



A Pedagogical Methodology for Active Learning of DEVS Concepts

Item Type	text; Electronic Dissertation
Authors	Ariyananda, Lahiru
Citation	Ariyananda, Lahiru. (2022). A Pedagogical Methodology for Active Learning of DEVS Concepts (Doctoral dissertation, University of Arizona, Tucson, USA).
Publisher	The University of Arizona.
Rights	Copyright © is held by the author. Digital access to this material is made possible by the University Libraries, University of Arizona. Further transmission, reproduction, presentation (such as public display or performance) of protected items is prohibited except with permission of the author.
Download date	20/12/2022 11:01:23
Item License	http://rightsstatements.org/vocab/InC/1.0/
Link to Item	http://hdl.handle.net/10150/667284

A PEDAGOGICAL METHODOLOGY
FOR
ACTIVE LEARNING OF DEVS CONCEPTS

by

Lahiru Ariyananda

Copyright © Lahiru Ariyananda 2022

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

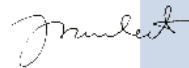
In the Graduate College

THE UNIVERSITY OF ARIZONA

2022

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Lahiru Ariyananda, titled A Pedagogical Methodology for Active Learning of DEVS Concepts and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.



Jerzy Rozenblit

Date: November 29, 2022



Tosiron Adegbiya

Date: November 29, 2022

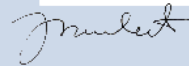


Kelly Potter

Date: November 29, 2022

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

We hereby certify that we have read this dissertation prepared under our direction and recommend that it be accepted as fulfilling the dissertation requirement.



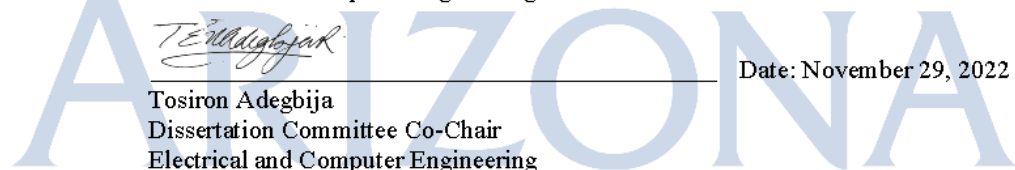
Jerzy Rozenblit
Dissertation Committee Co-Chair
Electrical and Computer Engineering

Date: November 29, 2022



Tosiron Adegbiya
Dissertation Committee Co-Chair
Electrical and Computer Engineering

Date: November 29, 2022



ACKNOWLEDGEMENTS

I am in much gratitude to Dr. Jerzy Rozenblit, Dr. Tosiron Adegbija, and Dr. Kelly Potter for the understanding, guidance, and especially the patience shown during a very difficult personal and historical passage of time which was instrumental in my finishing this dissertation. My respect goes to them.

I also would like to thank my former advisors, Dr. Bernard Zeigler, as the dissertation title/material was motivated by his initial work, and Dr. Roman Lysecky for his kind guidance and mentoring to complete my remaining Doctoral exams and for helping me to formulate and conduct a formal experimental study.

I will fail in my duty if I also do not specially mention Dr. Janet Sturman, former Associate Dean of Graduate College, and her present successor, Dr. Margaret Pitts, for their kind help and support.

DEDICATION

This dissertation is dedicated to my beloved Father, PF Ariyananda, who was my motivation and beacon of light till he was taken away from me at an early stage in my life. His dream was for me to get the higher educated as possible. Thus, I dedicate this effort to his fond memory.

TABLE OF CONTENTS

Contents

LIST OF FIGURES	9
LIST OF TABLES	11
ABSTRACT	12
CHAPTER 1	14
INTRODUCTION	14
1.1 What is Modeling and Simulation?.....	15
1.1.1 What is a Model?	15
1.1.2 What is an Entity?.....	16
1.1.3 What is a System?.....	16
1.1.4 What is Simulation?.....	18
1.2 Need for Modeling and Simulation.....	20
1.3 Challenges in Modeling and Simulation.....	21
1.4 A New Pedagogical Approach to M&S Using FDDEVS.....	22
CHAPTER 2	23
BACKGROUND	23
2.1 Current State of Modeling and Simulation Education.....	23
2.1.1 Broad Categorization of Modelers & “Simulationists”	23
2.1.2 M&S in Industry	24
2.1.3 Modeling & Simulation as a Discipline	27
2.1.4 M&S Education at University Level	29
2.2 Existing M&S Formalisms, Frameworks, and Finding a Base for M&S Education	33
2.2.1 Types of M&S Frameworks and Formalisms.....	33
2.2.2 Why Was DEVS Chosen?	36
2.2.3 Discrete Event System Specification.....	37
2.3 Existing DEVS Pedagogical Approach	41

2.3.1	Traditional DEVS Learning Approach.....	41
2.3.2	Challenges	43
2.3.3	Need for a Newer Approach.....	44
CHAPTER 3		45
INTRODUCTION TO FDDEVS, SYSTEM ENTITY STRUCTURES AND THE NEW PEDAGOGICAL APPROACH.....		45
3.1	DEVS and M&S Education Challenges Revisited	45
3.2	Motivations for this Research	46
3.3	A Brief Introduction to FDDEVS and Its Extensions.....	47
3.4	System Entity Structures.....	49
3.5	The Usage of FDDEVS in the New Pedagogical Approach.....	51
3.6	The Research Plan and Methodology	54
CHAPTER 4		56
PRELIMINARY IDENTIFICATIONS, REQUIREMENTS, AND guidelines FOR THE NEW PEDAGOGICAL APPROACH.....		56
4.1	Preliminary Content Requirement Assessment and Guidelines	56
4.1.1	Identification of Traditional M&S Tool-Based Education Shortcomings.....	56
4.1.2	Utilization of Hierarchy of System Specification Levels & a Student as a System..	57
4.1.3	Systems-Based Problem-Solving Types.....	59
4.2	Research Plan Task 1 : Formulating Background Insight for a Pedagogical Approach...	61
4.3	Research Plan Task 2 : GUI Development	62
4.4	Research Plan Task 3 : Guidelines for Active learning Content Development.....	63
CHAPTER 5		65
FDDEVS BUILDER GUI AND INSTRUCTIONAL TUTORIAL INTRODUCTION.....		65
5.1	Chapter Introduction	65
5.1.1	Pedagogical Material and Software.....	65
5.2	Supporting Software	66
5.2.1	FDDEVS Builder Software	66
5.2.2	FDDEVS Builder GUI and Software Features.....	67
5.2.3	Software Installation and Notes.....	71
5.3	Supporting FDDEVS Pedagogical Material	73
5.3.1	Material Organization.....	73
5.3.1.1	Introductory Material	74
5.3.1.2	Advanced/Intermediate Material.....	78
5.3.2	Content Distribution Formats.....	80

CHAPTER 6	82
FDDEVS ANIMATOR	82
6.1 FDDEVS Animator Introduction.....	82
6.2 FDDEVS Animator Dependencies, Work-Folders and Inputs	84
6.3 FDDEVS Animator Outputs.....	85
6.3.1 Analytical Info/Content Breakdown of an FDDEVS Input Model	85
6.3.2 Model Behavior Simulation via Auto-Executed OpenOffice Presentation Format ...	89
6.3.2.1 Model Behavior Simulation Without Selectable Options.....	89
6.3.2.2 Model Behavior Simulation With Selectable Options Enabled.....	92
6.3.3 Comparison of Two FDDEVS Models	97
6.4 More Features, Caveats, and Challenges Faced.....	102
6.4.1 Additional Features.....	102
6.4.2 Caveats.....	105
6.4.3 Challenges Faced.....	106
6.5 Utilizing the Tool for Effective Pedagogy/Learning and Future Work.....	107
CHAPTER 7	109
EXPERIMENTAL DESIGN, STUDY RESULTS, AND ANALYTICAL DISCUSSION.....	109
7.1 The Study Background	109
7.1.1 Study Goals.....	109
7.1.2 Study Prerequisites	109
7.2 Study Methodology.....	110
7.2.1 Participant Eligibility.....	110
7.2.2 Participant Recruitment	110
7.2.3 Participant Allocation	110
7.2.4 Study Administration.....	111
7.2.5 Study Sample Size and Expected Participant Time Commitment	111
7.3 Assessment Material	112
7.3.1 Study Introduction and Participant Academic Background	112
7.3.1.1 Study Group A - Following the Traditional Approach	112
7.3.1.2 Study Group B - Following the New Pedagogical Approach.....	114
7.3.2 Participant Knowledge and Skill Assessment	116
7.3.2.1 Study Group A	116
7.3.2.2 Study Group B.....	117
7.3.2.3 Knowledge Assessment Questions 1 -14 for Both Group A and B	118

7.3.3	Participant Feedback.....	124
7.4	Assessment Results and Analytical Discussion.....	125
7.4.1	Participant Background Assessment	125
7.4.1.1	Graduate vs Undergrad Participant Composition.....	125
7.4.1.2	Participant Familiarity with Object-Oriented and Java Programming.....	126
7.4.1.3	Participant Exposure to a Prior Simulation Course.....	127
7.4.2	Participant Knowledge Assessment.....	129
7.4.3	Participant Feedback.....	130
7.4.4	Study Results Analysis and Conclusion	132
7.5	Challenges Faced Concerning the Study	134
CHAPTER 8		136
CONCLUSIONS.....		136
CHAPTER 9		138
CHALLENGES FACED AND FUTURE WORK.....		138
9.1	Challenges Faced During this Research/Study.....	138
9.2	Future Work	140
APPENDIX A : IRB PROJECT RELATED DOCS		142
APPENDIX B : TUTORIAL SAMPLES.....		145
REFERENCES		170

