Simulation Modelling Practice and Theory 29 (2012) 193–206

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Formal verification and validation with DEVS-Suite: OSPF Case study

Ahmet Zengin *, Muhammed Maruf Ozturk

Department of Computer Engineering, Sakarya University, Faculty of Technology, Turkey

ARTICLE INFO

Article history:
Received 28 February 2011
Received in revised form 9 February 2012
Accepted 2 May 2012

Keywords:
Verification and validation
DEVS
DEVS-Suite
OSPF
Ns-2

ABSTRACT

Validation is a degree of which how correct a model represents the behavior of its system counterpart from the perspective of intended use of the model. The degree of representation of the model or abstraction is determined by the modeler according to user demands and objectives. Whenever the modeler and simulation user's demands are satisfied, the model is considered as valid.

In this paper, verification and validation of the DEVS models in DEVS-Suite environment are discussed. A case example called OSPF–DEVS simulator is applied and verification and validation tests are performed on it to show usefulness of DEVS formalism. Performed verification and validation tests are followed using a technique developed by Forrester and Senge. Particular attention is paid to reliability and maintainability in view of the state-of-the-art network simulator `ns-2`. Results are documented to lend confidence to simulation users and to show DEVS-Suite environment's capabilities not to increase model infrastructure.

1. Introduction

Confidence in simulation models is provided by verification and validation (V&V) tests. Validation is a level of which a model mimics the behavior of its system counterpart [32]. The degree of representation of the model or abstraction is determined according to the model objectives. Whenever a simulation user demands are met, the model can be considered as valid and has confidence. Validation is an important, a highly required and also a difficult phase in developing simulation models [3]. For the sake of a valid and correct model, quantitative and qualitative tests must be performed [17,10]. The success of a specific test does not prove that the model is valid. No model can be validated, verified and accredited with passing specific tests [17]. If behaviors of both the model and its system counterpart are within acceptable tolerance, the model is said to be valid [32].

Validation is a difficult and inconceivable process that many modelers either do not perform or leave incomplete the task. The reason of incomprehensibility of the validation stems from a number of issues in verification and validation process [22,21]. These issues include how V&V level is needed, demand for automated and formal tools, selection and inaccuracy of real world data and time limitation [21,3]. Main motivation was to develop and implement modeling and simulation application to bring solutions to the issues above and to show suitability of the `DEVS-Suite` simulator for V&V process. In this work, the Open Shortest Path First (OSPF) model [33] on top of `DEVS-Suite` environment was applied to formal validation model in order to exemplify environment's power on validation process. A verification and validation technique proposed by Forrester and Senge [10] is chosen in the process from many techniques available for modeling and simulation study [4,5].

* Corresponding author.
E-mail addresses: azengin@sakarya.edu.tr (A. Zengin), muhammedozturk@sakarya.edu.tr (M.M. Ozturk).

1569-190X/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2012.05.013

DEVS-Suite is a general-purpose, discrete event simulation environment which supports validation, visualization and tracking capabilities [14,23]. This is the new generation of the DEVSJAVA simulator [1] based on DEVS formalism [32]. This simulator also supports variable structure modeling [12]. The DEVS-Suite user-interface provides a consistent, an efficient, an integrated hierarchical component-based representation of models with run-time I/O and state trajectories and tabular data visualization. These properties of the DEVS-Suite simulator facilitate to process outcomes to be documented completely to establish confidence and correctness over simulation models [15].

One of the main purposes of the performed work is to answer the key question about the validation process: How validation level is needed for a specific modeling and simulation study? In order to determine the validation level, requirements are specified, associated DEVS experimental frame is developed and validation tests are performed. Specified requirements of the modeling process help to define the scope of the evidence needed to show that the model is validated for its intended use. Therefore, the scope of the experimental frame helps to determine the scope of the validation level easily. Performed work implements adopted V&V method from early phases of the modeling process to the end, i.e. it covers model objective specifications. Many studies in this area use V&V methods at the last phases of the application. The main purpose of V&V is to ensure that the simulation tool is made according to the requirements of the users and does indeed meet the intended purpose.

The performed work includes an application of the formal V&V method to the DEVS formalism as a formalized system specification. The design of many modeling and simulation V&V tools is usually non-formalized. Lack of any formal definition causes a low confidence, performance, bad-scalable and non-repeatable approaches. Well-formalized V&V process facilitates verification of optimal system design and efficiency. Instead of non-formalized V&V process, stage-by-stage design, verification, validation and final integration are more preferable [34]. DEVS formalized modeling approach has many important traits for V&V process by which validation is done in a robust, rapid and system theoretic way [32]. The developed study takes advantages of the DEVS formalism in doing formal validation process.

The need for automated V&V tools is a big challenge in modeling and simulation process [21]. In this study, results show that DEVS-Suite simulator runs faster, increases V&V coverage and decreases the costs. It has improved visualization and easy-to-use interface which makes learning automated V&V process testing.

The remainder of this paper is organized as follows: Section 2 gives background in which it briefly reviews the validation of simulation models, gives brief introduction about DEVS-Suite, ns-2 and OSPF protocol. Section 3 elaborates validation tests performed in DEVS-Suite. Section 4 covers evaluation of the developed framework. Section 5 presents conclusions and future works.

2. Background

2.1. Validation of simulation models

2.1.1. What is validation?

Validation is a degree of which how correct a model represents the behavior of its system counterpart from the perspective of intended use of the model [32]. The degree of representation of a model or abstraction is determined by the modeler according to the user demands and objectives. Whenever the modeler and simulation user's demands are satisfied, the model is considered to be valid. In other words, model validation must be evaluated for its condition of usefulness instead of perfectness. Validation is a highly required and an integral part of the entire simulation lifecycle by which the model is credible and accredited. Model validation process is intended for building the correct model and it helps to find the right model [3]. Simulation validity is related to simulation design and simulation usage purpose. Validation process has to be performed by both the modeler and the simulation user [17].

In order to validate a model, quantitative and qualitative tests can be performed [17,10]. In DEVS framework, an experimental frame is used to perform validation tests. If behaviors of both the model and its system counterpart are within acceptable tolerance, the model is valid [32]. Validation tests are various rather than a single detailed test by which confidence of the model increases as it passes them. Passing a test does not mean the model is valid, on the other hand, the failure of any test allows the modeler to make decision on the model redesign process.

2.1.2. Validation process

Fig. 1 shows verification and validation processes in discrete event modeling and simulation study with its experimental frame. Whole process is exemplified using the OSPF–DEVS [33] model together with its experimental frame for verification and validation. In DEVS modeling, the experimental frame is used to make deeper decisions on the model. Though its main function is to transfer outcome measures to variables, it is also used in evaluating how well model objectives are achieved. Fig. 1 helps to map V&V onto whole modeling process rather than giving an idea that V&V process is a singular phase or step in modeling. V&V is a continuous process confluent with modeling and simulation activity [4].

As shown in Fig. 1, modeling activity starts with problem statement and description of objectives. Upon development of the conceptual model, conceptual model validation is conducted to determine detail level of the proposed model. Validation of the conceptual model means that the detail of proposed model is sufficient and the performed assumptions are accurate.

