DIVAs 4.0: A Framework for the Development of Situated Multi-Agent Based Simulation Systems

(Demonstration)

Frederico Araujo, Junia Valente, Mohammad Al-Zinati, Dane Kuiper, Rym Zalila-Wenkstern
University of Texas at Dallas
800 West Campbell Road
Richardson, Texas, USA
{frederico.araujo, juniavalente, mohammad.al-zinati, kuiper, rymw}@utdallas.edu

ABSTRACT

In this paper we present DIVAs 4.0, a framework that supports the development of large-scale agent-based simulation systems where agents are situated in open environments. DIVAs includes high-level abstractions for the definition of agents and open environments, a microkernel for the management of the simulation workflow, domain-specific libraries for the rapid development of simulations, and reusable, extendable components for the control and visualization of simulations. We illustrate the use of DIVAs through the development of a simulator where virtual agents are situated in a virtual city and an office environment.

Categories and Subject Descriptors

I.6.8 [**Simulation and Modeling**]: Visual; D.2.13 [**Software Engineering**]: Reusable Libraries

Keywords

multi-agent simulation, framework, open environments

1. INTRODUCTION

Multi-Agent Based Simulation Systems (MABS) has been an active area of research for over two decades. Although many approaches have been proposed for different application domains, very few platforms support the development of MABS where the environment is *open* (i.e., inaccessible, non-deterministic, dynamic, and continuous) and agents do not have access to global environmental data [3, 2]. In addition, none of the existing MABS allow for the dynamic access and modification of environment and agent *properties* and *behavior* at run-time.

Over the past several years we have developed DIVAs[1], a framework that supports the specification and execution of large-scale real-time simulations for non-trivial domains (e.g., crowd evacuation, traffic). The framework is based on the premise that agents and environment play an equally important role in the simulation. In DIVAs, agents are situated in an open environment, which is totally decoupled from the agents. Agent and environment interact through the action-potential/result (APR) model. At run-time, DIVAs allows for the modification of the environment model (e.g., centralized, decentralized) and the agent properties (e.g., sensor parameters, velocity, heading, goal modification). DIVAs simulates the propagation of external events and incorporates a novel agent perception combination algorithm [1] that combines the information perceived from the agent's multiple sensors into useful knowledge.

2. HIGH-LEVEL ARCHITECTURE

DIVAs (see Figure 1) was designed to provide a flexible, extensible architecture whose components can be easily reused and extended in various simulation domains. The framework is fully implemented in Java and is based on state-of-the-art enterprise-scale technologies (e.g., JMS, OpenGL, JavaFX).

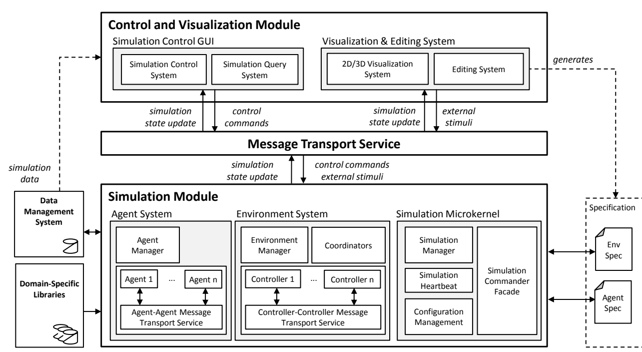

Figure 1: DIVAs framework high-level architecture

The architecture includes three main modules. The *Simulation Module* creates large-scale simulation instances. The *Message Transport Service* (MTS) provides a messaging infrastructure that allows different elements of the simulation to communicate. The interactive *Control and Visualization Module* receives information from the MTS, renders 2D/3D images of the simulation, and allows simulation interaction (e.g., specify environment, trigger events, change agent and environment properties) and modification of simulation parameters at run-time.

DIVAs' main constituent, i.e., the Simulation Module, consists of three subsystems: the *Agent System* creates and

[1]http://youtu.be/bUB5yJr9REM

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

manages agents in the simulation, the *Environment System* creates and manages a dynamic environment in which the agents are situated, and the *Simulation Microkernel* encapsulates the most important core services of the framework (e.g., simulation heartbeat), manages the workflow of the simulation and provides mechanisms for loading/storing agent and environment specifications to/from persistent storage (e.g., xml file).

Also, DIVAs includes a *Data Management System* for storing and processing information collected from the simulation and *Domain-Specific Libraries* for the rapid development of simulation platforms. Currently, DIVAs embeds models for virtual agents and traffic simulation domains.