LIST OF FIGURES

Figure 1 : Example graph to show continuous-time vs discrete-time changes.....	17
Figure 2 : M&S life cycle process [8, p. 18].....	19
Figure 3 : An infrastructure for the Establishment of M&S Discipline.....	28
Figure 4 : Basic systems specification formalisms.....	35
Figure 5 : Sample of research and development in DEVS.....	37
Figure 6 : DEVS Formalism with port specifications.....	38
Figure 7 : Coupled DEVS Specification/Formalism.....	40
Figure 8 : DEVS Simulation Protocol.....	40
Figure 9 : DEVS Toolkits.....	42
Figure 10 : Basic structural composition of a SES.....	51
Figure 11 : DEVSJava code of a Tennis Player.....	52
Figure 12 : The flow of the traditional approach for DEVJava M&S.....	52
Figure 13 : FDDEVS natural language description of the same model.....	53
Figure 14 : FDDEVS natural language model input and auto Java code generation.....	53
Figure 15 : Student as a system.....	61
Figure 16 : GUI interface.....	62
Figure 17 : FDDEVS Builder “Natural Language Tab.....	67
Figure 18 : FDDEVS Builder “SES” Tab.....	68
Figure 19 : FDDEVS Builder “SES with Prune” Tab.....	69
Figure 20 : FDDEVS Builder “Files” Menu option.....	69
Figure 21 : FDDEVS Builder “Tools” Menu option.....	70
Figure 22 : Sample content page from an installation tutorial.....	72
Figure 23 : Static screenshot from an interactive installation guide presentation.....	72
Figure 24 : Introductory material listed on the website.....	74
Figure 25 : Static screenshot from a dynamic presentation content slide.....	75
Figure 26 : Learning by example 1.....	76
Figure 27 : Learning by example 2.....	76
Figure 28 : Reminders drive home concepts better!.....	77
Figure 29 : Reminders to make people remember!.....	77
Figure 30 : Advanced topics found under the “SES” tab.....	78
Figure 31 : Advanced topics found under the “SES” tab.....	79
Figure 32 : Revisiting past concepts to refresh memories.....	79
Figure 33 : Revisiting past examples to refresh memories.....	80
Figure 34 : FDDEVS Animator added as an extension to FDDEVS Builder.....	83
Figure 35 : FDDEVS Animator GUI.....	83
Figure 36 : FDDEVS Animator “Choose File” option.....	86
Figure 37 : Selectable FDDEVS XML model list.....	86
Figure 38 : Analytical breakdown of FDDEVS model.....	87
Figure 39 : Detailed analytical info generated by FDDEVS Animator.....	88
Figure 40 : FDDEVS Animator presentation execution sans selectable options.....	89
Figure 41 : A static screenshot from the Tennis PlayerA presentation animation.....	90

Figure 42 : Another static screenshot from the Tennis Player A presentation animation	90
Figure 43: Software will generate an auto-mouse-click to demo an Internal transition.....	91
Figure 44: Presentation presets/selectable options	93
Figure 45: A stick figure is added to denote the “wait” state in “TennisPlayerA” model.....	94
Figure 46: A stick figure is added to denote the “play” state in “TennisPlayerA” model.....	94
Figure 47: Dynamically rotating gears were added to denote the “process(ing)” state.....	95
Figure 48: A dynamic stick figure was added to denote the “walk” state in “StoryofMan”	95
Figure 49 : Feature to compare 2 FDDEVS models	97
Figure 50 : Model selection and comparison interface.....	98
Figure 51 : 2 FDDEVS models are selected for comparison.....	98
Figure 52 : Clicking the “Compare Models” button to enable the model comparison feature.....	99
Figure 53 : Example analytical information generated by the model comparing tool.....	100
Figure 54 : Picking the same model 2 times demo they are identical.....	101
Figure 55 : Clear button feature	102
Figure 56 : A complex FDDEVS model.....	103
Figure 57 : A screen capture from the presentation of fig 56 model	104
Figure 58 : A screenshot with extra educational information is provided.....	104
Figure 59 : Error generated when no TA is defined	105
Figure 60 : Study Group A introduction and goal specification.....	113
Figure 61 : Question 1 on participant's academic background	113
Figure 62 : Questions 2-3 on participant’s academic background.....	114
Figure 63 : Group B Study introduction and goal specification	115
Figure 64 : Scanned introductory material from the Theory of Simulation	116
Figure 65 : Introductory material from the new pedagogical approach.....	117
Figure 66 : Knowledge assessment questions 1-3	118
Figure 67 : Knowledge assessment questions 4-5.	119
Figure 68 : Reference screen for questions 6-10	119
Figure 69 : Knowledge assessment questions 6-8	120
Figure 70 : Knowledge assessment questions 9-11	121
Figure 71 : Reference screen for questions 12-14	122
Figure 72 : Knowledge assessment questions 12-14	123
Figure 73 : Group A participant feedback questions	124
Figure 74 : Group B participant feedback questions	124
Figure 75 : Group A academic standing	125
Figure 76 : Group B academic standing	126
Figure 77 : Group A computer programming background	126
Figure 78 : Group B computer programming background	127
Figure 79 : Group A exposure to a prior Simulation course.....	127
Figure 80 : Group B exposure to a prior Simulation course.....	128
Figure 81 : Group A response to the question “was the traditional approach easy to grasp?” ...	131
Figure 82 : Group B response to the question “was the new approach easy to grasp?”	131

LIST OF TABLES

Table 1 : Possible applications of M&S and hypothetical questions to be addressed	21
Table 2 : Education for Industry Contrasted with Academia	25
Table 3 : Career growth of a Boeing real-time simulation engineer.....	26
Table 4 : Currently offered PhD programs in M&S	29
Table 5 : Currently offered Master's level programs in M&S.....	30
Table 6 : Currently offered Undergrad level programs in M&S	31
Table 7 : Natural Language Semantics and how to use them	48
Table 8 : Systems Specification Hierarchy vs Student as a System	58
Table 9 : System problem-solving types.....	60
Table 10 : Sample topics and learning objectives for DEVS/FDDEVS instruction.....	64
Table 11 : Group A participant responses.....	129
Table 12 : Group B participant responses.....	129
Table 13 : Legend for Tables 11 and 12	130
Table 14 : Group A participant scoring table with study completion times	133
Table 15 : Group B participant scoring table with study completion times	133

ABSTRACT

DEVS (Discrete Event System Specification) is a formalism that was introduced in the mid-1970s by Bernard Zeigler, for modeling and analysis of discrete event systems. DEVS is essentially a formal mathematical language for specifying complex systems through models that can be simulated and has been executed in object-oriented software, DEVSJava being prominent. In the present day, it serves as a robust and popular engine for simulation-based design in areas such as manufacturing, transportation, communication, military, etc. Here lies the first obstacle. DEVS and Modeling and Simulation, in general, is multi-disciplinary in nature. Any student or industry personnel who desire to learn modeling and simulation through DEVS will need a solid mathematical and programming background to get a good grasp. Though powerful as it is, there could be significant overhead involved in learning, and depending on the skill level of the learner it may even deter some from further studies. To date, DEVS is usually taught at universities as a Graduate level course.

Hence, the objective of this dissertation is to present a simplified pedagogical content/methodology that would cater to a wider audience including undergraduates, industry workers, or even high school students with minimal background in programming and mathematics to get exposed to modeling and simulation via DEVS concepts. Utilization of a unique software environment is introduced to define DEVS models via an innovative easy-to-learn natural language format that encourages active learning along with automatic model generation, testing, and experimentation. The broader educational aspects of this dissertation will lead to general understanding of ways to bridge the gap between concept acquisition and the use of the concepts,

and extend some of the learning to pay the way for use of DEVS in other fields of study beyond Engineering and Computer Science.

CHAPTER 1

INTRODUCTION

Imitation is the sincerest form of flattery they say. But in the world of Engineering and related Sciences, producing such “imitations” may go well beyond flattery, so much to save time, costs, and even lives maybe. Hence, Modeling and Simulation (M&S) is already a recognized area of study and practice, especially in Engineering Sciences for approximately representing (or imitating) real-world systems, entities, or phenomena. The NSF Blue Ribbon Panel on Simulation-Based Engineering Science report [1], which drew input from a wide constituency, showed the real potential of M&S to revolutionize the way engineering and science are conducted in the twenty-first century.

In this dissertation, we inherently recognize the broad and multi-disciplinary nature of M&S and find the need to introduce the discipline to minimally prepared students without the usual costs and complexities of learning such. Hence a non-proprietary, generic M&S environment, DEVS (Discrete Event System Specification) is chosen as the base. But we identify that there are several complexities in learning DEVS via conventional study methods, especially for minimally prepared students. To achieve our end goal, a new pedagogical approach and software developed for such purposes are introduced to learn DEVS concepts via active engagement.

The first 2 chapters of this dissertation will give a general introduction to Modeling and Simulation, the need, related work, and the background for the research undertaken. Next, we will go into introduce and describe the innovative pedagogical approach itself and the incorporated material in greater detail. To validate the effectiveness of the new pedagogical approach, an Institutional Review Board (IRB) approved research study was conducted with volunteers from the ECE Department at the University of Arizona. The experiment, the challenges, and the results are presented with analysis in the latter part of this dissertation before a conclusion is made.

1.1 What is Modeling and Simulation?

1.1.1 What is a Model?

A Model, in the simplest form, is “a thing used as an example to follow or imitate” as the Oxford dictionary would describe it. Singh [2, p. 9] describes “Model of a system is the replica of the system, physical or mathematical, which has all the properties and functions of the system.” Zeigler et al [3, p. 30] broaden the view of a model as any physical, mathematical, or logical representation of a system, entity, phenomenon, or process.

Though a definition of a model can differ depending on the area of focus or perspective, in the context of our interest, we like to describe a model as a structure to produce behavior claimed to represent a real-world “entity” or “system”.

1.1.2 What is an Entity?

In reality, an entity essentially could be anything in the real world with independent existence such as a chess piece, a customer, or a queue for example. But at this early stage, it is important to note an entity such as a chess piece is also static and does not display behavioral change over time. Though a physical model of it can be made, it is of no real use or interest to modeling and simulation as far as we are concerned. Hence dynamic entities (which change with time) and show behavior will be of more interest for our purposes. For example, a tennis player could be an entity of interest for a model.

1.1.3 What is a System?

Wu [4] defines a system as “a collection of components which are interrelated in an organized way and work together towards the accomplishment of certain logical and purposeful end.” But for our purposes, we would like to define a system as a collection of entities (parts) that work collectively to display a certain behavior over time. Analogously, a person could be seen as an entity and a team as a system. Sometimes depending on our modeling need and objective, there could be a thin line when deciding whether an object is an entity or a system. For example, an alarm clock could be seen as an entity if we consider only the alarm, but if we need to be more detailed, it could be broken down into a collection of individual entities such as a timer, clock, or chiming alarm etc.

It was previously mentioned that entities could be dynamic or static with respect to time. Adding further, it should also be noted that dynamic entities or systems could display behavioral changes (let us call them events) over time to be Continuous or Discrete.