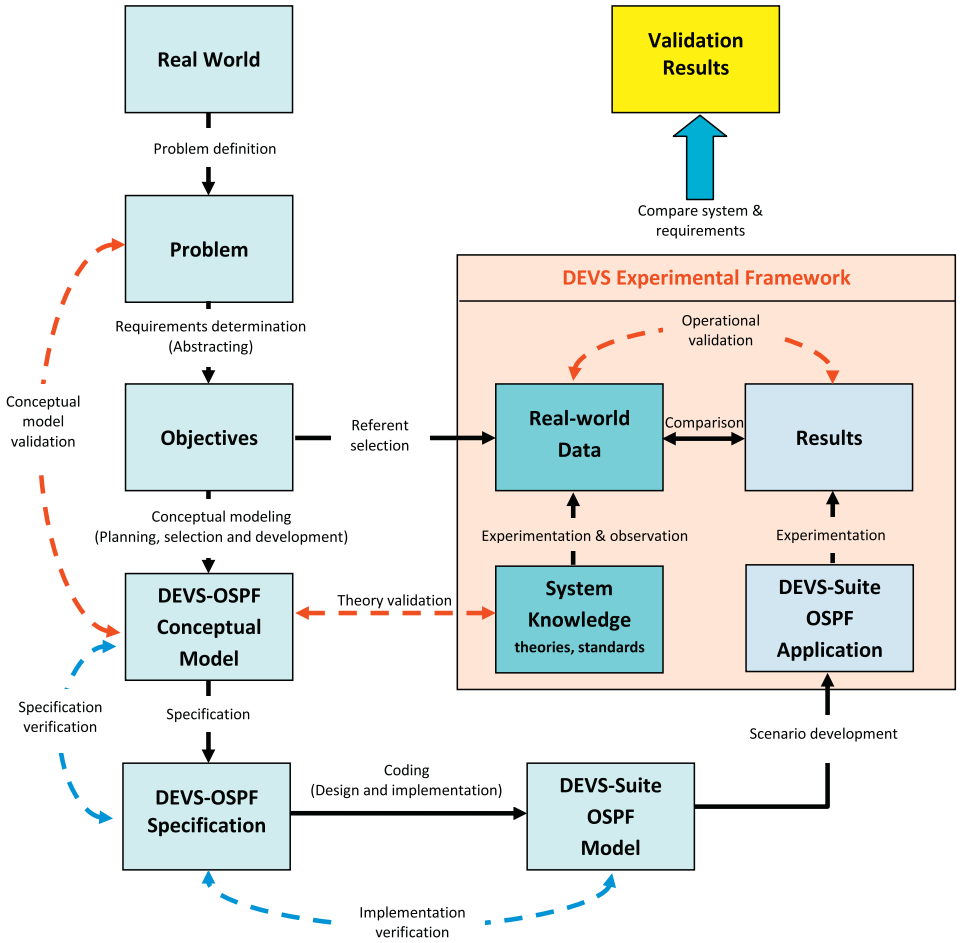

Fig. 1. Verification and validation in DEVS modeling process and experimental frame.

The theory validation is about to the technical depth of the developed model, its underlying formalisms, approaches, standards and theories. For example, the OSPF protocol model is developed according to the RFC 2328 standard [9]. The theory validation requires to ask some questions such as which the theory is exploited in the models? and which theories are combined? [16].

Operational validation refers to determining whether model outputs are sufficiently correct according to the data obtained from the real world. Operational validation activity includes validation of the required model representations, their associated knowledge bases, simulation algorithms, formalisms, models and their associated data [28]. Operational validations are directly related to the model credibility [17].

Besides kinds of validation above, two aspects of verifications can be applied: specification and implementation verification. Specification verification ensures to truly specify the model behavior defined in the conceptual model, while implementation verification deals with all specifications are coded in the model run on the DEVS-Suite environment as built.

2.1.3. Validation challenges

Though many advantages are benefited in modeling and simulation study, there are number of issues in verification and validation process. This section summarizes challenges about V&V process which are well defined in [22,21]. The Article in [21] describes major modeling and simulation V&V challenges and proposes new solutions. In the following, some of these challenges are summarized:

- *V&V level*: Determining how much V&V is needed is very crucial since it depends on the user requirements. The model V&V process continues to meet its user objectives. Therefore, how much V&V tests are needed is the primary issue in M&S process.
- *Demand for automated tools*: Automated V&V techniques are required in modeling and simulation process to perform large V&V experiments. Enhanced V&V capabilities are also essential for better confidence.

- *Demand for formal tools*: The design of most modeling and simulation V&V approaches and tools are usually non-formalized. Lack of any formal definition causes a low performance, bad-scalable and non-repeatable V&V methods. Well-formalized V&V process facilitates verification of optimal system design and efficiency. Instead of non-formalized V&V process, stage-by-stage design, verification, validation and final integration are more preferable [34]. The formalized design approaches such as DEVS have many important advantages for V&V process [32].
- *Universality of V&V*: As already mentioned, model validation and model objectives are tightly coupled. A model can only be validated as required as its design objectives. It is impossible to validate a model generally. A model could only be validated if its validity for every purpose was proven [21]. Such a model requires much cost, a great deal of codes, data and computational resources.
- *Selection of the real world data*: Since view points of the modelers on the real world may vary, a model is valid according to the modeler, while it may not be valid to another.
- *Inaccuracy of real world data*: In V&V process, comparison of model outputs against the real world data is an important step. In case of less accuracy or lack of the real world samples, V&V value of the model is affected inversely. On the other hand, if a model is developed to research on alternative methods and systems, no real world data exists.
- *Time limitation for V&V process*: The scope of V&V process must be covered every aspect of a model. More tests on a model increase confidence on it. In most cases, the developer has not enough time to verify and validate everything.

2.1.4. Validation techniques

Many verification and validation techniques are available for modeling and simulation study [4,5]. Most of these techniques are related to modeling and simulation process. Various V&V techniques are summarized in [2] and an approach proposed by Forrester and Senge [10] is given in particular. Validating and verifying DEVS simulation models are presented in [15].

1. *Forrester and Senge* [10]: Qualitative and quantitative models can be validated using this method. This validation approach defines sets of tests and elaborates their contribution to the model confidence. Fig. 2 shows these tests in three categories. These categories are system structure tests, model behavior tests and policy implications tests. Tests of system structure evaluate structure of the system and its parameters. Tests of the model behavior assess indirectly model structure by analyzing a model's behavior. Tests of policy implications evaluate the response of a real system to a changed policy.
2. *Whitner and Balci* [30]: Information about the structure of a model, modeling techniques and practices employed, data and control flow within the model, and syntactical accuracy can be evaluated using this technique.
3. *Khazanchi* [13]: Qualitative and conceptual models can be validated using this approach. The approach includes a set of criteria and some of them are related to plausibility, feasibility and effectiveness of the model.

