Superdense Time Trajectories for DEVS Simulation Models

Hessam S. Sarjoughian
Arizona Center for Integrative Modeling & Simulation
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA
hss@asu.edu

Savitha Sundaramoorthi
Arizona Center for Integrative Modeling & Simulation
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA
savitha@asu.edu

ABSTRACT
Many scientific and engineering applications generate data that are well-suited to be studied using time series charts. Two types of time series that define input, output, and state dynamics of DEVS models are piecewise constant and event charts. In this paper, time series capable of displaying both linear and superdense time segments and trajectories are conceptualized and formulated. These lend themselves for visualizing behavior of parallel atomic and coupled DEVS models. The concept of superdense time segments is realized as plug-ins as part of the Eclipse BIRT (Business Intelligence and Reporting Tool) framework. They can receive time-based alphanumerical data sets from external static and dynamic sources, including the DEVS-Suite simulator. As standalone plug-ins, time series can be used to create static plots and used in BIRT reports. These plug-ins are also integrated into the DEVS-Suite simulator where each model component's behavior can be customized and dynamically plotted. Time series charts simplify and complement tabular logging of data sets for developing simulation models that exhibit zero-time transitory state transitions.

Author Keywords
BIRT, Data Trajectories; DEVS-Suite Simulator; Dynamic Visualization, Event Chart, Linear Time; Piecewise Constant Chart, Superdense Time.

ACM Classification Keywords
D.2.2 [Software Engineering]: Design Tools and Techniques; D.2.5 [Testing and Debugging]: Debugging Aids; D.2.6 [Programming Environments] Graphical Environments; I.6.1 [Simulation Theory]; I.6.4 [Model Validation and Analysis]; I.6.5 [Model Development]; I.6.6 [Simulation Output Analysis]; I.6.8 [Types of Simulation]: Visual.

TMS/DEVS 2015, April 12 - 15, 2015, Alexandria, VA, USA

INTRODUCTION
In many component-based modeling approaches, dynamics of the simulated components and their interactions are defined in terms of time. Computations and communications between any two components must occur either sequentially or concurrently with respect to a clock. The measure of time between any two consecutive time instances can have unbounded accuracy (as in continuous time) or bounded accuracy (as in discrete-time). The notion of linear time is concisely defined as an algebraic structure $\langle T, \prec \rangle$ where T is a set and $\prec$ is an ordering relation on the elements of T. Given T and a set of arbitrary values V with the constraints that for any $t \in T$ there exists only one value $v \in V$, trajectory charts with linear time can be easily plotted. For modeling approaches that can support multiple, contiguous instantaneous state changes the concept of linear time needs to be augmented with superdense time where two simultaneous time instances are defined to be contiguous with time instances $t[n_1]$ and $t[n_2]$.

A discrete-time and discrete-event models can be defined in terms of input and output sequences. In these system-theoretic modeling formalisms [10,18,19], time is mathematically defined to be either discrete or continuous. These models may allow multiple inputs (or outputs) to occur at an instance of time due to a model being in one or multiple states for a zero time duration (e.g., parallel DEVS [2]). The functions responsible for processing these inputs, changing states, and producing outputs can be mathematically specified in terms of superdense time [11]. However, to visually render trajectory charts for a finite number of contiguous state changes at an instance of time (i.e., instantaneous state transitions) and thus multiple inputs and outputs occurring an instance of time, visual representations accommodating superdense time need to be conceptualized, formulated, and implemented. They are particularly important for understanding transient state transitions (state changes in zero time period). State (with input/output trajectories) are very useful for understanding time-based dynamics of simulation models and especially identifying subtle timing issues which may be due to design flaws and/or implementation errors.

A modeling formalism that supports sequential and parallel model execution and communication is known as Discrete Event System Specification (DEVS) [2]. The theory of DEVS modeling is well-suited for conceptualizing superdense time. Trajectory charts with superdense time can be systematically defined and developed for tools such as DEVS-Suite simulator [5,8].

The DEVS formalism, grounded in system-theory [18], is based on a strong notion of I/O modularity where data belonging to any model component can only be shared with another model component via their coupled input and output ports. Complex, large-scale coupled models can be composed via tree-structure composition of atomic and coupled models subject to satisfying the closure under coupling principle. Inner dynamics of any atomic model (defined in terms of time and state) can be observed via I/O variables. Operations in all atomic and coupled model components can be defined (directly or indirectly) in terms of a logical time base. As complexity and scale of atomic and coupled model increases, it is useful to conveniently choose and observe their dynamics of interests in alternative, and complementary textual, tabular, and visual forms. Plotting trajectory charts for models that undergo contiguous, instantaneous state transitions is desirable although non-trivial to conceptualize and formulate given the restrictions imposed for creating input, output, and state trajectories in finite spatial spaces.

Every model can have input and output variables with corresponding input and output ports. The data variable types for inputs, outputs, and states can be simple (e.g., `int` and `enum` primitive data types) or complex (e.g., `Integer` and `Pair` class types). Variables such as `Pair` have complex structure (e.g., `(key,value)`, `key=(phase,sigma)`, `value=String`) from the standpoint of displaying them visually. Consequently, crude simplifications are often employed to display such complex data types. To visually view these data trajectories requires defining representations that can capture simultaneity for input, outputs, and states.

