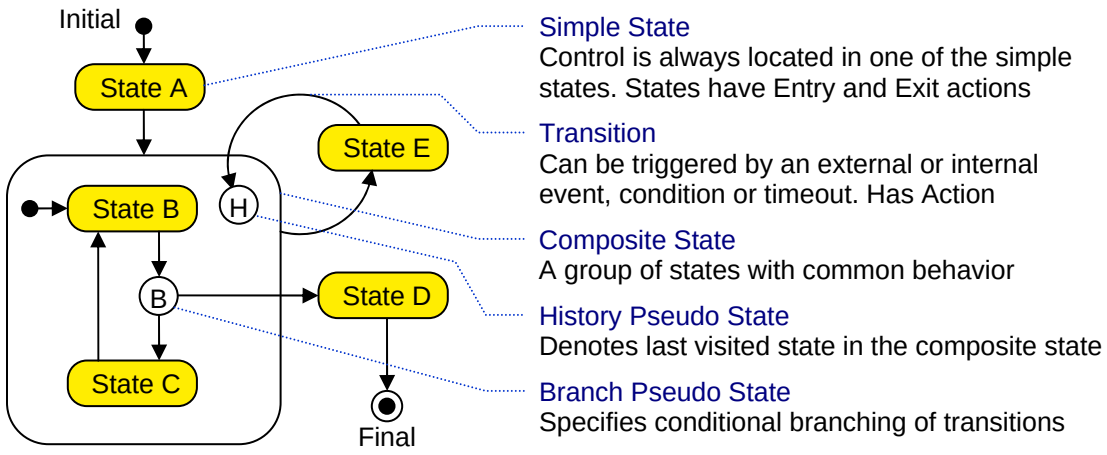

Figure 8: UML Statecharts: A Language for Specification of Event- and Time-Driven Behavior

Let us consider the SD model as the chain of stocks and flows between them and the "decision rules" part that controls the flows, see Figure 9. The key starting point is to "disaggregate" the stocks, i.e. to *look at the stocks as if they are not "tanks with liquid" but "boxes" containing discrete items, e.g. balls*. These balls will become agents. Put yourself inside a single ball and watch what is happening. You will discover that you only have two states *State A* and *State B* corresponding to the stock/box you are currently in, and if you are in *State A* you will eventually transition to *State B*. The moment of time at which this happens (intuitively) depends on the flow rate. The AB model that shows behavior corresponding to the SD model will consist of a number (*Stock A + Stock B*) of agents each executing such statechart. The transition between the states can be implemented in several ways, two of them shown in Figure 9: *synchronous*, when the decision is made every time step *dt*, and *asynchronous* when the time delay associated with the transition is calculated once upon entering *State A* and may be re-calculated later on depending on the *Rate*. Note that in the latter case the notion of time step is not present in the agent based model at all. *Asynchronous agents may be much more efficient computationally than synchronous ones*. In the examples below we will be using asynchronous agents whenever possible. For more formal description of the technique please see Addendum A.

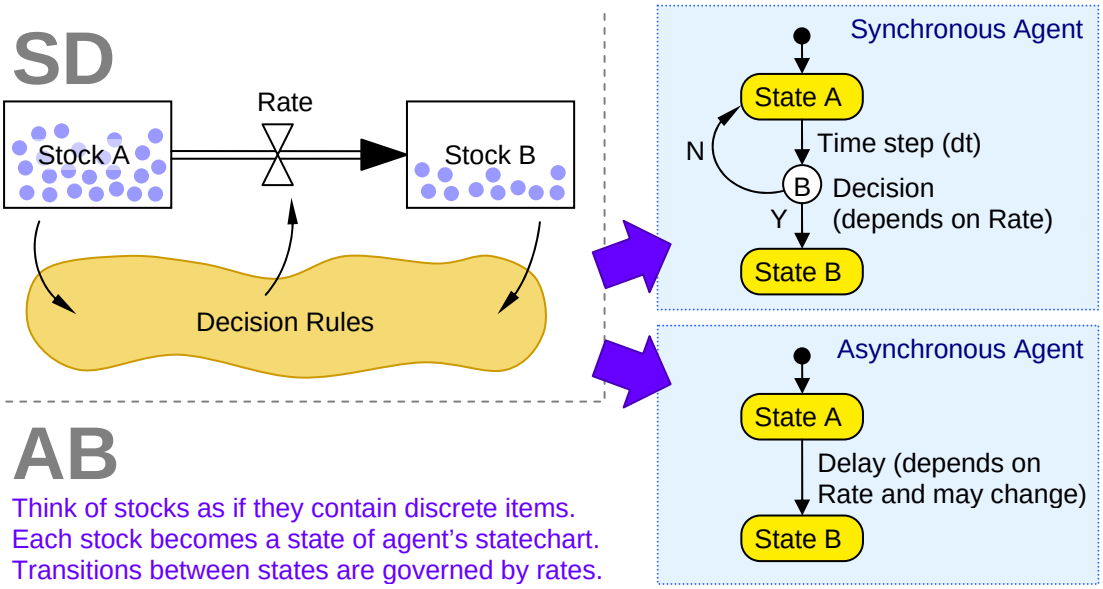

Figure 9: Re-Conceptualizing a System Dynamics Model into Agent Based Model. General Scheme

Bass Diffusion – Agent Based Version We will now illustrate this technique on a classic textbook SD model – Bass Diffusion described earlier, see Figure 10. This is simple:

Step 1. For the two stocks we create two states of an agent: *Potential Adopter* and *Adopter*.

Step 2. The two summands of the *Adoption Rate* will be modeled separately. For the *Adoption from Advertising* part we will create a transition from the *Potential Adopters* state to *Adopters* state that is triggered by an exponentially distributed time delay with the mean value of *Advertising Effectiveness*. This models the constant percent of people becoming adopters each time unit because of advertising, see Addendum for more information.

Step 3. For the *Adoption from Advertising* part of the flow we will create a periodic transition for each person being an adopter. That transition will model his contacts with other people with the *Contact Rate*. During each contact the adopter will pass a message *"Good stuff – buy it!"* to another guy. In case the other one has not yet adopted, i.e. is in *Potential Adopter* state, he will transition to *Adopter* state with the corresponding probability, hence the second transition from *Potential Adopter* to *Adopter*. Already adopted people will just ignore this message. Please note that we have implemented this word of mouth less straightforwardly but much more efficiently by modeling only a successful fraction of contacts (the rate of cyclic transition is multiplied by *Adoption Fraction*) and making the effect of receiving the message by a Potential Adopter deterministic.

