Extensible Intelligent Simulator Architecture for the Development of Cyber-Physical Systems

Daegeun Choe[1], Yiwen (Chris) Shen[1], Bien Aime Mugabarigira[1], and Jaehoon (Paul) Jeong[2]
[1]Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, Republic of Korea
[2]Department of Interaction Science, Sungkyunkwan University, Suwon, Republic of Korea
Email: {cdg1994, chrisshen, bienaime, pauljeong}@skku.edu

Abstract

This paper proposes an architecture for extensible Cyber-Physical Systems (CPS) simulator based on the open-source software. There are many useful open source systems but hard to integrate. To solve integration problems, we design an open-source based simulation architecture that can cooperate with another open-source application easily. Our simulation architecture consists of a physical simulator, a simulation manager, and external applications. The simulation manager constructs a simulation scenario and objects properties. Also, through the simulation manager, we can efficiently connect objects in the CPS simulator with external applications. Using this simulation architecture, we can make open-source based CPS simulator that can work efficiently with other open-source applications

Keywords: Intelligent Cyber-Physical Systems (iCPS), Simulation, Open Source.

1. Introduction

In research of Intelligence Cyber-Physical Systems (iCPS), to make and operate good simulation is important issue because testing of iCPS systems in real-world environments requires high cost and dangerous components such as vehicles. Also, to follow the pace of development to technology, we need to research through simulation. Because of these reasons, many researchers used various applications for build-up Cyber-Physical simulation system. For example, the Veins framework combines OMNet++ and SUMO to make vehicular network simulations [1-3]. However, it is difficult efficiently to connect with other applications. In this paper, we propose an extensible simulation system through merging these open-source based applications that supplement each other. To connect two or more applications, we should made a simulation manager that manage interface between application and controls simulation states.

The rest of the paper is organized as follows. Section 2 and 3 describes the architecture of iCPS simulation system. Next, Section 4 presents system scenario of the system. Finally, conclusion is provided in Section 5.

2. System Architecture

The main components of iCPS simulation are physical simulator, simulation manager, and external applications. The simulation manager can connects with two or more open-source applications to construct extensible simulating environment for conduct of iCPS simulation system.

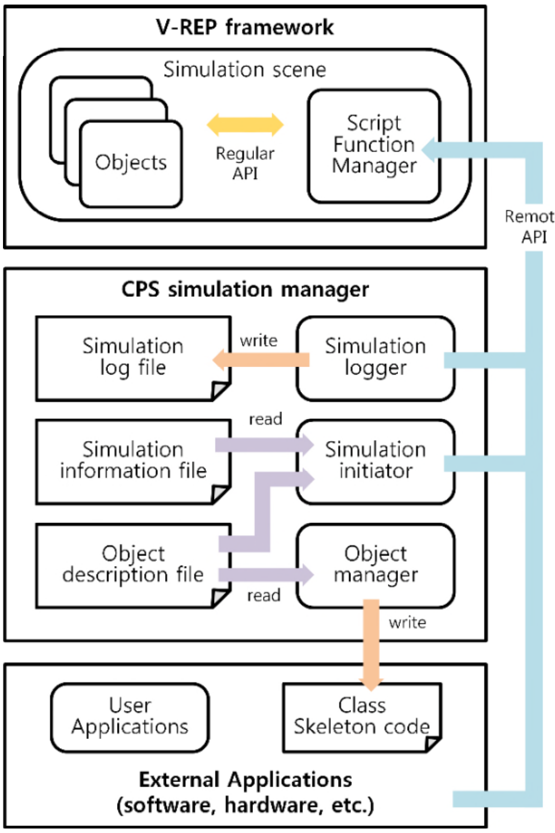

Figure 1: Architecture of iCPS Simulation

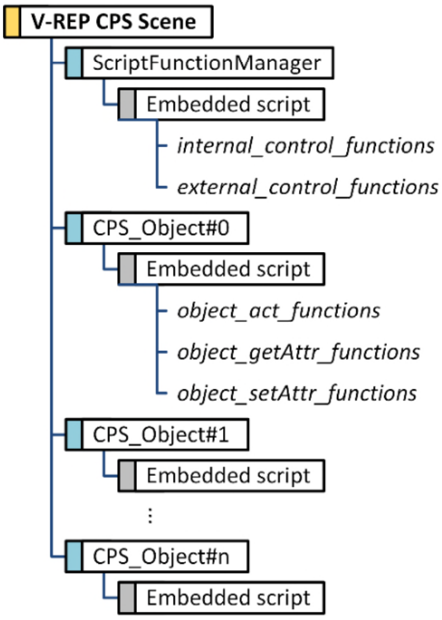

Figure 2: Structure of V-REP Simulation

The simulation manager has three roles; simulation initiator, simulation logger, and object manager. Reading simulation information file and object description file written by JSON, the simulation manager constructs simulation environment and controls simulation state such as start or stop. During the simulation, the simulation manager logs simulation information such as position or velocity of objects to review some simulation results. Also, simulation manager makes skeleton code that contains object definition and interface for external applications.