The time base used for visual data trajectories can be restricted to be linear even though the model can undergo internal or external transition in zero logical time. Limiting the time base of visual data trajectories to be strictly linear prevents visualizing transient state changes and generation of simultaneous events. The dynamics of a model instantaneously responding to input events or generate output events cannot be visually created and thus inaccessible (visualized) while the model being executed.

Another useful capability for visualizing the dynamics of any atomic model is to display variables belonging to a component in a "synchronized" fashion. Although an arbitrary model's behavior is asynchronous (i.e., inputs, outputs, and states may occur at distinct time steps during a simulation cycle), it is useful for the time axes of all selected data trajectories to be aligned. Such a visualization allows precise visual observation (evaluation) of the behavior of any atomic model including its I/O and state changes in response to internal, external, and confluent transition functions. Observation of the I/O and state variables can be extended to include the atomic simulator's state variables such as time of last event and time to next event. Alignment of the model's and simulator's visual data trajectories offers the modeler the ability to examine and understand simple and complex relationships that exist among external, internal, output, and time advance functions.

In the DEVS modeling formalism, two key concepts for the atomic model are allowing time to the next event to be defined as infinity or zero for the external or internal transition functions. The former allows future input events to arrive at arbitrary time instances. This requires visualizing time to next event data trajectories having the infinity value. With the latter, transitory state transitions can be allowed. The consequence is that (input, output, and state) trajectory charts for all legitimate DEVS models needs to be supported – i.e., dynamics of any atomic model that has a finite number of state transitions with zero time durations – can be plotted. Supporting visualization of superdense time results in creating trajectory charts for atomic and coupled models that can receive multiple inputs or outputs at the same time instance.

TRJACTORIES WITH LINEAR AND SUPERDENSE TIME BASES

States, inputs, and outputs of any dynamical model can be defined in terms of linear [19] and superdense time [11]. Time can be represented by a variable $\boldsymbol{t}$. We can specify every time-dependent input, state, or output of a model represented in terms of a variable $\boldsymbol{v}$ and variable $\boldsymbol{t}$.

Definition 1. $t^{[\mathsf{C}]} = t \times [\mathsf{C}], t \in \mathbb{R}^{+}_{0,\infty}, \mathsf{C} \in \mathbb{N} \setminus \infty$ represents a collection of time instances that are monotonically ordered with respect to time t. Time is a linear variable – i.e., given time duration $\delta t_i \doteq t_{i+\mu} - t_i = \mu$ and shifted by $\xi \in \mathbb{R}^{+}$, then $\delta t_j \doteq t_{j+\mu} - t_j = \mu,\ j = i + \xi$. The reference value for t is defined to be zero; it can be shifted by any positive or negative finite value. For any time instances t_i and t_j that are equal (i.e., $t_i - t_j = 0$), they are distinguished using the index term C – e.g., given $t^{[0]}, t^{[1]}, t^{[2]}$, time instant $t^{[1]}$ occurs after $t^{[0]}$ and before $t^{[2]}$ denoted as $t^{[0]} \preccurlyeq t^{[1]} \preccurlyeq t^{[2]}$. The operator $\preccurlyeq$ defines weak simultaneity of time which means two consecutive time instances $t^{[n]}$ and $t^{[n+1]}$ are unequal. The index term n is required if at least there are two equal time instances. For totally ordered time instances belonging to t, they degenerate to $\cdots, t_g, t_j, t_p, \cdots$. Therefore, the time base can be $\cdots, t_j, t_k^{[0]} t_k^{[1]}, \cdots, t_k^{[m]}, t_\ell, \cdots$ where $m < \infty$. Totally and partially ordered time instances are illustrated in Figure 1.

Definition 2. $g = t^{[\mathsf{C}]} \to v$ is an onto, but not a one-to-one function. $v \doteq \{a, \cdots, z\} \cup \{A, \cdots, Z\} \cup \mathbb{Z} \cup \mathbb{R} \cup \pm\infty$ can represent alphanumeric values for state, input, and output variables. Every alphanumeric value has a finite length. Values at any time instance for input, output, and state trajectories are determined using $g\left(t^{[\mathsf{C}]}\right) = v$. Variables

with numeric values are linear. Alphanumeric values have an arbitrary sequence. The negative and positive infinity values are the smallest and largest values. When a variable with alphanumeric values is assigned infinity, the values form a linear relationship with respect to the negative and positive infinity values as being the only smallest and the only largest values. A piecewise constant trajectory with a time that has both linear and superdense time is illustrated in Figure 2.

A trajectory is defined to be a concatenation of one or more segments. Two distinct time instance values are required to define the duration of a segment. The duration of a segment can be defined to range from zero to infinity inclusively. A segment can have a duration of zero, in which case its start and end time instances are partially ordered $t_i^{[k]} \preccurlyeq t_i^{[k+1]}$ (see Definitions 1 and 2). A segment with infinity duration is linearly ordered – i.e., $t_i < t_j$ or $t_i^{[q]} < t_j$ where $t_i \neq \infty, t_i^{[q]} \neq \infty, t_j = \infty$ and $q > 0$. A segment with infinity duration is a mathematical artifact which cannot be exactly displayed.

