

Fuzzy Reinforcement Learning for System of Systems (SOS)

Hamid Berenji, IEEE Fellow

IIS Corp

MS: 566-108, NASA Research Park
Moffett Field, CA 94035

Mo Jamshidi, IEEE Fellow

University of Texas at San Antonio
ACE Center, ECEC Dept.
San Antonio, TX 78249

Abstract— The System of Systems (SOS) technology is an advanced technology for Intelligent Systems that is developed with multiple intelligent systems. Recently, there has been a growing interest in a class of complex systems (robotic swarm as an example) whose constituents are themselves complex. Performance optimization, robustness and reliability among an emerging group of heterogeneous systems in order to realize a common goal have become the focus of various applications including military, security, aerospace, space, manufacturing, service industry, environmental systems, and disaster management, to name a few. In this paper, we discuss how Fuzzy Reinforcement Learning (FRL) can be used in SOS.

Keywords— *System of Systems (SOS); Fuzzy Reinforcement Learning (FRL); robots; cooperation; Unmanned Ground Vehicles (UGVs)*

I. INTRODUCTION

This paper proposes SOS-Robotics, a software-hardware environment for planning and simulating a group of complex subsystems for management all within a System of Systems (SoS) environment. SOS-Robotics uses DEVS-Java for mathematically simulating the performance of a group of Unmanned Automated Systems (UAS) and assures the correctness of the group performance. The SOS-Robotics will be in demand by many of NASA and DOD organizations that have to manage the performance of groups of highly complex air, land, and undersea robots. SOS-Robotics generates plans for achieving a goal assigned by an operator and can quickly re-plan due to some unexpected events such as weather changes. It will also find many applications in group operations such as response to threats and robotic searches.

A. System of Autonomous Rovers

The system of autonomous rovers, at the IIS Corp-ACE Center, the University of Texas, San Antonio, comprises four components: Base Rover, Swarm Robots, Sensors and a Threat. The sensors represent a standalone system able to sense a Threat and ability to communicate with the Base Rover. The Threat is spatially dynamic and it is assumed that it is detectable by the chosen sensors. Fig. 1 shows a system of two rovers working outdoor to coordinate Threat detection collectively. The navigation of this system of rovers was done using GPS along with other sensors. In order to facilitate reasonable outdoor navigation, a task-relevant augmentation of

the track rover (Fig. 1), with various sensory and control systems has been realized ([14]). The rover's onboard sensors include a GPS for absolute positioning in outdoor environments, wireless video for a view of the remote environment, ultrasonic range-finding for obstacle detection, 3-axis gyro for orientation, 3-axis accelerometer for motion capture and encoders for monitoring the vehicles velocity. The rover's computation is achieved through an on-board Windows-based PC.

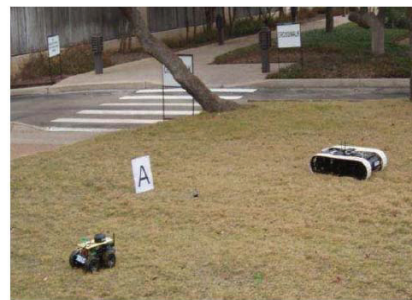


Figure 1. A system of two rovers performing a joint task on a field (Courtesy, ACE Center, University of Texas, San Antonio)

B. Navigation and Task Distribution for a System of Rovers

It is a state-of-the art approach for the coordination of a large number of robots. Fig. 2 shows such a scenario where ground robots are working together with a UAV. The UAV can survey a particular zone of interest. If it finds an object of interest (or a potential threat) in this zone then it can communicate to the ground robots and instruct them to further check it out. The ground robots can also communicate and coordinate among themselves to accomplish the task properly. They can also send the status report of their individual tasks back to the UAV or to a manned supervisory station for further instructions. To achieve such coordination capabilities, swarm robotics have many underlying issues that need to be addressed in order to create a practically applicable system. Robot communication, path planning, tracking, workload distribution, choice of sensors, system reliability, is just some of the most basic issues that need to be addressed. This work addresses the path planning, tracking and navigation so that mobile robots can reach to the object of interest effectively. In order to interact and distribute the task among the members, it is needed to create a communication system with a common protocol.