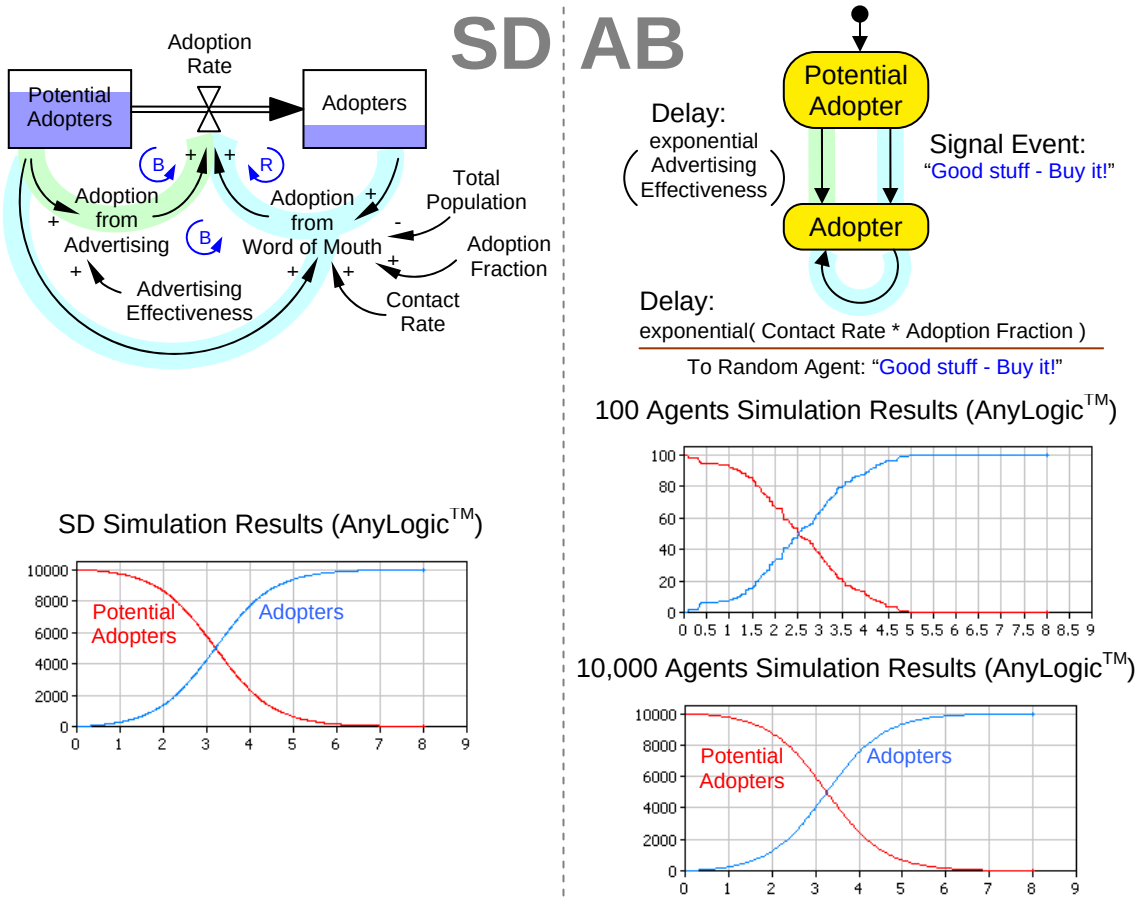

Figure 10: Bass Diffusion Model Converted From System Dynamics to Agent Based

The SD model generates the well-known S-shape, and so does the AB model. While the number of agents is low, the discrete (*thus closer to reality and more accurate!*) nature of the model shows well on the chart. If we increase the number of agents, the chart will be virtually indistinguishable from the SD one.

> One of the frequently asked questions is *"How many agents can I efficiently simulate?"* This certainly depends on software and hardware one uses, but much more interesting question is *"How many agents do I **need** to simulate?"* For example, if one needs to model the population of a country (like 300,000,000 people), does he need 300 million agents? The answer is: in most cases not necessarily! There are several techniques of reducing the number of agents to make the simulation computationally efficient yet giving the correct results. This exciting topic is however outside the scope of this paper.

The re-conceptualization procedure we have just described only makes sense if you wish to further develop the agent based model, to add more details, e.g. individual memory (history), agent communication, etc. *If the items that are stocked in the SD model are naturally passive and indistinguishable, you will probably not benefit at all from converting them to agents.* A notable example would be stocks of money: we normally are not interested in individual dollar history and (at least in the sense of this paper) dollars do not show any active behavior :)

Let us now refine our model a bit. Let the word of mouth influence of a particular person (i.e. the fraction of people who will adopt as a result of a contact with that person) depend on how long ago he has purchased the product. We will assume that soon after the purchase the adopter "promotes" the product quite actively, and then this influence goes down and stabilizes at some moderate level, like shown in the lookup function chart in Figure 11.

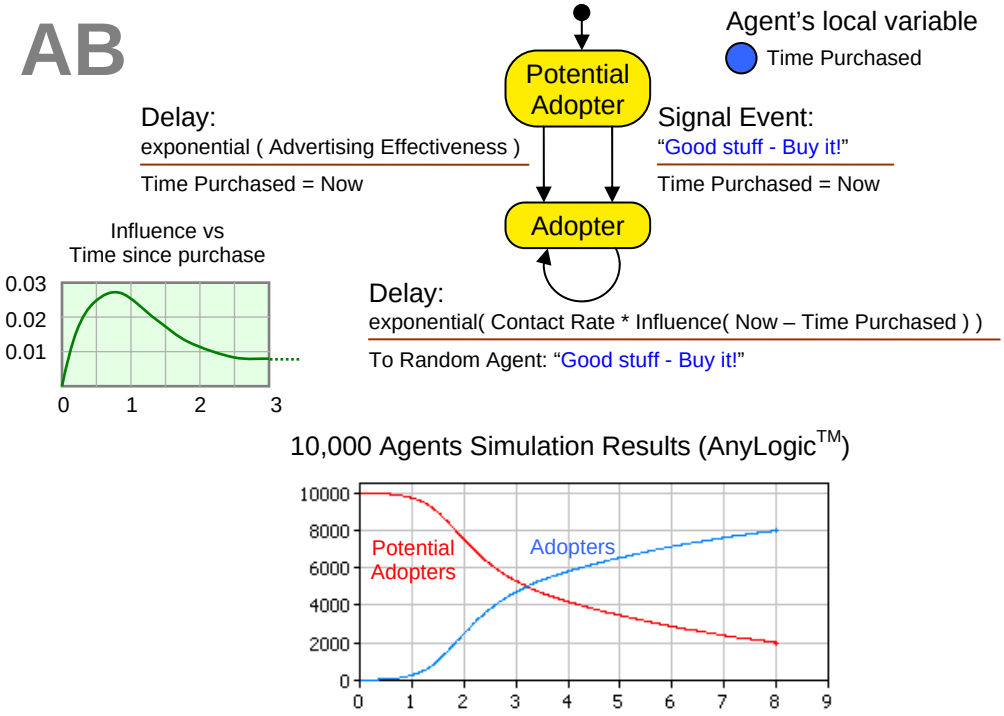

Figure 11: Agent Based Bass Diffusion Model: Capturing Dynamically Changing WOM Influence

Our agent needs only small modifications to capture this: first, we introduce the variable *Time Purchased*, then we remember the time when a person becomes an adopter (see actions of the two transitions from *Potential Adopter* state) and finally use a lookup function *Influence* instead of the constant *Adoption Fraction*. The resulting behavior is shown in Figure 11. The curve shapes certainly differ from the previous ones because of different value of adoption through word of mouth.

Now, here's the question: can you capture such behavior in SD? The word of mouth contribution into the *Adoption Rate* may now be different for any two adopters, and it also changes over time for an adopter. Therefore aggregating the adopters into one (or any reasonable finite number) of stocks will distort the results.