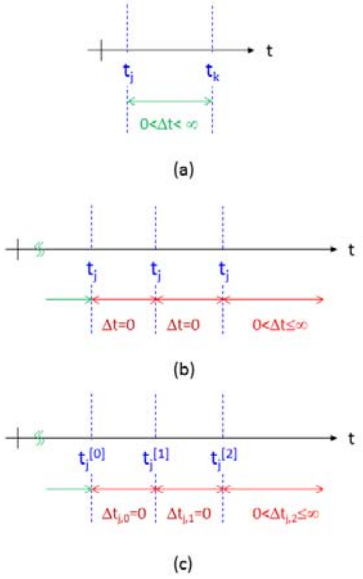

Figure 1: (a) Time segment satisfying the $<$ linear time ordering relationship, (b) contiguous time segments each having zero time duration, (c) contiguous time segments transformed to superdense time.

A segment can be piecewise constant (i.e., variable v has a constant value for the duration of the segment). We also have event segments where there exists a single between t_i and t_j $(t_i, t_j \mid t_i < t_j)$ or $t_i^{k[\ell]}$ and $t_i^{k[\ell+1]}$ (see Definitions 1 and 2). Values for a segment can be arbitrary (i.e., values for segments that have durations greater than zero and less than infinity can be defined using any function or relation). A trajectory can be created by concatenating segments using [,) rule. Therefore, trajectory $\omega_0 \circ \omega_1$ for consecutive piecewise constant $\omega_0 = v_0$ and $\omega_1 = v_1$ segments shown in Figure 2, ω_0 has value $v_0$from $t_0$until $t_1$and $\omega_1$has value v_1 from $t_1$until t_2. Event trajectories follow the same concatenation rule. Specifically, every event segment in an event trajectory has its values at the start of the segment with no values defined for the rest of its time duration (e.g., ω_0 has value v_0 at $t_0$with no values until t_1).

DEVS I/O AND STATE TRAJECTORIES

The inputs and outputs X, Y of any DEVS atomic model $M = \langle X, S, Y, \delta_{ext}, \delta_{int}, \delta_{conf}, \lambda, ta \rangle$ and coupled model $N = \langle X, Y, D, \{M_d\}, IC, EIC, EOC \rangle$ are strictly defined as event segments and trajectories. The state of any atomic model are also strictly defined as piecewise constant segments and trajectories. Event segments and trajectories may be transformed to piecewise, continuous, or other forms using appropriate mapping functions.

The values for trajectories can range from negative to positive infinity. Therefore, it is necessary for charts to support visualizing the full range of values a variable may have. An example of a variable that can have positive infinity value is time to next event in DEVS atomic model. Clearly, there may also exist some variable with negative infinity value. Therefore, input, output, and state piecewise constant and event trajectories with infinity values are important to be visually rendered.

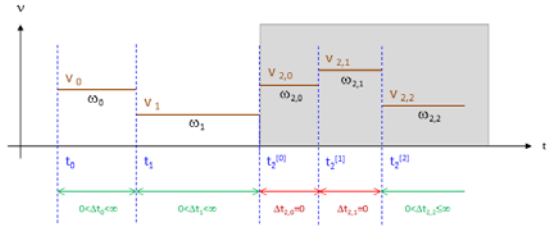

Figure 2: Piecewise constant data trajectory with superdense time base.

Input trajectories

An input is defined as either endogenous or exogenous. Inputs received from other atomic models are *endogenous*. Time-based input variables are defined as $g = t^{[C]} \rightarrow v$. Every input value corresponds to a unique time instance (see Definition 1). Input trajectories generated by sources other than atomic or coupled model (communicated via IC, EIC, EOC couplings) are *exogenous*. An example of an exogenous input trajectory is a data file having all its entries satisfying Definitions 1 and 2. These input trajectories can become endogenous via "autonomous" atomic models that read such data files and generate output events which in turn become input events.

Output trajectories

All outputs are *endogenous* and are generated by atomic models or other non-simulatable components [15] that belong to atomic models. Strictly speaking, coupled models cannot generate outputs on their own due to the closure under coupling principle.

State trajectories

Any state trajectory can become an output trajectory. The state S of any atomic models is defined as a set of variables.

States of a model can be specified as $phase \times \sigma \times v_1 \times \cdots \times v_q \mid \sigma \in \Re_0^{+\infty}$. The values for σ are computed using $ta(s)$. Together $phase$ and σ define the minimum state set for a model. They are referred to as the *primary* states. Phase must have at least two values in order to define state-based behavior (i.e., a model must at least be in two distinct states) [18]. Other variables (i.e., $v_1 \times \cdots \times v_q$) are called secondary states. There are no restrictions on what secondary variables may represent, except any variable that is related to time must be consistent with σ as defined in time advance function $ta(s)$. The state variables are commonly analyzed for run-time simulation verification and model validation.