2.2. DEVS-Suite validation framework

DEVS-Suite is an object-oriented implementation of the DEVS formalism and an open source, discrete event, general-purpose simulation environment [14]. `DEVS-Suite` can simulate models specified using the DEVS formalism [32]. It is a new generation extended from the DEVSJAVA simulator and the DEVS Tracking Environment. The main modules of the `DEVS-Suite` are as follows:

- *Simview* [1]: Simview which was developed for the DEVSJAVA supports component viewing of DEVS source code as block components. In `DEVS-Suite`, execution of the models can be animated in terms of the input/output messages for coupled models and the state changes for the atomic models. The simulation experiments can be triggered with test inputs – predefined inputs for every model can be selected from its dialog box at the beginning of the simulation. At the end of the

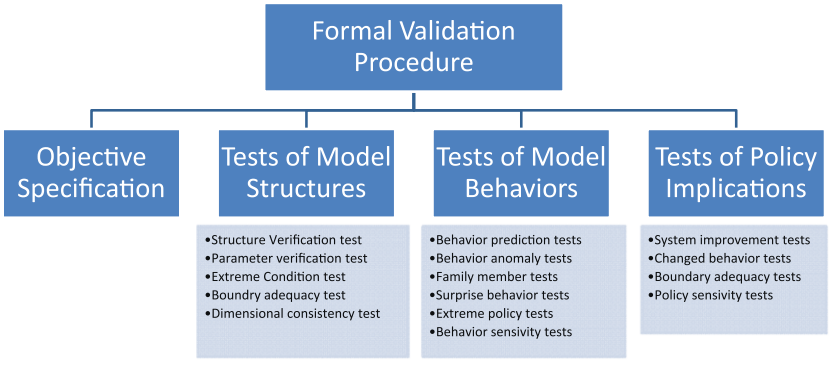

Fig. 2. Verification and validation methods of Forrester and Senge [10].

simulation, user-defined statistical metrics for any of the model components can be obtained with the so-called transducer models. The simulator also has an option window while loading the model, which includes simview and tracking options.

- *Tracking Environment* [24]: The architecture of the DEVS-Suite simulator environment is the Model Facade View Control (MFVC) [24] by which simulation data can be displayed with its animation and viewing of time trajectories generated by the parallel DEVS abstract simulator.
- *Timeview* [23]: Every atomic and coupled model component can have its own time-based trajectories and log files for inputs and outputs as well as the common phase and sigma state variables for atomic models.

The DEVS-Suite has some important treats for successful validation. The DEVS-Suite has very powerful and robust simulation engine which enables the modeler to simulate flexible application of a model's steps. It also provides easy access to parameters/settings via menus and dialog boxes in the simview. The DEVS-Suite's object orientation supports easy update and/or rerun of the model chain. Its tracking capability provides more detailed logging/tracking of operational assumptions, easy replication of simulation runs. Simulation data can be presented in tables, logs and files.

The DEVS-Suite has integrated reporting and validation table generation and support batch or step by step operation. In Fig. 3, DEVS-Suite and its associated technologies are depicted.

2.3. Ns-2 network simulator

The open source network simulator ns-2 [19] is an object-oriented, discrete event network simulator. It is written in C++ and uses OTcl as a command and configuration interface. It is focused for research studies in local and wide-area network protocols. This simulator implements network transmission protocols such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), traffic source behavior such as File Transfer Protocol (FTP), Telnet, Web, CBR and VBR, router queue management mechanism such as DropTail, RED and CBQ, and other routing algorithms [19]. It supports simulation of TCP, routing, and multicast protocols for wired and wireless network models and is primarily used for simulating local and wide area networks. The network simulator ns-2 supports modeling, simulation, and visualization of the network technologies and functionalities:

- Wired, wireless and satellite networks with their routing algorithms such as DV, LS, PIM-DM, PIM-SM, AODV, and DSR [8,27].

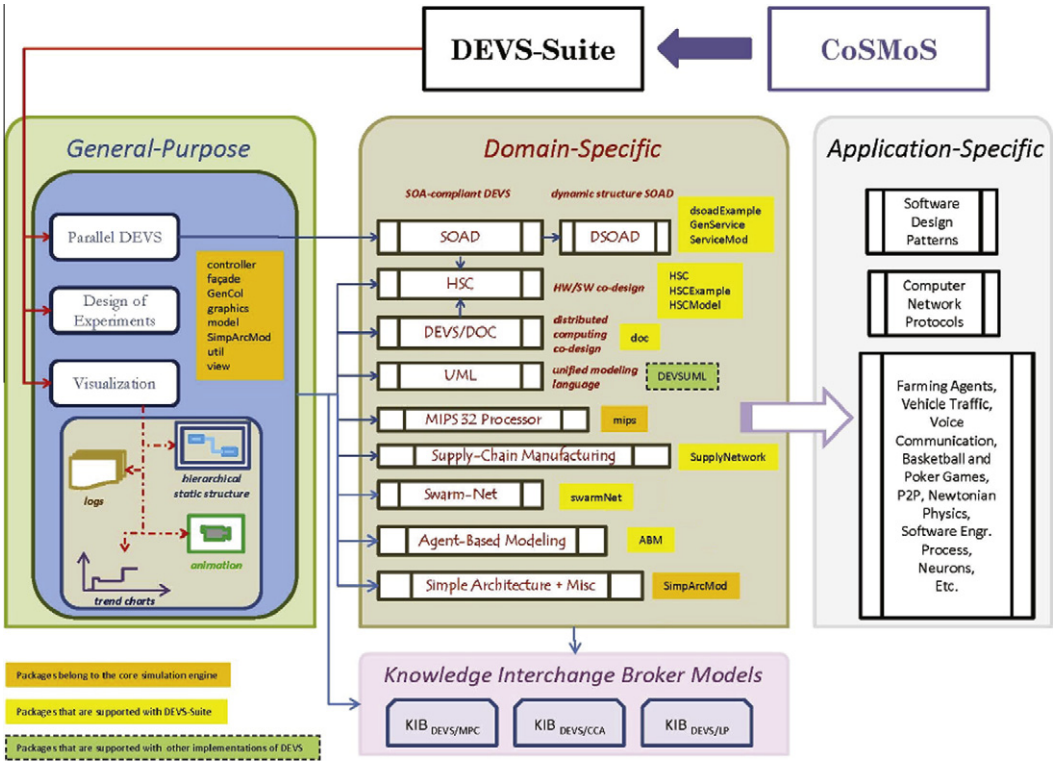

Fig. 3. DEVS-suite simulator infrastructure [23].

- User traffic sources such as HTTP, FTP, Telnet, and CBR.
- Failure models such as deterministic, probabilistic loss, and link failure.
- Queuing disciplines such as DropTail, RED, FQ, SFQ, and DRR and QoS such as IntServ and Diffserv.
- Packet flow, queue build up, and packet drops.
- Protocol behavior: TCP slow start, self-clocking, congestion control, fast retransmit, and recovery.
- Node movement in wireless networks.
- Annotations to highlight important events.
- Protocol state (e.g., TCP congestion window).
- Automatic traffic and topology generation.

