Cellular Automata: A Novel Visual Modeling Approach

Hessam S. Sarjoughian
ACIMS
Department of Computer Science & Engr.
Arizona State University
Tempe, AZ 85281-8809
sarjoughian@asu.edu

Sajjan Sarkar
PeopleNet Inc.
7000 Central Parkway Ave NE Suite 650
Atlanta,GA-30328
sajjan.sarkar@peoplenet.com

Gary R. Mayer
Department of Computer Science
Southern Illinois University
Edwardsville, IL 62026-1656
gamayer@siue.edu

Abstract
Many important scientific and engineering problems are studied using Cellular Automata (CA) models. Mathematical formulae with component-based modeling concepts are used to specify CA models. Computer simulation tools support viewing the dynamics of CA simulation models. Existing approaches with their tools are restrictive from visual model development perspective since models must be specified in mathematical or programming languages or exist as pre-built models. In this paper, the novel concept of spatial-specialization CA modeling is developed and introduced into CoSMoS (Component-based System Modeler and Simulator), a logical, visual, and persistence modeling and simulation framework. Modelers can visually create and manipulate structures of a family of hierarchical component-based CA models. The tool supports semi-automatic translation of component-based CA models to DEVSJAVA simulation code. An illustrative landscape erosion model is developed to highlight the basic capabilities of CA modeling in CoSMoS.

Keywords: Cellular Automata, Component-based modeling, CoSMoS, DEVS, visual modeling

1 INTRODUCTION

A variety of problems and solutions – concerned with application domains such as landscape erosion dynamics, disaster relief planning, and computer networks – are commonly studied using Cellular Automata (CA) models [6,17]. CA is considered to be a collection of homogeneous cells that have specific spatial topology or connectivity pattern. One or more cells may change state according to a set of rules that govern CA dynamics. Every cell is identical to every other cell in terms of its spatial dimension, shape, attributes and behavior. A simple CA example model consists of pieces of land (cells) that collectively represent a farmland with varying characteristics and dynamics.

Visual modeling tools are desirable for specifying simple to complex systems. Users with different degrees of expertise use component-based modeling and simulation tools across numerous application domains. Non-CA Tools such as Matlab®/Simulink® [8] have been highly effective in simplifying model development and use – i.e., creating models for the individual components of a system and synthesizing them in different ways to represent alternative models of the system. The existing approaches and available tools that can lend themselves for CA model development are restrictive. In particular, modeling approaches such as Cellular DEVS [15,18,19] which are based on classical paradigm of formal, logical model specification can support visualization of models once they are implemented in computer programming languages. Other approaches and tools such as NetLogo [16] and Mathematica [7] use the concept of generic pre-built models which can be specialized and synthesized to specify user-defined models, often within special application domains.

In this paper, we propose a novel approach aimed at visual development of CA models. The approach introduces a new relationship called spatial-specialization between a CA and one or more sets of the CA's cells. It enables visual, flexible creation and manipulation of cellular automata models. A realization of this approach is implemented by extending the Component-based System Modeler and Simulator (CoSMoS) framework [12]. The tool supports storage of a family of models in relational databases and automated generation of cells and CA model components and their instances. The main benefits of the visual modeling tool are exemplified using a basic landscape model that undergoes a simple erosion process.

2 BACKGROUND

The importance of visual model representation as exemplified with UML is well recognized across many scientific and engineering domains. Tools have amply demonstrated the significance of visual modeling for simulation and system design.

Tools that support visual model development (e.g., Matlab®/Simulink®) are important in reducing model development effort including design specifications and documentation. Indeed, visual modeling is recognized as a fundamental capability for model conceptualization, development, design of experiments among other modeling and simulation lifecycle activities. This is due to the ability to synthesize hierarchical system models from alternative sub-system models in various configurations without resorting to the implementation of models as computer programs [8,12].

