ABSTRACT

Development of new functionality and smart systems for different types of vehicles is accelerating with the advent of new emerging technologies such as connected and autonomous vehicles. To ensure that these new systems and functions work as intended, flexible and credible evaluation tools are necessary. One example of this type of tool is a driving simulator, which can be used for testing new and existing vehicle concepts and driver support systems. When a driver in a driving simulator operates it in the same way as they would in actual traffic, you get a realistic evaluation of what you want to investigate. Two advantages of a driving simulator are (1.) that you can repeat the same situation several times over a short period of time, and (2.) you can study driver reactions during dangerous situations that could result in serious injuries if they occurred in the real world. An important component of a driving simulator is the vehicle model, i.e., the model that describes how the vehicle reacts to its surroundings and driver inputs. To increase the simulator realism or the computational performance, it is possible to divide the vehicle model into subsystems that run on different computers that are connected in a network. A subsystem can also be replaced with hardware using so-called hardware-in-the-loop simulation, and can then be connected to the rest of the vehicle model using a specified interface. The technique of dividing a model into smaller subsystems running on separate nodes that communicate through a network is called distributed simulation. This thesis investigates if and how a distributed simulator design might facilitate the maintenance and new development required for a driving simulator to be able to keep up with the increasing pace of vehicle development. For this purpose, three different distributed simulator solutions have been designed, built, and analyzed with the aim of constructing distributed simulators, including external hardware, where the simulation achieves the same degree of realism as with a traditional driving simulator. One of these simulator solutions has been used to create a parameterized powertrain model that can be configured to represent any of a number of different vehicles. Furthermore, the driver’s driving task is combined with the powertrain model to monitor deviations. After the powertrain model was created, subsystems from a simulator solution and the powertrain model have been transferred to a Modelica environment. The goal is to create a framework for requirement testing that guarantees sufficient realism, also for a distributed driving simulation. The results show that the distributed simulators we have developed work well overall with satisfactory performance. It is important to manage the vehicle model and how it is connected to a distributed system. In the distributed driveline simulator setup, the network delays were so small that they could be ignored, i.e., they did not affect the driving experience. However, if one gradually increases the delays, a driver in the distributed simulator will change his/her behavior. The impact of communication latency on a distributed simulator also depends on the simulator application, where different usages of the simulator, i.e., different simulator studies, will have different demands. We believe that many simulator studies could be performed using a distributed setup. One issue is how modifications to the system affect the vehicle model and the desired behavior. This leads to the need for methodology for managing model requirements. In order to detect model deviations in the simulator environment, a monitoring aid has been implemented to help notify test managers when a model behaves strangely or is driven outside of its validated region. Since the availability of distributed laboratory equipment can be limited, the possibility of using Modelica (which is an equation-based and object-oriented programming language) for simulating subsystems is also examined. Implementation of the model in Modelica has also been extended with requirements management, and in this work a framework is proposed for automatically evaluating the model in a tool.