We choose V-REP for physical simulator because of some reasons [4]. First, V-REP is open-source software that has useful online document. Second, V-REP support various programming language for flexibility and extensibility. Last, V-REP support external interface called by *Remote API*. For these reasons, V-REP can communicate efficiently with other applications. Using *Remote API*, simulation manager can controls simulation state and logs simulation information. In the same way, external applications cooperate with V-REP to help some work that V-REP cannot dose such as network simulation or complex calculation.

External applications use skeleton code and *Remote API* to communicate with V-REP. To implement and simulate, we use OMNet++ as an external application. Figure 1 describe the overall structure of our simulation system.

3. Simulation components

In this section, we explain each components of iCPS simulation system in detail.

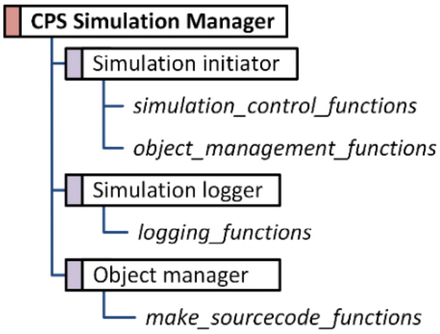

Figure 3: Structure of Simulation Manager

3.1. V-REP physical simulator

One V-REP simulation is represent by a *scene* that contains *scene object* and *embedded script*. *Embedded script* written in 'Lua' [5] script handles a particular function in a simulation. V-REP framework has two API categories; *Regular API* and *Remote API*.

Regular API is used to handle simulation within the simulator. Each object in V-REP simulation has associated scripts that contain functions such as *object_act_functions*, *object_getAttr_functions* and *object_setAttr_functions*. These functions written in *Regular API* handle act of associated object, modify and return attribute of associated object [4].

To efficiently communicate between V-REP and external applications, we made a special object called by *Script_Function_Manager*. This special object communicate with external applications through the calling to *Remote API*. If some applications want to control actions of objects or get information of objects, they need to send command message to *Script_Function_Manager*, and it decodes the command and calls object function such as *object_getAttr_functions()*. Figure 2 describe structure of V-REP simulation system.

3.2. Simulation Manager

As we explained in Section 2, the simulation manager consists of simulation initiator, simulation logger, and object manager. The simulation manager communicate with V-REP using *Remote API*. Also it makes skeleton code that contains *Remote API* for external applications. External applications can handle V-REP simulation same way through skeleton code that is made by simulation manager. Figure 3 describe functions of simulation manager.

3.3. External applications

External applications can be any applications if who can modify codes or add functions. To implement this, we choose OMNet++ for network simulation. To communicate between OMNet++ and V-REP, we add