Alternative approaches exist for specifying CAs. The concept of cellular automata introduced by Von Neumann [1] affords a simple, yet powerful approach for modeling a variety of complex systems [17]. Cellular Automata models may be specified in terms of general-purpose or specialized programming languages and modeling formalisms (for examples see [6,9,15,17, 19]. An attractive approach for describing CAs is to consider each cell as a component [18]. The cells, a topological network pattern, and rules form a spatial network of components. The components define the structure and behavior of a CA – i.e., the basic model elements are a cell and a topology which together support constructing families of hierarchical models. For example, NetLogo is an easy to use tool that supports visual modeling of agents and their environment specified as cellular automata. An advantage of this tool is its accessibility to users with limited programming or software engineering knowledge. Users can visually develop and manipulate CA models without requiring advanced simulation code development.

Our goal is to show that cellular automata is a class of models that may be developed visually using component-based modeling paradigm. In order to develop a visual CA modeling environment, we employ the conceptual and design principles that resulted in the Component based System Modeler and Simulation (CoSMoS) framework.

2.1 Cellular DEVS

Cellular Automata (CA) models have formal specifications and are generally realized in different programming and scripting languages. CA models can use the concept of component-based modeling where each cell is a component and each CA is a composition of cells. For example, Cellular DEVS which is an extension of the DEVS formalism is used to create and simulate cellular automata (e.g., [14,18]). The cells are defined as basic DEVS atomic models that are coupled to form multi-dimensional component models. The underlying basis of Cellular DEVS is the spatial multi-component DTSS described in [18,19].

Cellular DEVS simulation tools such as CD++ [15] and DEVSJAVA [4] support model development in terms of set-theoretic mathematical specification and programming code. The rules that account for individual cells and their combination are defined within atomic and coupled DEVS model components. In addition the DEVS "closure under coupling" property is enforced. This means every CA model is a kind of coupled DEVS model where a special connectivity pattern (e.g., Von Neumann and Moore neighborhood) among the cells is enforced. Cells are essentially atomic components with structured inputs/outputs, time-based state transitions, and I/O connectivity pattern. The difference between one cell and another is its state at a given time instance and its location with respect to the CA's topology.

2.2 CoSMoS

CoSMoS (Component-based System Modeling and Simulation) as a framework is aimed at a unified logical, visual, persistence component-based simulation model specification [12]. It supports component-based modeling approaches such as DEVS and XML Schema. Its underlying premise is to support both simulatable (e.g., DEVS) and non-simulatable (XML) models. Logical model specification is defined for primitive and composite DEVS and XML models. A set of axioms is provided to ensure consistency of a family of alternative hierarchical model specifications.

CoSMoS provides the facility to design the models at three different levels of abstraction called Template Models (TM), Instance Template Models (ITM) and Instance Models (IM). A Template Model specifies primitive and composite models with hierarchy depth of two. Every atomic and composite component can be specialized and can have a finite number of input and output ports. The I/O ports have names and can receive and send data, respectively. Types for input and output data can be specified. The modeler may also specify data and type for state variables of the primitive components. An Instance Template Model is defined to have a finite hierarchy of depth greater than two. It may not specify multiplicity of a model component within a composite model. An Instance Model is an instantiation of an Instance Template Model where the multiplicity of model instances is specified.

The logical models are completely specified visually and stored in one or more relational databases. The models in the databases can be transformed to target simulation models – in particular models can be translated to simulation code for DEVSJAVA or [4] DEVS-Suite [5]. Various behavioral and structural metrics can be queried for any primitive or composite model component. DEVS-Suite is integrated into CoSMoS in order to support simulating the generated models with animation, TimeView, and Tracking Log. The translated simulation models must be completed – i.e., transition, output, and timing functions are implemented – before they can be simulated. Such capabilities are useful for reducing model development effort and making modeling accessible to a larger group of users.

