MODELING AND SIMULATION OF DYNAMIC TOPOLOGY MODELS: AN OVERVIEW OF APPLICATIONS

Fernando J. Barros
Department of Informatics Engineering
University of Coimbra
P-3030 Coimbra, Portugal
E-mail: barros@dei.uc.pt

ABSTRACT

Many systems undergo changes in their topology. The ability to mimic these changes into systems representations produces models that are easier to understand and to maintain when compared with the corresponding static topology models. The Heterogeneous Flow System Specification (HFSS) is a formalism able to represent hybrid systems with a dynamic topology. HFSS supports the representation of topologies that can reconfigure themselves with the addition/removal of components and their interconnections. In this paper we present several systems modeled with dynamic topologies. We consider the representation of switched systems, mobile components, and unbounded cellular automata. We also provide the HFSS representation of spatially moving entities, a particular class of models where dynamic topologies can provide an effective representation.

1 INTRODUCTION

Many systems modify the topology during their lifetime. An adequate representation requires the ability to map these changes into systems models. Such representations lead, generally, to more intuitive and easier to maintain models. Examples of such systems include sensor networks **[11**], reconfigurable computers **[12**], fault tolerant systems **[8**], electrical circuits **[5**], forest fire propagation, manufacturing systems **[16**], **[18**], biological systems **[17**], forest fire propagation **[20**], artificial neural networks **[19**], and robot team formation **[13**]. Models with a dynamic topology enabling the creation/destruction of entities and their relationships offer, thus, an excellent framework for representing these kind of dynamic systems.

The Heterogeneous Flow System Specification (HFSS), is a modeling and simulation formalism able to represent continuous/discrete systems with a time-varying topology. The formalism defines changes in a broad manner that includes the topology modification in both composition and coupling. Changes in HFSS model topologies are made incrementally making it possible to represent models that evolve through a virtually unbounded set of topologies, since it does not require building all possible topologies in advance.

In this paper we provide an overview of modeling and simulation with dynamic topology models through the presentation of several examples that illustrate families of systems. In Section 3 we present a filling system where bottles are modeled as components that can be connected/unconnected to/from a filling controller. In Section 4 we present an electrical circuit that is modeled as a switched system where its current equivalent model depends on the state of a diode. Section 5 describes the use of dynamic topologies to represent virtually unbounded cellular automata. This representation exploits the HFSS ability to represent the active cells to describe very large automata. Section 6 presents a space vehicle with the ability to update the velocity controller. This update is achieved through the use of a particular form of topology adaptation involving component mobility. The last example is presented in Section 7 that models spatially mobile entities through a series of topology adaptations involving the creation of components to detect the interactions when entities are within some vicinity. This kind of system has been commonly represented using publish/subscribe communication and supported by simulation frameworks like the High Level Architecture (HLA) **[14**]. HFSS enables a representation based on peer-to-peer communication, promoting models that are easier to develop. Spatially moving entities are illustrated with the simulation of two airborne radars in the proximity of one aircraft.

2 THE HFSS FORMALISM

We briefly provide a description of the *Heterogeneous Flow System Specification* (HFSS), a formalism that provides a representation for hybrid systems with a dynamic topology. The formalism defines two types of models: basic and network. The latter is a composition of both basic and other network models. HFSS combines generalized sampling **[2]** with discrete events **[21]** to achieve a description of hybrid systems. HFSS capability of topology adaptation is based on the concept of dynamic structure/topology network **[1]**.

2.1 HFSS Basic Model

Formally, a HFSS basic model is defined by

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

where

$X = Xc \times Xd$ is the set of input flow values

Xc is the set of continuous input flow values

Xd is the set of discrete input flow values

$Y = Yc \times Yd$ is the set of output flow values

Yc is the set of continuous output flow values

Yd is the set of discrete output flow values

S is the set of partial states (p-states)

ρ: $S \rightarrow \mathbf{H}_0^+$ is the time-to-input function

τ: $S \rightarrow \mathbf{H}_0^+$ is the time to output function

with $\mathbf{H}$ the set of hyperreal numbers

$Q = \{(s, e) | \ s \in S, 0 \le e \le \nu(s) \ \}$ 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

