NOVA: A Novel Tool for Collaborative Agent-Based and Dynamic Systems Modeling

Nancy Darling

Richard Salter

Ian Burns

Darling, N., Salter, R. M., & Burns, I. R. D. (in press). Nova: A novel tool for collaborative agent-based and dynamic systems modeling. In W. F. Porter, J. Zhao, L. Schmitt Olabisi, & M. McNall (Eds.), *Innovations in Collaborative Modeling*. Lansing, MI: MSU Press.

NOVA: A Novel Tool for Collaborative Agent-Based and Dynamic Systems Modeling

Solving complex problems requires teams of people who bring different skills and expertise to the table. Interdisciplinary collaboration requires its own skillset. Not only do people working in different disciplines often use different vocabularies, they may have different paradigms for understanding and producing knowledge and have different priorities about what they believe is practical, important or necessary in attacking a problem. Finally, people in different fields may work at different levels of analysis. For instance, the problem of infant malnutrition can be approached at the cellular level, in terms of food consumption and composition, as a dyadic problem of feeding in the parent-infant relationship, as a problem of family dynamics, education, shopping choices, or city-wide access to food. Understanding each level of a problem can be important, but it can be difficult for teams to communicate across levels and to understand the complexities of how each set of processes feeds into the other.

Models and simulations can be useful in articulating the linkages between different levels of analysis and areas of expertise. By making assumptions explicit, they also help to facilitate communication between collaborators. Nicolson and colleagues (2002) point out that the specialization of knowledge in, for example, ecological and environmental science demands a problem-solving approach that is interdisciplinary and collaborative. Ecological and environmental science provides a particularly cogent example because the level of analysis used to approach different components of a problem may run from the genetic to the individual to the landscape. Almost by definition, solutions require understanding how different systems interact over time. Simulation models are particularly effective for integrating disciplinary knowledge of stakeholders with different areas of expertise. When the diverse nature of the participants' knowledge is reflected in the model structure, both building and running models allows them to see how different systems work together (Nicolson et. al, 2002). The ability to develop a series of models that can be tested and refined (rapid prototyping, Starfield & Salter, 2010) is facilitated by software and other tools that are easy to use and provide readily understood graphic output. Running live models that can be manipulated to simulate different assumptions allows stakeholders to have a better understanding of the dynamic processes under discussion. Looking at concrete results of a prototype model provides stakeholders with 'real' data to discuss and facilitates productive exchanges and refinement of conceptual models. Success is more likely when the software system used facilitates easy editing, rapid prototyping, sensitivity analysis, and modular construction.

Having teams use software to turn conceptual models into computational models can provide additional benefits. In this chapter, we use the term 'computational models' to refer to computer simulations that illustrate how different processes unfold over time. Teams can use computational models in different ways to achieve different ends. Collaborative teams can (a) manipulate pre-built models to gain insight into their problem (b) develop progressively more specific understandings of a problem through moving from a 'sketch' to a finished model, or (c) develop new models from scratch. The goal of this chapter is to describe a freely distributed software package, Nova (Getz et. al, 2015, Starfield and Salter, 2010, Salter, 2013), that can be used to facilitate collaborative modeling of a wide range of problems.

Nova was developed as part of an NSF initiative for introducing dynamic systems thinking to novice modelers across disciplines. It is unique among computational modeling platforms in that it:

1. Naturally supports the creation of models in the system dynamics, spatial and agent-based modeling paradigms in a single desktop application.
2. Uses a visual toolbox to express model design, providing automatic conversion of such models to script form for execution.
3. Promotes hierarchical design, code reuse, and extensibility through the use of plug-ins.

This chapter uses an example drawn from the social sciences to illustrate how Nova works: student-teacher interactions in a classroom. This very simple set of nested deterministic models is used because it illustrates how Nova models can be built up by different stakeholders with knowledge of different areas, because it illustrates the advantages of using dynamic system modeling to understand interactive processes that unfold over time, and because it clearly operates at different levels of analysis. Although our example is drawn from the social sciences, Nova comes with an easily accessible library of models drawn from the biological, wildlife management, and physical science literatures (Salter, et. al., 2014).

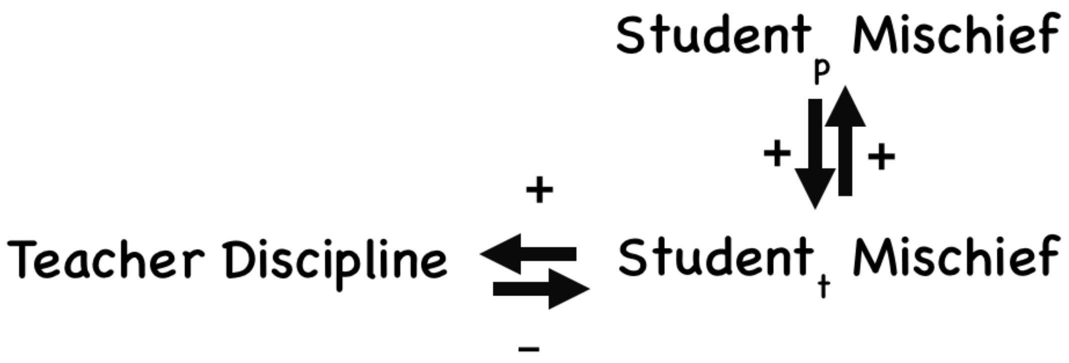
An Example: Dynamic Systems Theory in Developmental Science.