The `ns-2` is usually used for design, testing, and comparison of new network algorithms, protocols and technologies. It is based on the seven-layer network synthesis and designed as packet-based, which means that all packet interactions are in focus. The simulator becomes relatively straightforward to use once the users gain an in-depth knowledge of its concepts and implementation. However, development of the new models can be challenging due to factors such as few user-friendly manuals and complicated installation. The `ns-2` simulator is said to be extended with new visualization and other capabilities to support teaching computer network theory and principles [11]. However, details describing such capabilities are not provided. The `ns-3` simulator, the next generation has been under development since 2006 and the latest stable release 3.14 was available by June 5, 2012 [20].

2.4. OSPF protocol overview

Routing protocols in the network systems can be split into two main categories: *link state routing* and *distance vector routing*. Currently, and in particular for the Internet, while distance vector protocols are used for inter-gateway interactions, link state protocols are used for intranet [26]. Open Shortest Path First (OSPF) is an important link state routing protocol. It is an open standard routing protocol and is an efficient interior gateway (IGP) routing protocol that is faster than the routing information protocol (RIP), which belongs to the distance vector protocols family. It employs the Dijkstra algorithm while estimating the shortest paths [7]. The OSPF protocol is defined to have three phases in order to capture its behavior as a discrete event model. These phases are described as follows:

- *Startup phase*: As soon as a router connects to the network, it broadcasts the Hello messages to all neighbors, later each router tabulates own neighbor tables.
- *Update phase*: Each router casts an update message at fixed time intervals called "link state advertisements (LSA)" by which topology database is formed. All routers must have same topology database of a local network.
- *Shortest path tree phase*: Routers then estimate a "shortest path tree" that depicts shortest paths to every destination routers.

3. Validation tests of the OSPF–DEVS model

Validation is the most important phase in development of a model. Validation tests are needed in a modeling study since a model cannot be accepted unless it passes them [5,3]. Validation schemes are generally framework based methodical procedures as well as dynamic processes which have to be applied by the modeler during the model development. There are many model validation schemes summarized in [5,17]. From these schemes, a widely approved validation scheme is presented in [10] and followed in this study. In this scheme, confidence in the model increases as model passes more tests. This validation scheme is adopted and applied in this study due to its appropriateness for dynamic models such as distributed systems and networks. Applied validation scheme is mainly divided into four phases. These phases are (1) specifying model objectives, (2) validating the model structure, (3) validating the model behavior and (4) policy implications.

Adopted approach includes performing each of the recommended tests above as fully as possible in the DEVS-Suite, documenting and reporting the results with each test's evaluation, assessing and capturing the number of insights gained from the test and contrasting the tests in terms of their benefits/costs. In the following sections, these steps are followed to validate the applied model and summarized.

3.1. Test simulation environment and configurations

The OSPF model is applied to several processes to improve its confidence. These processes are summarized in previous sections. In these processes, some environmental parameters are configured similarly as close as possible to the real world settings. Section 3.3.2 briefly tabulates these configuration parameters and Table 2 compares selected parameters with the real world Internet network parameters. By doing so, it is claimed that simulation experiments are carried out under some sense of reality, though exact reality cannot be established on limited capacity computing environments (for example, large-scale network simulations have been scaled up to hundred thousands size, whereas Internet has billions size computers [6]). Table 1 shows simulation experiment parameters in V&V tests such as used environment and approaches. While comparing

Table 1
Simulation model parameters of ns-2 and DEVS-Suite.

Parameters for verification and validation process	
Approach	Forrester and Senge [10]
Structure verification	OSI layers
Parameter verification	Internet
Traffic density	442 Kbps to 4.9 Mbps
Network size	4–10 thousands nodes
Behavioral validation tools	DEVS-Suite 2.1 and ns-2 2.32
Used machine	2.66 GHz Core2Duo processor and 4 GB RAM

Table 2
OSPF–DEVS simulator parameters and Internet.

Parameters	OSPF–DEVS simulation	Internet
Exterior routing protocol	BGP	BGP
Interior routing protocol	OSPF	OSPF & RIP
IP address model	IPv4	IPv4 & IPv6
IP address size	4 Byte	4 Byte
Message type	Advanced IP Packet	IP Packet
MTU	552 Bytes	576(fragmented) to 65,548 Bytes(unfragmented)
Header size	20 Bytes	20 Bytes
LSA max age	Infinity	1 h
Number of hops	15 hops	15 hops
Node processing time	1 s	1 Mb/s to 40 Gb/s
Queue length	200 KB (362 packets)	32–500 packets
Link bandwidth	25 Mb/s	25 Mb/s to 10 Gb/s
Link delay	3 ms	3–16 ms
Layer	2	2 (Internet core and autonomous systems)
Scale	10,000 nodes/hosts	Hundreds of millions of hosts and users
Model	Waxman	N/A
Observation time	1000 s	N/A
Inter arrival time	1 s	N/A

the OSPF model, some small differences exist because of different internal parameter settings and/or different level of modeling detail in the simulation tools.

3.2. Model objectives specification

It is important to set objectives correctly when starting to model a system. In the previous sections, some background about motivation and problem entities are given. Once again, it is good to list main objectives to model such a system. It is aimed to develop a network simulator having such kind of properties;

- system theoretic design with modular and hierarchy,
- easy to deploy,
- good visualization,
- high performance and scalable,
- highly tractable,
- advanced testing framework,
- parallel and distributed capability.

Except for the last one, the model has all properties above. Parallel and distributed capability of the model is still ongoing research and will be done by DEVS/HLA framework [25].

3.3. Validation of the OSPF–DEVS model structure

Structure validation tests are performed during the modeling process in which the modeler tests the model during build up phase. These tests are very crucial in order to design a reasonable and correct model. These tests require to refer theoretical definitions and knowledge [2]. Empirical and theoretical structure tests are experimented with the model to show its structural confirmation. These experiments are separated into two categories: (1) network structure and (2) the OSPF confirmation tests. Network structure confirmation test was accomplished by several theoretical tests including appropriateness of the OSPF model to network theory, for example how Open System Interconnections (OSI) layers are represented on

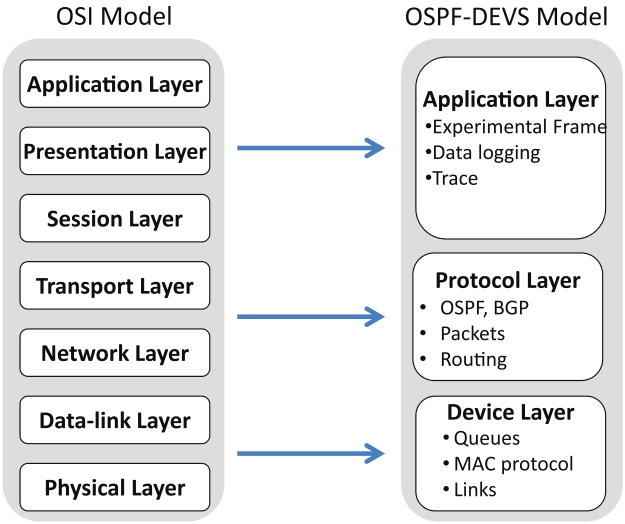

Fig. 4. OSI network layers mapping to OSPF–DEVS.

the DEVS-Suite. OSPF protocol confirmation tests are performed by empirical tests and the OSPF protocol is confirmed via graphs and tabular data. Empirical tests of the OSPF–DEVS model includes comparison of the OSPF model structure with information obtained directly from the real system.