3 APPROACH

Different methods including Cellular DEVS can be used to specify cellular automata logical models. Such methods use common techniques to render CA models visually as interconnected components with implicit or explicit connections. Example tools are Cell DEVS, DEVSJAVA, and NetLogo. These approaches and tools lack the basic concept of components-within-components visual model development) that has been developed for non-cellular automata models. As a result, visual creation of cellular models is restricted to rendering of models that are already implemented (e.g., DEVSJAVA) or limited means for model manipulating (e.g., loading a pre-built model in a tool such as NetLogo).

The concept of *spatial-specialization* for cellular automata specification is proposed and introduced into the CoSMoS framework. The concept affords a basis for visual development of cellular automata models. Specifically, logical cellular automata models can be visually created and manipulated with support for database persistence and automatic creation of partial simulation models. CA models with different connectivity patterns can be created, deleted, and modified as in CoSMoS. Cells and CAs are the primitive and composite model components in the same way as in non-cellular hierarchical component-based models. That is, the CoSMoS modeling syntax and semantics for creating, deleting, modifying, and copying are extended for cellular automata using the spatial-specialization concept. The spatial-specialization concept is formulated based on the following definition for cellular automata.

Every CA uses a cell which is treated as the base model component. The cell is the fundamental, atomic unit of a cellular automaton network. Every cell has states, behavior, shape, dimension, and inducers. Every state has a name with an associated data type. In principle the state value can be of any type including double, integer, and string. The behavior of a cell is specified via rules subject to the generic atomic DEVS state-transition functions – next state value(s) are computed based on previous state value(s), the rules, and input from neighboring cells. We use the term nearest neighbor to represent cells that are immediately connected to one another via input and output ports. This is to differentiate between these cells and those that have i^{th} neighbor relationship [6] – i.e., two cells may be interacting via one or more intermediary cells. Every cell and thus CA can have a finite number of dimensions. Most common dimensions are 1D, 2D, and 3D. The inducers of a cell (called influencees) are its neighbors (called influencers). The interactions among cells (i.e., sending and receiving messages via output and input ports) are specified in the CA model. Thus every cell has only knowledge of itself – i.e., a cell does not know the cells to which it is connected to. Input port of an influencee is used to receive data from an influencer and thus the influencee may change its state based on the input and rules. Output port of an influencer is used to inform the cell's influence about the influencer's state change.

3.1 Cellular Automata Model

For Cellular Automata, we define Instance Template Model (ITM) and Instance Model (IM). The Template Model that is defined for non-cellular component-based models is not used. This is due to homogeneity of cellular automata models and no need to specify multiplicity for cells. The Instance Template Model specification includes hierarchy depth, frame of reference, spatial dimension, and network connectivity pattern. The elements of the CA are a primitive (or base) cell and any collection of cells that have strong neighbors relationship and enclosed within the CA. Each cell is defined to have name, scale (also referred to as dimension), and frame of reference. Scales for (x, y) are defined to be 1×1 and frame of reference is defined as (x_c, y_c, orientation) where x_c and y_c are x and y coordinates with positive integral values and each of the x and y has an enumerated orientation value of either North, South, West, or East. The orientation can be eliminated given a fixed (x, y) orientation. The Instance Template Model for cellular automata is defined as follows:

- Uses a single primitive cell. The cell's x and y scales are 1 and 1.
- All cells are connected to one another using one type of I/O connectivity scheme (Moore or Von Neumann). Each cell's connectivity is defined by CA.
- CA is homogeneous. CA has two or more cells – i.e., at least the scale of one of x or y is greater than 2. All cells share the specification of the primitive cell – i.e., the same set of attributes and set of rules is shared among all cells. All changes to the CA are based on the specification and set of rules defined for the primitive cell.
- Each cell and CA can have specializee/specialization relationship (see Table 1).

- Each cell and CA has its own frame of reference subject to spatial-specialization relationship.