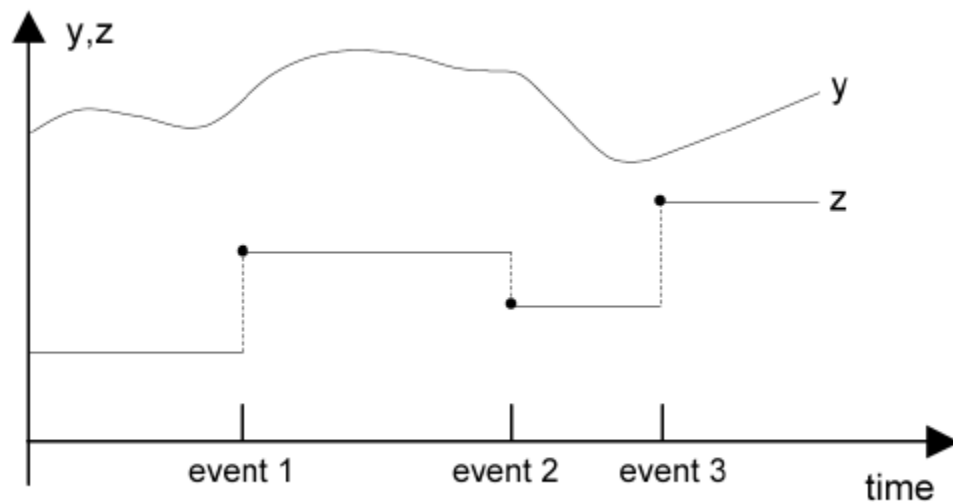


Figure 1 : Example graph to show continuous-time vs discrete-time changes

As shown in figure 1 [5], the graph y display continuous time. For example, if one is to analyze the amount of water flow over a dam, it is a continuous process with behavioral changes continuously over time. Graph z in the above figure shows events happening over distinct time changes (non-continuous). For example, a person arriving or leaving a queue can mark an event, which can change the status/behavior of the queue (an entity) at discrete instances of time.

1.1.4 What is Simulation?

As Simulation might be a rather common word people might hear in day-to-day life, it is important to understand the word in the context of this dissertation.

Oxford dictionary states Simulation is “the technique of imitating the behavior of some situation or system by means of an analogous model, situation, or apparatus, either to gain information more conveniently or to train personnel.” In the above definition “some situation” corresponds to a source (entity or system), and an “apparatus” would be a simulator. Also from the above definition, it can be gathered there are loosely 2 kinds of simulation objectives: one is to gain information and the other is to train (or entertain) someone. As Fujimoto [5] observes the former is often called an analytic simulation and the latter a virtual environment simulation. A flight simulator or car driving simulator is an example of a virtual environment simulation.

As far as our objectives are concerned, we can simplify Simulation to be the process to execute a model to generate its behavior. But as Sokolowski and Banks further explain [6, p. 22] simulation “allows for the repeated observation of the model; and that *analysis* facilitates drawing conclusions, V & V (verification and validation), and recommendations based on various iterations/simulations of the model.”

It should also be noted, at times, authors can singly use the word “Simulation” to encapsulate the intermediate step of model building too. In a broader sense, Fishwick [7] defines

computer simulation as “the discipline of designing a model of a system, simulating the model on a digital computer, and analyzing the execution output.”

Hence in academia and industry, a generally accepted approach for M&S could encapsulate, the process of designing a model of a real-world entity/system, simulating the model on a digital computer, and visualizing and analyzing the execution results. Figure 2 gives a general idea of M&S life cycle.

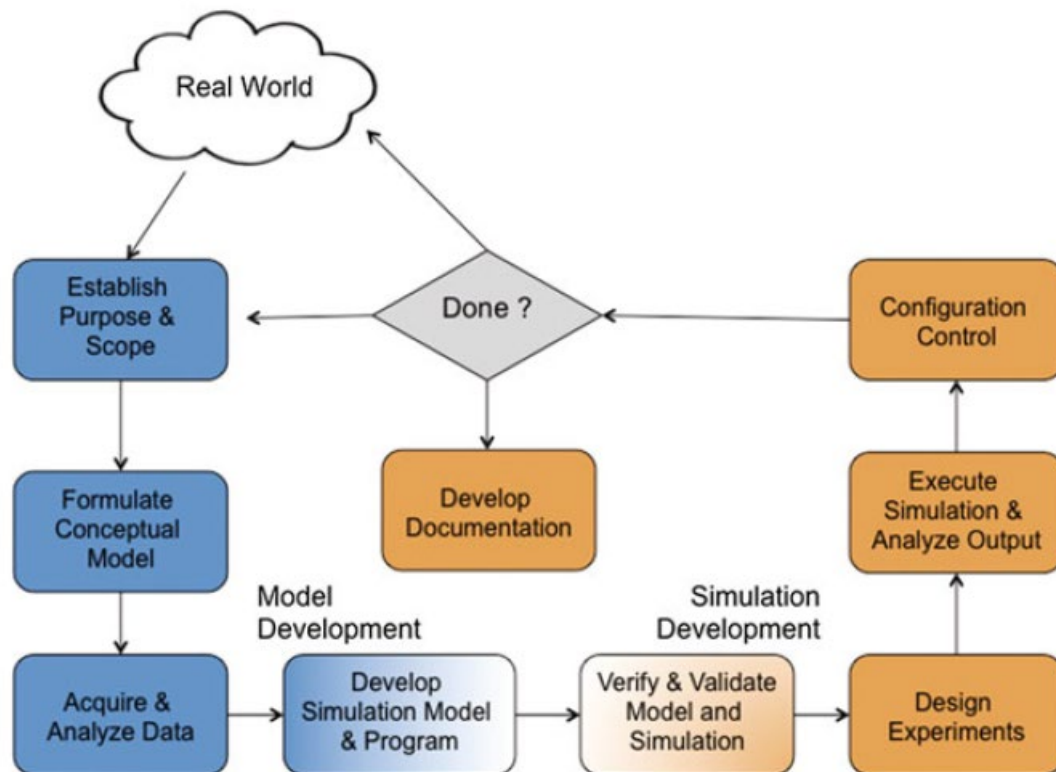


Figure 2 : M&S life cycle process [8, p. 18]

1.2 Need for Modeling and Simulation

M&S is used in multi-disciplinary fields as mentioned elsewhere. From fields such as manufacturing [9], transportation [10], meteorology [11], communication [12], healthcare and surgery [13] [14], computer-aided engineering design [15] , military and defense [16] [17], logistics [18] etc, the list goes on and on. NSF Blue Ribbon Panel [1] identifies M&S to “play a remarkable role in promoting developments vital to the health, security, and technological competitiveness of the nation.” In the article “Do We Need M&S Science?” [19], 3 experts from various science and engineering backgrounds form a consensus as to why M&S should be established as a Science itself.

The Defense Science Board in 1990 recognized the importance of (modeling and) simulation in reducing system acquisition costs by focusing on the testing required for such decisions as Horowitz [20] revealed.

In fact, for certain real-life situations, as Choi and Kang explain [21, p. 8] “simulation may be the only means to tackle the problems. In practice, simulation may be needed because experimenting with the real-life system is not feasible; your budget does not allow you to acquire an expensive prototype; a real test is risky; your customer wants it “yesterday”; your team wants to test several solutions and to compare them; you would like to keep a way to reproduce its performances later.”

Allen [22, p. 2] notes in Table 1 below some extended uses of M&S from capacity planning to test plan design.

Simulation applications	Example questions
Capacity planning	How many machines will be needed to meet a new order? How many airplanes will be needed to achieve strategic objectives?
Purchasing decision-making	How many regular and super-machines should we purchase? And how much raw material of different types will be consumed?
Project cost justification	In a six sigma project, if we reduce the defect rate to 2% will the reductions in inventory and rework cause the project to save money?
Strategic planning	What are the long term implications of a decision to use only air power in an attack plan?
Technology planning	If a machine has its processing or service time reduced by 20%, would the technology be worth the cost?
Training and gaming	How can I train my staff inexpensively? (There is really no clear line between discrete event simulations and gaming models for training)
Test plan design	What should the test specifications be to meet the performance requirements?

Table 1 : Possible applications of M&S and hypothetical questions to be addressed

1.3 Challenges in Modeling and Simulation

Challenges in Modeling and Simulation are numerous and extensive. Taylor et al [23] go on to define what is called the “Grand Challenges” in M&S for it to achieve a wider scale. Fujimoto et al [24] published the results of a workshop whose goal was “to identify and build consensus around critical research challenges in modeling and simulation related to the design of complex engineered systems”

But the basis for this dissertation is to address one of the most important but very much less explored challenges in M&S which is at the grass-root level; how to get a start on learning basics concepts of M&S without much exposure to complicated math, programming skills, time or high costs involved. It should be noted many of the software available for M&S can be field-

dependent, specialized, or proprietary and may also need prerequisite skills or considerable time to learn not to mention the costs involved. This will hamper any prospective student who is interested to get his/her feet wet in the field and has limited resources or exposure. More of this issue will be discussed in Chapter 2 of this dissertation.

1.4 A New Pedagogical Approach to M&S Using FDDEVS

DEVS though powerful it may be, has its limitations especially when it comes to learning its concepts and implementing models. Usually, DEVS is taught only at graduate level at universities with the need for prerequisite coursework on undergraduate math and computer programming. If one wants to self-study, resources are limited, and the existing material they find is catered to higher skill levels. Hence, for beginners, the experience could be challenging and demotivating.

Finite Deterministic-DEVS is a sub-class of DEVS. Though restrictive in full DEVS rigor, FDDEVS has properties useful from an education perspective and provides room for automation using a restricted Natural Language which maps it to DEVSJAVA. Hence programming skills will be kept to a minimum and we believe this will help beginners with no exposure to programming. More on such will be discussed in chapter 3.

CHAPTER 2

BACKGROUND

2.1 Current State of Modeling and Simulation Education

As previously mentioned in section 1.2, Modeling and Simulation is used in multi-disciplinary fields nowadays and. hence the demand for such professionals are ever-increasing making it an ever-growing need. The primary catering sources for educating such can be identified at the university level, by companies that manufacture M&S software, at on-the-job training, and at vocational level colleges. Needless to say, the latter 2 avenues usually prepare “workers” for job-specific fields. Hence, in this chapter, we will discuss in brief the present status of M&S education, a survey on M&S programs offered at the university level, existing M&S formalisms/frameworks, finding a suitable base for expanding M&S education to give adequate background to suffice with our primary goals previously stated.