δ: $Q \times (Xc \times Xd^{\phi}) \rightarrow S$ is the transition function where

$Xd^{\phi} = Xd \cup \{\phi\}$ and

ϕ represents the absence of value

Λc: $Q \rightarrow Yc$ is the continuous output function

λ: $S \rightarrow Yd^{\phi}$ is the partial discrete output function

The discrete output function, Λd: $Q \rightarrow Yd^{\phi}$, is defined by

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

The output function, Λ: $Q \rightarrow Yc \times Yd^{\phi}$, is defined by

$\Lambda(q) = (\Lambda c(q), \Lambda d(q))$

Basic models permit the representation of sampling and discrete events. Sampling is explicitly modeled by the time-to-input function ρ. A basic model specifies a continuous/discrete output flow and thus it can be used to represent continuous variables. The processing of continuous inputs is made through a sample-based representation, where the sampling rate can be independently specified by each basic component. HFSS basic models can be used to specify numerical integrators and detectors **[4]**, enabling a modular representation of hybrid systems. A detailed description of HFSS semantics can be found in **[3]**, **[9]**.

2.2 HFSS Network Model

HFSS networks are a 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. 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

$X_N = Xc_N \times Xd_N$ is the set of input flow values

Xc_N is the set of continuous input flow values

Xd_N is the set of discrete input flow values

$Y_N = Yc_N \times Yd_N$ is the set of output flow values

Yc_N is the set of continuous output flow values

Yd_N is the set of discrete output flow values

η is the name of the dynamic topology 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 topologies

γ: $Q_\eta \rightarrow \Sigma^*$ is the topology function

The network topology $\Sigma_{j,e} \in \Sigma^*$, 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

HFSS networks have their topology specified by the executive η. Changes in the executive state are mapped into changes in network topology through the topology function γ. These changes include the ability to add/remove components and their interconnections.

HFSS is currently implemented in JUSE **[10]**, a programming language based on independent and reusable software units, a concept inspired in General Systems modeling formalisms **[1]**.

3 DYNAMIC TOPOLOGIES

To show the ability to adapt network topology, we consider the bottle filling system of Figure 1, **[4]**.

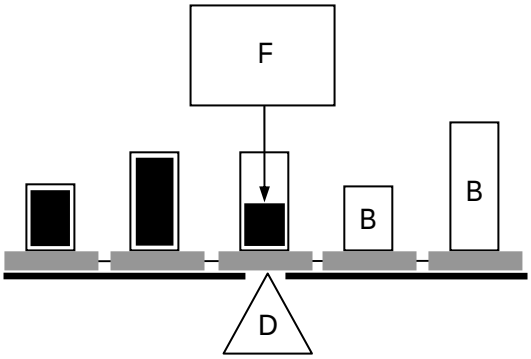

Figure 1**.** Bottle filling system.

The filling system is composed by the filler **F** and the detector **D**, a scale that detects filled bottles. Bottles are inserted bellow 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 the bottle volume 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 this system is represented in Figure 2, where rectangles represent HFSS components and circles represent input functions. An arrow indicate a component influencee. The filler is modeled by component **F**, a digital proportional controller. The scale 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 modeled in the executive that can add or remove bottles to the system. When a bottle is currently being filled the model topology is represented by Figure 2. The bottle is an integrator that receives as input the filling rate from the controller **F**.

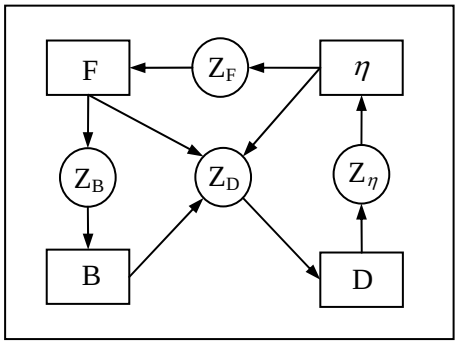

Figure 2. Filling a bottle.

The detector **D** signals the executive when the bottle is filled. The executive then removes the bottle and the network topology becomes represented by Figure 3. After conveyor transport time the executive inserts a new bottle and the model topology becomes again represented by Figure 2. Changes in topology correspond to the insertion of empty bottles and the removal of filled bottles. Given that each bottle represents a differential equation, these changes correspond to modifications in the set of equations describing the model.