CA can have composition and spatial-specialization relationships (see Tables 2 and 3). The composition refers to a CA containing cells or other CAs subject to definition given above. The spatial-specialization relationship is defined in terms of specializee and specialized elements. The spatial-specialization relationship is defined as follows:

- The specializee cell attribute values are not initialized and the specialized cell attribute values are initialized.
- Two specializee CAs differ in their assigned attributes' values and/or their scales.
- Specializee cell or CA does not have frame of reference. Each specialized cell or CA has a frame of reference.
- There exists only one specializee cell from which all specialized cells can be defined. The specialize cell can be specialized once or as many times as desired subject to the maximum number of its appearances in an instance model. Every specialized cell's attribute values are unique with respect to all other specialized cells.
- The x and y dimensions of any specialized CA is greater than one and less than or equal to the x and y dimensions of the CA that contains it.

Properties	Cell	Specializee CA	Specialized CA	CA
Attributes	Y	Y	Y	Y
(x, y) Dimensions	1×1	> 1×1	> 1×1	> 1×1
Coordinates	Y	N	Y	Y

Table 1: Elements of CA Models

Composition Relationship (Column to Row)	Cell	Specializee CA	Specialized CA	CA
Cell	N	Y	Y	Y
Specializee CA	N	N	N	N
Specialized CA	N	N	N	Y
CA	N	N	N	Y

Table 2: CA Composition Relationships: Cell, Specializee CA, and Specialized CA Elements

Spatial-Specialization Relationship (Column to Row)	Cell	Specializee CA	Specialized CA	CA
Cell	N	Y	Y	Y
Specializee CA	N	N	N	N
Specialized CA	N	Y	N	N
CA	N	N	N	N

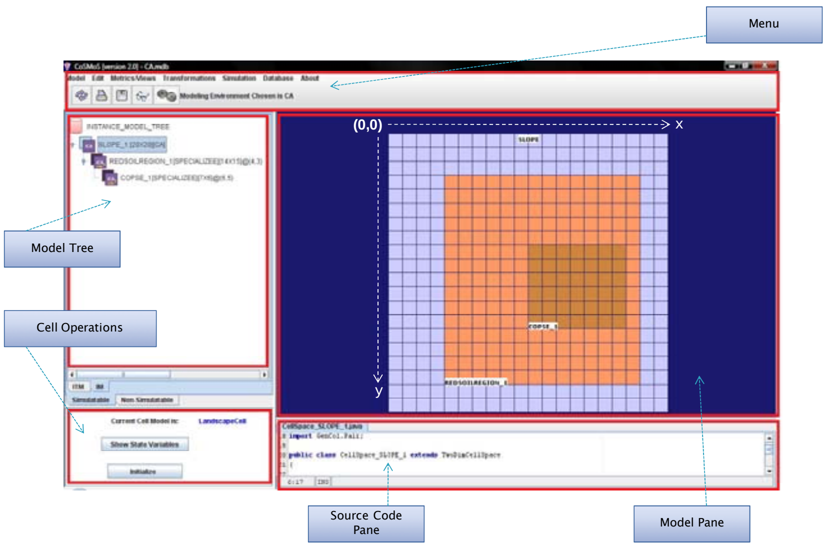
Table 3: CA Spatial-Specialization Relationships: Cell, Specializee CA, and Specialized CA Elements

The Instance Template Model can be used to define instance of Cells and CAs based on their distinct attributes, dimensions, and frames of reference. An Instance Template Model can be used to define any number of unique CA instance models. The Instance Model refers to the collection of all CA instance models. Primitive cell can only be instantiated as elements of one or more CAs (i.e., it cannot be independently instantiated). In other words, the cells are bound atomically to their CAs in order to distinguish between a cellular model and a non-cellular model. The Instance Model is defined to have composition relationship among its cells. Thus, given an instance template model, its instance models define the specialized cells' values and the frames of reference for specialized cells and CAs. The network pattern (i.e., von Neumann and Moore) is retained from the instance template model. Any cell that is not specialized also retains default attribute values and frame of reference. Every change in the primitive cell cascades through all specialized cells and the CAs that use it (see Section 4).