The communication protocol needs to be dynamically expandable where additional robots can be added easily during a real-time application. Exact requirements of data types in the communication protocol can vary depending upon the swarm mission. On the ground domain, two wheeled robots, one track-robot and four small scout robots work together (two Pioneer P2-AT8 slip-drive robots from ActivMedia Mobile Robotics and one custom built track-robot). To coordinate commands within the ground swarm, one robot acts as the master. These are equipped with GPS receivers and radio modems for proper navigation and communication. The aerial domain consists of one fixed-wing aircraft (KADET, LT-40) with radio modem, GPS and a camera for proper communication, navigation and surveillance. Utilizing the GPS, camera and the processing suite on-board the aircraft, an estimated location of an object of interest is identified. This estimated GPS location can be sent to the master on the ground in order to investigate the object of interest. Once the master of the ground swarm evaluates the transmitted commands to investigate GPS coordinates, it directs the slave robots to search the surrounding areas of the object of interest. When the slave robots reach their destination, the mission task can be performed as directed by the ground master. Fig. 3 shows the real time GPS based navigation of Pioneer II P2AT-8 robot.

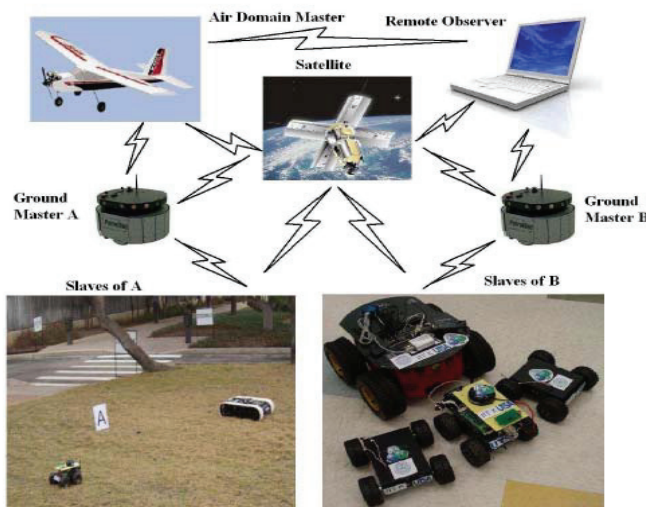


Figure 2. Hierarchy of multi-domain system of systems (Courtesy, ACE Center, University of Texas, San Antonio)



Figure 3. Pioneer II P2AT-8 Detector-Robot, Loader-Robot, Carrier-Robot and custom built robots at ACE Lab. (Courtesy, ACE Center, University of Texas, San Antonio)

In this approach, the environment and the Robotic agents will be simulated including real Robots in the environment. Without a real UAS in the field, a simulation of UAST specific scenarios cannot generate realistic test and evaluation parameters and results. Thus, in the proposed approach, the environment and UAS agents need be modeled accurately such that the simulation environment can have real UAS agents as well as virtual agents. This is extremely critical if the emergent or social behaviors of a team of UAS are being evaluated. For example, a UAS agent can be evaluated in a swarm of UAS agents by creating virtual swarms in the simulation environment and assigning a task to the swarm. The same approach can be used when a multi-modal decision making capabilities of a team of UAS is being evaluated.

II. TECHNICAL APPROACH

The problem of threat detection in an environment is considered. Three systems, comprising of robots and sensors, are proposed to form a system of systems to find a solution to this problem. In the land part of the proposed work, three wheel robots and a track robot comprise a system of rovers. The small wheel robots are in charge of detecting spots which are considered as threatening spots and they communicate with the big track robot to report the fire to the master robot (the track robot) and send and receive data through the designed communication scheme in order to do the desired tasks as a part of a system of systems.

A set of independent robots managed by System of Systems (SoS) architecture and programmed in DEVSJAVA [15-16] is proposed to serve as long term. The IIS Corp- University of Texas at San Antonio's Autonomous Control Engineering (ACE) lab has the objective to build and deploy Land, Air and Sea autonomous robotic vehicles that can independently operate toward a common pre-programmed objective.

In real world systems, the problem is addressed in a higher level where the systems send and receive data from other systems in the SoS and make a decision that leads the SoS to its global goals. Let's take the military surveillance example where different units of the Army collect data through sensors to locate a threat or to determine the identity of a target. In this type of situations, army command center receives data from these heterogeneous sensor systems such as AWACS, ground RADARS, and submarines. These systems are the parts of the SoS that make the decision; say the command and control station. However, they may be developed with different technologies. This will create a huge barrier in data aggregation and data fusion using the data received from these systems because they would not be able to interact successfully without hardware and/or software compatible [14]. In addition, the data coming from these systems are not unified which will add to the barrier in data aggregation and cooperation among the SoS components.