Like ecology and environmental science, the field of human developmental science has historically been interdisciplinary in nature and spanned research embracing biological, psychological, sociological, historical, and epidemiological approaches (e.g., Bronfenbrenner, 1979). The dynamic, interdisciplinary nature of the field is apparent in how the *NICHD Child Health and Behavior Branch* describes the developmental processes that underlie child health and behavior. They say child health and development is "best . . . studied as a variable process in which individual differences in cognitive, social, emotional, language, neurobiological and physical maturation, environment, life experiences, and genetics interact in complex ways" (NICHD, 2015). Despite this conceptual emphasis on multiple processes interacting over time, the techniques used to study these processes most often 1) are linear, 2) assume that within person change can be inferred from between person change (the problem of ergodicity) (Gayles & Molenaar, 2013), and 3) are not well suited for studying multiple simultaneous processes that are reciprocal, recursive, or self-regulating over time (Kunnen, 2012; van Geert, 2012). In other words, they are most often approached using a statistical methodological framework rather than a systems dynamic one. Finally, it has become increasingly apparent that development and, particularly, medical and behavioral interventions, are best understood in terms of individual patterning of co-existing and self-reinforcing processes and behaviors (Darling & Cumsille, 2003; van Geert, 2012). It is exactly this type of relationship that dynamic systems techniques are useful for understanding (van Geert, 2012).

Despite this, computer based dynamic systems modeling has been rarely used to study human development or related health-related behaviors. For example, only one of 41 chapters in the Handbook of Developmental Research Methods (Laursen, Little, & Card, 2012) discusses dynamic systems methodology. A brief search yielded only 11 article using dynamic systems modeling published in *Child Development* and *Developmental Psychology* since 2000, and several of these papers focus on use of simultaneous differential equations for model development, rather than on the use of computational modeling per se (e.g., Howe & Lewis, 2005). Underutilization of dynamic systems modeling as a tool results from two major factors. First, sociology, psychology, and medicine have traditionally relied on sophisticated statistical models, with researchers receiving relatively little training in other methodologies such as modeling (cognitive science and epidemiology are exceptions). Second, software developed to serve the dynamic systems modeling community (e.g., Stella, Vensim, Netlogo) and the modeling libraries developed to teach this software have not reflected the research needs of most developmentalists, instead, focusing on ecology, physics, and chemistry (e.g., http://www.iseesystems.com).

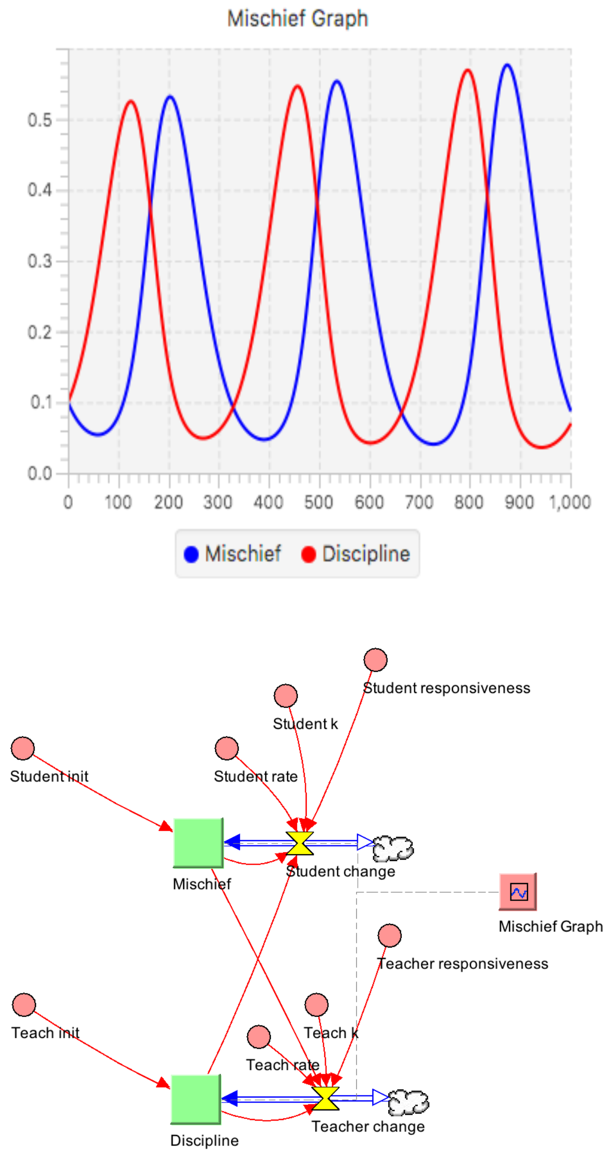
How can computational models help teams solve problems? Computational models focus on how things influence each other (processes) and how these processes change over time. For example, Figure 1 is a simple conceptual model of our example. In this model, we have a focal dyad: a teacher and student$_t$. The teacher and student$_t$ influence each other reciprocally. As teacher discipline increases, student$_t$ mischief decreases. As student$_t$ mischief incr

Figure 1: Conceptual model of teacher:student interactions. This model focuses on the target teacher:student$_t$ dyad. In this model, teacher's discipline and student$_t$ mischief are reciprocally related. Teacher discipline increases in response to student$_t$ mischief. Student$_t$ mischief decreases in response to teacher discipline. The target student$_t$ influences and is influenced by other students (Student$_p$).



eases, teacher discipline increases. Importantly, this model is not static: it assumes that these reciprocal processes unfold over time. Because the influences of discipline on mischief and mischief on discipline are not mirror images of one another (the former is negative, the latter is positive), how these processes will unfold over time is not immediately clear. One possible scenario is illustrated in Figure 2. This complex oscillating pattern that slowly amplifies over time was not immediately obvious from the initial conceptual relationship and depends on the exact assumptions of the model. This is where the collaborative team comes in and where computational models become useful. For example, three important parameters in determining how this system unfolds would be are (1) how big the influences of discipline and mischief are on each other (responsiveness), (2) how closely synchronized discipline and mischief are in time, and (3) the behavioral range or flexibility that teachers have in their discipline and students have in their mischief (k). Although a group of collaborators might generally agree on the conceptual model, useful discussions of each of these

Figure 2. Student-teacher dyadic behavior over time. The top of this figure illustrates how student mischief declines in response to teach discipline. As mischief declines, teacher discipline relaxes, resulting in a subsequent resurgence in mischief. Note the time lag between teacher and student behaviors. This output is derived from the Nova model in the middle. The equations determining change in student and teacher behavior per unit of time are below. Change in Mischief and Discipline depend on current levels of behavior (Mischief and Discipline), how often behaviors are emitted (Rate), how responsive teachers and students are to each other (Responsiveness), what their partner is doing, and how close they are to the floor and ceiling of their behavioral ranges (k).