> Of course, system dynamics have faced that kind of problems long ago, and have even suggested a certain partial solution, to be more precise – a workaround: a stock containing objects with sufficiently different properties is decomposed into an array of buckets, and objects move between the buckets as their properties change. Consider however objects having not one, but several such properties. The array of buckets grows as a product of dimensions and, after a few steps *the number of cells in the array may easily exceed the number of individual objects in the real world we are modeling* [Keenan and Paich 2004]. This obviously makes such system dynamics model senseless (and terribly slow), while the agent based model will always contain as many objects as needed.

Predator Prey – Agent Based Version We will use different (and maybe more practical) approach to create an agent based version of another classic SD model – Predator Prey [Lotka 1925 and Volterra 1926], see Figure 12.

> **Predator Prey Model** is composed of a pair of differential equations that describe predator-prey (or herbivore-plant, or parasitoid-host) dynamics in their simplest case (one predator population, one prey population). It was developed independently by Alfred Lotka and Vito Volterra in the 1920's, and is characterized by oscillations in the population size of both predator and prey, with the peak of the predator's oscillation lagging slightly behind the peak of the prey's oscillation. The model makes several simplifying assumptions: a) the prey population has unlimited resources and prey only die when eaten up by the predator; b) prey is the only source of food for the predator and predators only die because of starvation; c) predators can consume infinite quantities of prey; and d) there is no environmental complexity (in other words, both populations are moving randomly through a homogeneous environment).

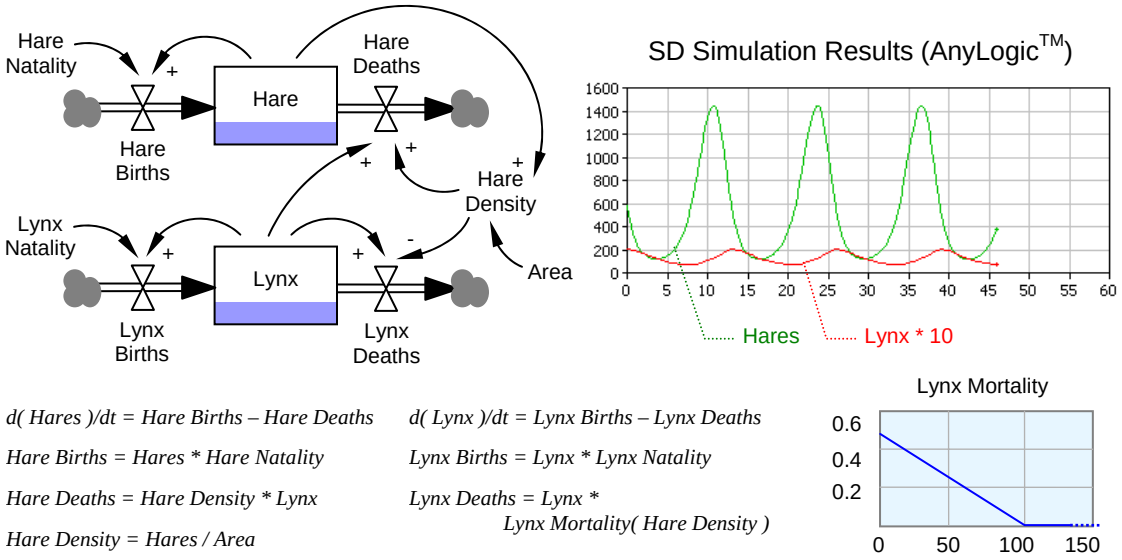

Figure 12: The Classic System Dynamics model of Predator Prey

Instead of reproducing this dynamics literally in AB model (as we did with Bass Diffusion), we will build an agent based model with richer content "directly" from a bit more realistic set of assumptions. In our model a) hares and lynx both have life expectancy, so they die because of age as well as because of being eaten up or starving; b) hares and lynx live in 2D space (are "*space-aware*"); c) hares density is limited, and they only breed if there is enough space around; d) lynx can only reach hares within a certain geographical range; e) lynx hunt at a certain rate; f) if lynx cannot find a hare during a hunt it moves; and e) if lynx does not eat a hare within a certain period of time, it dies.

In Figure 13 both Hare and Lynx agents have variable *Location* that stores their current location in 2D space; initial location is random. It is updated when the agent moves and affects its behavior. During their lifetime lynx and hares have babies with a certain frequency – this is modeled by the cyclic timers *Births* that create new agents; in case of hare births depend upon local density of hares. The hare statechart is quite simple: two states *Alive* and *Dead* and two transitions between them corresponding to the two reasons of dying: age and being killed by the lynx (message sent directly from a lynx to the hare).

Lynx statechart is a bit more complicated. Lynx hunts each *Lynx Hunting Period* and if it finds no hares (that probabilistically depends on the hare density in the lynx neighborhood), it moves (changes *Location*) but stays in the *Hungry* state. In case it kills a hare (sends the "*I ate you!*" message), it exits and immediately re-enters the *Hungry* state, which (according to the statecharts semantics) causes "restart" of the delay transition *Lynx Hunger Death Threshold*. Therefore lynx needs no less than one hare per *Lynx Hunger Death Threshold* to survive.

