Dynamic Structure Multiparadigm Modeling and Simulation

FERNANDO J. BARROS
Universidade de Coimbra

This article presents the Heterogeneous Flow System Specification (HFSS), a formalism aimed to represent hierarchical and modular hybrid flow systems with dynamic structure. The concept of hybrid flow systems provides a generalization of the conventional concept of hybrid system and it can represent a whole plethora of systems, namely: discrete event systems, multicomponent and multirate numerical methods, multirate and multicomponent sampling systems, event locators and time-varying systems. The ability to join all these types of models makes HFSS an excellent framework for merging components built in different paradigms. We present several examples of model definition in the HFSS formalism and we also exploit the ability of the HFSS formalism to represent mutirate numerical integrators.

Categories and Subject Descriptors: I.6.1 [**Simulation and Modeling**]: Simulation Theory—*systems theory*

General Terms: Design, Experimentation, Performance

Additional Key Words and Phrases: Dynamic Structure systems, hybrid systems, multirate sampling, variable step integration

1. INTRODUCTION

Hybrid systems are commonly defined as systems exhibiting both continuous and discrete behavior. While discontinuities are represented by discrete event systems, the continuous behavior is usually described by differential equations. There has been an intense research in these types of systems, and examples can be found in Alur et al. [2001], Antsaklis [2000], Barros [2002a], and Branicky and Mattson [1997].

Although differential equations are widely used for representing continuous systems, they are not exclusive; for example, control and signal areas use a representation based on sampling. Besides the traditional approaches based on single-rate sampling, there has been research on multirate systems. Numerical

This work was partially funded by the Portuguese Science and Technology foundation under Project PRAXIS/EEI/14152/98.

Author's address: Departamento de Engenharia Informática, Universidade de Coimbra, Pólo II, PT-3030 Coimbra, Portugal; email: barros@dei.uc.pt.

methods for solving differential equations also rely on sampling, and multirate integration has been subject of research [Engstler and Lubich 1997; Howe 1998].

To achieve an unified representation of all these types of systems, we have developed the *Heterogeneous Flow System Specification* (HFSS) [Barros 2002a]. This formalism relies on time-varying and multicomponent sampling for providing a common representation of all kinds of continuous systems in digital computers. Multirate numerical methods for solving differential equations can also be represented in the HFSS formalism. This formalism also provides support for discrete event systems and event detection. The HFSS formalism provides, thus, a unified framework for representing digital control, signal processing, numerical integration and hybrid systems. Because all these types of systems share the same underlying representation, they can be arbitrarily connected to build complex models.

In addition, the HFSS formalism offers the ability to represent systems with a time-varying structure. Examples of these types of systems include mobile agent systems [Barros 2001] and switching systems [Barros 2003].

To simplify the specification of systems of Ordinary Differential Equations (ODEs), we present the *Differential Equation Discrete Flow System Specification* (DEDF) formalism, which is aimed to directly represent hybrid systems described by ODEs and discontinuities. This formalism can encode multirate integration methods.

Traditional modeling formalisms for continuous or hybrid systems are based on ideal representation of continuous signals. These modeling approaches are based on analogue or hybrid state machines that have a counterpart in analogue and hybrid computers. Examples include modeling formalisms for differential equations [Zeigler 1976] and for hybrid systems [Praehofer 1991]. These modeling approaches, while being powerful abstractions for describing systems, do not provide the necessary constructs to represent continuous systems in digital computers. Or, in other terms, we can say that these approaches are pure *modeling* formalisms in opposition to *modeling and simulation* formalisms. A different situation occurs in the discrete event field where modeling and simulation formalisms, like the DEVS formalism [Zeigler 1976], were earlier created, providing not only modeling constructs but also their representation in digital computers. The simulation relationship was formalized in the realm of discrete event systems by the concept of abstract simulator [Zeigler 1984], a construct to extract model dynamic behavior.

Recently, there has been a growing interest in developing modeling and simulation formalisms to represent hybrid systems in digital computers [Barros 2002a; Zeigler and Lee 1998]. These formalisms permit to discuss the efficiency of the numerical simulators that represent continuous variables, and also, and perhaps more importantly, they enable model interoperability. This last characteristic cannot be discussed in the realm of pure modeling formalisms. The realization of pure modeling formalisms is made in an *ad hoc* basis. This casuistic implementation renders impossible to achieve model interoperability for there is no universal underlying discrete state machine that supports their

implementation. This is particularly pertinent when events detectors need to be used [Praehofer 1991].

While implementation of discrete state machines in digital computers is a trivial task, the implementation of continuous systems in digital computers has proven to be a challenging one. Actually, the opposite situation would probably arise if modern computers were analog. In this case, continuous systems would have a trivial implementation while discrete event systems would need some kind of, albeit nontrivial, approximation. This situation has changed with the creation of the Continuous Flow System Specification (CFSS) [Barros 2002b], which has established the use of multicomponent and multirate sampling for representing continuous signals.

The unification of sampled based systems with discrete event systems was made by the HFSS formalism [Barros 2002a] that has introduced a new paradigm for simulating hybrid systems in digital computers. Although sampling, especially single-rate sampling, has been extensively used in many fields like control and signal processing, a unified modeling formalism has not been created before. The HFSS provides sounds semantics for representing and simulating discrete and continuous signals on digital computers. For example, it is possible to describe numerical integrators as part of the formalism and not as an external construct that needs to be incorporated into a pure modeling formalism so it can be realizable in a digital computer. HFSS precise semantics enable the interoperability of hybrid systems in a similar manner existing for discrete systems.

To illustrate the use of the HFSS, we present examples of hybrid systems with both static and time-varying structure modeled in the CHAOSTALK environment, a Smalltalk implementation of the HFSS formalism.