3.3.1. Structure verification tests

Theoretical structure tests of the OSPF model are carried out with comparison model structure with generalized knowledge about the system. The DEVS network model is developed in [33] according to the network OSI standard with several abstractions. Since protocol implementation and education are in focus, first abstraction is to flatten seven segment OSI layers to three layers (see Fig. 4).

3.3.2. Parameter verification tests

Just after the structure verification test that resembled the target system, parameters are configured to form model behavior that synchronous to reference data. This process is highly required and the benefits obtained from these tests were high. This kind of verification includes comparison of the model parameters with the real system [10]. In Table 2, the OSPF–DEVS model parameters are compared to the Internet system. For example, queue lengths, delays and bandwidths are selected as real as Internet network. However, some parameters such as scale are not modeled since computational devices are limited as depicted in boundary adequacy test. Besides, IP address and data packets are very realistic.

3.3.3. Extreme conditions tests

Extreme conditions test is performed to reveal simulated system's normal operating boundaries. Extreme conditions test is important in terms of the model structure and behavior. It can be used for detecting model structure defects and also can be used for enhancing model's usefulness by specifying operating conditions [10].

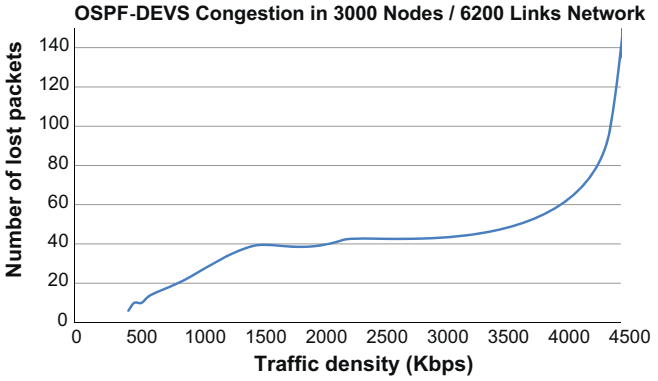

Fig. 5. OSPF–DEVS extreme condition test giving boundaries of the simulated model.

In the OSPF case example, it is tried to determine operating conditions by experimenting the model in heavily conditions. An example network having 3000 nodes and 6200 links is experimented under varying traffic conditions ranging from 442 Kbps to 4.9 Mbps. As seen in Fig. 5, the OSPF–DEVS model deserves its stability till approximately 4.4 Mbps traffic density. After that value, the model looses its preciseness and correctness. For example, packet lost ratio was 96% in heavily traffic condition test.

Extreme condition tests are tightly related with sensitivity analysis. Upper and lower limits for the traffic density were established. Fig. 5 shows the lower and upper limits for the OSPF–DEVS model. When the value for a traffic density is set outside of 4.4 Mbps, the model behavior cannot be relied upon. These tests are very crucial and give much knowledge about clarification of the model's useful domain of applicability.

3.3.4. Boundary adequacy tests

Boundary adequacy tests examine the model boundary for the objectives specified at the beginning. First of all, it is important to determine the boundaries of the model. Inspecting model diagrams, model equations may help to identify constants affected the system. At the beginning of the model design, several assumptions and decisions are made for expected outcomes when specific inputs are given. The benefits from these tests were high (see Table 5).

3.3.5. Dimensional consistency tests

Dimensional consistency test copes with dimensional analysis of the OSPF–DEVS model. First, network models up to ten thousands nodes are developed and measured the efficiency of the networks. The efficiency is estimated as performed network tasks such as message delivery, scheduled events and routing databases' correctness. In Fig. 6, the efficiency trajectory of the OSPF model is shown. For small scale models, the efficiency is ideal (i.e. simulator is running with highly correctness), however in large models, simulator is deviating from reality (for example, 99.5% for 10,000 nodes).

To show dimensional consistency of the OSPF–DEVS simulator, its resource consumption is measured and compared to the memory demands of several simulators. From these values, how much is being used to describe the traffic and architecture can be revealed. Nicol [18] reports that ns-2 demands 52.2 KB per connection, and the JavaSim demands 17.3 KB per connection. In Table 3, these values are compared to the OSPF–DEVS Simulator. Generally, memory demands are caused by routing table implementation and visualization. Due for lightweight Java GUI capability, DEVS visualization load is surprisingly low in the OSPF–DEVS simulator.

3.4. Validation of the OSPF–DEVS model behavior

Besides structural validity tests, certain tests are needed to measure how accurately the model can reproduce the major behavior patterns shown by the real system. These tests are performed both with the DEVS-Suite Tracking Environment and the ns-2. The Tracking Environment provides detailed imprints of the model by means of time diagrams.

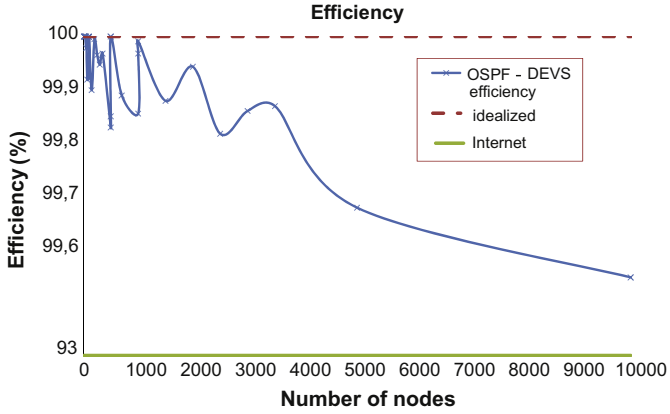

Fig. 6. Efficiency tests of the OSPF model.

Table 3
Per-connection core memory demands of the simulators.