Teacher_change = Discipline*(Teach_rate-Teach_rate*Discipline/Teach_k+Teacher_responsiveness*Mischief)
Student_change = Mischief*(Student_rate-Student_rate*Mischief/Student_k-Student_responsiveness*Discipline)

parameters could result in a much clearer understanding of how this dyadic system works. Looking at the output of the model makes it much easier to talk about how the results of the model differ from collaborator expectations, why that might happen, and whether it results from a problem in the conceptual model or the way it is operationalized mathematically.

Models are based on rules about how things change over time. That's what collaborators need to hammer out. These rules are based on relationships that are articulated either mathematically or algorithmically. For example, in the model of student and teacher behavior illustrated in Figure 2, changes in student behavior per unit of time get incrementally smaller as their behavior gets closer to maximum mischievousness[1]. In other words, we see a decelerating increase in mischief over time as they get more and more deviant. In some ways, that makes intuitive sense: as the student's behavior first begins to deviate from acceptable norms, changes may be relatively large. However, there may be only so far the student can go, thus their incremental increases in mischief become smaller and smaller as they get closer to that maximum level of deviance. Different stakeholders might disagree on exactly what those curves should look like. An alternative perspective might be that students start with small increases in mischief but increases in mischief accelerate as they become more deviant. As currently modeled, student change also decelerates as they get closer

[1] This and all models in this chapter are available as browser resident apps through http://www.cs.oberlin.edu/~rms/icm/. Full Nova models are also available for download through that site. The Nova software is downloadable through novamodeller.com. These and many other models are available through that site as well. Equations and supporting Novascript underlying the models are saved with each Nova model. Equations can be viewed using the *Edit Component Equations* command under *Tools* or exported using the *Save Equations* function under the *File* tab.

to expected normative behavior (low mischief). Teacher and student decelerate at the same rate. Are these assumptions correct? Discussion of such issues are exactly what the precision required by the modeling process facilitates. Mathematically, one can develop equations that express any of those behavioral patterns (and many more). We will return to this model later to see how it can be further articulated.

Barriers to Modeling and the Advantages of Nova

Equation based relationships can generally be referred to as a dynamic systems or stock and flow models. An alternative approach to modeling is algorithmic. One can write rules that say that if the people closest to you are more mischievous than you are, you will move away from them in virtual space. Or you can create a rule that says a tree has a 75% chance of burning if the one next to it is on fire. These are rule-based models where complex patterns derive from the behavior of individual agents (agent-based and spatial modeling). As can be seen in these examples, the development of a computational model requires very explicit articulation of relationships. Useful discussions are often spurred when stakeholders make their assumptions and beliefs clear, revealing differences. Building computational models together helps to spark those conversations. Underlying assumptions need to be articulated and agreed upon. Broad statements of how one stock or process affects others need to be quantified (for linear effects) or articulated (for algorithmic ones).

Computational modeling has many advantages, but requires software. Learning any software platform has costs. In addition, collaborators come to a project with their own preferences and requirements for hardware - e.g., they will typically already be working with hardware running Windows, Mac, Android, Linux, or iOS operating systems. Nova is versatile in that it runs custom built models of various types on multiple platforms. Because it can run dynamic systems stock and flow models, agent-based models, network, event-driven and spatial models using the same interface, it minimizes learning time. Thus rather than deciding whether to use an agent-based or stock and flow model based on the available software, models can be developed to fit the problem. Importantly, as we will illustrate below, different levels or aspects of a problem can use different types of models. For example, the interaction of a teacher and student can be modeled with a stock and flow model. This stock and flow interaction can then form the basis of a spatial model with multiple students influencing each other, or of an agent-based model in which students are attracted towards others like themselves and reinforce each other's characteristics. This minimizes the learning time involved in mastering multiple platforms. In addition, because Nova models can be built on machines running Windows, Mac, and Linux operating systems and can be run and explored using smartphones or tablets, it increases accessibility to broader user communities.

Nova as a modeling platform: An example

Stock and Flow Models. The following example illustrates key features of Nova and its use for a range of problems. Although the mathematical model on which this example is based is hypothetical, in a real-world application, it would be built upon theory and parameter estimates taken from the extant literature. Figure 2, which we examined earlier, shows a simple dyadic model in which *student mischief* and *teacher discipline* are reciprocally related. In other words, teacher discipline rises in response to student mischief, which declines in response to discipline. Both student and teacher *base rates* and *responsiveness to partner behavior* can be manipulated.

Several features of this model differentiate it from a traditional statistical model. First, the model illustrates how stable patterns of behavior can emerge from reciprocally related behaviors that change over time (the increasing oscillations to the right of the graph). In this model, all variables are reciprocally related and moderate one another. In addition, the changes in both student and teacher behaviors slow as they reach the top and bottom ranges of the measures (something not typically modeled using traditional statistics, which have difficulty making accurate predictions near the model floors and ceilings). There is another important paradigmatic distinction between traditional statistical models and computational models. While a statistical model parsimoniously summarizes observed data, a model such as this one would be used to **generate** data. It allows modelers to explore many different 'what if' scenarios. For example, collaborators could systematically vary teacher and student characteristics (mischief and discipline base rates and student and teacher responsiveness to each other's behavior) to see what patterns result. This can be done in Nova using batch processes running through researcher specified permutations. Because it was designed to facilitate modeling of biological and social systems, different types of distributions for inputs (e.g., normal, Poisson) can be specified for each variable. Once this simulation data is generated, it can be analyzed and compared to predictions from theory or from empirical data to provide a stronger test of specific hypotheses.