Every simulation scenario for an atomic model with a finite number of state transitions must have the same number of segments as there are state transitions. The simulation scenario for an input trajectory can have at most the same number of input segments as there are state transitions. The same holds for output trajectory. A state transition can be transitory or non-transitory. Zero logical-time refers to a simulation cycle (due to either an internal or external transition function) with $ta(s) = 0$. Such segments are called zero-time segments and conform to Definition 2. In the context of a logical-time atomic DEVS model, a zero logical-time segment corresponds to a transitory state ($ta(s) = 0$). For any legitimate DEVS model, there may only be a finite number of consecutive zero logical-time segments in any trajectory. Furthermore, only a finite number of non-transient and transient state transitions may exist for a simulation scenario having a finite duration. It is also assumed there is no ordering among simultaneous input and output events that occur at the same time. The order of contiguous transient states, however, is determined in the external transition function which can be deterministic or not. There is ordering among simultaneous output events.

Given simulation cycle periods specified by $ta(s)$ in a model's external or internal transition function, a simulation scenario can be categorized to have either *non-transient state transitions:* Every simulation cycle has a duration greater than zero and less than infinity or *mixed non-transient and transient state transitions:* Every simulation cycle can have zero, finite, or infinite duration.

Restrictions

To plot trajectories for input, output, or state variables, they must be a primitive type. The variables that can be plotted over a time period can be numbers and strings. Numbers can be integer, real, or others (e.g., bytes). Variable type String can also be plotted. In principle, symbols (e.g., $\diamond$ and $\odot$) can be used instead of numbers or strings. If a variable has a complex type, it must be decomposed to its primitive parts (numbers and strings) in order to be plotted. If complex data types can be mapped to symbols, they can be displayed too.

DEVS-Suite simulator variables

It is sometimes useful to plot time of last event t_L and time to next event t_N. These variables belong to the simulation protocol used in the DEVS-Suite simulator. These variables specify time duration for the input, output, and state segments. For an atomic model, $t_N$is the time for the internal transition firing and can also be the time instance at which outputs are computed and sent out. For a coupled model, $t_N$is the shortest time duration from t_L for one or more atomic models which have either an internal transition, an input scheduled to arrive, or both. Time instances belonging to t_N and t_L can have linear or superdense relationships.

MIXED LINEAR AND SUPERDENSE TIME TRAJECTORIES

The time axis and spatial axis are distinct. The numerical time axis has to be mapped to string values, each of which is spatially displayed using pixels. One approach to display both linear and superdense time is shown in Figure 3. In this setting, time axis is composed of linear and superdense time segments where time instances are labeled as defined above.

Mapping numeric time and value axes to their spatial counterparts

The duration for linear time segments are defined to be greater than zero and less than infinity. Time labels for these segments are unique for any single trajectory. These numerical numbers map one-to-one to pixels. In contrast, the duration for every superdense time segment is zero. Therefore, their time labels are indexed as defined above. A segment with zero time duration (measured in terms of the model and simulation time) is defined to have a finite spatial length which can be in pixels. For example, given the segment between $t_1[0]$ and $t_1[1]$ with $\Delta t = 0$ (i.e., $t_1[0] \preccurlyeq t_1[1]$) is mapped to x_2 and x_3 with $\Delta x = \chi$, which must be a finite number of pixels (see Figure 3). The x_2 and x_3 spatial values are strictly ordered (i.e., $x_2 \prec x_3$) even though their corresponding time values are weakly simultaneous. The spatial length for all superdense time segments for all input, output, and state variables is assumed to be the same.

Composing linear and superdense time segments as just described has few shortcomings. Adding or removing superdense time segments causes charts to expand or shrink. This is undesirable when the charts are being plotted dynamically. An alternative approach is shown in Figure 4. In this setting, the linear and superdense time axes are separated. This is important as it simplifies visualization of the charts even though more display space is required. Another benefit of this approach is that charts can scale. When there are several series of superdense time segments, they are separated and therefore simpler to view and evaluate.

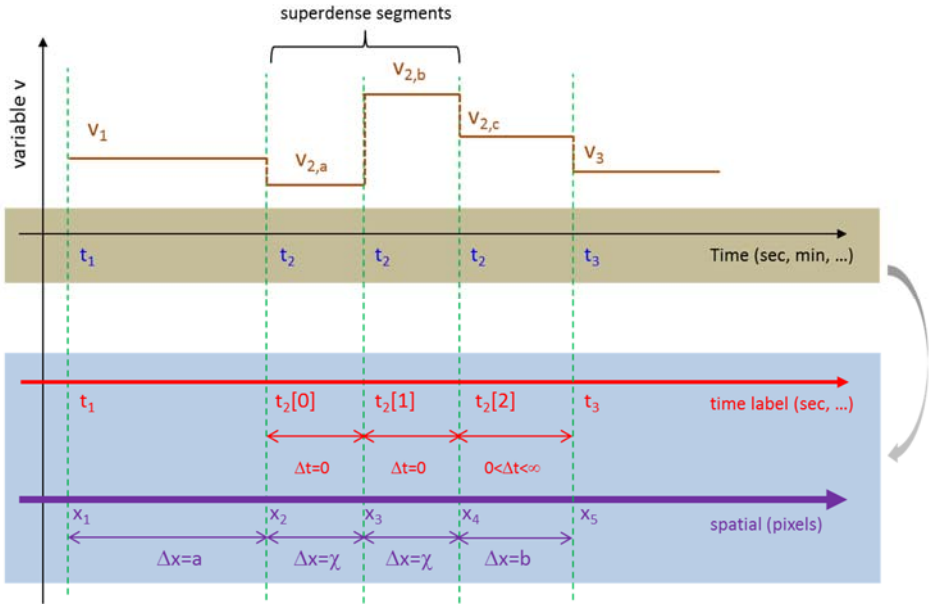

Figure 3. Combined linear and superdense time segments with their spatial counterparts.