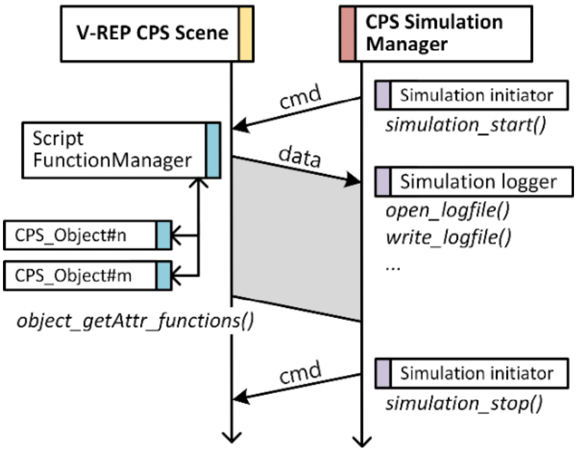

Figure 4: Sequential Diagram of Simulation Manager

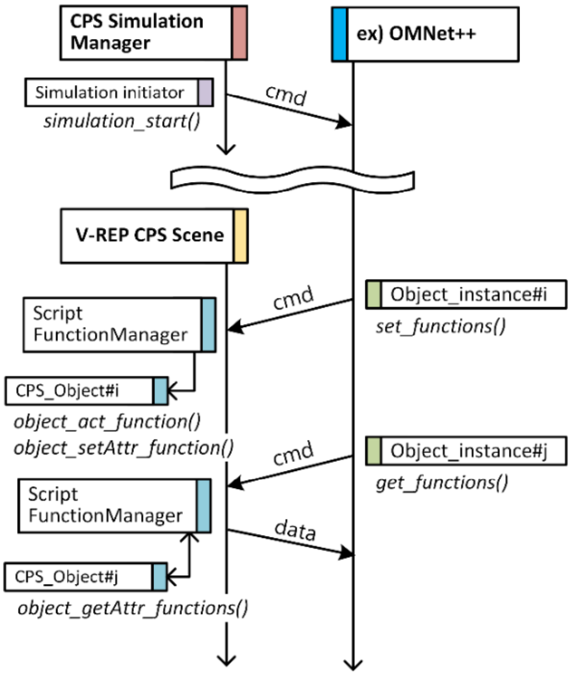

Figure 5: Sequential Diagram of External Applications

the skeleton code to OMNet++. For example, OMNet++ can get coordinate of objects or change trajectory of objects.

4. Scenario of Simulation

In this section, we show two scenario; one is simulation manager case and the other is external application case.

4.1. Scenario of simulation manager

Simulation manager sends command to V-REP simulator and other applications to initiate iCPS simulation. The start point of simulation is synchronized by simulation manager. On the other hand, V-REP synchronizes progress of simulation during simulation. Simulation manager read JSON files that include simulation information and objects description to make V-REP scene. When the simulation is started, *Script_Function_Manager* gets objects information and sends to simulation manager for logging. Simulation manager save log file as some logging options in JSON files. When the specific condition is satisfied, simulation manager sends command to V-REP to stop the simulation. Figure 4 show this scenario.

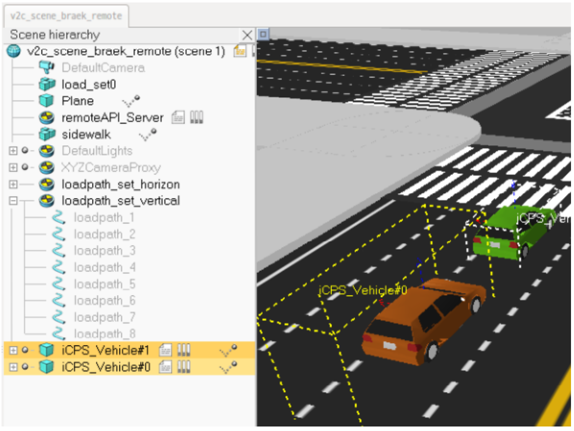

Figure 6: V-REP Simulation of Intelligent Transportation Systems

4.2. Scenario of external application

External applications such as OMNet++ are started by simulation manager. These applications handle V-REP simulation through *Remote API*. For example, the applications change coordinate of objects or add new objects in scene. At that same time, these applications participate in simulation using *Remote API*. Also, external applications get information from V-REP and do their work such as complex calculation or other simulation. All these works are processed by *Script_Function_Manager*. External applications send come command to *Script_Function_Manager* through *Remote API*. Figure 5 show the scenario of external applications.

5. Implementation for Intelligent Transportation Systems

In this section, we show intelligent transportation system made by our architecture as an implementation example. The system can simulate physics and network environment. To build V-REP scene, the simulation manager read JSON file that describe vehicles and crossroads. V-REP simulates virtual vehicles on roads. In the simulation system, each vehicle measure coordinate using virtual GPS. OMNet++ simulates network communications between vehicles. OMNet++ gets coordinate of vehicle from V-REP and simulate packet exchange between two vehicles. The following vehicle calculates a distance from leading vehicle. If the distance is smaller than safety distance, the following vehicle stops to avoid collision. Figure 6 is screenshot of V-REP simulation of intelligent transportation system.

6. Conclusion

In this paper, we purpose extensible simulation architecture for iCPS projects. We design simulation manager to provide extensibility of simulation systems. The simulation manager conduct the iCPS simulation. It control simulation states, record the simulation information, and make skeleton code that contain interface for external applications. Also, we choose V-REP simulation for physical simulation because it support many functions such as *Remote API*. To demonstrate that many applications can be integrated efficiently, we connect V-REP with OMNet++ as networking simulator. We expect this simulation architecture will help many research who want to combine many open-source application for iCPS projects.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7065980).

References

[1] Veins, Vehicles in Network Simulation, “http://veins.car2x.org/”.

[2] OMNet++, Discrete Event Simulator, “https://www.omnetpp.org/”.

[3] SUMO, Simulation of Urban Mobility, “http://sumo.dlr.de/”.

[4] V-REP, Virtual Robot Experimentation Platform, “http://www.coppeliarobotics.com/”.

[5] Lua, The Script Programming language, “https://www.lua.org/about.html/”