Nested Models. Many important problems involve nested data. For example, cellular processes are nested within organs, conflict behaviors are nested within dyads (e.g., Darling, Cohan, Burns, & Thompson, 2008), or students are nested within classrooms (Raudenbush & Bryk, 2002). Nova facilitates modeling of nested processes because it is designed so that submodels can be saved as 'capsules' and aggregated at higher levels. Figure 3 shows the results of a model of a four-student classroom in which each student has a different initial mischief level. Their different trajectories in response to teacher discipline are seen in the graph. All students share a common teacher, whose store of patience is determined by recent mischief in the classroom as a whole. Models of individual teacher-student dyads were imported from the model discussed in Figure 2, with dyads now nested in a classroom. Importantly, nested models can be used to examine individual trajectories under different conditions (for example, a well-behaved student in a classroom of mischievous peers). Using a team approach, classroom-level processes, such as patience, can be modeled by one group, while dyadic processes, such as student-teacher dyads, could be modeled by another.

Spatial Models. Spatial models are those in which individual agents are nested within groups and are influenced by others near them. These can be literally 'near', as when fire spreads from one tree to the next, or it can be figuratively 'near' as when memes spread through social networks. Spatial models can also be used to model neural nets. Some agent-based models allow individual agents to move, allowing modeling of, for example, social contagion, disease spread, or peer selection and reciprocal influence.

Figure 3. Four students in a shared classroom. The top graph illustrates the mischief of four students over time nested within a classroom with one teacher. It is the results from the model pictured at the bottom of the figure. In this model, each student-teacher dyad is represented as a cyan chip which contains the full model illustrated in Figure 2. The mischief of individual students varies because they have different behavioral base rates, so receive different discipline from the teacher. This nested model has one additional component: teacher patience. Because the same teacher is working with four students, the store of teacher patience (latency to discipline in response to student mischief) is depleted by all four students. These types of nested models allow collaborators to better understand how a student high in mischief might fare differently in a classroom with many difficult students compared to one with few. This full model, including the equations, is available from novamodeller.com.

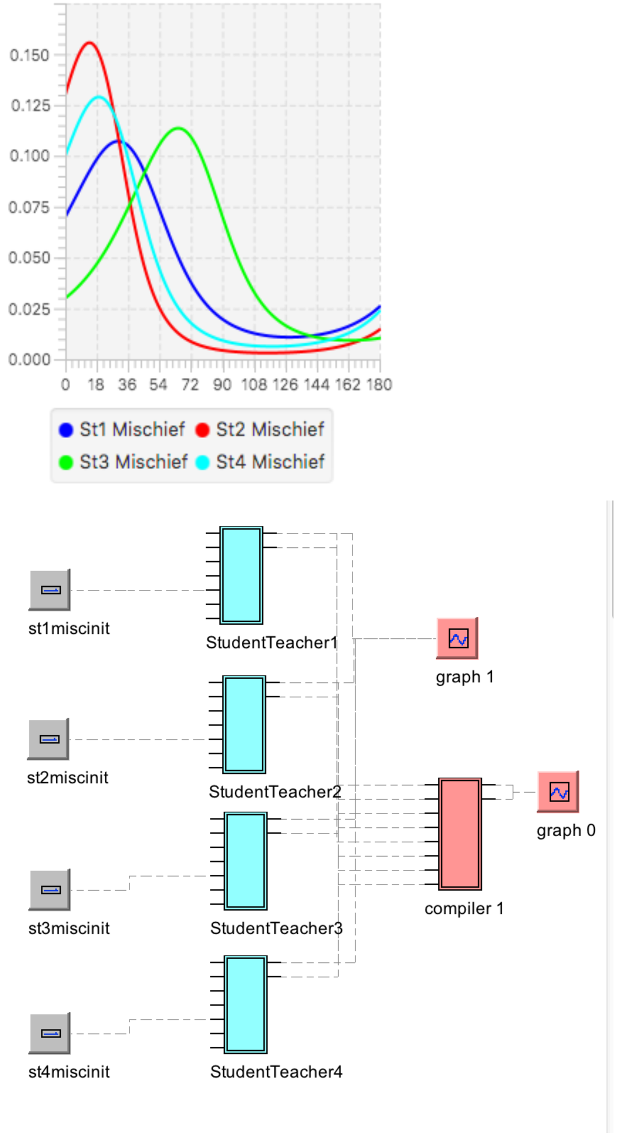
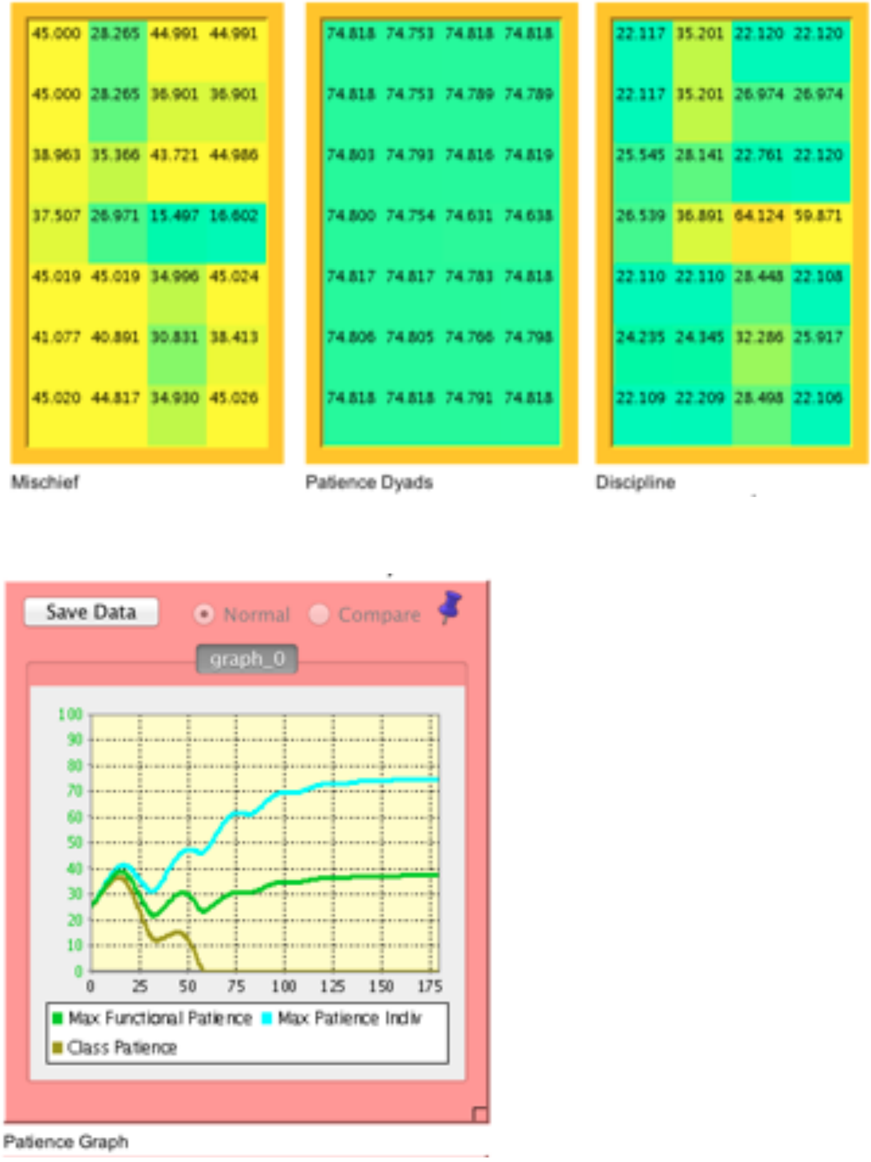