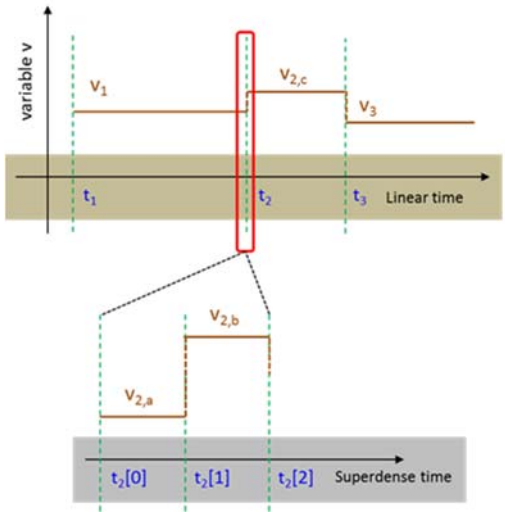

Figure 4. Stacked linear and superdense time segments.

Considering the positive and negative values, they too must be mapped to spatial representations. Since infinity needs to be plotted relative to a finite value set, it must be mapped to a number larger than the largest number in the set. If the value set is known a-priori, it is relatively simple to assign a finite value to infinity, which can be visually rendered well relative to the numbers (or strings) in the set. However, in a dynamic setting, the largest number cannot be known. Therefore, the (positive or negative) infinity value can be defined to be a percentage larger than the largest (positive or negative) value in the value set. If the range of values is very large (e.g., 1^{-7} to 1^{+19} and 1 to 1^{+3}), then the numerical infinity band (defined as the difference between the largest number in the set and the assigned value to infinity) can change dramatically. The data value set including the assigned finite value to infinity can be mapped to spatial axis rendered linearly in pixels.

CREATING MIXED LINEAR AND SUPERDENSE CHARTS

Several professional grade plotting tools exist for creating a different kinds of charts including line charts. Among them are the Business Intelligent and Reporting Tool [1,17], JFreeChart [6], and d3.js [3]. The Business Intelligence and Reporting Tool (BIRT) offers a flexible framework for extending or creating new kinds of charts. It supports reporting and business intelligence capabilities for rich client and web applications, especially those based on Java and J2EE. Main components for BIRT are its design, report, chart, and script engines. BIRT has report designer for creating BIRT reports within the Eclipse IDE [4] , Chart Builder, and Report Viewer. These can be deployed in any Java application. BIRT supports different kinds of charts such as Bar, Pie, Line, Scatter, and Gantt charts among others. One of its major capabilities is its flexible, rich chart customization wizard. As BIRT is built using the Eclipse plugin framework, one can extend the BIRT API using extension points to develop new chart types.

Piecewise constant and event charts

The approach taken for implementing the piecewise constant (shown in Figure 4) and event charts is to extend several of the BIRT plug-ins [16]. For example, the “data point definition” extension point from the Design engine is extended to process data sets and their mapping to pixels and creating superdense segments. The package for the piecewise constant is shown in Figure 5. Definitions for infinity bands and transitory state transitions among many others are

specified in the `PiecewiseConstantImpl` class. Other important capabilities (e.g., switching on and off display of superdense time segment) for visualization of complex content are also supported. Considering the rendering of the charts, Figure 6 illustrates use of the BIRT plug-ins. An important focus of these plug-ins is to display charts from arbitrary data sets that are generated dynamically. These and other classes and packages developed for Piecewise Constant charts are used for Event charts with appropriate modifications.