2. HETEROGENEOUS FLOW SYSTEMS

The Heterogeneous Flow System Specification (HFSS) is a formalism intended to represent piecewise constant partial state systems that accept and produce both continuous and discrete input flows. The HFSS achieves the representation of continuous flow systems based on sampling [Barros 2002a]. Discrete event representation is based on the DEVS formalism [Zeigler 1976]. The HFSS formalism can be used as a basis for the representation of numerical integration methods aimed to solve differential equations, as we showed in the Section 2.3.

2.1 HFSS Basic Model

A Heterogeneous Flow System Specification (HFSS) is defined by

$$HFSS = (X, Y, S, \rho, \tau, q_0, \delta, \Lambda_c, \lambda),$$

where

$X = X_c \times X_d$ is the set of input flow values
X_c is the set of continuous input flow values
X_d is the set of discrete input flow values

$Y = Y_c \times Y_d$ is the set of output flow values
Y_c is the set of continuous output flow values
Y_d is the set of discrete output flow values
S is the set of partial states (p-states)
ρ: $S \to \mathbf{R}_0^+$ is the time to input function
τ: $S \to \mathbf{R}_0^+$ is the time to output function
$Q = \{(s, e) | s \in S, 0 \leq e \leq \nu(s)\}$ is the state set and
$\nu(s) = \min\{\rho(s), \tau(s)\}$ is the time to transition function
$q_0 = (s_0, e_0) | \in Q$, is the initial state
$\delta : Q \times (X_c \times X_d^{\phi}) \to S$ is the transition function
where
$X_d^{\phi} = X_d \cup \{\phi\}$ and
ϕ represents the absence of value
$\Lambda_c : Q \to Y_c$ is the continuous output function
$\lambda : S \to Y_d^{\phi}$ is the partial discrete output function

The discrete output function, $\Lambda_d : Q \to Y_d^{\phi}$, is defined by

$$\Lambda_d(s, e) = \begin{cases} \lambda(s) & \text{if} \quad e = \tau(s) \\ \phi & \text{otherwise} \end{cases}$$

The output function, Λ:$Q \to Y_c \times Y_d^{\phi}$, is defined by

$$\Lambda(q) = (\Lambda_c(q), \Lambda_d(q))$$

Figure 1 represents typical trajectories of a HFSS component. At time t_0, the component is in state (s_0,e_0) when it receives a discrete input x_{d_0}.

It changes then to the p-state $s_1 = \delta(s_0, e_0, (x_{c_0}, x_{d_0}))$. During the interval $\rho(s_1)$, no discrete input arrives and, at time $t_1 = t_0 + \rho(s_1)$, the system changes to p-state $s_2 = \delta(s_1, \rho(s_1), (x_{c_1}, \phi))$, where x_{c_1} is the value of the continuous flow at time t_1. At p-state s_2, the time to input function is equal to the time-to-output function and the time-to-transition is scheduled to time $t_2 = t_1 + \nu(s_2)$. During this interval, there is no discrete flow, but the discrete value x_{d_2} arrives at the end of the interval. The component changes at time t_2 to p-state $s_3 = \delta(s_2, \nu(s_2), (x_{c_2}, x_{d_2}))$. The time-to-transition function is now equal to the time-to-input function. The component is scheduled to change at time $t_3 = t_2 + \tau(s_3)$. The component changes then to p-state $s_4 = \delta(s_3, \tau(s_3), (x_{c_3}, \phi))$ because there is no discrete flow. The continuous output is always defined whereas the discrete output is only nonnull at times t_2 and t_3, when the elapsed time equals the time to output function δ.

The HFSS provides a general framework for modeling arbitrary hybrid flow systems in digital computers that, contrarily to analog computers, can only deal with a representation based on piecewise constant p-states. The simulation of HFSS basic models can be made using the abstract simulator described in Barros [2002c].

Example 1. *Triangular Wave Generator.* Consider a triangular wave generator that produces discrete outputs at wave extreme values. The generator is described by four parameters: maximum value ($vMax$), minimum value ($vMin$);

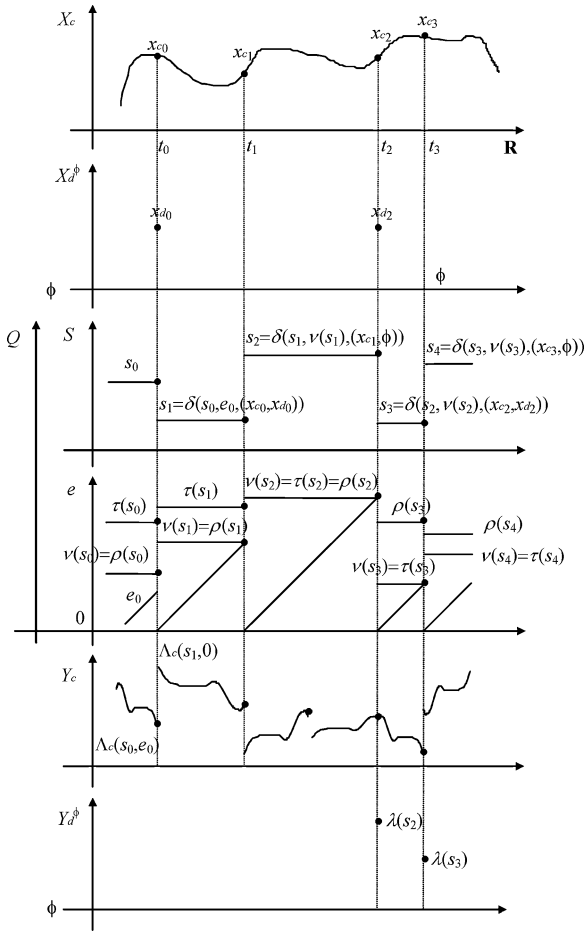

Fig. 1. HFSS trajectories.