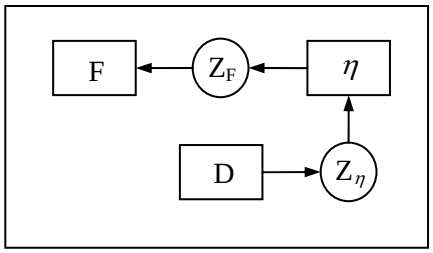

Figure 3. No bottle to fill.

The volume of a sequence of 4 bottles is given at Figure 4. This figure reflects the dynamic topology approach employed. The absence of volume value indicates that currently no bottle is being filled corresponding to a model without the bottle integrator.

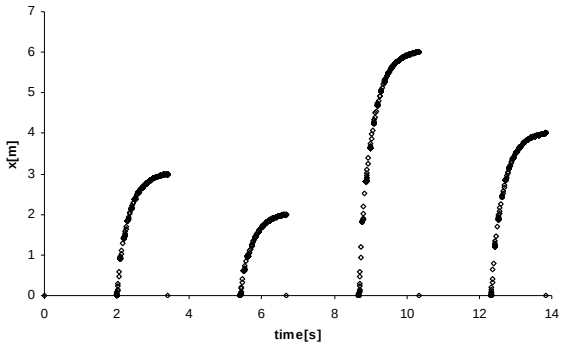

Figure 4. Bottle volume.

4 SWITCHED SYSTEMS

Switched systems can be broadly defined as systems represented by a set of models. At each moment a switched system has just one active model depending of the current system conditions. For example, electrical circuits with diodes will have different models for each combination of open/closed components. In this case the substitution of model is determined by changing the condition of every component. Whenever one of these elements changes it status, a new circuit is used.

The HFSS provides a natural framework for representing switched systems. A simple representation of these systems would require developing a network executive that would choose the adequate model for every situation. However, a better representation can be achieved if models are developed modularly, for in this case we can take the advantage of just handling the variations between the current and the new model. Models can thus become simpler and more efficient for the common components and interconnections remain. Keeping some components while changing the overall model requires also a more reduced number of initialization operations.

To illustrate the dynamic topology approach to modeling and simulation of switched systems we consider the simple half rectifier circuit represented in Figure 5, **[5]**. This circuit is composed by a capacitor C, a resistor R and a diode D and it is connected to a resistor R_L. The diode is a non-linear electrical component and is modeled as a switch with two positions (open and closed).

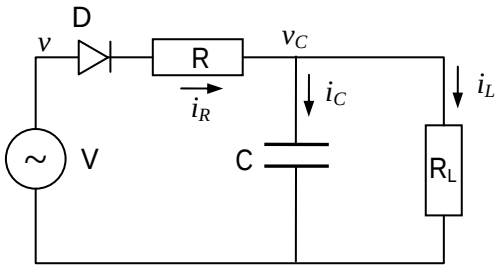

Figure 5. Half-wave rectifier.

Figure 6 represents the circuit when $v \geq v_C$ corresponding to the situation of a closed diode.

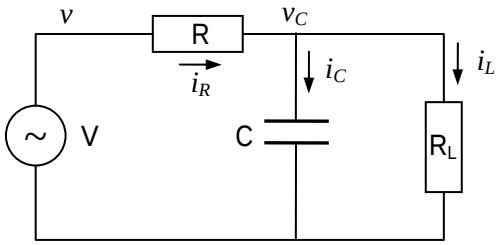

Figure 6. Equivalent circuit when $v \geq v_C$.

To allow the representation and simulation in the HFSS formalism the electrical circuit needs to be translated to a canonical form. The corresponding HFSS network is represented in Figure 7.

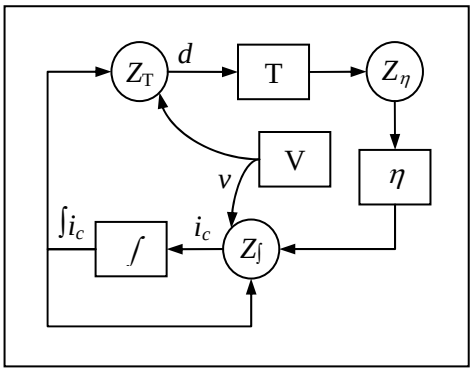

Figure 7. Simulation model for closed diode.