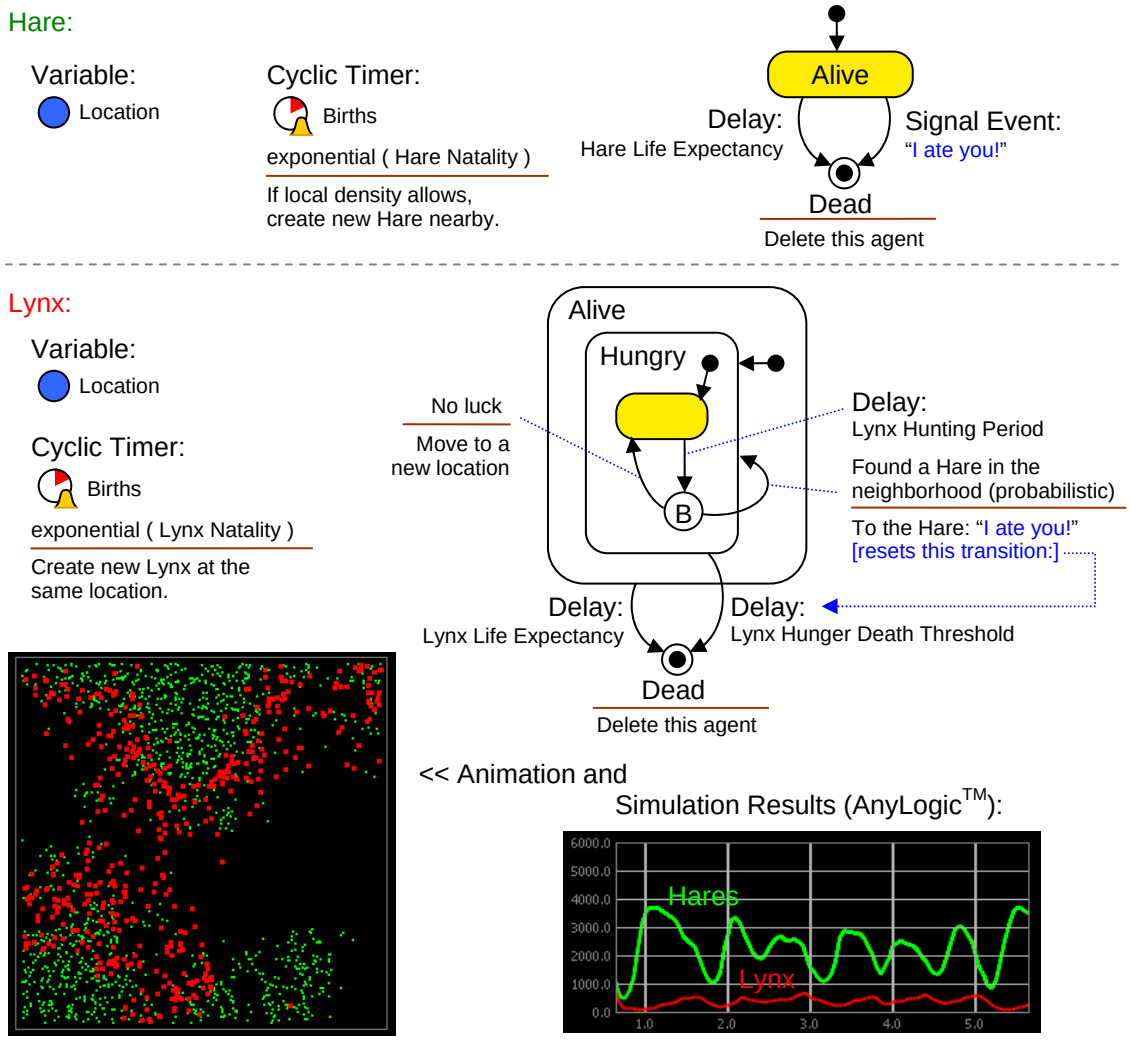

Figure 13: Agent Based Predator Prey Model: Making More Realistic Assumptions

The simulation shows the oscillating behavior similar to the SD model with lynx population peaks lagging behind the hare population peaks. Subject to the model parameters, lynx (or hares and lynx together) may become totally extinct, which never happens in the SD model due to its continuous nature. The oscillations are stochastic because of the random and spatial nature of the model. In this agent based model you are able to view the 2D picture and trace a single hare, lynx, their families, or generations.

3. Correspondence between DE and AB Models

We will now make a (less detailed) investigation into the relationships of Discrete Event and Agent Based modeling. In DE model we already have individual entities, which makes life a bit easier: those entities can naturally become agents. The DE entities are however described as passive objects and the rules that drive the system are concentrated in the flowchart blocks. The exercise is to describe the process from the entity's viewpoint, thus decentralize (some of) the rules. Again, this only makes sense in case you wish to model some extra individual behavior later on that is hard to capture in DE style.

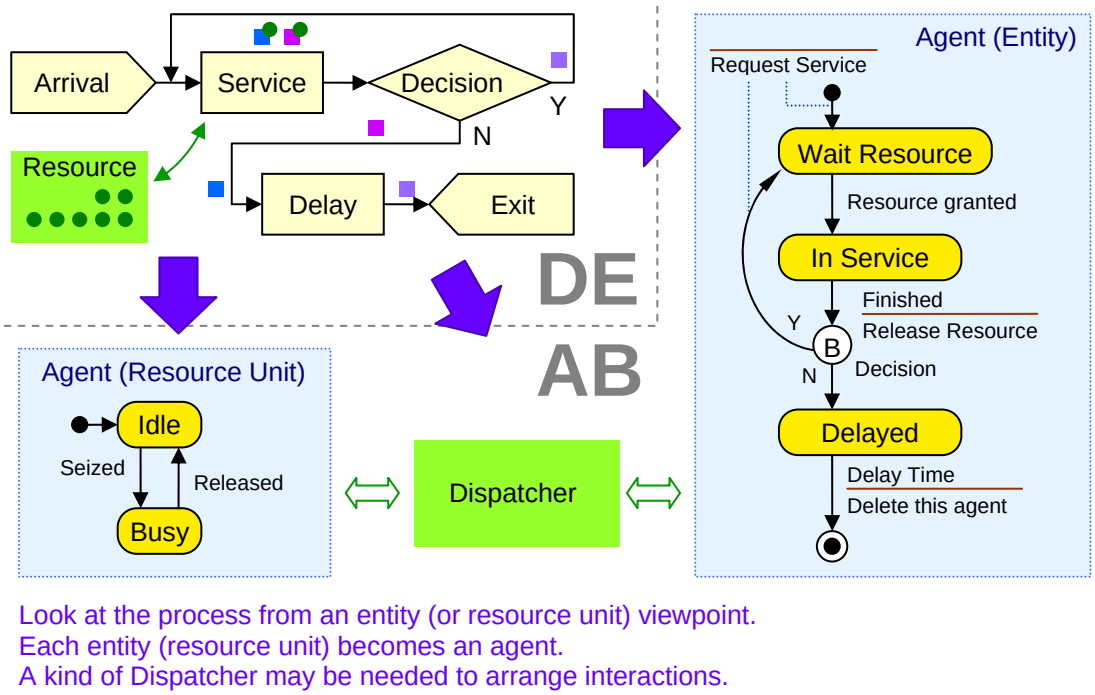

Figure 14: Re-conceptualizing a Discrete Event Model into an Agent Based Model. General Scheme

Let us consider a simple service system Figure 14 where entities (people, transactions, etc.) enter the system, get serviced (resource unit required) one or more times depending on their properties, then make a delay and then exit the system. Entities will become agents. The event of entity generation corresponds to the creation of an agent. Upon creation, the agent will request service (but not necessarily get it immediately) and go to the state *Wait Resource* (corresponds to the entity waiting in the queue at the block *Service*). When the resource is granted, the agent proceeds to state *In Service* (corresponds to the service delay) and, when finished, decides whether to request service again or proceed to the *Delayed* state. When completed, the agent destroys itself, which corresponds to the entity exiting the flowchart.

Resource units could also be modeled as agents if needed, in which case each resource unit will have e.g. two states: *Idle* and *In Use*. To co-ordinate agent's access to the resources a central Dispatcher may be needed. Such dispatcher may be considered as a part of the environment model that implements indirect communication between agents, e.g. queuing.

Alternatively, agents may “see each other” and communicate directly to manage resource access.