time to reach the maximum value (tUp) and time to reach the minimum value ($tDown$). This generator can be described by

$$T = (X, Y, S, \rho, \tau, q_0, \delta, \Lambda_c, \lambda),$$

where

$X = \{\} \times \{\}$
$Y = \mathbf{R} \times \mathbf{R}$
$S = \{(phase, beta, vMin, vMax, tUp, tDown) |$
$\quad phase \in \{\#\text{down}, \#\text{up}\}; vMin, vMax \in \mathbf{R};$
$\quad beta, tUp, tDown \in \mathbf{R}^+\}$ and $vMin < vMax$
$\rho(phase, beta, vMin, vMax, tUp, tDown) = \infty$
$\tau(phase, beta, vMin, vMax, tUp, tDown) = beta$
$q_0 = (\#\text{down}, down, min, max, up, down), 0)$
$\delta((\#\text{down}, beta, vMin, vMax, tUp, tDown), e, (\phi, \phi)) =$
$\quad (\#\text{up}, tUp, vMin, vMax, tUp, tDown)$

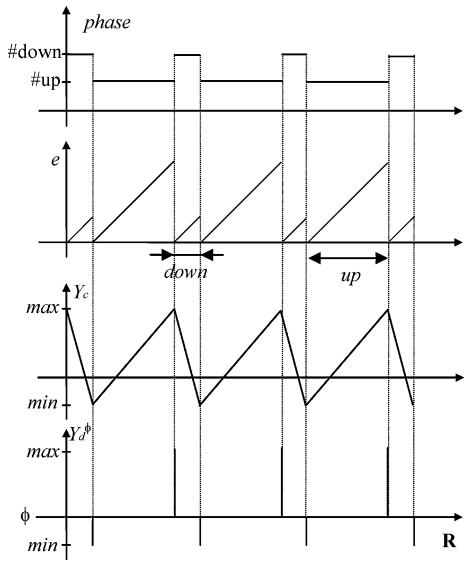

Fig. 2. Triangular wave generator trajectories.

$\delta((\#\text{up}, beta, vMin, vMax, tUp, tDown), e, (\phi, \phi)) =$
$(\#\text{down}, tDown, vMin, vMax, tUp, tDown)$
$\Lambda_c((\#\text{down}, beta, vMin, vMax, tUp, tDown), e) =$
$vMax + e(vMin - vMax)/tDown$
$\Lambda_c((\#\text{up}, beta, vMin, vMax, tUp, tDown), e) =$
$vMin + e(vMax - vMin)/tUp$
$\lambda(\#\text{down}, beta, vMin, vMax, tUp, tDown) = vMin$
$\lambda\ (\#\text{up}, beta, vMin, vMax, tUp, tDown) = vMax$

Figure 2 represents generator output trajectories for the parameters set in the initial state q_0. Generator initial value is *max* and its initial phase is #down. During time *down*, generator continuous output flow decreases from *max* to *min*. At the end of the *down* time interval, the discrete output of value *min* is produced. The component then changes its phase to #up and, during time *up*, it produces a linear-increasing output value. At the end of this interval, the discrete output value *max* is produced.

The continuous output flow of the generator is the triangular wave and the discrete output flow corresponds to the extreme values of the wave. Every component connected to this generator can sample the triangular wave at their own sampling rate, for the generator output is continuous, whereas they all receive the information of wave extremes values conveyed by the generator.

Example 2. *Proportional Controller.* A first-order proportional controller with a fixed sampling rate of R seconds can be described by the structure

$$P = (X, Y, S, \rho, \tau, q_0, \delta, \Lambda_c, \lambda),$$

where

$X = \mathbf{R} \times \{\}$
$Y = \mathbf{R} \times \mathbf{R}$
$S = \{(phase, r, x_1, x_0) | \ phase \in \{\#\text{run}, \#\text{out}\}; r \in \mathbf{R}^+; x_1, x_0 \in \mathbf{R}\}$
$\rho(phase, r, x_1, x_0) = r$
$\tau(\#\text{out}, r, x_1, x_0) = 0$
$\tau(\#\text{run}, r, x_1, x_0) = \infty$
$q_0 = (\#\text{run}, R, 0, 0), R)$
$\delta((\#\text{run}, r, x_1, x_0), e, x) = (\#\text{out}, r, x_0, x)$
$\delta((\#\text{out}, r, x_1, x_0), e, \_) = (\#run, r, x_1, x_0)$
$\Lambda_c((r, x_1, x_0), e) = x_0 + e \times (x_0 - x_1)/r$
$\lambda(phase, r, x_1, x_0) = x_0$

The controller is a hybrid system that receives continuous input flows and produces both continuous and discrete output flows. It samples the input at every time interval specified by variable r, and it keeps the current and the past input values at variables x_0 and x_1, respectively. The output is piecewise linear, and it is computed from the previous two samples. When a new sample is read, the new control level does generally correspond to the predicted control level and a discontinuity at the output signal occurs. Thus, the controller sends a discrete flow to signal the discontinuity. Hybrid integrators connected to the controller use this signal to perform correct computations over discontinuities.

For simplicity, we have considered that the proportional factor equals to the unit. We have also simplified controller initial conditions considering current and past values of the controller to be zero.

From the example, it is evident that changes in the sampling rate can be trivially achieved by changing variable r within the transition function.