Figure 4 is the final result of a model of a 28-student classroom. Each student's mischief is modeled using the same capsules as in Figures 2 and 3 and teacher patience is again modeled in response to the mischief aggregated across the classroom. What has been added to this model is that student mischief is now affected not only by their own base rates and by teacher discipline, but also by the mischief of their neighbors. We now have the full conceptual model sketched out in Figure 1. Different numeric and color values represent levels of mischief and of teacher patience and discipline at the dyadic level at the model end point. This spatial model includes behavioral contagion of students seated next to one another. It was initiated with a random normal distribution of student mischief levels. However, researchers could specify both initial levels and variability and create individual differences in, for example, students' susceptibility to peer behavior. In this way, researchers might model the behavior of a student with ADD in different types or classrooms and with different types of teachers. The bottom of Figure 4 illustrates one consequence of allowing reciprocal processes to unfold at both the individual and classroom levels. The teacher's store of patience with the classroom as a whole falls precipitously as they spend more time disciplining the subset of students who engage in high levels of mischief. Figure 4 also shows the teacher's patience with the best behaving student in the classroom. Individual patience with this student rises over time because this student is seldom disciplined. However, the teacher's functional patience with this student is the patience that determines her discipline behavior. It is influenced by both her dyadic interaction with the student and her overall store of patience. In this classroom, the teacher acts with less patience with her best student than would be so in a less mischievous classroom. In addition, the

Figure 4: A spatial model of 28 students in a classroom. Mischief of each student is represented in the first grid and the teacher's patience and disciplinary behavior with each child is the second two grids. Each cell in the grids enacts the full model shown in Figure 2 plus Teacher patience. Numbers represent values of the individual in that cell and are reflected in color values. You also see the effects of behavioral contagion: students' mischief is also affected by that of their neighbors. Teacher patience with each student is affected by past student mischief, then combined across the full classroom. In this model, teacher disciplinary behavior with each student is determined jointly by patience with the individual student and patience with the class as a whole. The bottom graph shows how an individual student would be affected. Teacher patience with the class falls rapidly (dark green line). Although patience earned by an individual well-behaved student is quite high and rises over time (cyan line), in this classroom context, the teacher will respond with patience as if they had misbehaved more (bright green line).



45.000	28.265	44.991	44.991
45.000	28.265	36.901	36.901
38.963	35.366	43.721	44.986
37.507	26.971	15.497	16.602
45.019	45.019	34.996	45.024
41.077	40.891	30.831	38.413
45.020	44.817	34.930	45.026

Mischief

74.818	74.753	74.818	74.818
74.818	74.753	74.789	74.789
74.803	74.793	74.816	74.819
74.800	74.754	74.631	74.638
74.817	74.817	74.783	74.818
74.806	74.805	74.766	74.798
74.818	74.818	74.791	74.818

Patience Dyads

22.117	35.201	22.120	22.120
22.117	35.201	26.974	26.974
25.545	28.141	22.761	22.120
26.539	36.891	64.124	59.871
22.110	22.110	28.448	22.108
24.235	24.345	32.286	25.917
22.109	22.209	28.498	22.106

Discipline

Patience Graph

functional patience of the teacher is not perfectly linked to the behavior of the individual student. As in real classrooms, students are affected not just by their own characteristics and their own interactions with the teacher, but by how their neighbor behaves and by the discipline that other students experience. As with all models, Nova allows modelers to run through a series of theoretical or empirically derived values for each part of the process, output the results, and analyze them in an analysis package of their choice.

Agent Based Spatial Models. Nova is capable of agent-based spatial modeling by layering models of 'individuals' on top of models of their 'ground'. For example, the model of an individual student can accept input from models of nearby students. This information can then interact with the model of the student's corresponding 'ground' to simulate movement within space. Grounds can be either two or three dimensional. The final model in this series shows the spatial model of the classroom above. In this model, students move towards others who are similar in mischief. This creates pockets of mischief in a classroom. Because mischief is contagious, difficult students become more mischievous over time and the least mischievous students less so.