The equations that describe the circuit are given by:

$i_C = v/R - (R + R_L)\cdot\int i_C /(C\cdot R\cdot R_L)$

$v_C = \int i_C/C$

$i_L = v_C/R_L$

$d = v - v_C$

The diode D is replaced by a detector component T that is responsible to signal when the diode com-

mutes its state. This detector receives the difference $v - v_C$, and informs the executive when this value is 0. The executive then commutes to the circuit of Figure 8 that corresponds to the open diode.

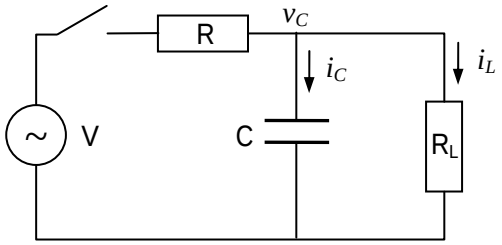

Figure 8. Equivalent circuit when $v < v_C$.

This circuit has a simulation model represented in Figure 9 which current in the capacitor i_C is given by:

$$i_C = -\int i_C/(C \cdot R_L)$$

The new model can be obtained from the previous one by just coding the differences. Thus all components remain in the state existing just before the model has changed. The difference from the previous network occurs in function Z_T that computes the voltage *d*.

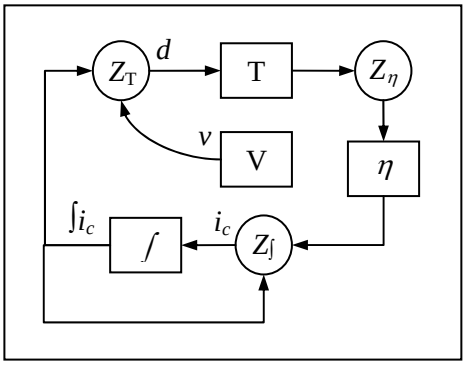

Figure 9. Simulation model for open diode.

Changes of topology from a closed to an open diode involve the removal of the source V as an influencer of the integrator (∫) and changing the corresponding input function accordingly. The initial topology corresponding to the closed diode, is obtained by adding the source V to the influencers set of the integrator (∫).Simulation results are represented in Figure 10.

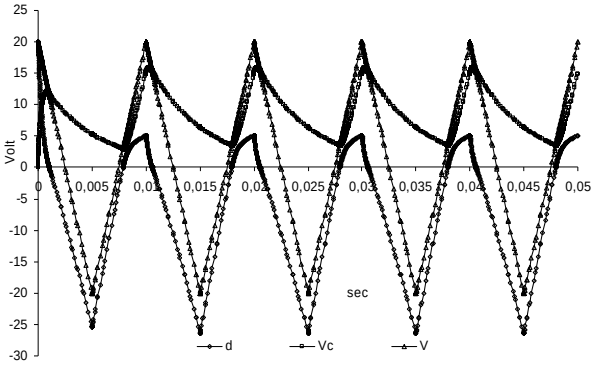

Figure 10. Voltages in the half-wave rectifier circuit.

5 UNBOUNDED CELLULAR AUTOMATA

Dynamic topologies can offer an efficient representation of very large systems. This is the case, for example, of potentially unbounded cellular automata used in the simulation of forest fire. For these systems, an efficient memory utilization can be achieved by keeping in the model only the cells that are currently active.

We consider a simple scenario of forest fire propagation in homogenous conditions using a dynamic topology cellular automata. Figure 11(a) shows that the simulation starts with no cells. At $t = 0^+$, one cell is ignited as demonstrated by Figure 11(b). At this point only one cell is active in the cellular automata. The simulation clock advances to the time of next event, $t = t_1$. At this time North, South, East and West cells are created as depicted in Figure 11(c). The first cell is still active because it can still start fire in other cells. The time of next event, the time when the remaining cells surrounding the starting cell are ignited, is $t = t_2$. At this time NE, NW, SE and SW cells are created and the starting cell is removed because all of its neighbors are burning, as depicted in Figure 11(d). With this procedure, the number of cells is only a fraction of the total number of cells of the overall model. Dynamic topologies enables thus the simulation of very large automata.