One solution to the problem is to modify the communication medium among the SoS components. Two possible ways of accomplishing this task are:

A. Create a Software Model of Each System

In this approach, each component in the SoS talks to a software module embedded in itself. The software module collects data from the system and through the software model generates outputs and sends to the other SoS components. If these software modules are written with a common architecture and a common language, then the SoS components can communicate effectively regardless of their internal hardware and/or software architectures [14].

Create a common language to describe data, where each system can express its data in this common language so that other SoS components can parse the data successfully.

The overhead that needs to be generated to have software models of each system on a SoS could be enormous and must be redone for new member of the SoS. In addition, this requires the complete knowledge of the state-space model of each SoS components, which is often not possible.

Consequently, recent research on SoS concepts focus on the study of coming up with a common language to describe the data of each system in a SoS in order to create an effective communication medium in the SoS. There are some research activities on using a language based on Extensible Markup Language (XML) in order to wrap data coming from different sources in a common way [13-14]. In these research activities, the XML is used to describe each component of the SoS and their data in a unifying way. If XML based data architecture is used in a SoS, only requirement for the SoS components is to understand/parse XML file received from the components of the SoS [13-14].

B. Integration

Integration is extremely important for both SoS and robotic swarms since both needs to expand and shrink with ease during the operation. In a SoS, it is critical to have the ability to bring a system into the SoS without any major problem such as redesign, restart, and/or reprogram the system. Integration in a SoS has two main components. The first one is to have adaptive communication schemes which allow the inclusion of a new system member. The second one is to have a common language which any new system can send its data to other systems in the SoS without any difficulty or major effort. The common language such as XML is a critical necessity for a SoS. The architecture for such as language can be incorporated with a discrete event simulation tool so that the SoS concepts can be simulated before they are applied to real problems. Consequently, the integration characteristics of a SoS require effective and adaptive communication among the systems. When the systems in a SoS are heterogeneous in hardware and software, the XML like common language becomes very essential. However, since all the members in a swarm have either identical or very similar hardware and software components, necessity of a common language is not so critical. Thus, in a robotic swarm, the design need not have an XML like common language. Instead, a common protocol or a common format will suffice.

The communication scheme needs to have ability to adapt itself for the population changes in the swarm. In order to achieve this goal, processes for accepting new members to the

swarm and removing a member from a swarm need to be implemented as part of the communication scheme.

III. FUZZY REINFORCEMENT LEARNING (FRL) IN SOS

In this section, we discuss how fuzzy control can be used for an unmanned Air Vehicle. This is based on the works of Gomez and Jamshidi [8]. There are a total of six output variables that need to be fuzzified and they are: roll, pitch, airspeed, airspeed error, heading error and altitude error. The ranges of these variables are given in Table I.

TABLE I. VARIABLE RANGES

Variable	Actual Range		Range Used	
Altitude Error	-1 km	1 km	-0.5 m	0.5 m
Heading Error	-360°	360°	-1°	1°
Pitch	-180°	180°	-10°	10°
Roll	-180°	180°	5°	5°
Airspeed Error & Heading Control	-100 m/s	100 m/s	-100 m/s	100 m/s
Altitude & Throttle Control	-0.1	0	-0.01	0

It is possible to create a fuzzy controller that will encase the whole range of values for each variable, but it will take a large number of rules making the calculation of the control much slower. The range used to decide the placement of the membership functions was decided by the fact that we want to reduce the number of rules. By taking a small range and scaling the different variables allows for a smaller number of rules, and the precision will not be lost.