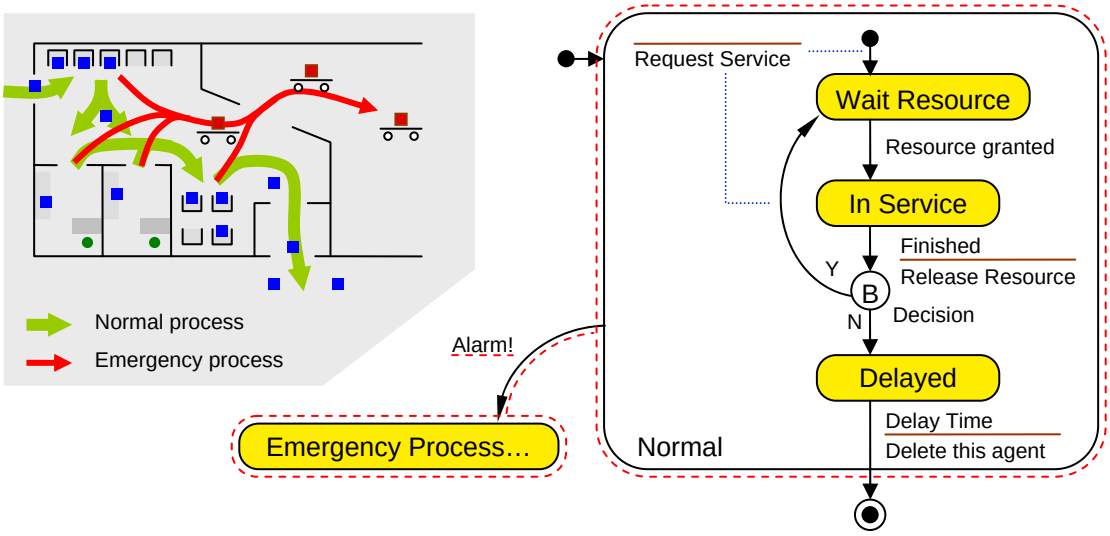

Figure 15: Agent Based Service Model: Capturing Events Associated with Individuals

Similarly to what we did with the Bass Diffusion model, we will now slightly change the definition of the problem in the service model. Suppose an event may occur for an agent and cause him to leave the system at any time regardless the stage of the process, see Figure 15. This may be e.g. a phone call, a heart attack, an alarm signal, etc. In the Agent Based model this is captured by simply enclosing all states in a composite state *Normal* denoting normal operation and adding a transition from that state to the *Emergency Process* state triggered by the event (of course, one has to define also cancellation of the resource requests issued by that agent where appropriate). In the flowchart languages such specification change would be either impossible to capture or require the modeler to insert code in many places across several blocks.

4. Tools Available. AnyLogic – Multi-Paradigm Simulation Tool

Almost all the existing tools for simulation modelers are designed to support some particular modeling paradigm, please see Figure 16. There are just about four tools for SD. There are tens of different tools for DE modeling. The latter is explained by the fact that Discrete Event modeling as a discipline is not as rigorously defined as SD and there are a lot of DE “dialects” tailored for particular applications. DS world is very much dominated by MATLAB Simulink. As for agents, until recently, they were toys played around within the academic community by means of coding in Java or C++, so there were no commercial tools, just university written libraries.

AnyLogic [AnyLogic] – the tool that we and our partners use – historically was developed not by simulation modelers but by people with background in distributed systems, concurrency theory and computer science. Therefore none of the classical simulation modeling paradigms was used as a foundation. Instead, approaches and languages designed

to handle complexity and adopted in the software engineering world were implemented. It turned out that stock-and-flow diagrams and flowcharts are naturally expressed in the object-oriented core language of AnyLogic, and there is a lot of value added even for those who stay within the classical modeling styles: compact structured representation, flexible data definition, etc. But the most exciting thing is the ability to rapidly compose industrial-strength agent based models within the same visual environment. AnyLogic supports ready-to-use constructs for defining agent behavior, communication, environment model, and has rich visualization capabilities. Moreover, AnyLogic enables one to specify different parts of the model using different paradigms thus providing for more adequate modeling of large and complex systems.

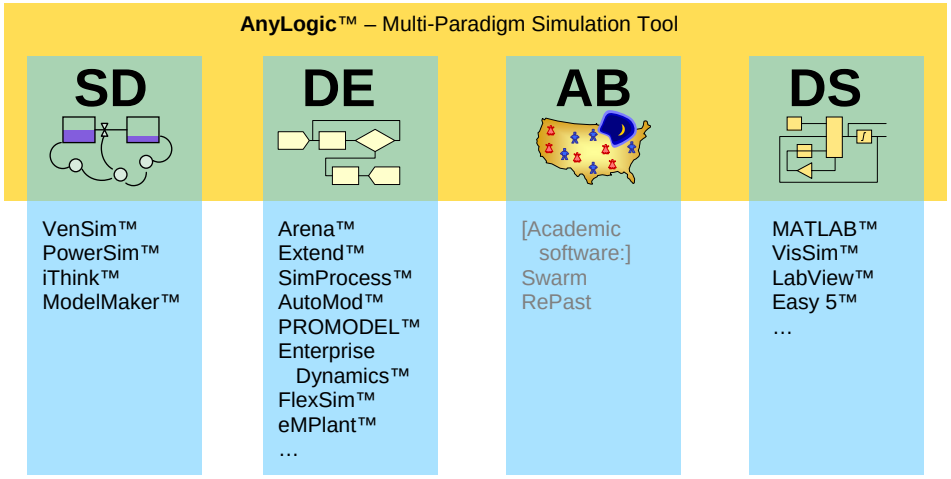

Figure 16: Tools for Simulation Modeling

5. Examples of Practical Applications

Alcohol Use Dynamics In this model developed in cooperation with Research Triangle Institute International [see also Bobashev et al 2004] we investigate people's attitude towards alcohol, life expectancy and the corresponding healthcare expenses. We distinguish between the four possible states of a person with respect to alcohol use: *Never User*, *Recreational User*, *Addict* and *Quitter*, see Figure 17. The transitions between the states are probabilistic delays. For example, Initiation delay is a sample of Initiation Time Distribution that, in turn, is formed on the basis of statistical data that was available, namely the probability of starting drinking alcohol at a certain age. Same applies for Death Time Distribution, but that may change dynamically during the person's lifetime depending on whether and how the person drinks. Please notice that in this model agents do not interact with each other.

Two population groups are being compared: one with "natural" dynamics of alcohol usage and another one being intervened. Intervention (that may be e.g. a change of law, a promotion campaign, etc. preventing people from starting using alcohol or convincing users to quit) is modeled as change in initiation and quitting probabilities.

Sample simulation results (the number of non-users, users, addicts and quitters vs. age) are shown in the two timed stack charts on the right. The intervened group in this case has twice lower initiation probabilities and twice higher quitting probabilities. People in the intervened group live longer and spend less healthcare resources, the exact numbers shown in the figure. Models like that can be used as decision support tools in the healthcare policy development.