Simulation tools	Per connection memory footprint (KB)
JavaSim	17.3
Ns-2	52.2
OSPF–DEVS (with SimView)	39.5
OSPF–DEVS (without SimView)	31.6

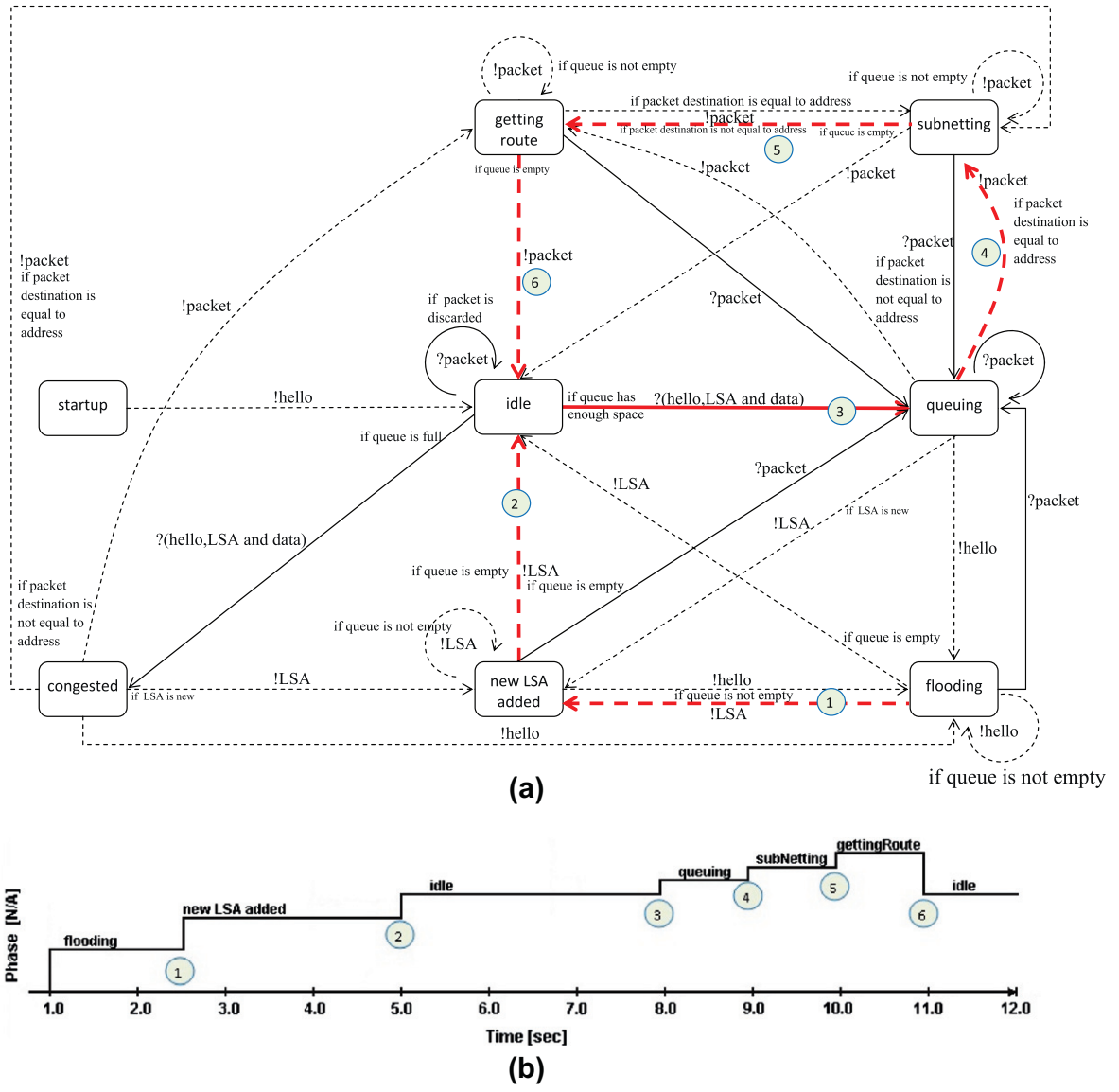

Fig. 7. OSPF–DEVS reproduction of the node's behavior.

3.4.1. Behavior reproduction tests

Behavior reproduction tests help to evaluate degree of matches between the model-generated behavior and the real system. These tests include symptom generation, frequency generation, relative phasing, multiple mode and behavior characteristic tests [10]. In Fig. 7, the behavioral reproduction comparison is given. In Fig. 7, a finite state machine of a node atomic model is shown. This state trajectories are formed theoretically in designing of such a network model. Fig. 7b is from time-view facility from the DEVS-Suite environment showing state transitions of a node's run. Fig. 7 proves that in the OSPF–DEVS simulator, the node model exactly follows the states as specified in its definition. In other words, node correctly mimics and reproduces behavior as such in its definition.

Another behavior reproduction test is done in comparison with the ns-2 network simulator. In order to validate behavior of the OSPF–DEVS model, first a small scale topology is considered (four routers and links). In this study, widely accepted network simulator 2 (ns-2) is selected instead of a real system, since the selected simulator has enough detail in the network domain. Furthermore, same configurations are used in the experiments in order to show the key structural differences between modeling the `DEVS-Suite` and the `ns-2` (see Table 1). Finally, comparison with analogous ns-2 test traces is included.

According to the configuration parameters listed in Table 1, simulation experiments with both the `ns-2` and the `OSPF-DEVS` simulators are performed for 10 s. The throughput and end to end delay results as a function of time are shown in Fig. 9.

As depicted in Fig. 9a, after a stabilization phase time (2 s), throughput curves converge to nearly the same average values, 822.4 KB/s. for the ns-2 and 1489 packets × 552 bytes = 821.9 KB/s. for the OSPF-DEVS. This shows that both simulators produce same outputs for same traffic configuration.

Another important performance metric is end-to-end delay. End-to-end delay is a time value as a packet's life time from its creation time in source node to its destination. Different from throughput trajectories, difference between end-to-end delay values was bigger than expected (see Fig. 9b). In the OSPF-DEVS environment, the router model is designed based on a simple processor model with a queue. The modeler selects the router model's processing time. Routing delay is a time period that router takes to route a packet and is equal to total passed time needed for evaluating the packet header, checking the routing table which varies with table size, and assigning the packet to its intended output port of the router model. Unlike OSPF-DEVS, ns-2 uses a single virtual clock to execute events that are scheduled in a list. By considering the part of the trace file in Fig. 8, the first line says at time 0.240667, router 3 handles a CBR packet which it receives. The second line depicts at the same simulation clock time that router 3 also makes available the processed packet to a link. Thus, no time is consumed from the time the router receives a packet to the time the packet is made available on the link – i.e., the routing delay time caused by process load of the router processor is not accounted for. Consequently, difference in Fig. 9b comes from

Event	Time	From node	To node	Pkt type	Pkt size	Flags	Fid	Src addr	Dst addr	Seq num	Pkt id
r	0.240667	2	3	cbr	1000	-------	2	1.0	5.0	0	0
+	0.240667	3	5	cbr	1000	-------	2	1.0	5.0	0	0
-	0.240667	3	5	cbr	1000	-------	2	1.0	5.0	0	0
r	0.286667	3	5	cbr	1000	-------	2	1.0	5.0	0	0

Fig. 8. A snippet of the ns-2 trace file.