Nova Nuts and Bolts: Understanding the Architecture

The previous series of examples illustrates some of Nova's potential for solving problems. How does it work? Nova is a Java-based modeling platform that naturally supports the creation of models in the system dynamics, spatial and agent-based modeling paradigms within a single desktop application that can be disseminated through browser-resident apps. Nova uses a visual, icon-based language to express model design, and provides automatic conversion for such models to script form for execution. Nova's architecture promotes hierarchical design, code reuse, and extensibility through the use of plug-ins. Nova models can be built on computers running Apple, Windows, and Linux operating systems. Nova models can also be exported and run as in-browser applications on any device that the browser runs in, including smartphones and tablets.

As described in our teacher: student examples, Nova is fundamentally a dynamic modeling system that is extended through hierarchical design to express spatial and agent-based relationships. Stock and flow dynamic systems models are build describing processes, as in Figure 2. Nova can easily multiply those individual models so they can be combined together at higher levels, as in the four-student classroom in Figure 3. Those models can be used to populate a fixed matrix, as in the 28-student classroom (Figure 4). Or an agent-based model can be developed in which dynamic systems models are used to control the interaction between individual agents and the ground on which they 'move'.

Stock and flow models are built using an icon-based visual language modelers manipulate on a canvas by pointing and clicking. The functions that determine how these icons interact can be entered by clicking and entering equations. More detailed editing and programming can be done directly through the equation editor. Users can use NovaScript – similar to JavaScript – to develop specialized functions. Nova also facilitates moving data into and out of other data analysis programs like R, SPSS, Excel, or SAS. Models, submodels, and scripts can be shared between models or between users.

Nova's Structure

The basic units for building Nova models are *capsules*, *chips*, and *aggregators* (Figure 5). Nova focuses on the creation of a modular unit called a *capsule*. Each capsule is a complete model that interacts with its environment through an interface consisting of input and output channels. The simplest capsule might contain a stock-and-flow model such as the teacher-student model in Figure 2. When these complete models are introduced into a superordinate model of a classroom, they become *chips.* Figure 3 shows four student-teacher dyads. Each student-teacher dyad is represented as a chip and contains a full model of that interaction – i.e the full functionality pictured in Figure 2. In this case, we have a 4-student classroom modeled in a capsule which contains four chips. Each of these chips has I/O channels that move data in and out of the dyadic, lower level capsules and can communicate with one another or with other capsules or model elements. One of the most powerful features of Nova is that capsules – i.e. full models – can be introduced as chips into other larger and superordinate capsules, maintaining their full functionality. Capsules may also be exported and reused in other projects.

Looking at Figure 3 in more detail, we see four student-teacher dyads represented as chips. The 7 input pins on the left of each chip represent potential input parameters defined by the modeler in the student-teacher capsule the chip represents. They include, for instance, prior student mischief, prior teacher discipline, and teacher patience. Similarly, the two pins on the right side of each chip represent potential outputs from the student-teacher capsules, e.g., current discipline and mischief. In other words, the two outputs the student-teacher dyad produce are determined by seven inputs. There is an additional chip in the classroom model labeled 'Compiler 1'.

This chip takes information from each student-teacher model (chip) and combines it together to create classroom-level data. Note that there are two outputs from each student-teacher chip (8 total) and there are eight potential inputs to the classroom chip. The classroom chip 'Compiler 1' has been configured to combine information from different inputs to produce two outputs: the maximum classroom Discipline and Mischief at each time point. These are output to a graph (the square labeled Graph 0 in Figure 3). We have also taken the output of each individual student's mischief from their respective chips and output them to a second graphing chip (the square labeled Graph 1). The output of this chip is shown at the top of Figure 3. These functions are available in the Nova graphic palette and can be configured by the user. This nesting of models could continue, with the classroom capsule in Figure 3 becoming a chip that might be embedded in a higher capsule environment representing, for example, a school.

The chip is one type of *container*. It contains all the functions of a more complex submodel. Spatial and agent-based models are constructed using array-like containers called *aggregators*. For example, the 28-student classroom in Figure 4 is a 4 x 7 cell array, each of which contains a capsule containing the Figure 2 model. The current implementation provides five types of aggregators that can be combined together to produce different model types (Figure 5).

AgentVectors are how Nova represents agents moving through 'space'. Technically, AgentVectors are one-dimensional arrays of agents. As described in Figure 4, where students in a classroom moved through space and moved closer to other students with similar levels of mischief to themselves, each agent is a model (capsule).

An AgentVector can be associated with an agent such that it includes a representation for location and movement within a two dimensional space (the grid in which student agents move). AgentVectors also manage dynamic creation and destruction of agents. For example, AgentVectors can be used to represent bacteria and white blood cells, where bacteria die when they encounter white blood cells. In this model, an AgentVector would be used to describe changes in a series of cells representing the movement of the bacteria through a spatial grid. It would also determine that when the bacteria move to a cell occupied by a white blood cell, its state would go from 'alive' to 'dead'.

CellMatrices are two-dimensional arrays of capsules, (e.g., the 28-student classroom). They provide a means for representing interacting capsules in a spatial array (cellular automata). CellMatrices can connect capsules using either a four-sided Cartesian or a hexagonal topology. CellMatrices represent the spatial structure within which information is exchanged between capsules. In agent-based models, they are spatial structures in which the agents of an AgentVector move. Figure 6 shows a simplified model of fire spreading through a forest. The forest is represented as a cellular automaton in a two-dimensional Cartesian array. Each cell represents either a tree or a firewall. If it is a firewall, it prevents transmission of fire. If it is a tree, it can catch fire from one of its neighbors, and subsequently ignite of its neighbors. Nova implements this by having each cell in the CellMatrix contain a Nova capsule with single state component ("tree") that has one of three states: unburnt, burning or burned. A 50x50 array of these capsules is contained in single CellMatrix, which facilitates the communication that allows a cell to detect the state of its neighbors.