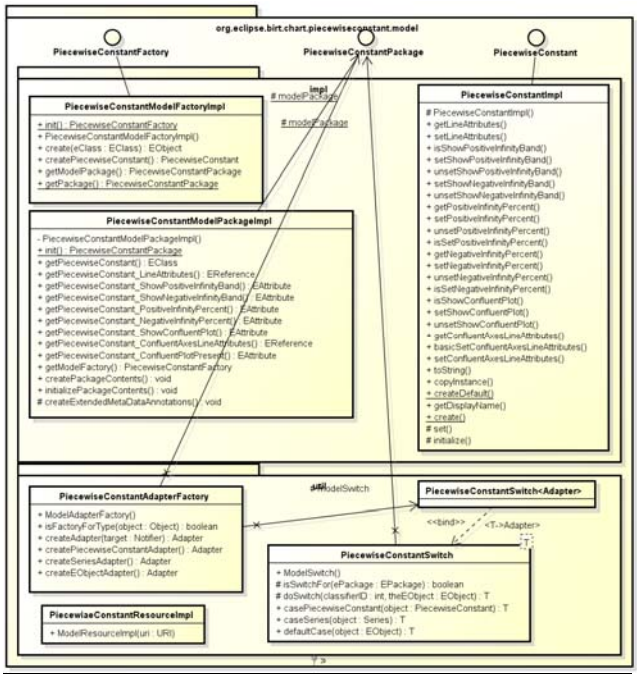

Figure 5. Classes specifying Piecewise Constant chart

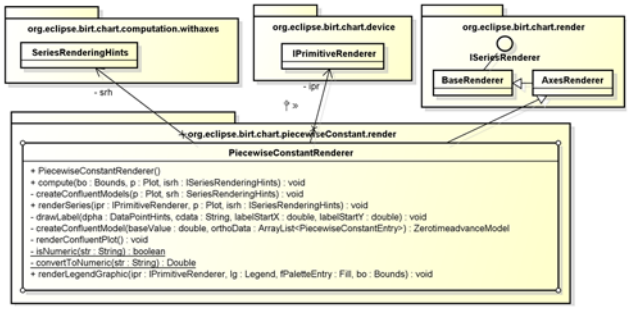

Figure 6. Classes for rendering Piecewise Constant chart.

Integration with the DEVS-Suite simulator

The DEVS-Suite simulator's charting engine has major limitations common to many classical and contemporary tools. Such simulation tools do not directly support the concept for superdense time, which is necessary for segments that have zero time duration.

The DEVS-Suite simulator supports generating run-time plots that have zero-time segments while allowing state, input, and output to have positive and negative infinity values.

The DEVS-Suite Modeling and Simulation tool, supporting Parallel DEVS modeling formalism, offers basic support for generating linear time Piecewise constant and Event chart types at run-time [8]. This simulator was built using the *Model-Façade-View-Control* (MFVC) architecture pattern [14]. The View component receives notification from the Model and can generate input, output, and state trajectories in tabular, piecewise constant and event charts. The piecewise constant (or event) chart consists of piecewise constant (event) segments. Similar to other classical and contemporary simulator tools such as Simulink [12] and Ptolemy II [13], superdense time segments and charts as defined in this paper are not supported. Given the separation of concerns in the MFVC, the *Timeview* of the DEVS-Suite simulator 2.1.0 is extended to support plotting piecewise constant and event charts with superdense time segments and positive and negative infinity bands at run-time. The top level design of the Timeview is illustrated in Figure 7 [16]. The Piecewise constant and event chart plugins that are developed in BIRT and BIRT's chart plug-in are used. The BIRT scripting is used in the Timeview. It affords configuring the attributes of the charts (e.g., label font and color, line type and thickness, tick marks, etc.). The scripting complements customization using BIRT's chart wizard.

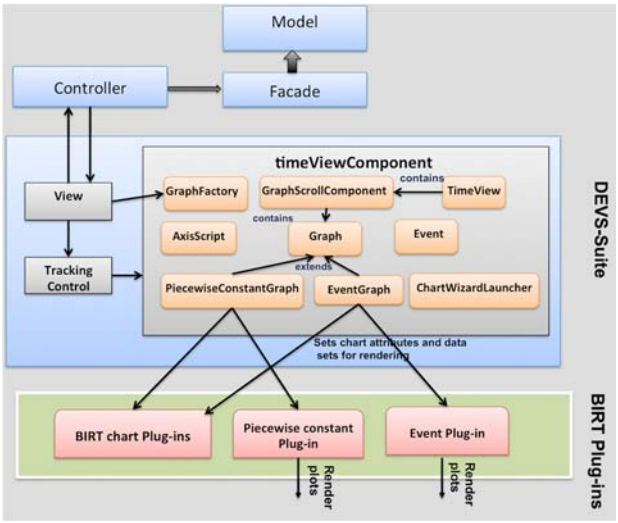

Figure 7. Illustration of integration of BIRT Piecewise Constant and Event Charts with DEVS-Suite's Timeview.

Exemplar linear and superdense time charts

As noted above, charts can have both linear with superdense time segments and infinity bands as shown in Figure 8. The *Phase* piecewise constant chart is generated from a model that undergoes transitory state transitions at time instances 10.0 and 20.0. In the linear time trajectory part, only the phase $send\_y$ is displayed. In contrast, in the superdense time trajectory, the phase $passive$ can also be viewed. At time instance 20.0, there are two contiguous transitory state transitions followed by a non-transitory state transition (phase $send\_out$ with $\sigma = 10.0$). The *sigma* piecewise constant chart shows the positive infinity band and at time

instance 30.0, phase is *passive* with $\sigma = \infty$. In Figure 9, event charts with simple and complex data types are shown.

All the elements in the piecewise constant (and event) chart can be customized. The switch for plotting superdense time segments is set to off and therefore events generated at time instances 10[0], 20[0] and 20[2] corresponding to the charts in Figure 8 are not shown. To avoid cluttering of the charts, the 20[0], 20[1], and 20[2] time instance labels are displayed as 20[0], [1], and [2].

The axes in the plots have default scales, but can be initialized before the simulator starts, or resized when the simulator is paused. The inputs, states, and outputs of the model components can be individually selected and tracked dynamically (see Figure 10). Furthermore, all plots for a component can be stacked as shown. This is useful to understand complex timings for state transitions, I/O events, separately in linear and superdense time to be aligned. Rudimentary linear time trajectories was implemented in Java AWT in DEVSJAVA [20], the predecessor to the DEVS-Suite simulator.