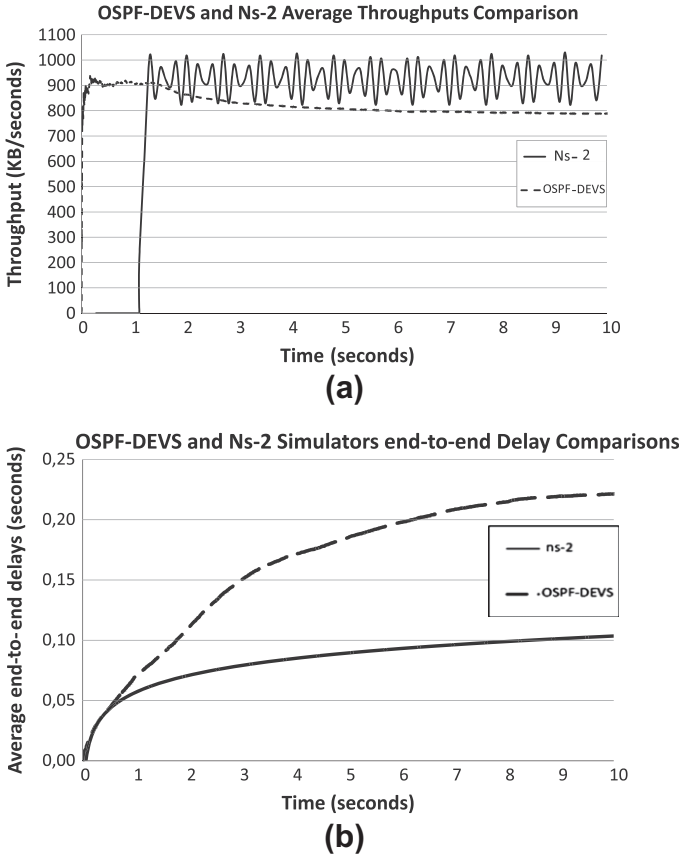

Fig. 9. Simulated performance measurements of ns-2 and OSPF-DEVS for throughput and delay.

Table 4
Event frequency of the selected simulators.

Simulator	DEVS-Suite	Ns-2
Number of events per second	1000	28,388

Table 5
Evaluation of the V&V test results.

V&V test	Difficulty	OSPF–DEVS success	Total benefit (success/difficulty)
Structure verification	L	H	VH
Parameter verification	M	H	H
Extreme condition	M	M	M
Boundary adequacy	L	M	H
Dimensional consistency	H	H	M
Behavior reproduction	H	VH	H
Ns-2 throughput comparison	M	H	VH
Ns-2 delay comparison	M	L	VL

VH – very high, H – high, M – medium, L – low, VL – very low.

processing time value of routers in the OSPF–DEVS. This value is zero in the ns-2 simulator while it is a value bigger than zero in the OSPF–DEVS simulator. Therefore, average end-to-end delays are different.

Routing tables are also observed for four router nodes to be consistent – the creation of tables was validated step by step.

In Table 4, event resolution of the DEVS-Suite and the ns-2 simulators are shown. Since ns-2 has lower level of abstraction it supports more realistic simulation of packets and their interactions, therefore large number of events are generated and stored. This is not suitable while scaling large networks. The DEVS formalism and its associated tools such as DEVS-Suite allow the modeler to create models with higher level of abstraction which leads to focus big picture omitting redundant details. By this fact, after completion of the simulation study, the total number of events (event frequency) in the ns-2 is greater than DEVS-Suite by a factor of 30 (i.d. DEVS can do same task by generating 30 times less events than ns-2).

4. Evaluation

In order to evaluate outcomes from verification and validation tests performed with the OSPF–DEVS simulator, some criteria are taken into consideration. Evaluation of the OSPF–DEVS simulator is the process of determining the appropriateness of it [29]. Evaluation is handled by three topics. These topics are test results, modeling approach and educational use evaluations. Test results evaluation includes the assessments of the capability, software accuracy, data accuracy and results accuracy of the OSPF–DEVS simulator.

4.1. Test results evaluation

In Table 5, results from OSPF–DEVS V&V tests are evaluated. Evaluation is done in terms of their difficulty and the OSPF–DEVS performance. Difficulty of a specific test is related to total effort, hardness and required time to perform it. The OSPF–DEVS model success is related to how OSPF model meets model objectives specified. Later, total benefit is calculated by a ratio of these values. Table 5 shows that most of these tests are beneficial except from end-to-end delay comparison tests. Since DEVS-Suite allows the modeler to develop high quality models, these results are not surprising.

4.2. Modeling approach evaluation

As noted earlier, OSPF–DEVS model emphasizes routing by supporting the packet-level abstract. This high level of abstraction affords better execution efficiency, for example when compared with the `ns-2`. This is due in part to having significantly less number of events generated and processed. DEVS approach allows the modeler to create highly tractable models. DEVS approach is developed as an extension of the Moore machine formalism [31]. DEVS formalism is a kind of finite state automaton approach in which states and outputs are created by the current state and the inputs. As contrary to other finite state approaches, DEVS has brought a new concepts to the simulation community such as state lifespan and a hierarchical concept. These innovations have rendered possible to create very efficient simulation software. In the DEVS component connection architecture, a basic entity is a component called `atomic model`. Each atomic model owns one or more communication ports. By the way, two atomic models are connected by "wiring" their ports together. DEVS formalism relies on network models to facilitate entity or message movement and decision making.

4.3. Educational usability evaluation

Ease of use of the simulator is tightly related to the question "Is the simulator user friendly and do the students benefit from the simulator?". In [35], the evaluation of the OSPF–DEVS simulator in classroom settings is presented. In that work, to evaluate the OSPF–DEVS simulator, graduate students of computer network course are asked to use it and the ns-2 simulator in three homework assignments. The students' evaluations and analysis have shown the extent to which each of these tools can support computer network protocol education. Following sections evaluate the simulator in terms of visualization, web access, reusability, deployment and user base.

4.3.1. Visualization

Visualization support in the OSPF–DEVS application includes tracking the creation and update of the neighbor table, LSA database, and finally, the routing table that is calculated using the shortest path algorithm. Although tools are provided to track the changes on routing tables, this level of visualization is not supported in the `ns-2` and it is difficult to add such capability due in part to the use of both C++ and Tcl programming languages. Using the `DEVS-Suite`, students can track and view the logic behind the routing protocol as well as the discrete event-based run of the network system.

4.3.2. Web access

The common way simulators are made available to students is to install them on personal computers or computers in laboratories at the universities. To avoid basic difficulties such as software installation and upgrades, it is useful to make simulators web accessible. This is desirable in addition to making simulators available to users based on anyplace and anytime principle. Using Java Web Start technology, the simulator can be run over the Internet, therefore it can be used for distance learning applications.

4.3.3. Reusability

Code reuse of the simulation models and components is a very good way to increase model development productivity and application maintainability. Instead of developing all the things, students should always look for well-designed models, application frameworks and customizable components. Reusability of the OSPF models in the `DEVS-Suite` environment allows students to make rapid changes, exploit new features globally, and spend less time testing and debugging in course practices.

4.3.4. Deployment

One of the problems in network education courses having practice with the `ns-2` is installation and deployment. Students are getting bored while installing the ns-2 network simulator on their machines. One of the reasons is that `ns-2` is using two programming language basis and hence requires infrastructure programs over tens. Another reason emerges from its hard platform-dependent structure. In our simulator, deployment is very easy with a executable jar file and it is also platform independent since Java has many running platforms.

`DEVS-Suite` has not problems neither in terms of background technology which is just need for the Java Virtual Machine, nor in terms of platform dependency which Java is already cross-platform technology. Consequently, process of managing and automating the packaging, testing, distribution and installation of the `DEVS-Suite` application even for its online version is easier and more affordable than the `ns-2`.