3.2 User Interface

The special characteristics of cellular automata require visualization that is distinct from those provided for non-cellular component models. Indeed, the motivation behind the CoSMoS CA modeling approach was to support visual model development. The concepts behind Instance Template Model and Instance Model are adapted for 2D CA visual representation. The user interface design shown in Figure 1 is based on the CoSMoS' non-cellular component-based modeling. The CoSMoS tool [3,13] has been extended to support logical, visual, and persistent CA model development. A brief description for the tool's user interface is provided below:

- *Menu Pane:* Handles to a variety of utility tasks like refreshing the Model Tree, launching the source code viewer, and printing the current model (see Cell Operations Pane).
- *Model Tree Pane:* Provides a hierarchical representation of the family of models (instance template model and instance models). Every instance template model's name and dimensions and in case of instance models their coordinates are shown. Each model can also be modified using menus.
- *Model Pane:* Provides the canvas for visualizing all CAs (instance template model and instance models) that are defined in 2D space. The x and y dimension directions with their assigned initial coordinates (0, 0) are overlaid in the user interface.
- *Cell Operations Pane:* Provides menus for operations such as viewing and editing cell. All cell operations affect every CA that is created using the cell.
- *Source Code Pane:* Displays the exported partial simulation code.


Menu
CoSMoS [version 2.0] - CA.mdb
Model Edit Metrics/Views Transformations Simulation Database About
Modeling Environment Chosen is CA
INSTANCE_MODEL_TREE
SLOPE_1 [20X20][CA]
REDSOILREGION_1[SPECIALIZED][14X15]@(4,3)
COPSE_1[SPECIALIZED][7X6]@(9,5)
Model Tree
Cell Operations
ITM
IM
Simulatable
Non Simulatable
Current Cell Model is: LandscapeCell
Show State Variables
Initialize
(0,0)
x
y
SLOPE
COPSE_1
REDSOILREGION_1
CellSpace_SLOPE_1.java
Source Code Pane
Model Pane

```
import GenCol.Pair;

public class CellSpace_SLOPE_1 extends TwoDimCellSpace
{
```

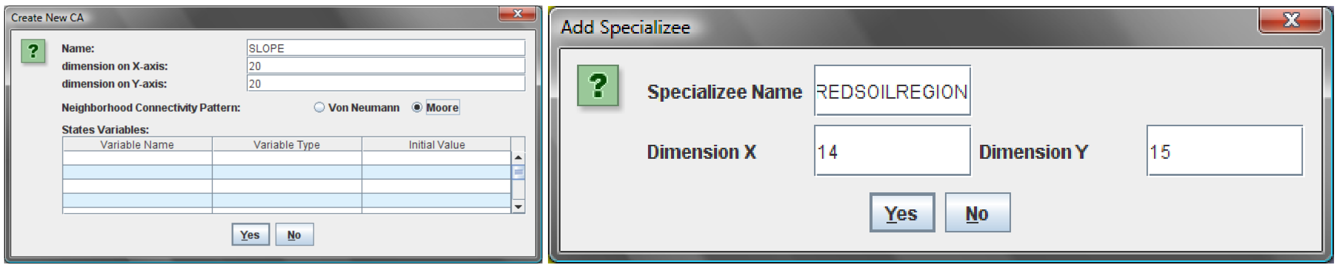
Figure 1. User interface for developing CA models in CoSMoS

4 EXAMPLE

A piece of landscape, for example representing side of a mountain in 2D space (20 × 20 cells), is used to highlight CA model development in CoSMoS. The dynamics of interest for the landscape could be soil erosion. Each landscape cell can be described in terms of R, K, C, and P attributes [2]. R is defined as rainfall intensity factor. K is defined as soil erosion resistance factor computed in terms of the percent of sand, silt, clay, and organic matter. P is defined as an erosion prevention practice factor. The values for the rainfall intensity, erosion resistance, and erosion prevention factor can be real or integer positive values.