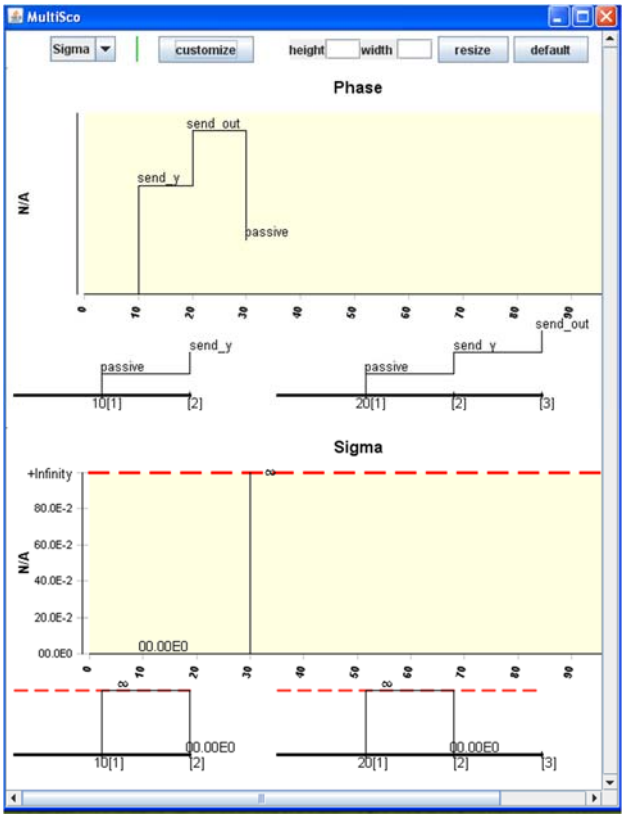

Figure 8. Piecewise Constant Charts with linear and superdense time segments.

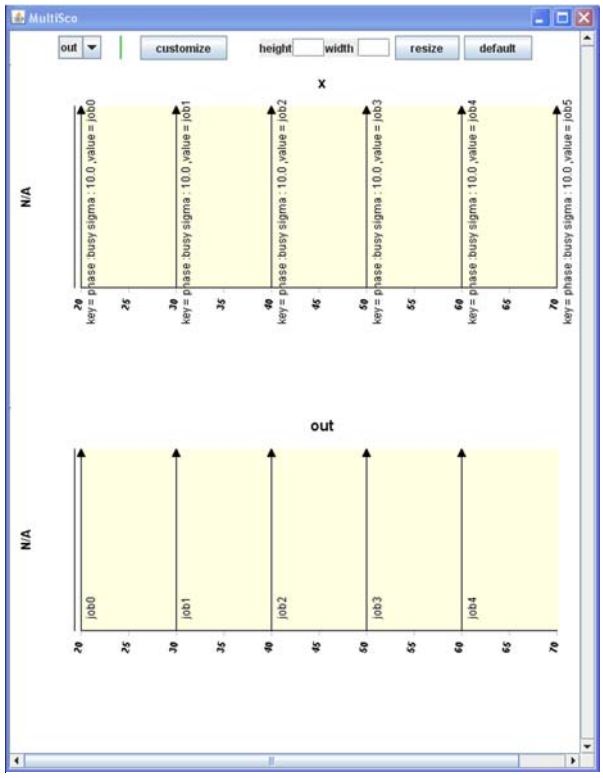

Figure 9. Event charts with linear time segments.

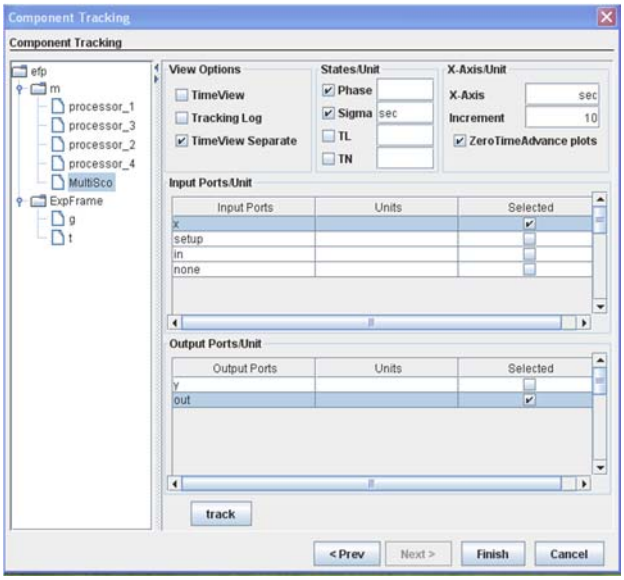

Figure 10. DEVS-Suite's Component tracking for Tracking Log and TimeView.