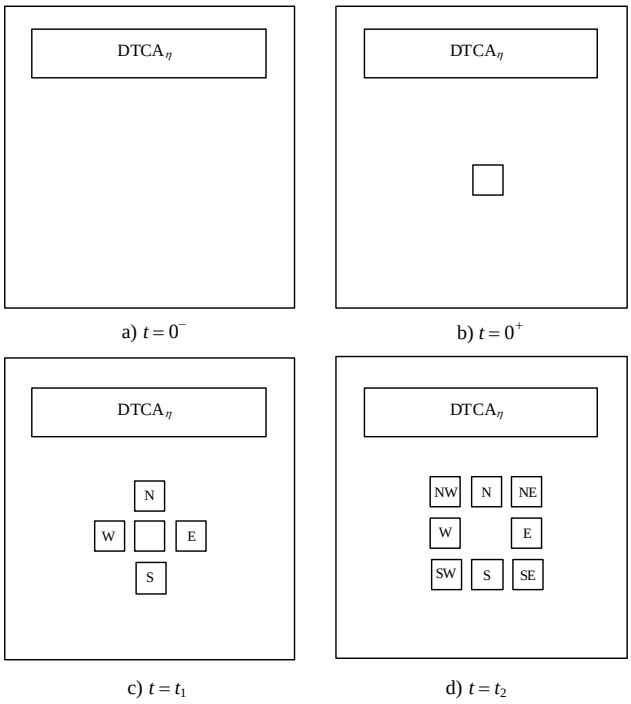

Figure 11**.** Dynamic topology cellular automata.

6 MOBILE MODELS

Mobility is a particular aspect of topology adaptation. It requires the ability to move a component between two networks. Mobility is a key operation to enable the modification of hierarchical models without breaking encapsulation. The network depicted in Figure 12 represents a system that includes a space vehicle **V** and a digital PID controller π_i **[7]**. The controller is responsible to keep vehicle velocity at a constant value v_R.

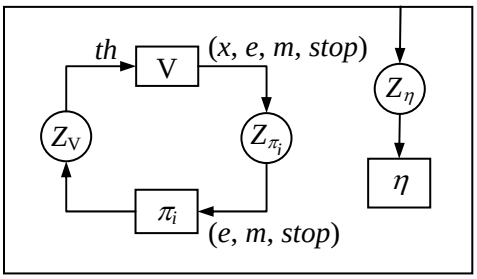

Figure 12. Digital controller and vehicle.

Vehicle dynamics are controlled by the model of Figure 13. This models comprises a 2nd order differential equation for describing vehicle position. Fuel consumption is taken into account and it is proportional to thrust *th*. A detector Δ is used to signal fuel depletion. When the network executive receives a mobile controller it replaces the current controller by the new one. The vehicle sends information about current velocity *v* and mass *m*. The controller receives the error $e = v - v_R$, vehicle mass and a discrete flow stop signal, and sets vehicle thrust.

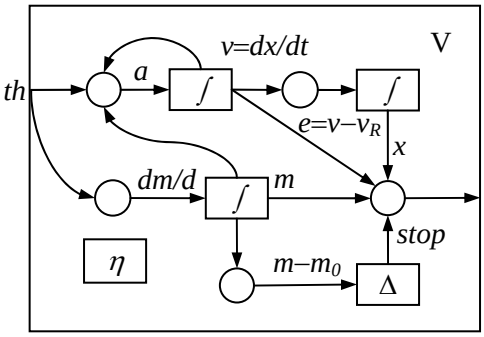

Figure 13**.** Space vehicle dynamics.

Mobility can be exploited to upgrade the controller while the vehicle is travelling in space. Two controllers were developed. Controller **A** is a PID with constant parameters. Controller **B** adjusts PID parameters according to vehicle mass. This controller was developed later and it was sent to the spaceship in order to upgrade controller **A**. This upgrade can achieved through component mobility supported by HFSS. Figure 14 depicts vehicle dynamics according to the different controllers. A mixed strategy, involving the replacement of controller **A** by a new controller **B** is also represented.