Assuming rain drops flowing from one side of the landscape to the other in a uniform pattern, the soil can begin to erode. The landscape changes are calculated using the supply of sediment and the topographic characteristics of the areas up and down slope from that point [2]. A 2D CA model of the landscape (called Slope) can be formulated in terms of the CA's x and y dimension and topology pattern given a primitive cell's with the above attributes and rules (Figure 2 shows specification of the CA's structure). One or more parts of the Slope model can be specialized using the spatial-specialization concept. The Red Soil Region CA is a specializee that has a part of it also defined to be the Copse CA specializee (see Figure 3). Therefore, the cells of the Soil model can be hierarchically specialized (assigned different initial values).

Given the instance template model (INSTANCE_TEMPLATE_MODEL) with SLOPE, REDSOILEROSION, and COPSE elements, any number of instance models such as SLOPE_1 can be generated by iteratively allowing the user to assign x and y coordinates as well as initial values to the specializee elements as shown in Figure 3 (for example, the x and y coordinates (7×6) for COPES is notated as @(7,6)). The Model Tree structure of the instance model SLOPE_1 (INSTANCE_MODEL_TREE) with its specialized parts is shown in Figure 4. The 2D visual representation of the instance template model Slope used for creating the SLOPE_1 model is depicted in Figure 1. This model can be transformed to simulation code for simulators. In particular, DEVSJAVA is used as the target simulation engine. The partial simulation code must be completed with rules before it can be simulated.



Create New CA

Name:	SLOPE
dimension on X-axis:	20
dimension on Y-axis:	20

Neighborhood Connectivity Pattern: Von Neumann [✓ Moore]

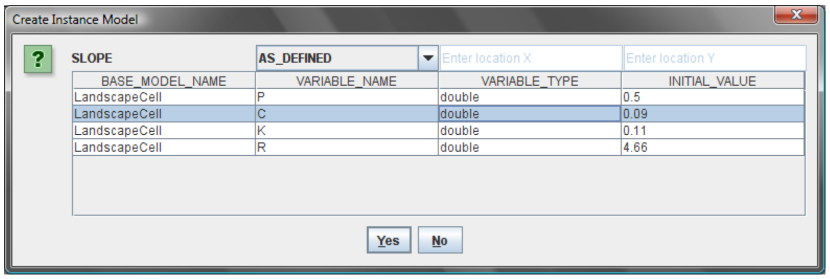
States Variables:

Variable Name	Variable Type	Initial Value

Add Specializee

Specializee Name	REDSOILREGION		
Dimension X	14	Dimension Y	15

Figure 2. Creating CAs: Slope and the specializee Red Soil Region models

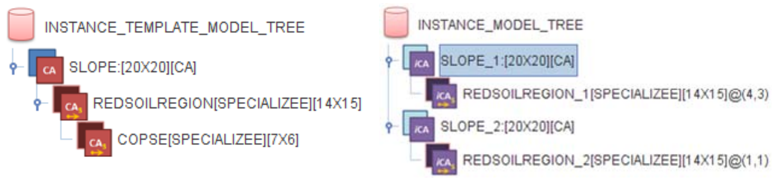


Create Instance Model

SLOPE AS_DEFINED Enter location X Enter location Y

BASE_MODEL_NAME	VARIABLE_NAME	VARIABLE_TYPE	INITIAL_VALUE
LandscapeCell	P	double	0.5
LandscapeCell	C	double	0.09
LandscapeCell	K	double	0.11
LandscapeCell	R	double	4.66

Figure 3: Defining CA x and y coordinates.



```
INSTANCE_TEMPLATE_MODEL_TREE
  SLOPE:[20X20][CA]
    REDSOILREGION[SPECIALIZEE][14X15]
      COPSE[SPECIALIZEE][7X6]
```

```
INSTANCE_MODEL_TREE
  SLOPE_1:[20X20][CA]
    REDSOILREGION_1[SPECIALIZEE][14X15]@(4,3)
  SLOPE_2:[20X20][CA]
    REDSOILREGION_2[SPECIALIZEE][14X15]@(1,1)
```