2.2 HFSS Network Model

HFSS networks are an arbitrary composition of HFSS components. Hierarchical composition is a key to represent complex systems by allowing a representation based on small components that can be independently developed and tested [Barros 1998]. Time-varying structure systems are also better represented by dynamic structure models. In this case, dynamic structure models offer a more intuitive representation of reality for they are able to mimic the dynamic creation and destruction of entities, and the dynamic nature of the relationship existing among entities within a system [Barros 1997]. Formally, a Heterogeneous Flow System Specification Network is a 4-tuple

$$HFN_N = (X_N, Y_N, \eta, M_\eta),$$

where

N is the network name
  $X_N = X_{cN} \times X_{dN}$ is the set of input flow values
  X_{cN} is the set of continuous input flow values
  X_{dN} is the set of discrete input flow values
$Y_N = Y_{cN} \times Y_{dN}$ is the set of output flow values

Y_{cN} is the set of continuous output flow values
Y_{dN} is the set of discrete output flow values
η is the name of the dynamic structure network executive
M_η is the model of the executive η

The model of the executive is a modified HFSS, defined by

$$M_\eta = (X_\eta, Y_\eta, S_\eta, \gamma, \Sigma^*, \rho_\eta, \tau_\eta, q_{0,\eta}, \delta_\eta, \Lambda_{c\eta}, \lambda_\eta),$$

where

Σ^* is the set of network structures
$\gamma.Q_\eta \to \Sigma^*$ is the structure function

The network structure $\Sigma_{j,e} \in \Sigma^*$, corresponding to the state $(s_{j,\eta}, e) \in Q_\eta$, is given by the 4-tuple

$$\Sigma_{j,e} = \gamma(s_{j,\eta}, e) = (D_j, \{M_{i,j,e}\}, \{I_{i,j}\}, \{Z_{i,j,e}\}),$$

where

D_j is the set of component names associated with the executive state $q_{j,\eta}$ for all $i \in D_j$

$M_{i,j,e}$ is the model of component i
$I_{i,j}$ is the set of components influencers of i
$Z_{i,j,e}$ is the input function of component i

For simplicity, we assume here that the models and input functions do not change with executive elapsed time e. Thus, $M_{i,j,e} = M_{i,j}$, $Z_{i,j,e} = Z_{i,j}$, and $\gamma.S_\eta \to \Sigma^*$. An example of a continuous variation of the input function is described in Barros [2000].

These variables are subject to the following constraints for every $s_{j,\eta} \in S_\eta$:

$\eta \notin D_j$
$N \notin I_{N,j}$
$M_{i,j} = (X_{i,j}, Y_{i,j}, S_i, \rho_i, \tau_i, q_{0,i}, \delta_{i,j}, \Lambda_{c_{i,j}} \lambda_{i,j})$ is a basic HFSS model, for all $i \in D_j$, with $\delta_{i,j} : Q_i \times (X_{c_{i,j}} \times X_{d_{i,j}^\phi}) \to S_i$
$Z_{N,j} : \underset{k \in I_{N,j}}{\times} Y_{k,j} \to Y_N$

$$Z_{i,j} : \underset{k \in I_{i,j}}{\times} V_{k,j} \to X_{i,j}, \quad \text{for all} \quad i \in D_j \cup \{\eta\},$$

where

$$V_{k,j} = \begin{cases} Y_{k,j} & \text{if} \quad k \neq N \\ X_N & \text{if} \quad j = N \end{cases}$$

The equivalence between HFSS networks and atomic models is established in Barros [2002a]. The abstract simulator for HFSS networks can be found in Barros [2002c].

2.3 Differential Equation/Discrete Flow System Specification (DEDF)

The HFSS formalism provides a general framework for representing systems with both discrete and continuous input/output flows. The formalism provides

a direct representation of systems based on sampling, and it is well suited for representing digital control and digital signal systems. However, such a direct representation can be cumbersome when specifying ODEs. The HFSS formalism can be used as a framework to create other formalisms to represent ODEs with discontinuities. These systems are commonly known as hybrid or combined systems. We create the *Differential Equation/Discrete Flow System Specification* (*DEDF*) to describe hybrid systems using a combination of ODEs and discrete event systems. Although the DEDF can be reduced to the HFSS formalism, it allows a direct representation of ODEs; thus, it offers a more friendly syntax for specifying some type of systems. For simplicity, we describe a 1st order version of the DEDF formalism. However, higher order methods can be developed. A *DEDF* is defined by

$$DEDF = (X, Y, S_d, \rho, \tau, q_0, \delta_d, f, \lambda),$$

where

$X = X_c \times X_d$ is the set of input flow values
$X_c \subseteq \mathbf{R}$, is the set of continuous input flow values
X_d is the set of discrete input flow values
$Y = Y_c \times Y_d$ is the set of output flow values
$Y_c \subseteq \mathbf{R}$, is the set of continuous output flow values
Y_d is the set of discrete output flow values
S_d is the set of discrete partial states
$S = X_c \times Y_c \times S_d$ is the set of partial states
$\rho : S \to \mathbf{R}_0^+$ is the time to input function
$\tau : S \to \mathbf{R}_0^+$ is the time to output function
$Q = \{(s, e) | s \in, 0 \leq e \leq \min\{\rho(s), \tau(s)\}\}$ is the state set
$q_0 = (s_0, e_0) \in Q$, is the initial state
$\delta_d : Q \times (X_c \times X_d^{\phi}) \to S$ is the discrete transition function
$f : X_c \times Y_c \to \quad Y_c$ is the derivative function
$\lambda : S \to Y_d^{\phi}$ is the partial discrete output function

We consider that both input and output continuous flows are based on the set of real numbers $\mathbf{R}$. This is not a limitation, for the ultimate goal is to use multirate methods where each variable is independently integrated. An extension to vectors of variables in $\mathbf{R}^n$ would be straightforward.