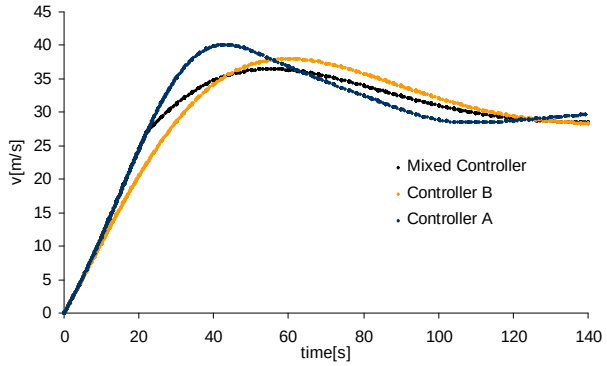

Figure 14**.** Vehicle velocity for the different combination of controllers.

Mobility permits to keep network encapsulation while enabling the access the network inner topology. Mobility plays thus an important role in the runtime adaptation of network components.

7 SPATIALLY DISTRIBUTED MODELS

In spatially distributed systems the interactions depend mainly on the entities' physical location. Since sensors, like radars or radio receivers, have a limited range, entities can only communicate when they are within some vicinity. When entities are too far apart they do not communicate. Thus communication depends on the physical location and implicitly defines a network with a dynamic topology that adapts according to the distance between moving objects. To illustrate an example involving moving entities we use a scenario with two airborne radars and one aircraft. Radar sweeping beam is modeled as a line moving at a constant angular speed. The aircraft is modeled as a point following some arbitrary trajectory. The simulation of radar echo is considered here as the problem to find the intersection of a moving line (the sweeping beam) with a moving point (aircraft). Both radar and aircraft can evolve following an arbitrary trajectory as described by Figure 15.

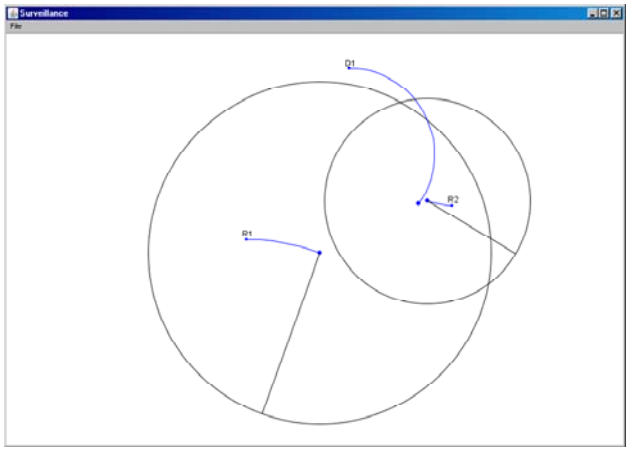

Figure 15. Radars range, beam and trajectory, and aircraft position and trajectory.

A detection signal is issued when the distance between the radar beam and the aircraft ≈ 0. We consider a scenario composed by two airborne radar R1 and R2 with ranges of 35 km and 15 km. respectively. The components that initially describe this system are depicted in Figure 16.

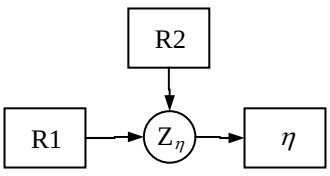

Figure 16. Initial configuration.

The network executive is responsible for handling aircraft creation and removal at random time intervals. We describe now the situation when one aircraft is present and out of radars range, as represented by Figure 17.

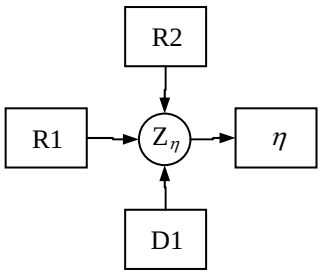

Figure 17. Aircraft creation.

When D1 becomes close to R1, the network becomes represented by Figure 18 that introduces component E1. E1 generates the interactions between the aircraft and radar R1, simulating the echoes produced by the radar.