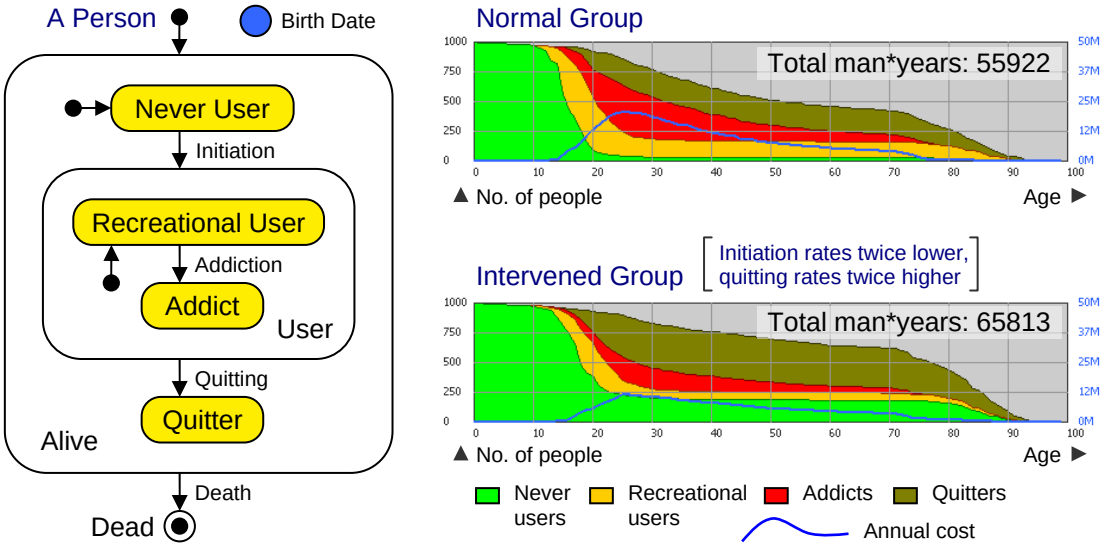

Figure 17: Agent Based Model of Alcohol Use Dynamics

Dynamics of Hispanic Population Acculturation and Behavior This model [Wallis, Paich and Borshchev 2004] was developed by Decisio Consulting [Decisio] for Synthesis Alliance. In recent US Census data widely reported in the press “Hispanics” have become the largest minority group in the US. Using simulation modeling technology the modelers look at some of the structural forces that shape the characteristics of the Hispanic population.

The model creates a simulated Hispanic population whose level of acculturation to the broader population of which it is a part dynamically varies according to individual choice. The modeling technique used draws on both System Dynamic and Agent based paradigms. The representative Hispanic population is disaggregated down to the individual level as individual agents. Each agent makes choices stochastically as modulated by its current state and the outside environment that it is in. The essence of being “Hispanic” is rooted in cultural attributes that are transmitted through the population by agent mobility and other mechanisms.

While some aspects of an agent’s state are represented discretely the modelers also draw on the well developed System Dynamics concepts of modeling “soft” variables to represent the accumulation and decay of cultural attributes within an agent. Also in the System Dynamics tradition the global level feedback structures that shape agent level behavior are identified. Finally, innovative data visualization is employed to expose the system’s dynamic behavior to the audience in a compelling fashion.

The model helps to find that dynamically complex behavior endogenously emerges in the population with temporally stable population sub-segments developing. The underlying dynamics that create these segments and the segments themselves are of interest to those who want a deeper understanding of the Hispanic population. In addition, most of the structures captured in this model are broadly applicable to studies of population and cultural dynamics and are not limited to the Hispanic population.

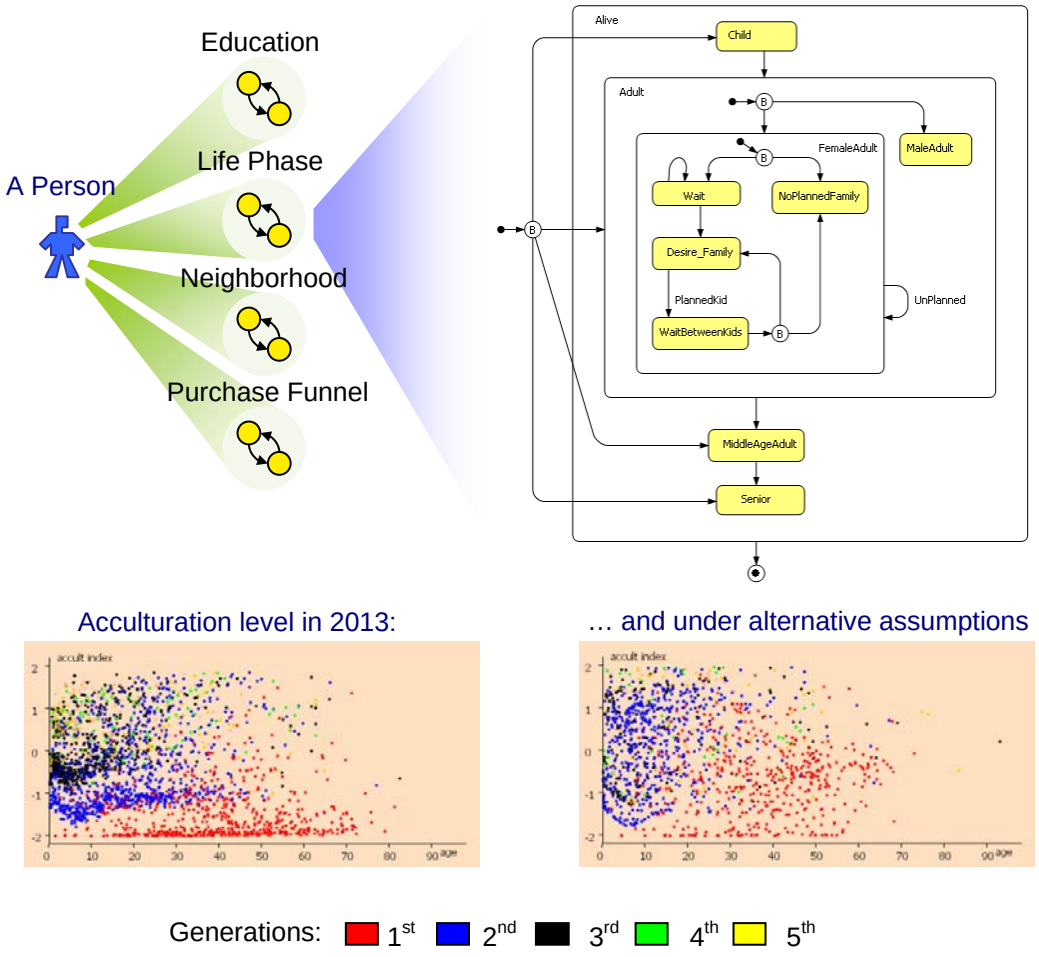

Figure 18: Agent Based Model of Dynamics of Hispanic Population Acculturation and Behavior

6. Conclusion. Which Approach to Use?

We have seen that in general using AB approach you are able to capture more real life phenomena than with SD or DE approach. This does not mean however AB is a replacement for SD or DE modeling. There are a lot of applications where SD or DE model can efficiently solve the problem, moreover, agent based modeling for many of such applications will not make much sense being less efficient, harder to develop, or simply not matching the nature of the problem. Whenever this is the case, traditional approaches

should be used without hesitation, and there are many off-the-shelf commercial tools supporting these approaches, AnyLogic being one of them.

Agent based modeling is for those who wish to go beyond the limits of SD and DE approaches [see also Keenan and Paich 2004], especially in the case the system being modeled contains active objects (people, business units, animals, vehicles, or projects, stocks, products, etc.) with timing, event ordering or other kind of individual behavior. For that kind of systems AnyLogic will not only enable you to develop agents efficiently with minimum coding required, but also support your partial or full migration from legacy stock and flow diagrams or process flowcharts to agents.