NodeNetworks are another way of connecting agents. NodeNetworks are an array of graph nodes with weighted directed connections. These can be used to model diverse phenomena such neural networks, but also can be used to model the relative likelihood of contagion under different conditions. Figure 7 shows a Susceptible-Infected-Resistance (SIR) model over a network of interconnected nodes (such a model is useful in modeling the spread of a computer virus, for example). Like the CellMatrix, the NodeNetwork maintains the array of individual nodes (each a capsule) and facilitates communication. (Note that, while this model limits connectivity to immediate neighbors in a cartesian grid, this is not an inherent limitation in the system.)

SimWorlds combine Agentvectors with Cellmatrices, so that agent locations correspond to matrix coordinates. The result is a virtual space of interacting agents and cells. For example, disease vectors can be represented as moving agents that can interact with different types of tissues.

NetWorlds are analogous to SimWorlds substituting a NodeNetwork for a CellMatrix as the space in which the agents operate.

Nova Tools

In addition to assets that constitute the structure, Nova includes components used for visualization, data entry, advanced computations and the like. These are implemented as **plug-ins**; that is, Java-coded objects that are added to the base set of model components. Nova ships with a basic set of useful plug-ins such as clocked chips, however a user familiar with Java may write additional plug-ins. These may be specialized to very specific domains. For example, Neural Net plug-in is available to model a feed-forward, back-propagating, multi-layer perceptron-based neural network configurable to any desired topology). Specialized plug-ins would likely be an important part of any component kit built for specific problem domains.

Finally, it may be necessary to provide algorithms in code to define the relationships among model components, or to execute state-changing commands. Nova provides a component called a **Codechip** for this purpose. A Codechip is added to Nova's design canvas like any other component; unlike other components, a new Codechip is a blank slate. It has no built-in content. Any benefit must come from the programming that it includes. Like the Chip component, the Codechip presents input and output pins to connect with the rest of the model. They are represented by variables in the Codechip's program, which uses the NovaScript language (an extension of the familiar and well-documented Javascript language). One example for a Codechip might be to compute the mean and variance of a set of data points. Once a Codechip has been created it enters the Codechip palette for the given model, and can be reused in multiple settings, or exported for reuse elsewhere.

At a deeper level, Nova's computational architecture comprises the semantics of NovaScript, a scripting language embedded in JavaScript. The NovaScript runtime environment is an extension of the ECMA 1.7 JavaScript standard. All Nova simulations are actually NovaScript programs executing on the NovaScript runtime interpreter. Nova has the flexibility that an experienced programmer can design many specialized components. These components can be saved, reused, and exported to other models and shared with other users.

Nova and Collaboration

Nova’s flexibility and architecture make it particularly well-suited for collaborative efforts. We previously discussed how Nova’s ability to export models to browser-resident apps allows stakeholders to manipulate, learn from, and refine models during collaborative work sessions. The graphic interface also facilitates experienced users working with teams to take ‘sketch’ models and refine them through collaborative discussion of model outputs. The modular nature of capsules also allows collaborators with different expertise to work on different parts of a model. This is particularly easy because capsules can be exported, reused, and shared.

Summing up some of the advantages we have discussed through previous examples:

Nova supports an eclectic set of modeling paradigms. Nova organically integrates system dynamics modeling with support for spatial geometries and mobile agents. Nova therefore provides a single platform for various models of different designs that may be produced by a single team. In other words, different team members can use Nova to develop stock and flow, spatial, agent-based, or neural net models without learning new software.

Nova’s architecture promotes modular design*.* Nova’s graphical design language encourages the creation of modular units (capsules) with well-defined interfaces. These *sub-models* aid in model design by “factoring complexity” and promoting sub-model reuse. They also permit simulations to be run at different levels. To use an analogy, one can build new models from previously developed capsules in the same way as one can build a new electronic devise from previously developed chips or integrated circuits. This makes it easier to integrate contributions from multiple authors.

Nova is extensible*.* Nova's *plug-in* feature extends the expressiveness of the graphical language with new functionality. A plug-in API allows users to design and deploy new plug-ins.

Nova supports abstraction*.* *Abstraction* is the process of extracting a set of interacting elements which together create a well-defined computation over a given set of inputs, and providing the ability to access that computation with different inputs from multiple points within the overall project. Nova's submodel and plug-in features are only two examples of its capability for abstraction. Unique to Nova are its *clocked chips* which abstract an entire simulation run, and *code chips*, which provide new functionalities expressed in *NovaScript™* as graphical components. In addition, submodel containers, or *aggregators*, manage large arrays of submodels used as agents or spatial elements in networks or cellular arrays.

Nova promotes asset reuse*.* Nova abstractions (submodels, plug-ins, clocked chips and code chips) may be exported and reused in other projects. It also becomes possible to create groups of reusable elements that address a particular problem area and distribute these "kits" to workers in those areas. The Nova Website is planning to support an exchange to facilitate sharing of such assets.

Nova integrates with other technologies*.* These include R, GPS and others in development.

Nova desktop integrates with its Web technologies*:* Nova models can be deployed remotely. *Nova On Line* can be used to distribute developed models through browser-resident apps. At the other end of the spectrum, *Nova On Server* can be used to implement large and complex models on servers or supercomputers.

The base model of Nova is freely distributed through NovaModeler.com. In addition to software, additional resources include a broad modeling library and a full series of training videos.

Conclusion

This chapter has presented Nova and shown how it can be used as a tool for collaborative modeling, and especially as a means of promoting computational modeling to solve complex problems. In addition to the models presented above, Nova has been used by teams of researchers in many successful projects in environmental science and epidemiology. Among these are 1) modeling for rhino herd population viability analysis (Getz, Muellerklein, et. al., 2016); 2) a model of the 2015 California measles epidemic (Getz, Carlson, et. al., 2016, Getz and Salter, 2015) and the 2014 African Ebola outbreak (Getz, Gonzalez, et. al., 2015, Getz and Salter, 2015); and 3) a genetic algorithm-base study of foraging patterns among herding mammals (Getz, Salter and Lyons, 2015). The population viability study has resulted in an on-line application that can be used by investigators around the world using their own sets of parameters (Getz, Muellerklein, 2016).