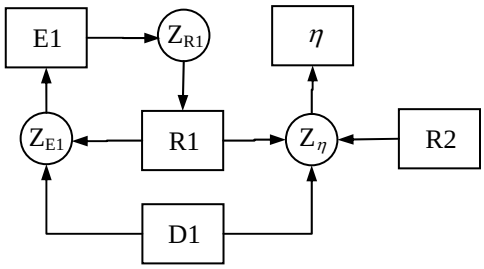

Figure 18. Aircraft becomes detected by R1.

When detected by R2, network topology becomes described by Figure 19.

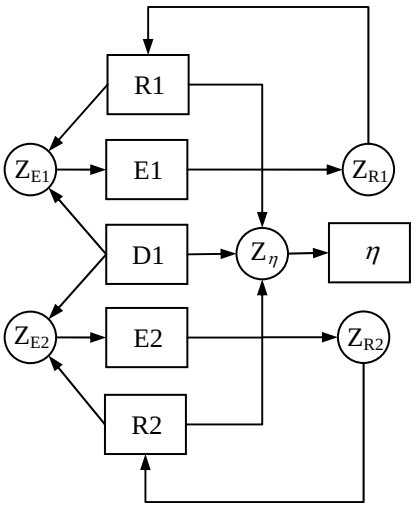

Figure 19. Aircraft detected by both R1 and R2.

Radars and the aircraft evolve following their own equation dynamics. The evolution includes changes in radar and aircraft positions, and the rotation of the radars' sweeping beams. The detailed representation of the radar beams plays a key role in determining the phase between radar echoes, enabling the fusion of radar data **[6]**. Radar echoes are represented in Figure 20.

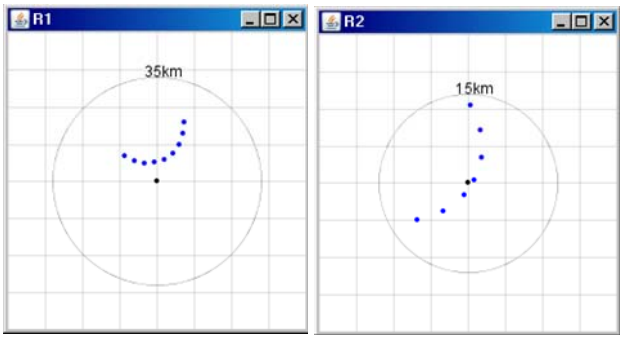

Figure 20**.** Aircraft echoes in radars R1 and R2.

The modular construction of simulation models permits the easy inclusion of new aircrafts and radars during a simulation run. This feature allows the representation of complex dynamic scenarios.

The common representation of moving entities uses publish/subscribe communication as used by the High Level Architecture **[14]**. HFSS provides an alternative representation based on peer-to-peer communication. Although we do not develop here a comparison between these two representations, we note that the HFSS enables the dynamic creation of entities, enabling simpler and more expressive models.

8 CONCLUSIONS

Dynamic topologies offer very expressive and flexible constructs to represent adaptive systems that undergo changes in their structure. The HFSS formalism provides a framework for modeling and simulation of hybrids models being able to represent continuous dynamic behavior along with discrete events that trigger changes in model topology. These changes have a broad scope and comprise the addition and removal of models and their interconnections during simulation runs. The HFSS formalism is thus able to represent arbitrary changes in model topology. The HFSS formalism can represent as particular cases families of models that are representative of a large class of systems. Examples include switched systems, spatial moving entities, mobile components and unbounded cellular automata. HFSS offers a unifying representation of these apparently unrelated systems, showing HFSS flexibility.

ACKNOWLEDGMENTS

This work was partially supported by the Portuguese Foundation for Science and Technology under project PTDC/EIA-EIA/100752/2008.

REFERENCES

[1] Barros, F.J. "Modeling Formalisms for Dynamic Structure Systems." *ACM Transactions on Modeling and Computer Simulation*, Vol. 7, No. 4, pp. 501-515, 1997.

[2] Barros, F.J. "Towards a Theory of Continuous Flow Models." *International Journal of General Systems*, Vol. 31, No. 1, pp. 29-39, 2002.