The time-to-transition function is used to compute the time-to-read the next input when no discrete values occur. Due to the modularity of the formalism, each model can have its own time-to-transition function and thus its own integration time step [Barros 2000]. The discrete transition changes the model p-state in the presence of discontinuities. For combining DEDF and HFSS models, we need to establish their equivalence. A DEDF $= (X, Y, S_d, \rho, \tau, q_0, \delta_d, f, \lambda)$ is equivalent to the HFSS $= (X, Y, S, \rho, \tau, q_0, \delta, \Lambda_c, \lambda)$, where

$$S = X_c \times Y_c \times S_d.$$

The continuous output function just assumes a linear interpolation and is given by

$$\Lambda_c((x, y, s), e) = y + e.f(x, y).$$

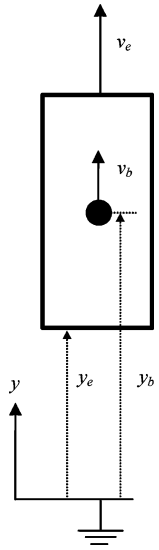

Fig. 3. Ball and elevator coordinate system.

The transition function uses first order Euler integration method for updating continuous values and it is defined by

$$\delta((x, y, s), e, (x_c, x_d)) = \delta_d((x_c, y + e.f(x, y), s), e, (x_c, x_d)).$$

Since DEDF models can be reduced to HFSS models, both types of models can be arbitrarily connected.

The creation of formalisms to represent other numerical methods, like an adaptive step-size 3rd order method used later in this article, can be made within the HFSS paradigm that offers, thus, a framework for mutirate numerical integration.

3. HYBRID SYSTEMS

Hybrid systems are commonly defined as systems ruled by ordinary differential equations (ODEs) except at points when states variables change abruptly. They can thus be viewed as a combination of ODEs and discrete event systems. Many types of hybrid systems have been described in the literature [Alur et al. 2001; Antsaklis 2000; Barros 2002a; Branicky and Mattson 1997; Deshpande et al. 1997; Praehofer 1991]. To illustrate the HFSS representation of hybrid systems, we consider a simple system composed by a bouncing ball confined to a 1-dimensional moving elevator of 2.5 m height. The ball is launched from a height of 0.4 m relative to the elevator ground, and with a null velocity relative to the ground. The ball bounces every time it reaches the elevator limits and it loses part of the relative velocity in every collision. The ball and elevator coordinate system is represented in Figure 3, where v_e and y_e represent elevator velocity and position relative to the ground and v_b and y_b represent ball velocity and position relative to the ground.

The ball can hit the elevator in two positions: ground and ceiling. For modeling these collisions, we use two detectors one for each type. The ground collision is defined as:

$$D1 \equiv y_e - y_b = 0.$$

For an elevator with 2.5 m height, the collision with the ceiling is defined by

$$D2 \equiv y_e - y_b - 2.5 = 0.$$

Before collision, the relative velocity is given by

$$v_r = v_e - v_b.$$

After a collision, the elevator velocity remains the same, but the relative velocity is given by

$$v_r' = v_e - v_b' = -kv_r,$$

where v_b' represents the ball velocity after the collision, and k, $0 < k \leq 1$ models the loss of relative velocity due to the collision. After collision, ball velocity is thus given by

$$v_b' = (1 + k)\, v_e - kv_b.$$

The HFSS model of the collision component is given by

$$C = (X, Y, S, \rho, \tau, q_0, \delta, \Lambda_c, \lambda),$$

where

$X = \mathbf{R}^2 \times \{\#\text{detect}\}$
$Y = \{\phi\} \times \mathbf{R}$
$S = \{(phase, k, out) |\ phase \in \{\#\text{run}, \#\text{out}\};\ k \in \mathbf{R}^+;\ out \in \mathbf{R}\}$
$\rho(phase, k, out) = \infty$
$\tau(\#\text{out}, k, out) = 0$
$\tau(\#\text{run}, k, out) = \infty$
$q_0 = (\#\text{run}, k = 0.95, \phi), 0)$
$\delta((\#\text{run}, k, out), e, (<v_b, v_e>, d)) = \#\text{out}, k, (1 + k) \times v_e - k \times v_b)$
$\delta((\#\text{out}, k, out), e, \_) = \#\text{run}, k, \phi)$
$\Lambda_c((phase, k, out), e) = \phi$
$\lambda(phase, k, out) = out.$

The collision receives a pair of input values corresponding to the ball and elevator velocity and the additional value d from one of the detectors.

The ball is a free-fall body whose position is given by

$$y_b'' = -g,$$

where g = 9.807 m/s^2 is the standard gravity acceleration at Earth's surface. The elevator position is given by

$$y_e = y_0 + \int v_e\, dt,$$

where y_0 is the elevator initial position and v_e is the elevator velocity (considered here to be a known function). Figure 4 represents the HFSS model of the ball—elevator system.

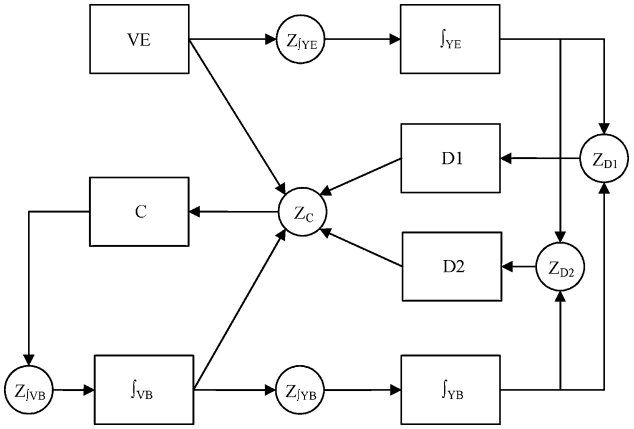

Fig. 4. Bouncing ball and elevator block diagram.