You should also consider using different modeling paradigms for different parts of the simulation model. Some examples of multi-paradigm model architectures are given in Figure 19. The top one may be found in many papers on supply network simulation [Schieritz and Grosler 2003]: the processes inside the company are modeled in system dynamics terms, whereas the communication between the companies is essentially discrete. In the middle the model of e.g. a hospital process is extended to capture the bigger lifecycles of patients and staff. At the bottom agents (people or households) live in the environment (jobs, housing, infrastructure, etc.) whose dynamics is described in SD terms.

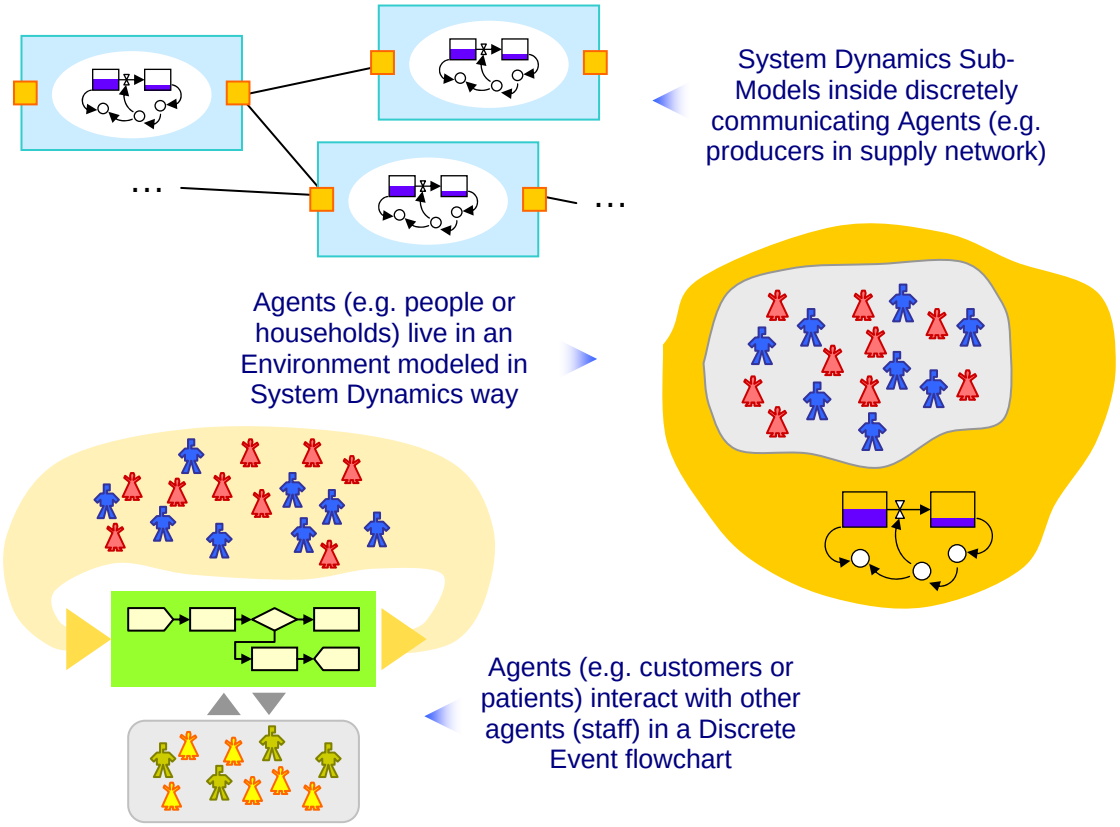

Figure 19: Multi-Paradigm Model architectures

7. References

Forrester, Jay. 1958. *Industrial Dynamics: A Major Breakthrough for Decision Makers*. Harvard Business Review, Vol. 36, No. 4, 37-66.

Forrester, Jay. 1961. *Industrial Dynamics*. Cambridge, MA: MIT Press.

Sterman, John. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. McGraw Hill.

Lotka, Alfred J. 1925. *Elements of physical biology*. Baltimore: Williams & Wilkins Co.

Volterra, Vito. 1926. *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*. Mem. R. Accad. Naz. dei Lincei. Ser. VI, vol. 2.

Gordon, Geoffrey. 1961. *A General Purpose Systems Simulation Program*. McMillan NY, Proceedings of EJCC, Washington D.C., 87-104.

Schelling, Thomas. 1978. *Micromotives and Macrobehavior*. W. W. Norton and Co.

Bobashev, Georgiy, Zule, William, Root, Elizabeth, Wechsberg, Wendee, Borshchev Andrei, and Filippov, Alexei. 2004. *Geographically-Enhanced Mathematical Models of HIV Dynamics*. NIDA Symposium on AIDS, Cancer and Related Problems, St. Petersburg, Russia.

Bobashev, Georgiy, Zule, William, Root, Elizabeth, Wechsberg, Wendee, Borshchev Andrei, and Filippov, Alexei. 2004. *Scalable Mathematical Models for Substance Use: From Social Networks to the Whole Populations*. The College on Problems of Drug Dependence 66th Annual Meeting, San Juan, Puerto Rico.

Wallis, Lyle, Paich, Mark, and Borshchev, Andrei. 2004. *Agent Modeling of Hispanic Population Acculturation and Behavior*. The 3rd International Conference on Systems Thinking in Management (ICSTM 2004), Philadelphia, Pennsylvania, USA.

Schieritz, Nadine, and Milling, Peter. 2003. *Modeling the Forest or Modeling the Trees - A Comparison of System Dynamics and Agent-Based Simulation*. The 21st International Conference of the System Dynamics Society, New York, USA.

Schieritz, Nadine, and Grosler, Andreas. 2003. *Emergent Structures in Supply Chains – A Study Integrating Agent-Based and System Dynamics Modeling*. The 36th Annual Hawaii International Conference on System Sciences, Washington, USA.

Solo, Kirk, and Paich, Mark. 2004. *A Modern Simulation Approach for Pharmaceutical Portfolio Management*. International Conference on Health Sciences Simulation (ICHSS'04), San Diego, California, USA. Available from http://www.simnexus.com/SimNexus.PharmaPortfolio.pdf.

Keenan, Philip, and Paich, Mark. 2004. *Modeling General Motors and the North American Automobile Market*. The 22nd International Conference of the System Dynamics Society, Oxford, England

UML – The Unified Modeling Language. http://www.uml.org.

AnyLogic. http://www.anylogic.com.

Decisio Consulting. http://www.decisio.net.