3. CASE STUDY

In this section we briefly discuss how DIVAs is used to develop simulators for the modeling of a virtual city and an evacuation scenario.

3.1 Agent Specification

DIVAs provides domain-independent abstractions and modules (e.g., sensor, tasks) that can be used to create instances of agents. In the case of social simulations, virtual agents are specified by creating a concrete class (based on the DIVAs' agent abstract class) that incorporates the modules fit for that type of agents. The modules available in the DIVAs' *social simulation* library include:

Interaction module. It contains a communication module to handle asynchronous communication with other agents and a perception module including vision, auditory, and olfactory sensors and their respective sensing algorithms. The developer may select to equip agents with some or all of the provided sensors.

Task module. It includes primitive tasks that the agent can perform such as walk, run and look. The user may select to include these tasks in the agent definition and/or define new tasks.

Knowledge module. It includes the agents' knowledge about the external world and itself. Agents have predefined knowledge (e.g., an explosion is dangerous) and knowledge that is acquired at run-time (e.g., the location of an exit).

Planning and control module. It serves as the brain of the agent, and defines how an agent reacts to critical situations, plans, initiates tasks, and achieves its goals. This module includes path finding and collision avoidance algorithms to be used by agents when planning their movement within the environment. Proactive and reactive features can be used to define the agent's desired behavior.

In addition, DIVAs provides libraries of predefined concrete agent types that can be instantiated by developers.

In this demo we show how a virtual agent is specified by creating a concrete *virtual-agent* class.

3.2 Environment Specification

The simulated environment can be created in one of three ways: 1) similar to virtual agents, environment objects can be defined through concrete classes that are instantiated into actual objects (e.g., building, tree, road). These objects are then used to construct the environment; 2) users can upload predefined environments; 2) the environment can be built using existing libraries. DIVAs provides a 3-dimensional graphical user interface that allows users to "drag-and-drop" environment objects into the simulation. In addition, the user can "click-to-edit" an object to modify its properties (e.g., size, position, rotation). By using libraries, a user can construct an entire virtual environment piece-by-piece without programming a single line of code. Finally environment events (e.g., fireworks, explosions, emergency sirens) can be selected from a set of predefined events and triggered during the execution of the simulation.

In this demo we show how a virtual city is created using a library of environment objects. We also show how an existing office space environment is uploaded.

3.3 Executing the Simulation

Once the simulation model is specified and the simulator is deployed, the user can use DIVAs' supporting tools to visualize, control and interact with the running simulation. Using the simulation control GUI, users can launch concurrent, synchronized visualizations (2D/3D) of the simulation. Users can interact with the simulation by modifying agent and/or environment object properties and triggering events in the simulated environment at run-time. In the demo we show how an agent's field of view can be altered at execution time. By activating the agent camera mode, we show how the environment is viewed from the agent's perspective. We also show how agents react to user-triggered events by combining the percepts acquired through their multiple senses. In the case of the crowd evacuation scenario, a bomb is triggered at the center of the office building. As a result emergency sirens are triggered and agents use their senses and knowledge to plan for the closest exit. We can observe emergent patterns created by the flow of agents towards the exits of the building.

4. CONCLUSION

Divas is a framework for the development of situated multi-agent simulation systems with continuous, non-deterministic, and dynamic environments. Agents do not have access to global environmental data and perceive their surroundings through perception sensors. The users of the simulation can modify agent properties and trigger external events at run-time. As more libraries are developed, DIVAs will provide a unique framework for the simulation of a variety of real world domains.

Acknowledgments DIVAs is supported by Rockwell-Collins grant 5-25143. The DIVAs project has been under investigation for several years. This work would not have been possible without the contributions of Travis Steel, Renee Steiner, and Gary Leask.

5. REFERENCES

[1] D. Kuiper and R. Wenkstern. Virtual agent perception combination in multi agent based systems. In *Proc. of the 12th Int. Conf. on Autonomous Agents and Multiagent Systems*. IFAAMAS, 2013.

[2] N. Pelechano, J. Allbeck, and N. Badler. Controlling individual agents in high-density crowd simulation. In *2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108, San Diego, California, August 02 - 04 2007.

[3] S. J. Rymill and N. A. Dodgson. Psychologically-based vision and attention for the simulation of human behaviour. In *Proceedings of Computer graphics and interactive techniques*, pages 229–236, Dunedin, New Zealand, November 29 - December 02 2005.