RELATED WORKS

Data visualization is crucial in many science and engineering areas. Techniques including data normalization, filtering, zooming, etc. are important for creating useful visual data representations such as 2D histograms [1]. Tools supporting time series that are commonly used in analysis applications do not support creating the kinds of charts developed in this

research. They are intended for understanding patterns, trends, etc. In contrast to these kinds of charts, for designing and experimenting with dynamical simulation models for concurrent systems, such charts are insufficient. For this reason tools such as D3 [3]. Other tools such as Graphite [7] are developed for real-time data storage with support for data visualization. Tools such as BIRT support 3D line charts where one axis can be used for time variable, one for continuous variable and one for discrete variable. Alternatively, two axes can be used for linear and superdense time with the remaining axis for a variable. One crucial difference is that 2D charts, unlike 3D charts, can be stacked and evaluated together. The concept of expanding a portion of the linear time axis with high resolution relates to our work. Users can zoom on any finite time segment of a 2D time series with high precision [9]. The spatial representation of a plot's time axis can be scaled uniformly unlike the superdense time axis with variable values that may have positive and negative infinity bands.

CONCLUSION

In this paper, we have described superdense time which is central for simulation models such as Parallel DEVS that exhibit concurrency. It is important to create trajectories that have superdense time segments. In this paper, we have detailed this concept with a formulation of it where visual representation of superdense time is defined based on its mathematical specification. We developed a formulation for developing complementary linear and superdense time piecewise constant and event chart types with positive and negative infinity bands. These can be used for visualizing input, output, and state trajectories for DEVS atomic models. They are implemented using the BIRT framework. As BIRT plug-ins, they can be used in RCP and Java applications. As part of the BIRT tool, the piecewise constant and event charts defined as $v \times time$ data sets (for example stored as CSV files) can be plotted as static charts in the BIRT Design Report. These plug-ins are used in the Timeview module of the DEVS-Suite simulator. Time series are dynamically plotted alongside I/O animation and tabular log files. These piecewise constant and event trajectories can be customized which is crucial when simulation data to be visualized is complex, numerous, and may not necessarily be known a-priori. Future work includes supporting other kinds of chart types such as continuous piecewise and $\{v_1, \cdots, v_q\} \times time$ and integrating the BIRT Design Report with the DEVS-Suite simulator. Another area of research is to offer greater degree of customization resulting in greater data modality and fewer data trajectories.

ACKNOWLEGEMENT

We are grateful to the anonymous referees for their helpful reviews.

REFERENCES

[1] BIRT 4.4, (2014), http://www.eclipse.org/birt/.
[2] Chow, A.C., Zeigler, B.P., (1994), "Parallel DEVS: a parallel, hierarchical, modular modeling formalism," Winter Simulation Conference, 1994.
[3] Data Driven Documents, (2013), https://ds3js.org.
[4] D'Anjou, J., Fairbrother, S., Kehn, D., Kellerman, J., & McCarthy, P., (2005), *The Java Developer's Guide to ECLIPSE* (2nd ed.), Addison Wesley.
[5] DEVS-Suite simulator 2.1.0, (2009), http://devs-suitesim.sourceforge.net.
[6] JFreeChart, (2013), http://www.jfree.org/index.html.
[7] Graphite, (2011), http://graphite.readthedocs.org.
[8] Kim, S., Sarjoughian, H. S., Elamvazuthi, V., (2009), "DEVS-suite: A Simulator Supporting Visual Experimentation Design and Behavior Monitoring," Proceedings of the 2009 Spring Simulation Multi-conference, San Diego, CA.
[9] Kincaid, R., (2010), "SignalLens: Focus+Context Applied to Electronic Time Series," IEEE Transactions on Visualization and Computer Graphics, vol.16, no.6.
[10] Lee, E.A., et al., (2014), *System Design, Modeling, and Simulation using Ptolemy II*, http://ptolemy.org.
[11] Manna, Z., Pnueli, A., (1993), *The Temporal Logic of Reactive and Concurrent Systems*, Springer, Berlin.
[12] Mathworks, Simulink, (2011), http://www.mathworks.com/products/simulink.
[13] PtolemyII, (2010), http://ptolemy.eecs.berkeley.edu/.
[14] Sarjoughian, H.S., Singh, R., (2003), "Building simulation modeling environments using systems theory and software architecture principles," Advanced Simulation Technology Conference, SCS, Wash. DC.
[15] H. S. Sarjoughian and V. Elamvazhuthi, "CoSMoS: a visual environment for component-based modeling, experimental design, and simulation", in Proceedings of the 2nd international conference on simulation tools and techniques, 2009.
[16] Sundaramoorthi, S., (2014), "Eclipse BIRT Plug-ins for Dynamic Piecewise Constant and Event Time Series," Master's Thesis, School of Computing, Informatics and Decision System Engineering, Arizona State University, Tempe, AZ.
[17] Weatherby, J., Bondur, T., & Chatalbasheva, I., (2011), *Integrating and Extending BIRT* (3rd ed.).
[18] Wymore, A.W., (1993), *Model-Based Systems Engineering*, CRC Press, Boca Raton.
[19] Zeigler, B.P., (1976), *Theory of Modelling and Simulation*, Wiley Interscience, New York.
[20] Zeigler, B.P., Sarjoughian, H.S., Au, V., (1997), "Object-oriented DEVS: object behavior specification", Proceedings of Enabling Technology for Simulation Science, Orlando, FL.