Addendum. Correspondence of SD and AB Models

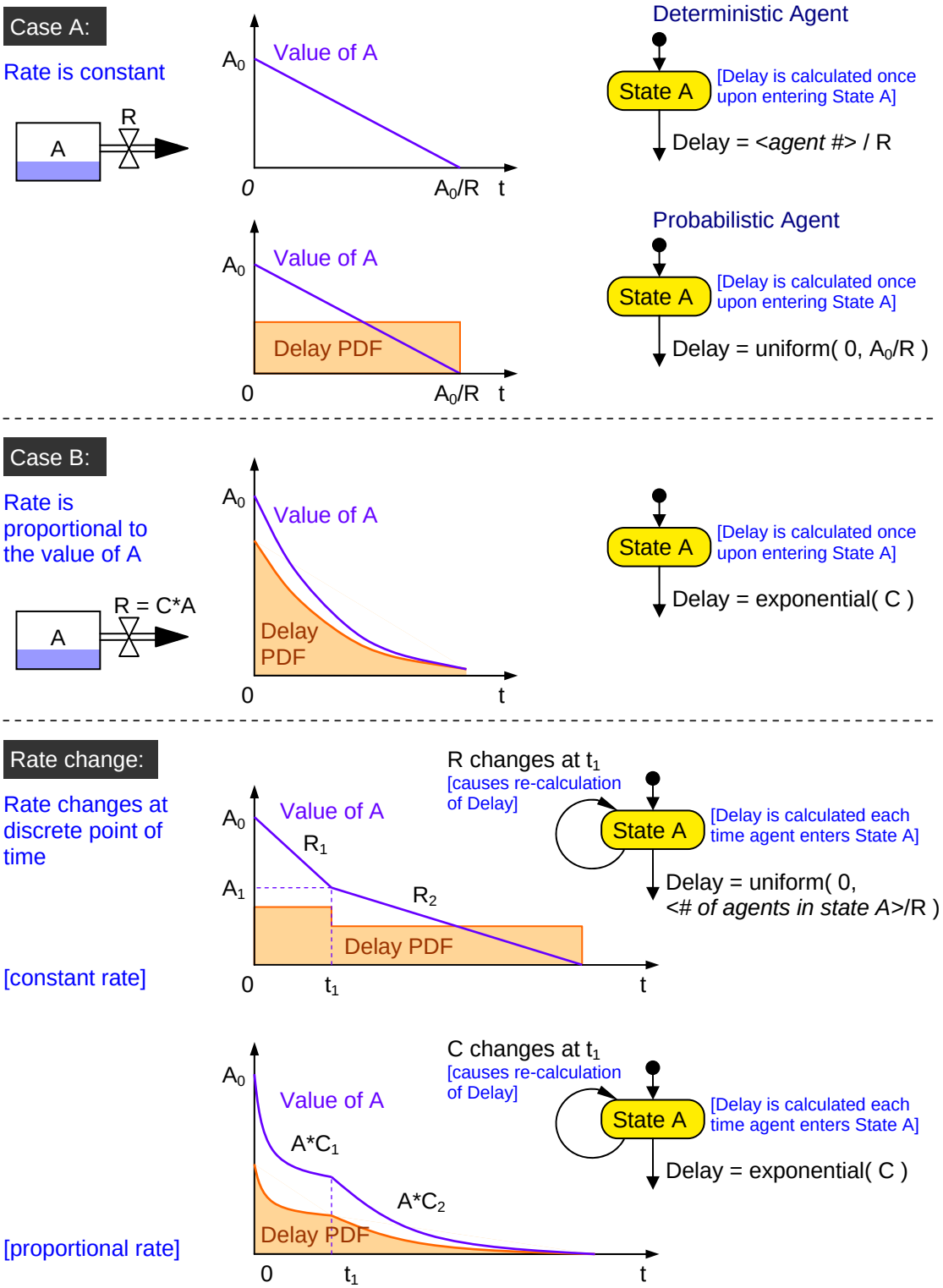

Figure 20: Correspondence of SD and AB Models. Constant and Proportional Rates

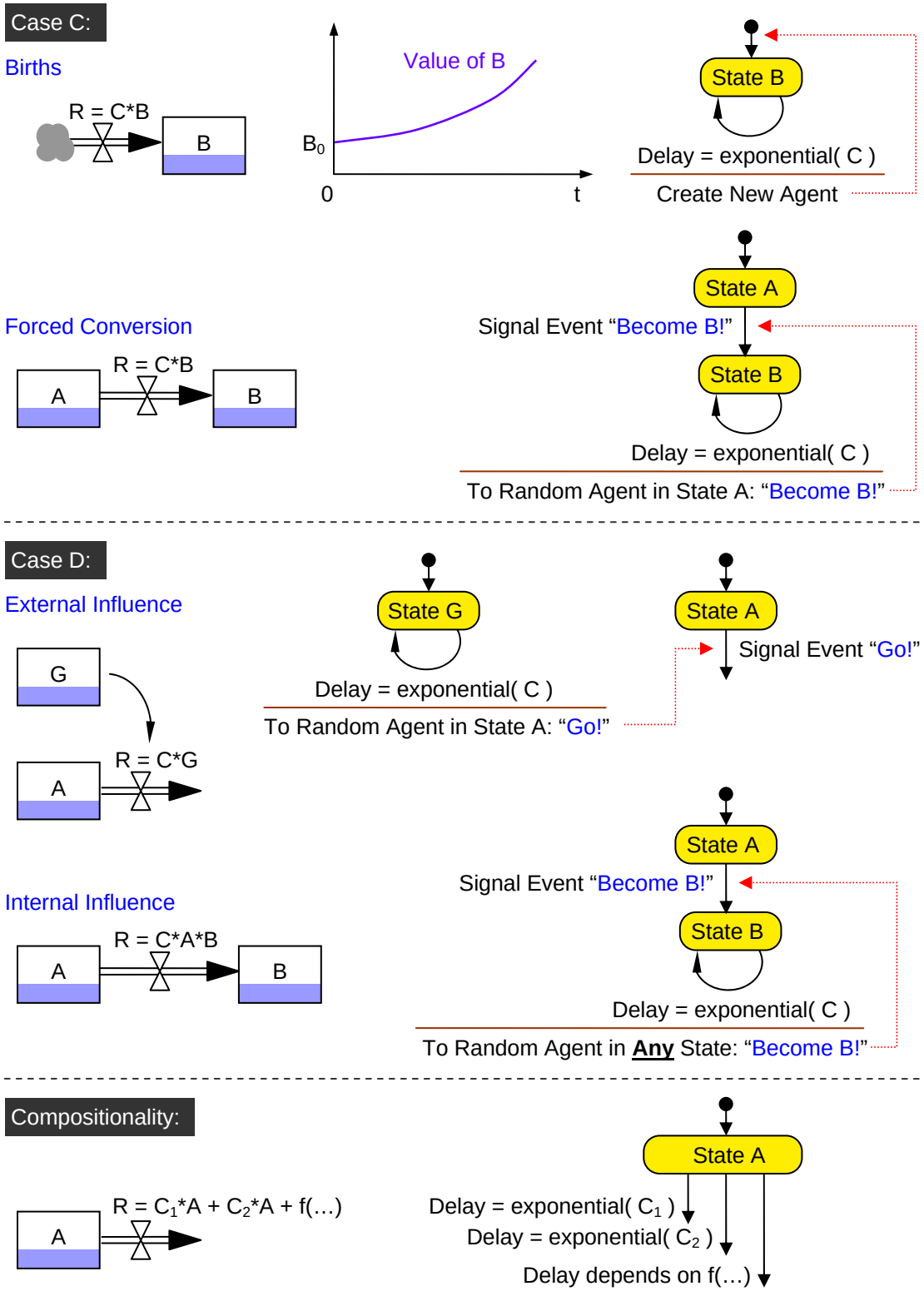

Figure 21: Correspondence of SD and AB Models. Multiple Stocks. Compositionality