(a) (b)

Figure 4: Model Tree Models: (a) Instance Template Model (b) Instance Models

5 RELATED WORK

The CoSMoS is compared with DEVSJAVA, NetLogo and Mathematica7 [7]. To limit the scope of comparison among the modeling supported by these tools, visual modeling, logical modeling, code generation, and model persistence aspects are considered. Visual Modeling determines whether the tool supports construction and viewing of complex cellular automata. Logical Modeling is defined in terms of support for the following:

- Families of models: Whether or not multiple CA models can be created, saved, and reused.
- Specialization: Whether cells can have different initialization values for their state variables. This is important to create spatially specialized regions.
- User-Defined Rules: Whether or not customized rules can be defined.
- Multi-Dimensional CAs: Whether or not cellular automata with more than 2 spatial dimensions can be defined.
- Model Translation is concerned with code generation given a visual model. Finally, model persistence determines the medium that is used for storing families of CA models.

NetLogo is aimed at CA-based multi-agent modeling. It supports developing cellular automata models from pre-built templates. It also has a visual interface where the user can assign initial values to cells' attributes and viewing simulation execution. Simple rules for the cells may be created using the native NetLogo programming language. However, NetLogo doesn't support specifying hierarchical CAs with the concept of spatial-specialization. Furthermore, there is no support to model cells as first class objects with explicit representation of inputs and outputs and thus coupling of cells to form CAs. Unlike CoSMoS where no programming language is used for visual model development, programming code is required for NetLogo.

DEVSJAVA as in CD++ supports cellular and non-cellular DEVS-based modeling. With DEVSJAVA or CD++, users typically formulate pseudo code, devise set-theoretic models, develop UML models, or write simulation code. The underlying concepts for these tools, unlike CoSMoS, do not consider visual model development. Thus, they also do not consider transformation of visually developed models to simulation code. They support viewing of simulation models. Due to lack of Instance Template Model and Instance Model concepts, DEVSJAVA and CD++ do not offer support for creating a family of models – i.e., supporting development and management of a group of alternative models. Unlike CoSMoS, models developed in these tools are stored and used as flat files.

Mathematica7 is a computational software tool that can be used for a variety of application domains. It supports model building using interpreted expressions. This tool consists of a kernel and a front end. The kernel interprets expressions (Mathematica code). Its front end supports document-centered user interface. However, there is no support for visual model development – the GUI shows the output of evaluated expressions. Mathematica offers an extensive rule sets, but user defined rules are not supported. Models are stored in flat files and given its interpretive concept, there is no need for model to code transformation. Other aspects such as availability of pre-defined domain-specific models, technical support, and user-base are not considered. Considering such aspects, Mathematica7 is superior.

Tool	Visual Modeling		Logical Modeling				Model Translation	Model Persistence
	View	*Create*	*Families of Models*	*Specialization*	*User-Defined Rules*	*Dimension (>2)*		
DEVSJAVA	Yes	No	No	Yes	Yes	Yes	NA	Flat file
NetLogo	Yes	yes	No	Yes	Yes	No	No	Flat file
Mathematica7	Yes	No	No	Yes	No	Yes	NA	Flat file
CoSMoS	Yes	Yes	Yes	Yes	Yes	No	Yes	Database