The following figures represent the membership functions of the linguistic variables used for the creation of the different controls. The higher the precision the better the fuzzy controller will function, but it also means more memory needed to store those values. The same can be said about the number of membership functions, a bigger number usually means a smoother control output. Thus, the increase in membership functions leads to an increase in the number of rules

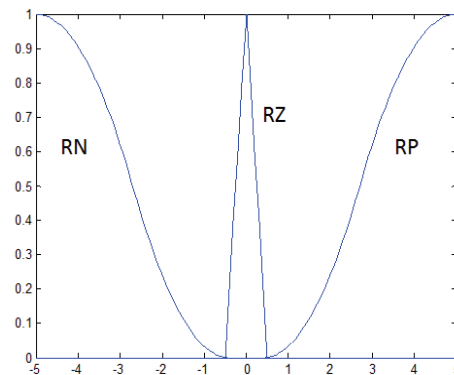


Figure 4. Roll membership functions

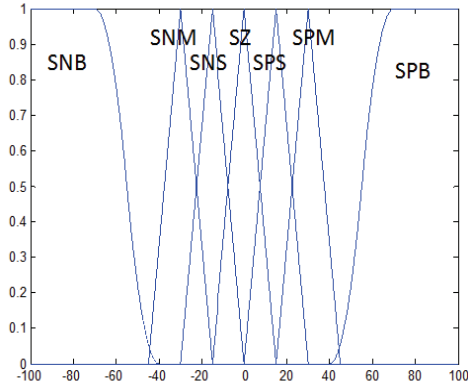


Figure 5. Control output membership functions

*If Heading Error is HP and Roll is RP
then Heading Control is SNB*

⋮

*If Heading Error is HN and Roll is RN
then Heading Control is SPB*

The relationship between the altitude error and the pitch angle create the control surface. The higher the precision the better the fuzzy controller will function, but it also means more memory needed to store those values. The same can be said about the number of membership functions, a bigger number usually means a smoother control output. Thus, the increase in membership functions leads to an increase in the number of rules

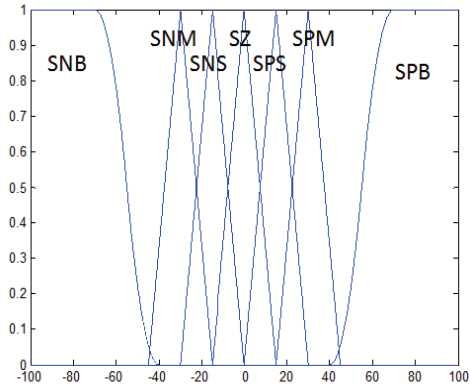


Figure 6. Control output membership functions

*If Heading Error is HP and Roll is RP
then Heading Control is SNB*

⋮

*If Heading Error is HN and Roll is RN
then Heading Control is SPB*

*If Heading Error is HP and Roll is RP
then Heading Control is SNB*

⋮

*If Heading Error is HN and Roll is RN
then Heading Control is SPB*

A simple graphical example of how the different rules give us the control output is shown in the figure below.

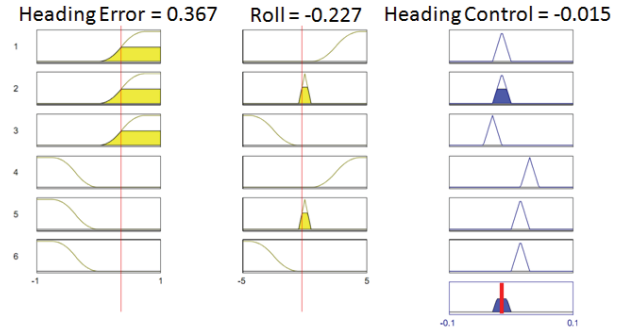


Figure 7. Visual representation of the altitude control rules

The same procedure was followed to create the heading control, six rules relating the heading error and the roll, and an extra rule to maintain the airplane from over rolling.

FRL can be applied for formation control of UGVs according to [3]. For navigation of UGVs, Fuzzy rules can be written according to the above and their parameters can be updated based on GARIC-Q [5,6] and MULTI-GARIC-Q [6]. The use of FRL is not limited to UGVs only and any system of our SOS can use FRL for updating its parameters. For example, [1] focuses on using the FRL for UAVs.

IV. SYSTEM OF SYSTEMS SIMULATION

In a System of Systems, systems and/or subsystems often interact with each other because of interoperability and over all integration of the SoS. These interactions can be achieved by efficient communication among the systems using either peer-to-peer communication or through central coordinator in a given SoS. Since the systems within a SoS are operationally independent, interactions between systems are generally asynchronous in nature. A simple yet robust solution to handle such asynchronous interactions (specifically, receiving messages) is to throw an event at the receiving end to capture the messages from single or multiple systems. Such system interactions can be represented effectively as discrete-event models [15]. In discrete-event modeling, events are generated at variable time intervals as opposed to some pre-determined time interval seen commonly in discrete-time systems. More specifically, the state change of a discrete-event system happens only upon arrival (or generation) of an event, not necessarily at equally spaced time intervals. To this end, a discrete-event model is a feasible approach in simulating the proposed SoS framework and its interaction [15]. Several discrete-event simulation engines [13-16] are available that can be used in simulating interaction in a heterogeneous mixture of

independent systems. We consider one such simulation framework – Discrete Event System Specification (DEVS) because of the available effective mathematical representation and its support to distributed simulation using DoD’s High Level Architecture (HLA).