[3] Barros, F.J. "Modeling and Simulation of Dynamic Structure Heterogeneous Flow Systems." SIMULATION: *Transactions of the Society for Modeling and Simulation International*, Vol. 78, No. 1, pp. 18-27, 2002.

[4] Barros, F.J. "Dynamic Structure Multiparadigm Modeling and Simulation." *ACM Transactions on Modeling and Computer Simulation*, Vol. 13, No. 3, pp. 259-275, 2003.

[5] Barros, F.J. "Modeling and Simulation of Switched Systems: A Dynamic Structure Approach." *Summer Computer Simulation Conference*, pp. 259-275, 2003.

[6] Barros, F.J. "Fusion of Radar Data: A Modeling and Simulation Approach." *Proceedings of the International Conference on Radar Systems*, 2004.

[7] Barros, F.J. "A System Theory Approach to the Representation of Mobile Digital Controllers Agents." In *Innovative Concepts for Autonomic and Agent-Based Systems*, LNCS 3825, Springer, pp. 321-333, 2006.

[8] Barros, F.J. "An Evolving Hierarchical & Modular Approach to Resilient Software." *Workshop on Software Engineering for Resilient Systems*, pp. 79-86, 2008.

[9] Barros, F.J. "Semantics of Dynamic Structure Event-based Systems." *Distributed Event-Based Systems*, pp. 245-252, 2008.

[10] Barros, F.J. "Increasing Software Quality through Design Reuse." *7th International Conference on the Quality of Information and Communications Technology*, pp. 236-241, 2010.
[11] Cerpa, A., and Estrin, D. "ASCENT: Adaptive Self-Configuring Sensor Network Topologies." *IEEE Transactions on Mobile Computing*, Vol. 3, No. 3, pp. 272-285, 2004.
[12] Compton, K., and Hauck, S. "Reconfigurable Computing: A Survey of Systems and Software." *ACM Computing Surveys*, Vol. 34, No. 2, pp. 171-210, 2002.
[13] Hu, X., Zeigler, B.P., and Mittal, S. "Variable Structure in DEVS Component-Based Modeling and Simulation." *SIMULATION: Transactions of the SCS*, Vol. 81, No. 2, pp. 91-102, 2005.
[14] Kuhl, F., Weatherly, R. and Dahmann, J. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall, 1999.
[15] Mei, B., Lambrechts, A., Mignolet, J.Y., Verkest, D., and Lauwereins, R. "Architecture Exploration for a Reconfigurable Architecture Template." *IEEE Design and Test of Computers*, No. 2, 90-101, 2005.
[16] Pawletta, T., Pawletta, S., and Drewelow, W. "A DEVS-Based Approach for Modeling and Simulation for Hybrid Variable Structure Systems." *In Modeling, Analysis and Design of Hybrid Systems*, LNCS 279, Springer, pp. 107-129, 2002.
[17] Regev, A., Panina, E., Silverman, W., Cardelli, L., and Shapiro, E. "BioAmbients: An Abstraction for Biological Compartments." *Theoretical Computer Science*, 325, pp. 141-167, 2004.
[18] Shang, H., and Wainer, G. "A Simulation Algorithm for Dynamic Structure DEVS Modeling." *Proceedings of the Winter Simulation Conference*, pp. 815-822, 2006.
[19] Vahie, S., N. Jouppi, N. "Dynamic Neural Ensambles: Learning Using Variable Structures." *Proceedings of the Sixth Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, pp. 148-157, 1996.
[20] Vasconcelos, M.J., Guertin, P., and Zwolinski, M. "FIREMAP: Simulation of Fire Behavior, a GIS Supported System." *Proceedings of the Effects of Fire in Management of Southwestern Natural Resources Conference*, pp. 217-221, 1990.
[21] Zeigler, B.P. *Theory of Modeling and Simulation*. Kluwer, 1976.

AUTHOR BIOGRAPHY

Fernando J. Barros is an assistant professor at the University of Coimbra in Coimbra, Portugal. His research interests include theory of modeling and simulation, dynamic topology models, and software reuse based on hierarchical and modular approaches**.**