Going forward, we hope that Nova will continue to facilitate the use of computational techniques in an expanding sphere of applications, and in doing so promote collaboration on many levels.

References

Darling, N., & Cumsille, P. (2003). Theory, measurement, and methods in the study of family influences on adolescent smoking. Addiction, 98(Supplement 1), 21-36.

Darling, N., Cohan, C. L., Burns, A., & Thompson, L. (2008). Within-family conflict behaviors as predictors of conflict in adolescent romantic relations. *Journal of Adolescence, 31*(6), 671-690. doi: DOI 10.1016/j.adolescence.2008.10.003.

Getz, W. M., Muellerklein, O., Lyons, A. J., Seidel, D. P. and Salter, R. (2016), "A Nova Web Application for Population Viability and Sustainable Harvesting Analyses" (June 15, 2016). World Conference on Natural Resource Modeling. Paper 28. http://scholarexchange.furman.edu/rma/all/presentations/28

Getz, W. M., Muellerklein, (2016), "Numerus PVA: An interactive Population Viability Analysis web app for single and multi-populations." http://www.numerusinc.com/webapps/pva

Getz W. M., Carlson, C. J., Dougherty, E., Porco, T. and Salter, R. (2016) "An Agent-Based Model of School Closing in Under-Vaccinated Communities During Measles Outbreaks". SpringSim-ADS, 2016, April 3-6, Pasadena, pp. 689-696.

Getz, W.M., R. M. Salter, R and N. Sippl-Swezey. (2015) "The Numerus platform - An innovative simulation and modeling building environment." In (L. Yilmaz, W. K V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.) *Proceedings of the 2015 Winter Simulation Conference*. IEEE Press, 2015.

Getz, W. M., R. M. Salter, and N. Sippl-Swezey (2015). "Using Nova to Construct Agent-Based Models for Epidemiological Teaching and Research". In (L. Yilmaz, W. K V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.) *Proceedings of the 2015 Winter Simulation Conference*. IEEE Press, 2015.

Getz, W. M., R. Salter, A. J. Lyons, N. Sippl-Swezey, (2015) "Panmictic and Clonal Evolution on a Single Patchy Resource Produces Polymorphic Foraging Guilds", *PLOS ONE* , 10(8): e0133732.doi:10.1371/journal.pone.0133732.

Getz, W. M., R. Salter R. and O. Muellerklein. (2015) "A Nova Model and Web App for Sustainable Harvesting and Population Viability Analyses in Teaching and Research" , *DIMACS MPE 2013+ Workshop on Management of Natural Resources* , Howard University, Washington DC, June 4-6, 2015.

Getz, W. M. and R. M. Salter. (2015) "NOVA transmission chain models for epidemic forecasting: Ebola and Measles", *Impact of Environmental Changes on Infectious Diseases*, Sitges, Spain, March 23-25, 2015.

Getz, Wayne M., Richard Salter, Dana Paige Seidel and Pim van Hooft. (2015) "Sympatric speciation in structureless environments", *BMC Evolutionary Biology* , 201616:50 DOI: 10.1186/s12862-016-0617-0, http://bmcevolbiol.biomedcentral.com/articles/10.1186/s12862-016-0617-0

Getz, W. M., J.-P. Gonzalez, R. Salter, J. Bangura, C. Carlson, M. Coomber, E. Dougherty, D. Kargbo, N. D. Wolfe, N. Wauquier, (2015) "Tactics and Strategies for Managing Ebola Outbreaks and the Salience of Immunization.", *Computational and Mathematical Methods in Medicine* , Volume 2015 (2015), Article ID 736507, 9 pages, http://dx.doi.org/10.1155/2015/736507 .

Gayles, J. G., & Molenaar, P. C. M. (2013). The utility of person-specific analyses for investigating developmental processes: An analytic primer on studying the individual. International Journal of Behavioral Development, 37(6), 549-562. doi: 10.1177/0165025413504857

ADDIN EN.REFLIST

Howe, M. L., & Lewis, M. D. (2005). The importance of dynamic systems approaches for understanding development. *Developmental Review, 25*(3-4), 247-251. doi: 10.1016/j.dr.2005.09.002

Laursen, B., Little, T. D., & Card, N. A. (Eds.). (2012). Handbook of Developmental Research Methods. New York: Guilford Press.

Kunnen, S. (Ed.). (2012). *A Dynamic Systems Approach to Adolescent Development*. New York: Routledge.

Nicolson, C., Starfield, A., K, Kofinas, G., Kruse, J., (2002) "Ten heuristics for interdisciplinary modeling projects", *Ecosystems* 5: 376-384.

NICHD. (2015). Child Development and Behavior Branch (CDBB). Retrieved 1/8/15, 2015, from http://www.nichd.nih.gov/about/org/der/branches/cdbb/Pages/overview.aspx

Raudenbush, S. W., & Bryk, A. S. (2002). *Hierarchical linear models: Applications and data analysis methods* (Second ed.). Newbury Park, CA: Sage Publications.

Salter, R. and N. Darling. (2015) "NOVA: A New Tool for System Dynamics, Agent-Based, and Spatial Modeling", *Innovations in Collaborative Modeling* , Michigan State University, June 4-5 2015.

Salter, R.M. (2013) Nova: A modern platform for system dynamics, spatial, and agent-based modeling. *Procedia Computer Science*. 2013;18:1784–1793.

Salter, R. et. al., (2014) *Nova Modeler Website,* https://www.novamodeler.com/ .

Starfield, A. M. and R. M. Salter. (2010) “Thoughts on a general undergraduate modeling course and software to support it”, *Transactions of the Royal Society of South Africa* , 65(2)(2010):116-121.

van Geert, P. (2012). Dynamic Systems. In B. Laursen, T. D. Little, & N. A. Card (Eds.), *Handbook of Developmental Research Methods* (pp. 725-741). New York: Guilford.