A. DEVS Modeling and Simulation

Discrete Event System Specification (DEVS) [13-16] is a formalism, which provides a means of specifying the components of a system in a discrete event simulation. In DEVS formalism, one must specify *Basic Models* and how these models are connected together. These basic models are called *Atomic models* and larger models which are obtained by connecting these atomic blocks in meaningful fashion are called *Coupled models*. Each of these atomic models has *imports* (to receive external events), *exports* (to send events), a set of *state variables*, *internal transition*, *external transition*, and *time advance functions*.

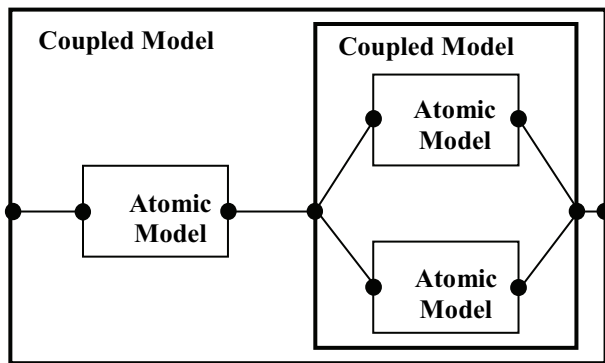


Figure 8. DEVS models for system and sub-systems

Next we present a simulation example using the DEVS simulation environment [13-16]. This is a scenario where a target is being tracked by a group of sensors deployed in the region of interest. The sensors are grouped to form clusters, with each cluster having a cluster-head that can send and receive information to a command center (base-station). The entire simulation can be performed by “coupling” all the DEVS atomic models together with meaningful interconnections. In order to test our SoS architecture in the DEVS environment, our XML based data representation needs to be embedded in to the simulation environment.

B. XML and DEVS

In DEVS, messages can be passed from one system (coupled or atomic model) to another using either predefined or user-defined message formats. Since the systems within a SoS maybe different in hardware and/or software, there is a need for a unifying language for message passing. Each system need not necessarily have the knowledge (operation, implementation, timing, data issues, and so on) of another system in a SoS. Therefore, one has to work at a high-level (information or data level) in order to understand the present working condition of the system. Fig. 9 describes conceptually a SoS simulation example to show the use of XML in the DEVS formalism.

C. SOS Simulation Framework

Based on the system of systems architecture, we have simulated robust threat detection data aggregation Scenario) where there is a base robot, two swarm robots, five sensors, and a threat. The sensors are stationed such that they represent a sensor network covering an area. The threat is moving in the area and is detected by sensors when the threat is sufficiently close to a sensor node. All sensors communicate with a base robot which itself does not actively seek threat. Instead, the base robot processes the information using sensor data and data sent by swarm robots. A command center can collect processed data (aggregated data) from the base robot without interacting with other components of the SoS such as sensors and swarm robots. Robust threat detection works as follows. When a sensor detects the threat, they notify the Base Robot. Then, the Base Robot sends the Swarm Robots the location of the threat based on the information sent by the sensors. This is crucial in order to get rid of false detections as they can cause inefficient usage of system components.

D. XML Format

As mentioned before, the hardware and/or software architectures of these systems will not be the same in reality. Thus, they may not be able to talk to each other successfully even though they can operate independently. We have implemented XML based SoS, message architecture in DEVJSJAVA software [15]. In this XML based message architecture, each system has an XML-like message consisting of their name and a data vector. The name of each system represents an XML tag. The data vectors are used to hold the data of the systems. The length of the data vectors in each system can be different based on the amount of data each system contains.