2.1.1 Broad Categorization of Modelers & “Simulationists”

In Chapter 1, what is a Model, and what Simulation is was introduced. The word *Simulationist* used here, though not a dictionary word as of now, is beginning to appear in simulation manuscripts and occasional conference presentations [25, 26] [27]. A definition for *Simulationists* appears in the Simulationist Code of Ethics [28] as professionals involved in one or more of the following areas: modeling and simulation activities, providing modeling and

simulation products, or modeling and simulation services. For the context of this research and on the topic of educating Modelers and Simulationists, we prefer to not separate them as 2 individuals but see him/her as one person performing the combined task. Rozenblit [29] better explains the unification as, “Simulation modeling is a professional, intellectual, and academic discipline whose primary concerns are the construction of real-world systems' models, computer simulation of the models, and analysis of simulation results.”

Before exploring the topic of educating such individuals, it is important to identify and categorize potential candidates for such purposes. Broadly speaking, we can identify such individuals as i) future modelers/simulationists and ii) current professionals who need or would benefit from M&S.

2.1.2 M&S in Industry

In industries or fields where M&S is used, for example, manufacturing, transportation, defense, or health care, the training involved will be domain-specific. Hence, the software used will also be task-specific and will most likely also be proprietary. For example, Autodesk Simulation [30] provides proprietary M&S software for mechanical event simulation and computational fluid dynamics. Arena Simulation Software [31] is used heavily in production and manufacturing-related industries. PSpice [32] is used in electrical/electronic engineering for circuit simulation and verification. Hence using or training in such software only produce Modelers and Simulationists who are domain and task-centric. Such individuals will not obtain a general education on M&S which would allow them to apply their training or knowledge to other areas if

needed. Thus, when talking about Modelers/Simulationist education, it is important to understand the difference between industry/field specific vs general M&S training.

Furthermore, it should be mentioned that there are other available modeling and simulation software such as Matlab which has the capability to handle complex model specification/development and simulation using a GUI, but such available approaches do not address the M&S pedagogy-related goals of this dissertation.

In table 2, Jerry Banks [33, p. 1643] discusses the differences of M&S education in industry training vs academia.

Consideration	Industry	Academia
Objective	Provide training for use of products	Support course on statistical aspects of simulation
Audienc	Non-homogeneous	Homogeneous
Preferred teaching method	By example	By example
Categories of students	Managers and engineers	Students only
Pace requested	Go slower	Go slower
Class size	Small	Large
Responsibility for information transfer	Instructor	Student
Evaluation	Little or none	Grades
Assignments	Solve in class	Solve out of class
Frequency	Respond to demand	Function of supply (of teachers)
Length of training	Short (one week)	One academic term
Quality	Must be high	Can vary
Use of training	High % solve real problems	Low % solve real problems

Table 2 : Education for Industry Contrasted with Academia

Tucker et al [34], illustrates in table 3, a matrix of capabilities used to describe the career growth of a Boeing real-time simulation engineer. They identify a middle-level journeyman as a person who is competent to work on large, complex tasks with infrequent supervision.

	Knowledge	Skills	Experience
Apprentice (entry level)	<ul style="list-style-type: none"> • Modeling • Simulation • Programming • Statistics • Data structures • Standards • Knowledge of a Domain 	<ul style="list-style-type: none"> • Basic computer and software tools • Basic Simulation tools • Verification 	<ul style="list-style-type: none"> • Multiple Computers • Multiple Software Tools • One or a Few Simulation Projects • Model Development
Journeyman (engineer)	<ul style="list-style-type: none"> • Systems Engineering • Object-oriented Design • Multiple Domains • System Architectures 	<ul style="list-style-type: none"> • Proposal Preparation • Validation • Simulation Requirements Elucidation • Fidelity Requirements Specification • Sensitivity Analysis • Object-oriented Programming • Customer Coordination 	<ul style="list-style-type: none"> • Multiple Projects within a Domain • Multiple Lifecycle Phases • Application of Simulation • One or more publications
Master (senior engineer)	<ul style="list-style-type: none"> • Simulation in Support of Operational Systems • Advanced Simulation Techniques • Alternative Simulation Concepts 	<ul style="list-style-type: none"> • Accreditation • Domain Evolution Planning • Simulation Program Management • Simulation Lifecycle Cost Analysis • Simulation Risk Management 	<ul style="list-style-type: none"> • Projects in Multiple Domains • Multiple customer Interfaces • National/ International Conference Committees • Simulation Innovation • Multiple Publications

Table 3 : Career growth of a Boeing real-time simulation engineer

More importantly, relevant to the main focus of this chapter, Tucker et al [34] also identify “the primary need for academia is to develop educational programs to speed the creation of journeymen.” In other words, it is inferred that there is an industry need from universities/ academia to produce “Apprentice level” Modelers & Simulationists or as previously mentioned, future Modelers /Simulationists who can be hired and trained to be a Journeyman in relatively quicker time.

2.1.3 Modeling & Simulation as a Discipline

In the late 1990s and early 2000s, several efforts were made to enhance Modeling and Simulation education as it was not much in existence at that time. Workshops organized by Ralph Rogers [35] [36], Yurick and Silverman [37] and individual efforts such as Fujimoto's [38] helped to initiate principles for creating M&S educational programs which would impact the realization of M&S as a discipline.

In the year 2000 Simulation conference proceedings, Szczerbicka et al [33] state "A general demand for modeling and simulation professionals can be observed in a large number of enterprises. However, computer science graduates are not adequately prepared for employment opportunities involving simulation as a tool for solving problems. Most computer science majors have very limited exposure to simulation. They gain experience in handling simulation problems through on-the-job training." In figure 3, Zeigler and Sarjoughian [33] propose an "infrastructure for the Establishment of M&S Discipline."

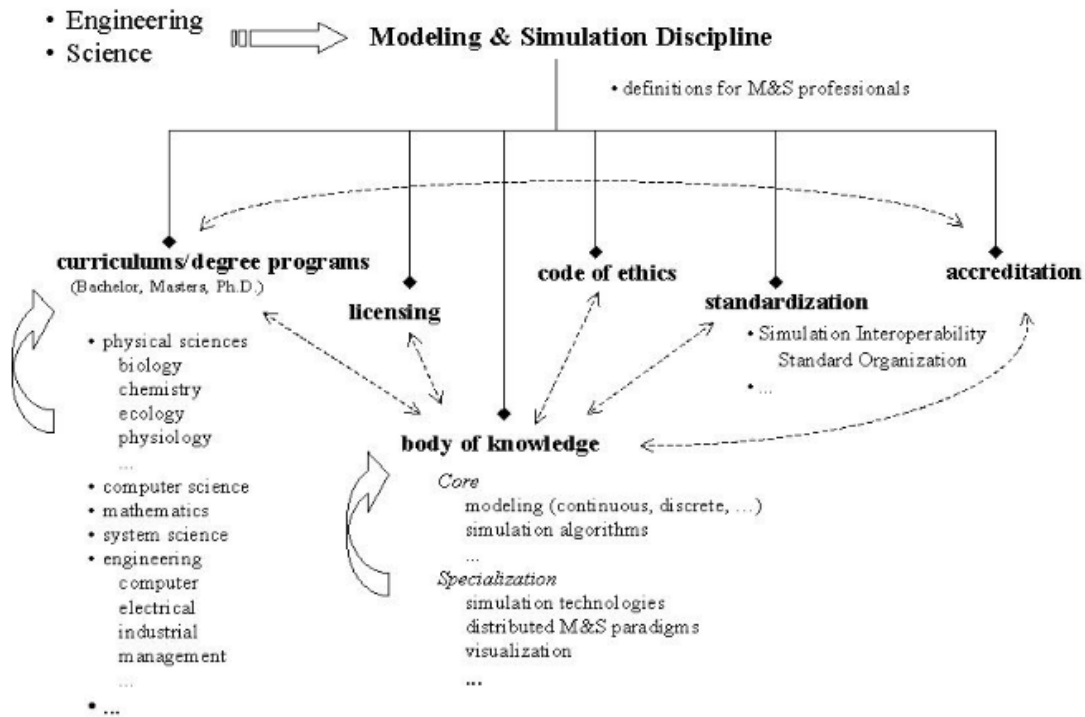


Figure 3 : An infrastructure for the Establishment of M&S Discipline

Hence it is safe to say that though modeling and simulation is needed in so many fields, it is only raising its head to become an established academic discipline since the recent past. And as we will discover in section 2.1.3, such programs in academia are still very limited and are found mostly at Master's or Ph.D. levels within science and engineering curriculums.

2.1.4 M&S Education at University Level

To understand and appreciate the present scope of M&S education at the university level, the below search-based survey was done and is presented in tables 4,5, and 6. The information is presented to the best of the research done by the author and may not essentially be complete.

Program	Academic institution	Weblink
PhD Level	University of Alabama in Huntsville (USA)	https://www.uah.edu/m-s-degree-program
	University of Central Florida (USA)	https://www.ucf.edu/degree/modeling-and-simulation-phd/
	George Mason University (USA)	https://cos.gmu.edu/cds/phd-in-computational-sciences-and-informatics/
	Naval Postgraduate School (USA)	https://my.nps.edu/web/cs
	Old Dominion University (USA)	https://www.odu.edu/cmse/academics
	University of Pennsylvania (USA)	http://cg.cis.upenn.edu/phd.html
	University of Pittsburgh (USA)	https://cmsp.pitt.edu/

Table 4 : Currently offered PhD programs in M&S

Program	Academic institution	Weblink
Master's Level	University of Alabama in Huntsville (USA)	https://www.uah.edu/m-s-degree-program
	Arizona State University (USA)	https://acims.asu.edu/education/
	University of Central Florida (USA)	https://www.ucf.edu/degree/modeling-and-simulation-ms-2/
	Columbus State University (USA)	https://online.columbusstate.edu/degrees.php
	Middle East Technical University (Turkey)	https://www.metu.edu.tr/graduate-programs-and-degrees-offered-odtu
	National University of Science and Technology (Pakistan)	http://www.nust.edu.pk/INSTITUTIONS/Centers/RCMS/Pages/default.aspx
	Naval Postgraduate School (USA)	https://my.nps.edu/web/cs
	University of New South Wales (Australia)	https://degrees.unsw.edu.au/master-vis-sim-im-des/
	Old Dominion University (USA)	https://www.odu.edu/cmse/academics
	Purdue University (USA)	https://academics.pnw.edu/grad-school/modeling-simulation-and-visualization/
	Savitribai Phule Pune University (India)	http://cms.unipune.ac.in/programmes/
	University of Wisconsin – Madison (USA)	https://www.engr.wisc.edu/department/mechanical-engineering/academics/masters-degree-mechanical-engineering-2-2/

Table 5 : Currently offered Master's level programs in M&S

Program	Academic institution	Weblink
Undergraduate Level with a specialization in M&S	Old Dominion University (USA)	https://www.odu.edu/cmse/academics
	Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (Pakistan)	https://www.giki.edu.pk/Faculties/FESUndergraduate/DegreeRequirement
	University of Alabama in Huntsville (USA)	https://options.pcs.uah.edu/searchResults.cfm?prgID=3
	Minnesota State University-Mankato (USA)	https://mankato.mnsu.edu/academics/academic-catalog/graduate/modeling-and-simulation/
	Portland State University (USA)	https://www.pdx.edu/sysc/graduate-certificates-in-systems-science
	Seminole State College (USA)	https://www.seminolestate.edu/computers/curriculum/simulation

Table 6 : Currently offered Undergrad level programs in M&S

Interestingly Delft University of Technology (Netherlands), had once claimed to have had a Master’s program for M&S. But the author could not trace the existence of such at the time of writing.