4.3.5. User base

Finally, the scale of the user-base of a tool is important. This is because large user communities help not only to sustain the simulator up-to-date, but also to improve and add new capabilities to it. `DEVS-Suite` is an open source project and the DEVS community continues to grow.

5. Conclusion and future work

According to the structural and behavioral validation tests, OSPF–DEVS simulator gives correct results and the logic of the system is also correct (see Table 5). The designed simulator also offers an improvement over the practices intended to supplement. Simulator is easy to learn and to become proficient on, and the simulator is useful as a training tool. Simulator is in fact maintainable by other modelers than the developers.

Structural tests and behavioral validation tests give at least some assurance that the structure and behavior of the OSPF–DEVS model is reasonable. DEVS-Suite more suitable for system level simulation as opposed to the ns-2 being suitable for the detailed network protocol designs (algorithms). The results of this study should help further development of the network simulator tools and may also lead to development of new kinds of simulators for the computer networks and possibly other types of the complex systems.

Future work includes developing automated V&V tools and evaluating them through some test cases. Additionally, the OSPF–DEVS simulator can be extended with data analysis features and more detailed network layer components. Another desirable addition is to use variable structure DEVS since in some cases it can better represent the dynamic nature of the

distributed computer networks. Another attractive capability is to visually develop models and automatically generate source code using the Eclipse and some other tools.

Acknowledgment

This work has been funded by Sakarya University Scientific Research Projects Agency under Contract 2007-05-02-001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Sakarya University.

References

[1] ACIMS, DEVSJAVA Modeling and Simulation Tool, 2012. <http://www.acims.arizona.edu/SOFTWARE>.
[2] O. Balci, Handbook of Simulation, Verification, Validation and Testing, John Wiley & Sons, New York, NY, 1997 (Chapter 10).
[3] O. Balci, R.G. Sargent, Validation of multivariate response models using Hotellings two-sample t2 test, Simulation 390 (6) (1982) 185–192.
[4] O. Balci, R.G. Sargent, Validation of simulation models via simultaneous confidence intervals, American Journal of Mathematical and Management Sciences 4 (3 and 4) (1984) 375–406.
[5] Y. Barlas, Formal aspects of model validity and validation in system dynamics, System Dynamics Review 12 (3) (1996) 183–210.
[6] J. Cowie, A. Ogielski, D. Nicol, The SSFNet Network Simulator, 2012. <http://www.ssfnet.org/homePage.html>, Renesys Corporation. <http://www.ssfnet.org/homePage.html>.
[7] E. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.
[8] J. Flood, Telecommunications Switching, Traffic and Networks, Prentice Hall, 1998.
[9] I.E.T. Force, Ospf Version 2 (rfc 2328) Internet Standards Track Protocol, 2009. <http://www.ietf.org/rfc/rfc2328.txt>.
[10] J.W. Forrester, P.M. Senge, Test for building confidence in system dynamics models, TIMS Studies in the Management Sciences 14 (1980) 209–228.
[11] J. Guo, W. Xiang, S. Wang, Reinforce networking theory with opnet simulation, JITE 6 (2007) 215–226.
[12] X. Hu, B.P. Zeigler, S. Mittal, Variable structure in DEVS component-based modeling and simulation, SIMULATION: Transactions of The Society for Modeling and Simulation International 81 (2) (2005) 91–102.
[13] D. Khazanchi, A framework for the validation of is concepts, in: Second Annual Association for Information Systems Americas Conference, Phoenix, Arizona, August 1996.
[14] S. Kim, H. Sarjoughian, V. Elamvazhuthi, DEVS-Suite: a simulator supporting visual experimentation design and behavior monitoring, in: Proceedings of the Spring Simulation Conference, San Diego, CA, March 2009, pp. 29–36.
[15] Y. Labiche, G. Wainer, Towards the verification and validation of devs models, in: Proceedings of 1st Open International Conference on Modeling and Simulation, 2005, pp. 295–305.
[16] C.M. Macal, Model verification and validation, in: Workshop on Threat Anticipation: Social Science Methods and Models, The University of Chicago and Argonne National Laboratory, Chicago, IL, April 7–9, 2005.
[17] M.S. Martis, Validation of simulation based models: a theoretical outlook, The Electronic Journal of Business Research Methods 4 (1) (2006) 39–46.
[18] D.M. Nicol, Scalability of network simulators revisited, in: Proceedings of the Communications Networking and Distributed Systems Modeling and Simulation Conference, 2003.
[19] NS2, The ns-2 Network Simulator, 2012. <http://www.isi.edu/nsnam/ns/>.
[20] NS3, The ns-3 Network Simulator, 2012. <http://www.nsnam.org/>.
[21] D.K. Pace, Modeling and simulation verification and validation challenges, Johns Hopkins APL Technical Digest 25 (2) (2004) 163–172.
[22] S. Robinson, Simulation model verification and validation: increasing the users' confidence, in: Proceedings of Winter Simulation Conference, 1997.
[23] H. Sarjoughian, DEVS-Suite Simulator, 2012. <http://sourceforge.net/projects/devs-suitesim/>.
[24] H.S. Sarjoughian, R. Singh, Building simulation modeling environments using systems theory and software architecture principles, in: Proceedings of the Advanced Simulation Technology Conference, Washington DC, April 2004, pp. 99–104.
[25] H.S. Sarjoughian, B.P. Zeigler, DEVS and HLA: complementary paradigms for modeling and simulation?, Transactions of the Society for Modeling and Simulation International 17 (2) (2000) 187–197.
[26] M. Steenstrup, Routing in Communications Network, Prentice-Hall, 1995.
[27] A.S. Tanenbaum, Computer Networks, third ed., Prentice Hall, 1996.
[28] Validation, US Department of Defence RPG Special Topic, 2004. <http://vva.msco.mil/Special_Topics/Validation/validation-pr.pdf>.
[29] W. Wakeland, M. Hoarfrost, The case for thoroughly testing complex system dynamics models, in: International System Dynamics Society Conference, Boston, MA, July 2005.
[30] R.B. Whitner, O. Balci, Guidelines for selecting and using simulation model verification techniques, in: E.A. MacNair, K.J. Musselman, P. Heidelberger (Eds.), Winter Simulation Conference, IEEE, Piscataway, NJ, 1989, pp. 559–568.
[31] B. Zeigler, Object Oriented Simulation with Hierarchical, Modular Models, Academic Press, 1990.
[32] B.P. Zeigler, H. Praehofer, T.G. Kim, Theory of Modeling and Simulation, Academic Press, New York, 2000.
[33] A. Zengin, Large-scale integrated network system simulation with DEVS-suite, KSII Transactions on Internet and Information Systems 4 (4) (2010) 452–474.
[34] A. Zengin, Modeling discrete event scalable network systems, Information Sciences 181 (5) (2011) 1028–1043.
[35] A. Zengin, H. Sarjoughian, Devs-suite simulator: a tool teaching network protocols, in: Winter Simulation Conference, Baltimore, Maryland, US, December 5–8, 2010.