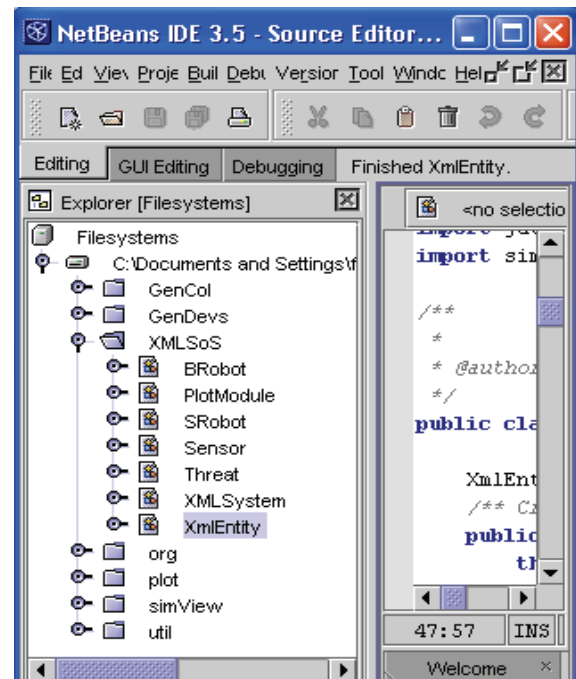


Figure 9. Atomic models and DEVJSJAVA components

REFERENCES

- [1] Berenji, H. R. and D. Vengerov and J. Ametha , "Perception based Reinforcement Learning for Sensor Allocation in Unmanned Aerial Vehicles", the 12th IEEE International Conference on Fuzzy Systems, 2003.
- [2] Berenji, H. R. and D. Vengerov, "Advantages of Cooperation Between Reinforcement Learning Agents in Difficult Stochastic Problems". In proceedings of the 9th IEEE International Conference on Fuzzy Systems, 2000.
- [3] Berenji, H.R. and D. Vengerov, "Convergent Actor Critic-based Fuzzy Reinforcement Learning Apparatus and Method", US patent 6,917,925 B2, Filed Dec. 21, 2001.
- [4] Berenji, H.R "Collaborative fuzzy reinforcement learning agents", International Fuzzy Systems Association World Congress, 1997.
- [5] Berenji, H.R. and D. Vengerov, "Cooperation and Coordination between teams of Fuzzy Reinforcement Agents", FUZZ-IEEE, Seoul, Korea, FUZZ-IEEE, Seoul, Korea, 1999.
- [6] Berenji, H.R. and S. Saraf, "Competition and Collaboration among Fuzzy Reinforcement Learning Agents", Seventh IEEE International conference on Fuzzy Systems, Anchorage, Alaska, May 1998.
- [7] Crossley, W.A., "System of Systems: An Introduction" of Purdue University Schools of Engineering's Signature Area."
- [8] Gomez, J. F and M. Jamshidi,"Fuzzy Logic Control of a Fixed Wing Unmanned Aerial Vehicle", WAC 2010, Kobe, Japan.
- [9] Jamshidi, M., "Theme of the IEEE SMC 2005", Waikoloa, Hawaii, USA".
- [10] Luskasik, S.J., "Systems, "Systems of Systems, and the Education of Engineers", Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, Vol. 12, No. 1, pp. 11-60, 1998.
- [11] Meilich, A., "System of Systems (SoS) Engineering & Architecture Challenges in a Net Centric Environment," Proc. of IEEE International Conference on System of Systems Engineering, Los Angeles, April 2006.
- [12] Sage, A. P. and C. D. Cuppan, "On the Systems Engineering and Management of Systems of Systems and Federations of Systems," Information, Knowledge, Systems Management, Vol. 2, No. 4, pp. 325-34, 2001.
- [13] Sahin, F. and P. Sridhar and B. Horan and V. Raghavan and Mo Jamshidi, "System of Systems Approach to Threat Detection and Integration of Heterogeneous Independently Operable Systems", " IEEE Conference on Systems, Man and Cybernetics, Montreal, Canada, October 2007.
- [14] Sahin, F. and M. Jamshidi, and P. Sridhar, "A Discrete Event XML based Simulation Framework for System of Systems Architectures,"IEEE International Conference on System of Systems, April 2007.
- [15] Ziegler, B.P. and H. Sarjoughian, "Introduction to DEVS Modeling and Simulation with JAVA A Simplified Approach to HLA-Compliant Distributed Simulations," ACIMS, <http://www.acims.arizona.edu>
- [16] Ziegler, B. P. and H. Praehofer. and T.G., Kim, "Theory of Modeling and Simulation", Second edition, Academic Press, Boston, 2000