It should also be noted that the author came across other academic institutions that offer various programs with keywords of Modeling, Simulation, and Visualization upon doing this survey. But the focus here was on general Modeling & Simulation coursework within computer /engineering programs at the university level but not on job-oriented or domain-specific programs. For example, such keywords could also be found in degrees in mathematics, computer animation, mechanical engineering, etc. Also in fields outside engineering, Gonczi [39] identifies “Simulation has proliferated in contexts where universities are charged with preparing graduates for work in specific professions, seen as a bridge between the classroom and the world of work.” The referenced SIGSIM website [40] will be useful to find some other institutions offering programs with the keywords of Modeling or Simulation.

With the above info presented, it will suffice to prove with very little ambiguity that M&S education as a discipline at the university level is still at a rudimentary stage and has a lot more room to expand.

2.2 Existing M&S Formalisms, Frameworks, and Finding a Base for M&S Education

Now that we have some background, it should be evident that M&S has a further need for innovative educational programs to interest and attract individuals into the field. It should also be clear, to introduce prospective M&S students to the field, an open-ended M&S educational framework is best suited as opposed to a domain-specific one. In this section, we proceed to identify some existing formalisms and frameworks before we establish the base for a new pedagogical approach.

2.2.1 Types of M&S Frameworks and Formalisms

A framework in simple terms is a methodology for realizing a defined goal. With respect to M&S, in practicality, if we explore the various commercially available domain-specific software itself (few such were mentioned in section 2.1.2) it will be quite expansive and yet we will find each of them carrying a unique and propriety underlying framework. Thus such will not meet our objective which is to introduce a pedagogical approach to introduce students to learn the basics of M&S that does not tie any student to a specific domain/field.

As we discussed in section 1.1.2, dynamic systems could display behavioral changes (or events) over time to be continuous or discrete. Hence in the simplest form, any system formalism could be divided to be either based on continuous or discrete-time (of course there can be systems that combine both).

Choi and Kang [21, p. 4] identify three types of dynamic systems where discrete or continuous-time modeling and simulation can be performed. Namely, Discrete-Event Systems, Continuous Systems, and Quantum Systems. They call a system at the subatomic level or cosmological level a quantum system, which is beyond the complexity and need for our purpose.

A formalism, in simple terms, is a way to describe some entity or system using formal mathematical notations or arguments. Zeigler [41] divides M&S formalisms into three main classes.

- Differential Equation System Specification (DESS)
- Discrete-Time System Specification (DTSS)
- Discrete Event System Specification (DEVS)

Zeigler et al [3, p. 6] state, “the traditional differential equation systems, having continuous states and continuous time, were formulated as the class of DESS.” In DESS the behavior of the system is generated in a continuous way using mathematical differential equations as the base for algorithms. Hence it produces an unlimited number of system states over the desired simulation period which creates a continuous curve if transferred on a diagram. A system state is basically a collection of data that contains all the information necessary to describe the system's behavior at a given time.

A second class, DTSS, can be formulated based on systems that operate on discrete-time such as Automata. DTSS also uses mathematical equations to produce the system state over time, but the difference is that this method does not produce an unlimited number of system states. Rather, it quantifies the system state in a limited number of time points that has a certain interval.

However, as our pedagogical base, we chose a 3rd class, Discrete Event Systems (DES), and its most popular formalism called DEVS (Discrete Event System Specification) which has the extendable potential to model and analyze discrete event systems as well continuous state systems, and hybrid systems (continuous state + discrete event systems). DES is used widely in many fields even outside engineering. For example, it is increasingly used in health-care services nowadays [42] [43] [44].

Figure 4 [3, p. 6] shows the 3 basic classes of system formalisms with graphs, reflecting the logic behind each and how the mentioned formalisms behave with respect to time.

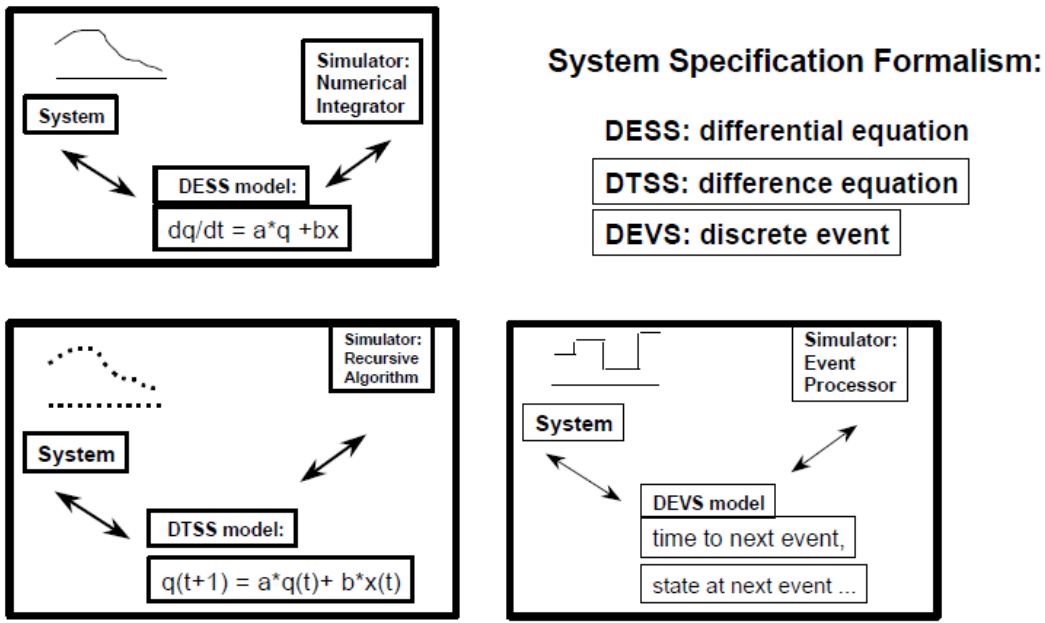


Figure 4 : Basic systems specification formalisms

2.2.2 Why Was DEVS Chosen?

Before introducing what DEVS is in more detail, we believe it is important to reiterate why we chose DEVS as the base to meet our purposes on M&S Education.

i) To start with, DEVS is a generic formalism. This generic format allows for non-domain/field-specific models to be developed and simulated which will not tie students to one area only.

ii) Many commercial M&S software are proprietary and will have licenses/heavy costs involved which might demotivate some students who want to self-learn or get their feet wet in general M&S education.

iii) Thirdly, DEVS has the capacity to analyze discrete event systems, continuous state systems, and hybrid systems which allows students to explore further once a basic foundation is obtained.

iv) DEVS allows the separation of models from the simulation program that executes them. The alternative, which is more commonly found in today's practice, is not to enforce such a clear separation of the two thereby complicating the task of a student who wants to explore the basics of modeling as an initial step.

v) Finally, DEVS is still very popular in academia and considerable research, extensions, and developments are ongoing [45, 46]. A representative, but not up-to-date sample of current research and development in DEVS is listed below under figure 5 [47]. For a student who wants to learn

the basics of M&S, DEVS provides a stepping stone to expand his/her goal to maybe go from an “Apprentice” to a “Master” as previously discussed in Table 3. Note that, the references given above are to serve as examples and there is no necessity to understand the terminology or depth of each topic at this point.

- **Theory**
 - Extension of the System Entity Structure for Hierarchical Abstraction
 - Formal Model-Checking Methods and DEVS
 - DEVS M&S of Multilayered Social Networks
 - Multicomponent Formulation of DEVS
 - Verification and Validation (Zeigler and Nutaro, 2016)
- **Simulation and Optimization**
 - DEVS Architecture for Simulation-based Optimization
 - Optimization of DEVS Distributed Simulations
 - Reproducibility of High Performance Stochastic Simulations
- **DEVS Development Environments**
 - VLE-Virtual Lab Environment²
 - MS4 Me³
 - DEVSIMPy⁴
 - GRADES (Graph-based and RANdom Discrete-event Simulator)⁵
 - PythonDEVS⁶
 - ProDEVS: Petri Net encoding of DEVS (Vu et al., 2015)
 - fwkDEVS: OO-Framework for DEVS (Bisgambiglia, 2016)
- **Applications**
 - Parallel Simulation of DEVS Spiking Neurons
 - DEVS Smart Phone Simulation
 - DEVS Encoding into Timed Petri Nets for Hardware Implementation
 - Multicomponent DEVS for Multi-agent Systems

Figure 5 : Sample of research and development in DEVS

2.2.3 Discrete Event System Specification

In a nutshell, DEVS is a modeling formalism. To recap, a formalism is simply a way to describe some entity or system using formal mathematical notations or arguments. As an analogy, just as arithmetic encapsulates addition, multiplication, and other calculation, DEVS allows for simulations of discrete event models. In other words, DEVS is a formal mathematical language for

specifying engineering-related or other complex systems through models that can be then simulated in virtual time and executable in real-time.

Shown in figure 6 below is the basic DEVS formalism with port specifications [3, p. 84]

$$DEVS = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$$

where

$X = \{(p, v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values

$Y = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and values

S is the set of *sequential states*

$\delta_{ext}: Q \times X \rightarrow S$ is the *external state transition function*

$\delta_{int}: S \rightarrow S$ is the *internal state transition function*

$\lambda: S \rightarrow Y$ is the *output function*

$ta: S \rightarrow R_{0, \infty}^+$ is the *time advance function*

$Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$ is the set of *total states*.

Figure 6 : DEVS Formalism with port specifications

In-depth details of this formalism, or explanations are not needed at this point as such will be addressed in sections 2.3.1, 2.3.2, and more exclusively in Chapter 3.