All the elements in the block diagram have an HFSS equivalent. Integrators are based on a DEDF like formalism that implements a 3rd order adaptive step-size integrator. Detectors operate using both variable sampling and the bisection method to detect zero-crossing. The velocity of the elevator is given by the triangular wave generator described in Section 2.1 that it is also HFSS equivalent. This equivalence is central to component integration and new integrators and detectors can be added as long as their equivalence to the HFSS formalism can be defined. For example, higher-order methods can be easily merged with the described models. Thus, the HFSS formalism provides a standard to model interoperability for it does not depend on any particular implementation.

Figure 5 illustrates the results produced in CAOSTALK, a Smalltalk implementation of the HFSS formalism. Elevator velocity has a maximum of 10 m/s and a minimum of -10 m/s. Time up and time down equal to 3 s. Elevator height is 2.5 m. We consider that initially the elevator ground is at position 0 m and that the ball starts at position 1 m with a null velocity. In every collision, the ball loses 5% of the relative velocity ($k = 0.95$). Figure 5 depicts ball and elevator positions. A third curve represents the ball height relative to the elevator ground. The complex behavior of the ball depends on the relative velocity at collision times and, thus, sometimes the ball hits the elevator ceiling, while other times the ceiling is not reached.

The velocities of the elevator and the ball are depicted in Figure 6. The discontinuities in ball velocity corresponding to the collisions are also represented in the figure.

Simulation was performed using a variable-step numerical integrator whose steps are independent and change accordingly to the local error. Figure 7 represents the time steps taken by the integrating that computes ball position.

Small time steps are used after collisions so the third order integrator can operate. Between collisions, the time steps can be quit large, keeping, however, the local truncation error within prescribed bounds. The possibility to change the integration step can lead to more efficient integration for, in general, the integrators will achieve the same prescribed accuracy using different step sizes.

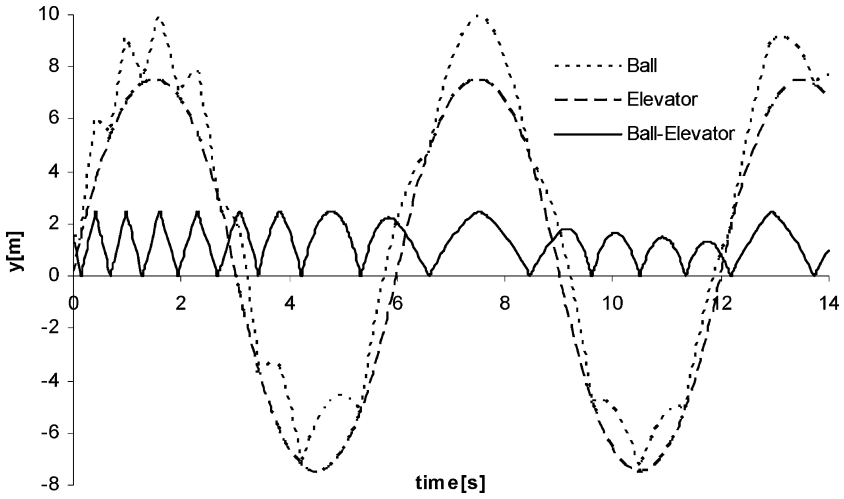

Fig. 5. Ball and elevator position.

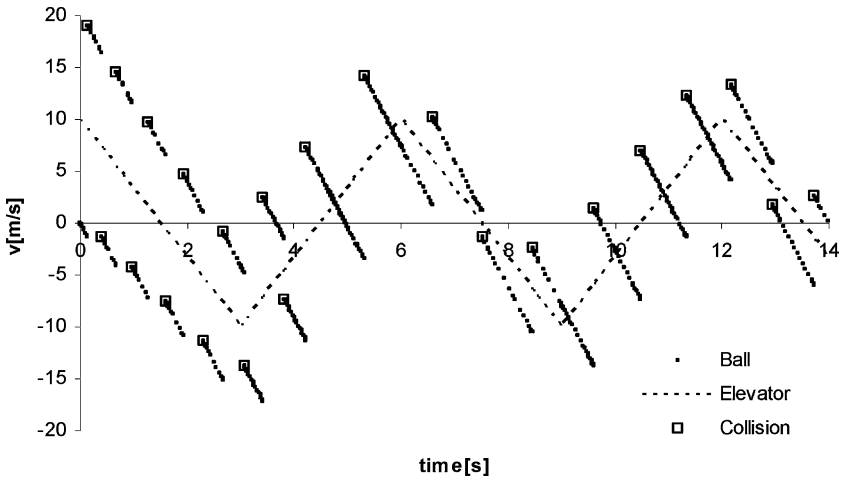

Fig. 6. Ball and elevator velocity.

4. HYBRID DYNAMIC STRUCTURE SYSTEMS

The hybrid systems considered so far have a static structure. We consider now the bottle-filling system of Figure 8 which is modeled as a hybrid system with a dynamic structure.

The filling system is composed by the filler F and the sensor S. Bottles are inserted below the filler by a conveyor. When the bottle arrives its capacity is read and the filler starts to fill the bottle. The scale below the bottle senses bottle volume weight and signals when the bottle is filled. When this happens, the conveyor removes it from the filler and brings a new empty bottle.