Table 4. A comparison of cellular automata modeling approaches with their tools

6 CONCLUSIONS

The main idea of this research has been to develop a basis for visual model development of component-based cellular automata models. The benefits of visual modeling and simulation are widely recognized and numerous tools have been develop across application domain such as computer networks, logistics, and automotive. Visual model development can simplify the entire modeling and simulation life-cycle development activities. Visual notation for Cellular Automata that is based on sound model abstraction concepts and principles is important for model development. Tools such as CoSMoS can aid in efficient, resourceful model development by supporting visually creating models that can also be automatically transformed to target simulation code. The CoSMoS framework and its tool have been extended to support specifying structures of cellular automata models. It's unified logical, visual, and persistent component-based model specification is generic and thus lends itself for use in arbitrary application-specific system modeling (e.g., [11]). The new concept of spatial-specialization is introduced and implemented for developing homogeneous geospatial specifications cellular automata in CoSMoS. Modelers can define any number of cells to be instantiated in terms of their desired spatial arrangements. Although the CoSMoS implementation does not support modeling behavior of cells, it can be extended to support visual specification of rules and thus increasing support toward complete code generation. Other desirable capabilities include support for multi-dimensional CAs [6] and composable CAs [9]. Another future work is for CA models to be simulated. Currently, the visual CA models cannot be animated since there is no support for linking a simulator's execution data to CAs and no ability to view simulation trajectories.

Acknowledgement:

This research was partially supported by Intel Research Council and NSF Grant #BCS-0140269.

References:

[1] Burks, A.W., 1970, Essays on Cellular Automata, Editor, University of Illinois Press, Urbana, Illinois, USA.
[2] Braun, J., A. M. Heimsath, and J. Chappell, 2001, "Sediment transport mechanisms on soil-mantled hillslopes," Geology, Vol. 29, No. 8, 683-686.
[3] CoSMoS, 2009, http://sourceforge.net/projects/cosmosim/.
[4] DEVSJAVA, Arizona Center for Integrative Modeling and Simulation, 2007, http://www.acims.arizona.edu/SOFTWARE.
[5] DEVS-Suite, 2009, http://sourceforge.net/projects/devs-suitesim/.
[6] Ilachinski, A., 2001, Cellular Automata: A Discrete Universe. World Scientific, New Jersey.
[7] Mathematica, http://www.wolfram.com/products/mathematica/index.html.
[8] Mathworks, 2007, http://www.mathworks.com.
[9] Mayer, G. R., H. S. Sarjoughian, 2009, "Composable Cellular Automata," Simulation Transactions, Vol. 85, No. 11-12, 735-749.
[10] Neteler, M. and H. Mitasova, 2004, Open Source GIS: A GRASS GIS Approach, 2nd Edition, New York, New York: Springer Science + Business Media, Inc.
[11] Natimo, L., X. Hu, and Y. Sun, 2008, "DEVS-FIRE: Towards an Integrated Simulation Environment for Surface Wildfire Spread and Containment," Simulation Transactions, Vol. 84, No. 4, pp. 137-155.
[12] Sarjoughian, H. S. and V. Elamvazhuthi, 2009, "CoSMoS: A Visual Environment for Component-based Modeling, Experimental Design, and Simulation," International Conference on Simulation Tools and Techniques, Rome, Italy.
[13] Sarkar, S., 2009, An Approach to Visual Modeling of Cellular Automata, Master's thesis, Computer Science and Engineering Department, Arizona State University, August, Tempe, AZ, USA.
[14] Wainer, G. and Q. Liu, 2009, "Tools for Graphical Specification and Visualization of DEVS Models," Simulation Transactions, Vol. 85, No. 3, pp. 131-158.
[15] Wainer, G., 2009, Discrete-Event Modeling and Simulation: A Practitioner's Approach, CRC Press.
[16] Tisue, S., and U. Wilensky, 2004, "NetLogo: Design and implementation of a multi-agent modeling environment," Agent Conference on Social Dynamics: Interaction, Reflexivity and Emergence.
[17] Wolfram, S., 2002, A New Kind of Science. Wolfram Media Incorporation.
[18] Zeigler, B. P., 1976, Theory of Modeling and Simulation, John Wiley and Sons, Reprinted in 1984 by Kreiger Publishing Company, Inc.
[19] Zeigler, B. P., H. Praehofer, and T. G. Kim, 2000, Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems, 2nd Edition, Academic Press.