After effectively learning this formalism, a modeler/student needs to correctly specify a DEVS model, which is essentially a description of a real-world dynamic entity specified in DEVS mathematical terminology. It should be noted at this stage that the most basic DEVS model (let's call it a building block) that can be specified using the above-mentioned formalism is called an “Atomic model”.

Basic “Atomic” models may be coupled in the DEVS formalism to form what is called a “Coupled model”. A coupled model tells how to connect or couple multiple Atomic models to form a new larger model. This larger coupled model can then be used as a component to thus create much larger and more complex systems using hierarchical construction. DEVS “Coupled model” specification [3, pp. 85-86], as shown in figure 7 below.

Once the DEVS model/system is specified correctly, as mentioned previously, the separation between the model and simulator allows the Simulationist to simulate the model. The flow that binds DEVS modeling and DEVS simulation services is the *DEVS Simulation Protocol*, as shown in figure 8, which is a restatement of what is called the DEVS Abstract Simulator. The DEVS Simulation Protocol is essentially an interface that the DEVS models need to present to the simulator and the rules governing how the simulator invokes the methods in the interface to execute a valid simulation.

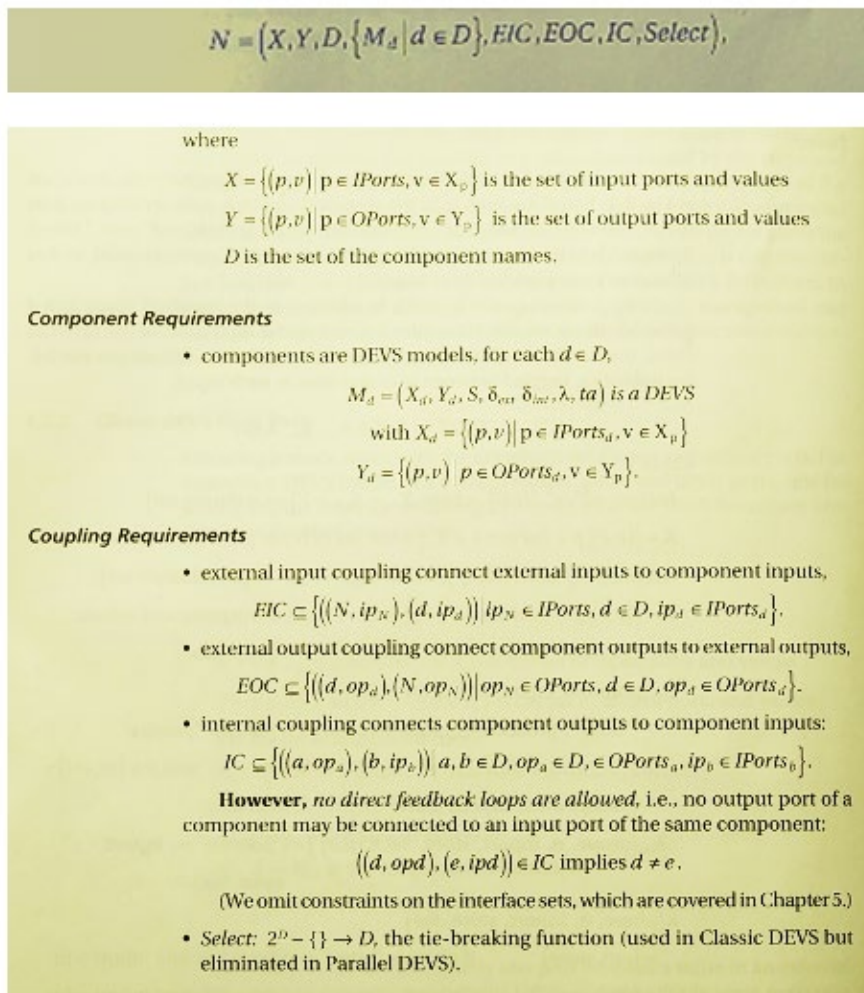


Figure 7 : Coupled DEVS Specification/Formalism

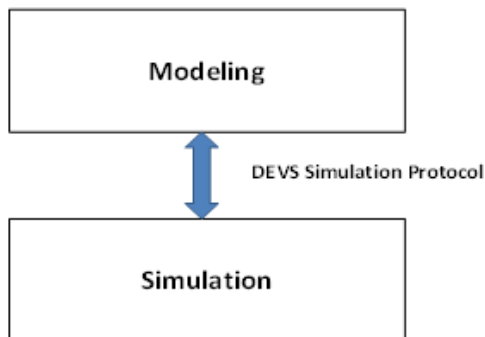


Figure 8 : DEVS Simulation Protocol

2.3 Existing DEVS Pedagogical Approach

2.3.1 Traditional DEVS Learning Approach

Sections 2.2.2 and 2.2.3 discussed why we chose DEVS as the base for our M&S pedagogical approach and introduced what DEVS is in a very concise way. However, what was not discussed is how a novice student who is interested in M&S can learn DEVS and the underlying technical knowledge needed for such.

It should be noted at the onset that the basics of DEVS is usually taught in the past at universities as a Graduate level course. Usually, a student would follow a semester-long course and use the below as standard references and resources. Any student interested in self-learning the basics of DEVS or M&S would usually search and resort to the same resources.

i) To learn the formalisms, specifications, and concept of DEVS, refer to :

“Theory of Modeling and Simulation” by Bernard Zeigler et al [3]

ii)To implement and simulate DEVS models:

use DEVS JAVA modeling and Simulation environment which is a software written in Java and supports parallel execution on a uni-processor. It also supports higher-level, application-specific modeling. [48]

It should be also noted, several other DEVS-based simulation engines and their environments and Toolkits exist below as listed by Wainer [49] in figure 9. (This list should not be taken as a very up-to-date version.)

- ❑ **ADEVS** (University of Arizona)
- ❑ **CD++** (Carleton University)
- ❑ **DEVS/HLA** (ACIMS)
- ❑ **DEVSJAVA** (ACIMS)
- ❑ **GALATEA** (USB – Venezuela)
- ❑ **GDEVs** (Aix-Marseille III, France)
- ❑ **JDEVs** (Université de Corse - France)
- ❑ **PyDEVs** (McGill)
- ❑ **PowerDEVs** (University of Rosario, Argentina)
- ❑ **SimBeams** (University of Linz – Austria)
- ❑ New efforts in China, France, Portugal, Spain, Russia.

Figure 9 : DEVs Toolkits

- ADEVs [50], a C++ library for developing discrete-event simulations based on the Parallel DEVs and DSDEVs formalisms. Developed by Jim Nutaro.
- DEVs SOA [51], is an implementation of DEVs within a Service Oriented Architecture (SOA) environment. The advantage of SOA DEVs is its effective mathematical representation and its support to distributed simulation using middleware such as DoD's High-Level Architecture (HLA).
- CD++ [52] [53] is a general toolkit written in C++, which allows the definition of DEVs and Cell-DEVs models. DEVs coupled models and Cell-DEVs models can be defined using a high-level specification language. Developed by Gabriel Wainer and his students (Carleton University, Canada; Universidad de Buenos Aires, Argentina).

Though all these DEVs-related M&S environments and Toolkits exist, it needs to be understood that for a student seeking to learn the basics of DEVs M&S, the previously mentioned textbook and the DEVsJAVA M&S environment serves as the primary and most popular resource.

Also, almost all of the other DEVS-based M&S environments could be seen as upgraded versions to include more complexities, specialized tasks, or features to enhance the capacity of DEVS. Hence the reason, DEVSJAVA should still prevail as the comparatively simpler M&S environment for DEVS.

2.3.2 Challenges

Though the “bible” of DEVS [3], and the DEVSJAVA environment documentation would suffice as a good educational source for an experienced individual, unfortunately, any novice undergrad or lower-level student or industry personnel who desire to learn M&S through DEVS will most likely meet a roadblock. A roadblock will come in the form of a solid prerequisite background in mathematics and programming to comprehend the resource material. Hence the reason DEVS was almost always taught in Graduate level programs.

One look at the mathematically challenging DEVS formalisms or the deeper algorithmic or programming knowledge needed to understand the Abstract DEVS Simulator algorithms or coding in DEVSJAVA could demotivate an average student.

In summary, as powerful as DEVS might be, there could be significant learning curves and overheads involved, and depending on the skill level of the learner it may even discourage them from further studies.

2.3.3 Need for a Newer Approach

In Chapter 3, we will visit in great depth, an innovative Pedagogical approach with newly developed learning material using a lesser stringent software approach to learn the basics of M&S using FDDEVS.

CHAPTER 3

INTRODUCTION TO FDDEVS, SYSTEM ENTITY STRUCTURES AND THE NEW PEDAGOGICAL APPROACH

In this chapter, we will revisit some previously mentioned issues/challenges and discuss the motivations, theoretical basis, and give background to the technology utilized for the new pedagogical approach presented in this research.

3.1 DEVS and M&S Education Challenges Revisited

In chapter 2, we discussed the capacity and capabilities of DEVS. The formal rigor and versatility of DEVS allow the development of a wide range of domain-specific models as discussed before and summarized by Kim and Kim [54, pp. 16-21].

But as powerful as DEVS is, it was discussed in chapter 2 that it presents a steep learning curve to many of the students and practitioners to understand the concepts, the mathematics behind it, and the implementation skills needed in programming to successfully tackle new problem-sets.

In Chapter 2, we explored in detail and discussed why Modeling and Simulation is an area of focus/discipline that has enormous potential and the pedagogical limitation it has to date.

3.2 Motivations for this Research

Though DEVS is widely used for domain-specific application development/simulation, and further research was/is done to expand its universe via diverse Environments and Toolkits as discussed in section 2.3.1 and [54], there is one area of focus that has not been researched or explored to date. That is how to utilize DEVS for Modeling and Simulation Education and to expand its reach to a wider audience as discussed in chapters 1 & 2.

Thus, it created an opportunity for this research to explore innovative ways to teach Simulation and Modeling using DEVS to introduce and cater to a wider audience who might not be otherwise proficient in programming or mathematical skills.

Further inspiration is drawn from observing two trends emerging in the computer games space. Namely, 1) the growing interest in the development of games that seek to teach more formal/traditional subjects rather than just entertain, and 2) the games that encourage players to figure out what the rules of the game are, rather than developing the skill to play a game with the provided rules.

Can we learn from these trends and our other motivations to develop effective educational technology for DEVS and systems concepts and utilize “learning by experimenting” in a more deliberate way to combine with traditional and innovative tutorials to gain knowledge in a shorter period compared to traditional approaches which more likely will expand the base of students interested in M&S?