The HFSS model of the described system is represented in Figure 9. The filler is modeled by component P, a 1st-order proportional controller. The scale

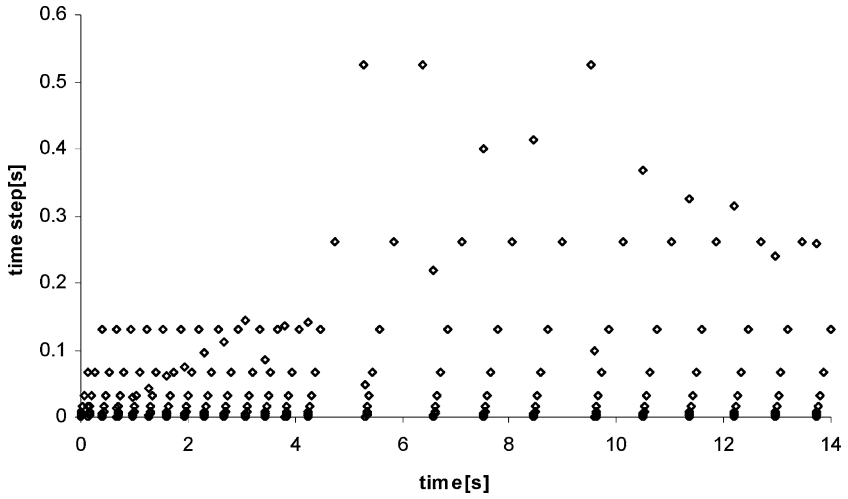

Fig. 7. Time steps for ball position integrator.

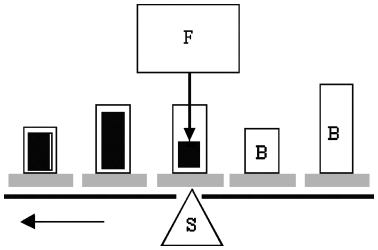

Fig. 8. Bottle filling system.

is modeled by the detector D. The bottle is represented by an active component that can give information about its current volume. The conveyor is represented within the executive model that can add or remove bottles to the system. When a bottle is currently being filled, the model structure is represented by Figure 9. The bottle is an integrator that receives as input the filling rate from the controller P.

The detector D signals the executive when bottle is filled. The executive then removes the bottle, and the network structure becomes represented by Figure 10. After conveyor transport time, the executive inserts a new bottle and the model structure becomes again represented by Figure 9. Structural changes correspond to the insertion of empty bottles and the removal of filled bottles. Given that each bottle represents a differential equation, structural changes correspond to modifications in the set of equations describing the model.

The controller P is described in Section 2 and it has a fixed sampling rate of 0.1 s. Controller input function is given by

$$Z_C = (\max - b) + 0.05,$$

where max is the bottle maximum volume, b is the bottle current volume and 0.05 cl/s is the minimum filling rate. Controller output is depicted in Figure 11

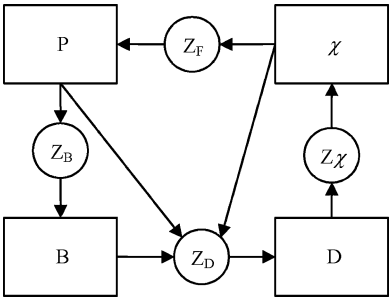

Fig. 9. Filling a bottle.

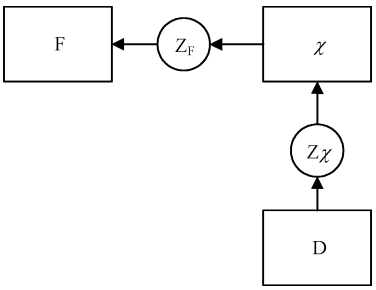

Fig. 10. No bottle to fill.

for a sequence of 4 bottles. During the bottle transit time of 2 s, the controller output is zero.

The volume of a sequence of 4 bottles is given at Figure 12. This value is computed by a hybrid integrator based on a variable-time-step 3rd-order algorithm. This algorithm needs to reinitialize every time the input has a discontinuity. Thus, time steps are reset whenever the controller issues a discrete flow value. This situation occurs every 0.1 s, at controller sampling points. This figure reflects also the dynamic structure model approach employed. The absence of volume value indicates that currently no bottle is being filled corresponding to a model without the integrator.

This example, although very simple, demonstrates the ability of the HFSS formalism to represent models usually seen as belonging to *different paradigms*, namely: digital control, hybrid numerical integrators, event detectors and dynamic structure models. The HFSS approach allows us to perceive these different paradigms as particular cases of hybrid flow systems.

5. CONCLUSIONS

The HFSS formalism provides a general representation of hybrid systems in digital computers based on a broad utilization of the sampling concept. Both time-varying and multirate sampling can be used. Numerical integration methods based on multirate and adjustable step can be described in the DEDF formalism. The equivalence between HFSS and DEDF formalism allows us to merge both models in the same network. The ability to change network

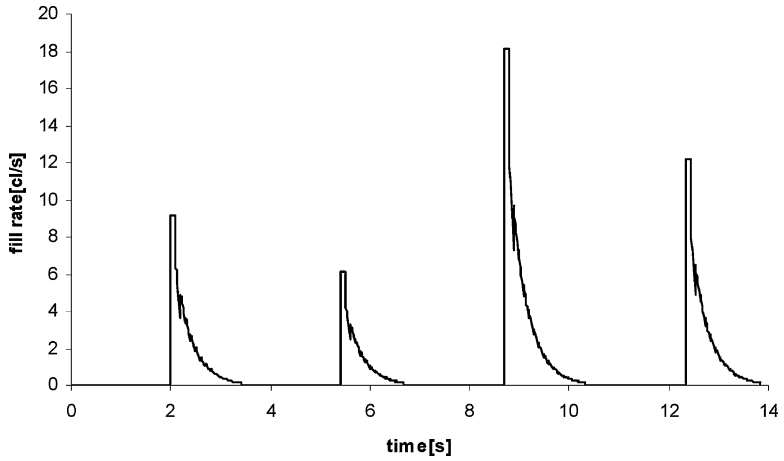

Fig. 11. Controller output.