A formal experimental study that was conducted under the approval of the Institutional Research Board and its results will be presented in Chapter 5.

3.3 A Brief Introduction to FDDEVS and Its Extensions

Finite Deterministic DEVS (FDDEVS) is a sub-class of DEVS that has properties that are very useful from an educational perspective. FDDEVS is relatively simple enough to provide an easier introduction to DEVS while still preserving the essential nature of the discrete event dynamic system properties such as events, timings, and model composition.

FDDEVS formalism was first introduced to restrict the class of DEVS models to allow for deeper analysis of verifiable properties for the restricted class by Hwang [55]. The simplicity of the FDDEVS formulation was subsequently recognized by Mittal et al [56] to provide a useful abstraction called XFDDEVS for the development of DEVS models using template-based design. This interface allowed for the creation of Java and XML-expressed FDDEVS models. A further extension called AutoDEVS was developed by Salas and Zeigler [57] to support a methodology to auto-generate models using a spreadsheet that contained requirement specifications.

As an alternative to the template-based design model generation, models also could be generated using a constrained natural language input that has the following statement format [56].

to start hold in PHASE for time SIGMA !

hold in PHASE for time SIGMA !

after PHASE then output MSG !

from PHASE go to PHASE' !

when in PHASE and receive MSG go to PHASE' !

To add further versatility and convenience, some of these statements can be compounded as phrases in the following statements:

hold in PHASE for time SIGMA then output MSG and go to PHASE' !

hold in PHASE for time SIGMA then go to PHASE' !

The natural language semantics and their sample usage are provided in table 7 below.

Natural Language Phrase	XFD-DEVS Specification	Example
to start hold in PHASE for time SIGMA! where SIGMA is 0, a positive real, or the symbol Infinity (other variants, such as Inf, are allowed)	TimeAdvanceTable(PHASE) = SIGMA and also set PHASE as initial state	to start hold in waitForJob for time Infinity! This can also be expressed as: to start passivate in waitForJob!
hold in PHASE for time SIGMA where SIGMA = 0, a positive real, or the symbol Infinity (other variants, such as, Inf are allowed)	TimeAdvanceTable(PHASE) = SIGMA	hold in sendOut for time 0.1 !
after PHASE then output MSG!	OutputTable(PHASE) = MSG	after sendOut then output Job !
from PHASE go to PHASE'!	InternalTransitionTable(PHASE) = PHASE'	from sendOut go to waitForJob !
when in PHASE and receive MSG go to PHASE'!	ExternalTransitionTable(PHASE,MSG) = PHASE'	when in waitForJob and receive Job go to processing!
when in PHASE and receive MSG go to PHASE' eventually !	ExternalTransitionTable(PHASE, MSG) = PHASE' with a mark to note that this input is schedule-preserving in this state	when in waitForJob and receive Job go to processing eventually !

Table 7 : Natural Language Semantics and how to use them

3.4 System Entity Structures

It is deemed necessary to provide a brief introduction to System Entity Structures (SESs) at this stage as this will be incorporated into the software toolkit we will be using and also pedagogical tutorial segments of the new approach.

SES is a knowledge representation framework for supporting the compositions of hierarchical and modular structures. SES can also be applied to describe conceptual hierarchical systems such as tree systems and special purpose languages [58] It should be also noted here that FDDEVS framework has the capability to use SES descriptions defined in a natural language to specify model couplings as will be seen in the tutorials.

Since SES originated from the representation of simulation model structures, it can be accommodated in modeling and simulation automation. The author of this dissertation, for this Master's thesis, successfully managed to map the SWT graphical library (Standard Widget Toolkit) into the SES format so that sufficient and quantitative knowledge was represented by it to auto-generate a GUI [59].

- **Entity** : could be anything that exist in the real world or maybe in imagined world.
 - Ex: Tennis Match, Book, Car, House, a Dragon! etc
- **Variable** : are an attribute of an Entity
 - Ex : Name of Player
- **Aspect** : represent ways of decomposing things into even smaller ones.
 - Ex : Tennis Match can be decomposed into Player A , Player B, Umpire etc.
- **Specialization**: represent categories or families of specific forms that a thing can assume.
 - Ex : Tennis Match can be Singles match or Doubles. Tennisball can be Green or yellow color.
- **Multi Aspect** : are aspects for which the components are all of the same kind.
 - Ex : Tennisballs

Though going into too much detail at this stage is not necessary, a brief synopsis on the basic structural composition of a SES is explained below and also shown in figure 10.

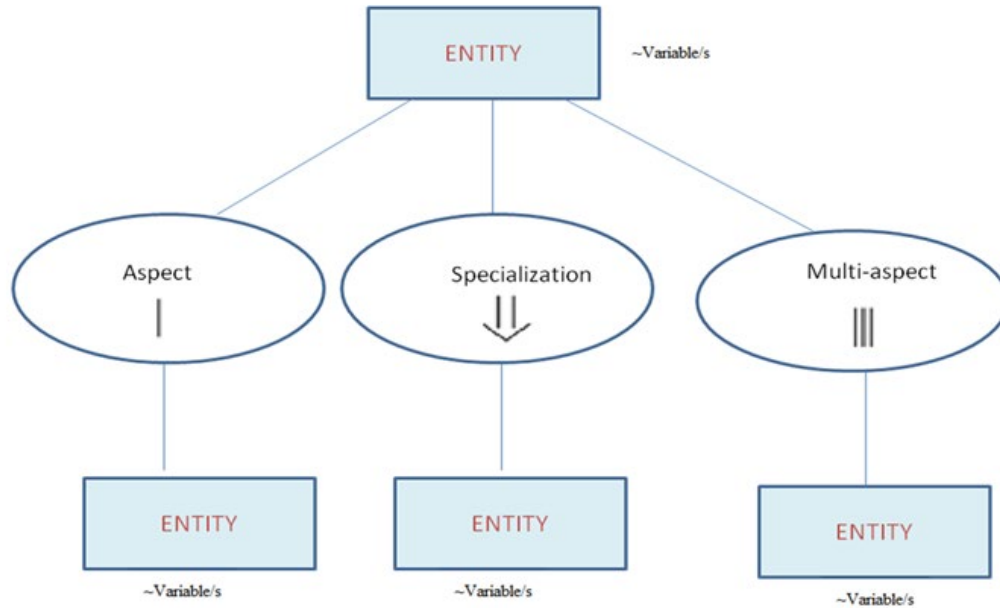


Figure 10 : Basic structural composition of a SES

3.5 The Usage of FDDEVS in the New Pedagogical Approach

As intimated above, by using FDDEVS it is relatively simpler to provide an introduction to DEVS while still preserving the essential nature of the discrete event dynamic system properties such as events, timings, and model composition. FDDEVS is amenable to automation in which a model defined in a restricted natural language can be transformed into a working DEVSTJava model which can be visualized and manipulated in a simulation viewer. Such is ideal to provide as a stepping stone to a novice who is keen to learn M&S but lacks programming skills.

As a quick example, note the DEVSTJava code needed to model a simple Tennis Player as shown in figure 11. Details of the model and the code content do not need to be looked at in-depth, but in a novice's eyes, the code complexity of even a simple model should be noted. Also, figure

12 shows the traditional approach a student would have needed to take to conceptualize, map, implement, and experiment with any model.

```
public class TennisPlayerA extends ViewableAtomic{
    double Infinity = INFINITY;
    public TennisPlayerA(){
        this("TennisPlayerA");
    }
    public TennisPlayerA(String nm){
        addInport("inShotB");
        addTestInput("inShotB",new entity());
        addTestInput("inShotB",new entity(),1);
        addOutport("outShotA");
    }
    public void initialize(){
        holdIn("play",0.5);
    }
    public void delTint(){
        if(phases("play")){
            passivateIn("wait");
        } else {
            passivate();
        }
    }
    public void deltext(double e, message x){
        Continue(e);
        for(int i=0; i<x.getLength(); i++){
            if(this.messageOnPort(x, "inShotB", i)){
                if(phases("wait")){
                    holdIn("Play",0.5);}
            }
        }
    }
    public message out(){
        message m = new message();
        if(phases("play")){
            m.add(makeContent("outShotA",new
            entity("ShotA")));
            return m;}
    }
}
```

Note : You do not need to know this code. Showing as a code example only.

Figure 11 : DEVJava code of a Tennis Player

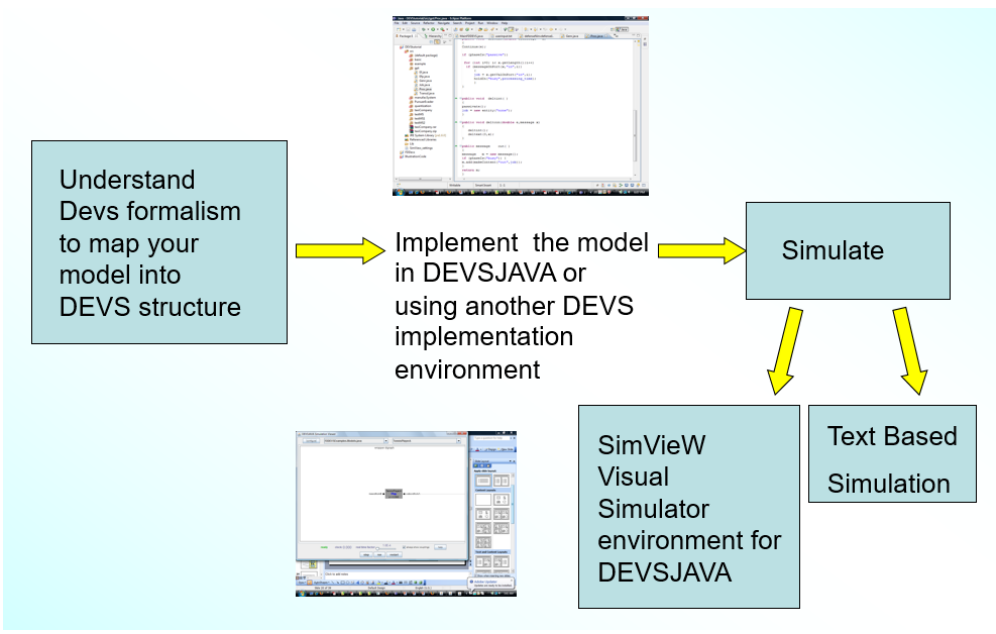


Figure 12 : The flow of the traditional approach for DEVJava M&S

Next, note the shorter FDDEVS natural language specification in figure 13 to auto-generate the same DEVSTJava code (refer to figure 14) that would shorten student modeling times.

to start hold in play for time 0.5!
after play output ShotA!
from play go to wait!

hold in wait for time Infinity!
when in wait and receive ShotB go to play!