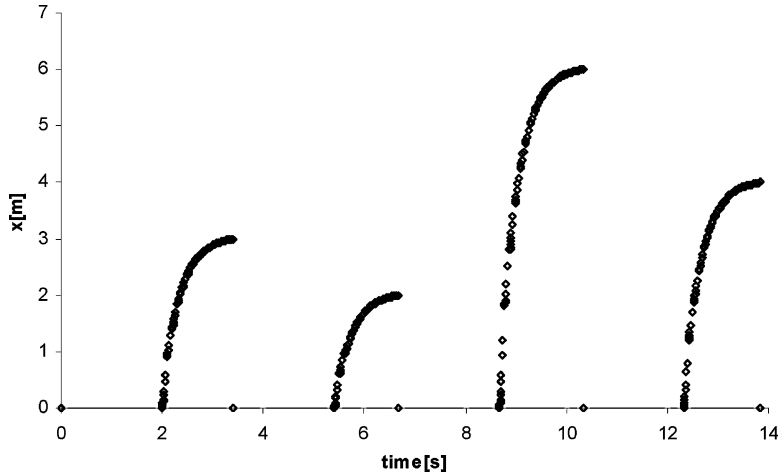

Fig. 12. Bottle volume.

structure can lead to more efficient and more understandable models. This feature is of crucial importance to represent very complex models. The HFSS formalism provides a comprehensive framework for combining models built in different paradigms. It is a general modeling formalism that can represent namely: dynamic structure systems, hybrid flow systems, multirate numerical methods and multirate sampling systems.

REFERENCES

ALUR, R., GROSU, R. LEE, I., AND SOKOLSKY, O. 2001. Compositional refinement for hierarchical hybrid systems. In *Hybrid Systems: Computation and Control. Proceedings of the 4th International Conference (HSCC'01)*. Lecture Notes in Computer Science, vol. 2034. Springer-Verlag, New York, 33-48.

ANTSAKLIS, P. J. 2000. A brief introduction to the theory and application of hybrid systems. *Proc. IEEE 88*, 7, 879–887.

BARROS, F. J. 1997. Modeling formalisms for dynamic structure systems. *ACM Trans. Model. Comput. Simulat. 7*, 4, 501–515.

BARROS, F. J. 1998. Hierarchical testing of dynamic structure models: A practical approach. *Trans. SCS 15*, 4, 181–189.

BARROS, F. J. 2000. A Framework for representing numerical multirate integration methods. In *Proceedings of the 2000 AI, Simulation and Planning in High Autonomy Systems* (Tucson, Az.). SCS International, 149–154.

BARROS, F. J. 2001. Modeling and simulation of mobile software agents in chaos. In *Proceedings of the European Simulation Symposium/DEVS Workshop* (Marseille, France). SCS International, 605–610.

BARROS, F. J. 2002a. Modeling and simulation of dynamic structure heterogeneous flow systems. *Simulat. Trans. SCS 78*, 1, 18–27.

BARROS, F. J. 2002b. Towards a theory of continuous flow models. *Int. J. Gen. Syst. 31*, 1, 29–39.

BARROS, F. J. 2002c. Abstract simulators for dynamic structure hybrid components. In *Proceedings of the AI, Simulation and Planning in High Autonomy Systems*. SCS International, 71–77.

BARROS, F. J. 2003. Modeling and simulation of switched systems: A dynamic structure approach. In *Proceedings of the 2003 Summer Computer Simulation Conference* (Montreal, Ont., Canada). SCS International, Simulation Series, *35*, 3, 426–431.

BRANICKY, M. S. AND MATTSON, S. E. 1997. Simulation of hybrid systems. In *Hybrid Systems IV*, P. J. Antsaklis, W. Korn, A. Nerode, and S. Sastry, Eds. Lecture Notes in Computer Science, vol. 1273. Springer-Verlag, New York, 31–56.

DESHPANDE, A., GOLLU, A., AND SEMENZATO, L. 1997. The shift programming language and run-time system for dynamic networks of hybrid automata. In *Proceedings of the 1997 NATO Workshop on Discrete Event and Hybrid Systems*. Lecture Notes in Computer Science. Springer-Verlag, New York.

ENGSTLER, C. AND LUBICH, C. 1997. Multirate extrapolation methods for differential equations with different time scales. *Computing 58*, 173–185.

HOWE, R. M. 1998. Real-time multi-rate asynchronous simulation with single and multiple processors. In *Proceedings of SPIE 12th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls: Enabling Technology for Simulation Science*, vol. 3369 (Orlando, Fla.). SPIE, 331–342.

PRAEHOFER, H. 1991. System theoretic foundations for combined discrete-continuous system simulation, Ph.D. Dissertation, Department of Systems Theory and Information Engineering, Univ. Linz.

ZEIGLER, B. P. 1976. *Theory of Modelling and Simulation*, Wiley, New York.

ZEIGLER, B. P. 1984. *Multifacetted Modeling and Discrete Event Simulation*. Academic Press, Orlando, Fla.

ZEIGLER, B. P. AND LEE, J. S. 1998. Theory of quantized systems: Formal basis for DEVS/HLA distributed simulation environment. In *Proceedings of SPIE 12th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls: Enabling Technology for Simulation Science*, vol. 3369 (Orlando, Fla.). SPIE, 49–58.

Received December 2001; revised June 2002, November 2002, April 2003; accepted June 2003