Figure 13 : FDDEVS natural language description of the same model

to start hold in play for time 0.5!
after play output ShotA!
from play go to wait!

hold in wait for time Infinity!
when in wait and receive ShotB go to play!

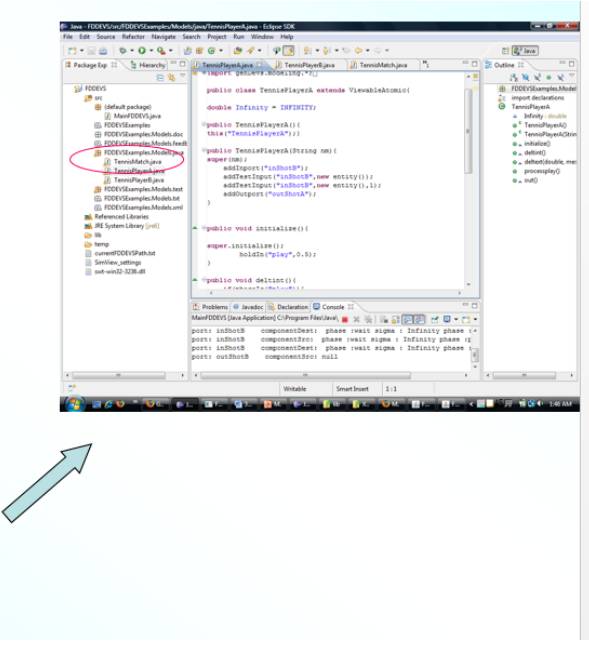
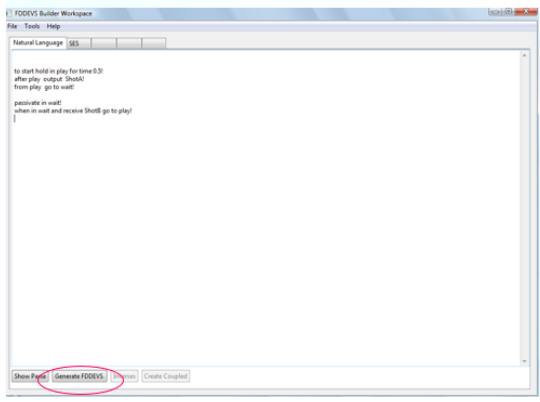


Figure 14 : FDDEVS natural language model input and auto Java code generation

Although limited in its expressive power, as we can fathom from above, FDDEVS models can serve as initial skeletons from which a full-scale DEVS model can be developed by adding more elaboration/complexity as needed. Hence, FDDEVS will serve as the mentor and backbone for the software tool and pedagogical approach used to meet the objectives of this study to significantly reduce learning curves, cut down on model development time, and make training/learning easier.

3.6 The Research Plan and Methodology

The overall long-term plan for this project to reach its full implementation capacity would be six-foldth, progressing from requirements formulation to implementation, application, and dissemination.

The sequence is outlined below:

Task 1: Develop requirements for active learning of DEVS and systems

Task 2: Develop/utilize GUI and toolset to meet requirements

Task 3: Develop innovative instructional support/tutorials to utilize the GUI/toolset to teach basic concepts of DEVS and SESs

Task 4: Introduce online and live engineering classroom instruction

Task 5: Formulate evaluation methods for online and live classroom environments

Task 6: Disseminate results and educate the workforce

However, due to resource, time, and scope limitations of this dissertation, the focus will be limited to tasks 1-3 to utilize a platform and to mainly develop educational support material/tutorials that would meet the basic objectives of this research as stated before. Please note this chapter has only attempted to give a brief overview of concepts, notions, and approaches that will be explored in more detail in Chapter 4.

CHAPTER 4

PRELIMINARY IDENTIFICATIONS, REQUIREMENTS, AND GUIDELINES FOR THE NEW PEDAGOGICAL APPROACH

This chapter will introduce and discuss in detail the preliminary identifications, requirements, and guidelines we formulated before implementing the new pedagogical approach to achieve our objective to teach DEVS modeling and simulation concepts using an innovative methodology. It should be mentioned at the outset that though the insightful material presented in this chapter was kept in mind to be incorporated much as possible into the new pedagogical approach, such concepts and ideas are elaborated more with the hope that it will be prudent for any possible future adaptors or developers of this approach. Before further discussions are underway, it should be mentioned that some insights and content were conceived with the help of Dr. Bernard Zeigler my initial advisor.

4.1 Preliminary Content Requirement Assessment and Guidelines

4.1.1 Identification of Traditional M&S Tool-Based Education Shortcomings

Further to what was discussed in section 2.3 and section 3.1, and before developing a new toolset and relevant pedagogical content, it was deemed important to identify any shortcomings faced from our initial experiences in teaching M&S using other tool-based instruction. Thus we, identify some shortcomings, and possible resolutions below that were kept in mind while developing material for the new pedagogical approach.

- ***Difficulty in introducing theory:*** help students in developing systems problem-solving skills along with systems theory concepts as facilitated by the new DEVS/System Theory based approach.
- ***Language and tool impediments:*** design our tools to reduce the complexity of their use by novices.
- ***Motivation:*** Overcome lack of motivation by novel approaches such as supporting and incentivizing “figuring out how the system works”
- ***Prior Preparation:*** Design tools that automatically generate as much as possible from user input at the modeling level, reducing the need to be computer-savvy at the language and operating systems level.
- ***Time Pressures:*** maximize the efficiency with which material is presented by optimal choice of topics and sequencing; decompose the material into atomic lessons that can be consumed in breaks between regular work; minimize the time spent in lessons by providing rapid responses with efficient software implementations.
- ***Concept Learning Vs Concept Application:*** from initial instruction attempts it became evident that students who have satisfactorily grasped a M&S concept still have difficulty in applying it to solve problems using corresponding tools. Hence concept introduction with practical experimentation which leads to deep learning needs to be explored.

4.1.2 Utilization of Hierarchy of System Specification Levels & a Student as a System

It was found innovative to use the hierarchy of system specification levels in discussing approaches to teaching students to learn the DEVS, and therefore system concepts. Table 8 below loosely scrutinizes the relevant specification hierarchy [3, pp. 131-133] with a scuba diver example that easily conveys many of the salient features. A scuba diver plans, before the dive to descend to

a certain water level and to stay at that level for some time, then to ascend to a second level for some period, and so on, until reaching the surface.

Level	Name	System Specification at this level	Scuba Diver Example	Student as a System
4	Coupled Systems	System built from component systems with coupling recipe.	Structural representation of Diver in terms of e.g., processing modules and their interaction	Structural representation of a student in terms of e.g., processing and learning modules and their interaction
3	I/O System Structure	System with state and state transitions to generate the behavior.	Diver's decision algorithm to execute planned dive	States represent student's learned skills and capabilities; transitions represent movement from a skill level to a higher one
2	I/O Function	Collection of input/output pairs constituting the allowed behavior partitioned according to the initial state of the system.	Diver's planned dive trajectory – levels and time at each level starting on the surface	For each learning state, the observable manifestation, viz., the pairs of input problems and output candidate solutions characterizing the student's response in the given state – input problem/output solution will differ as higher learning states are reached
1	I/O Behavior	Collection of input/output pairs constituting the allowed behavior of the system from an external Black Box view.	Diver's outputs under the surface over time in response to external inputs	Streams of input problems and associated output candidate solutions
0	I/O Frame	Input and output variables and ports together with allowed values.	Diver's receivable signals (inputs) and generatable signals (output)	Inputs – problem instances selected from systems problems types Outputs –candidate solutions in response to problems

Table 8 : Systems Specification Hierarchy vs Student as a System

The final column of table 8 shows how we can interpret the specification levels in modeling a student as a system. This ideology was also useful to structure the development of the new instructional approach for teaching DEVS.

4.1.3 Systems-Based Problem-Solving Types

As described by Klir and Elias [60], systems problems can be categorized into 4 types: system analysis, systems inference, systems synthesis, and systems diagnosis. These problem types can be characterized in terms of the sorts of transitions that they require to be made in the systems specification hierarchy [3, pp. 131-133] and the corresponding level of difficulty and creativity they require.

Table 9 below, formulates such systems problems in the context of DEVS-based pedagogy. In this manner, we can interpret the traditional systems problem types in terms of DEVS-based instruction to characterize the skills and creativity needed to solve problems. It should be mentioned for this research effort, developing all such metrics is beyond the scope. But developing metrics that distinguish skill attainment from the level of creativity exhibited remains a task for future adaptations and expansions of this pedagogical approach. Also note that we have added a problem type, “systems figuring out”, to the traditional categories.

Systems Problem	Is the system given by an explicit model or via its behavior? What is student trying to learn about it?	Which level transition is involved?	Skills needed/ Creativity level evoked
systems analysis	The model being analyzed is given for inspection. The student must try to understand its behavioral characteristics.	Moving from higher to lower levels, e.g., using simulation to generate a model's behavior	Skills: understanding FDDEVS concepts; knowing how to employ the tools available to generate and observe the behavior Creativity Level: low
systems inference	The model exists for experimentation. The student is trying to infer how it works from observations of its behavior.	Moving from lower to higher levels, e.g., having input/output data, finding a model structure to generate it	Skills: knowing how to employ the tools available to construct model that displays given behavior Creativity Level: moderate
systems synthesis	The model is not available for experimentation. The student is trying to come up with a structure that can generate given a description of its input/output behavior	Moving from lower to higher levels, e.g. finding a structure that can generate the given behavior by synthesizing it with components created or already existing	Skills: knowing how to employ the tools available to construct model that displays given behavior Creativity Level: high
systems diagnosis	The system exists but is not behaving correctly. The student has the correct model and the errant version. The student is trying to infer what is wrong by observations of its responses to selected inputs or structure changes	Moving from errant behavior to the possible causes as departures from the correct structure	Creativity Level: high
“figuring systems out”	The system exists but is not behaving correctly. The student has a description of the correct model and the errant version. The student is trying to infer what is wrong by observations of its responses to selected inputs or structure changes	Moving from correct and errant behavior to infer the correct structure	Creativity Level: very high

Table 9 : System problem-solving types

As the terminologies are now clearer, as discussed previously, modeling a Student as a system in the context of DEV-based instruction considering systems-based problem types can be viewed as shown in figure 15.

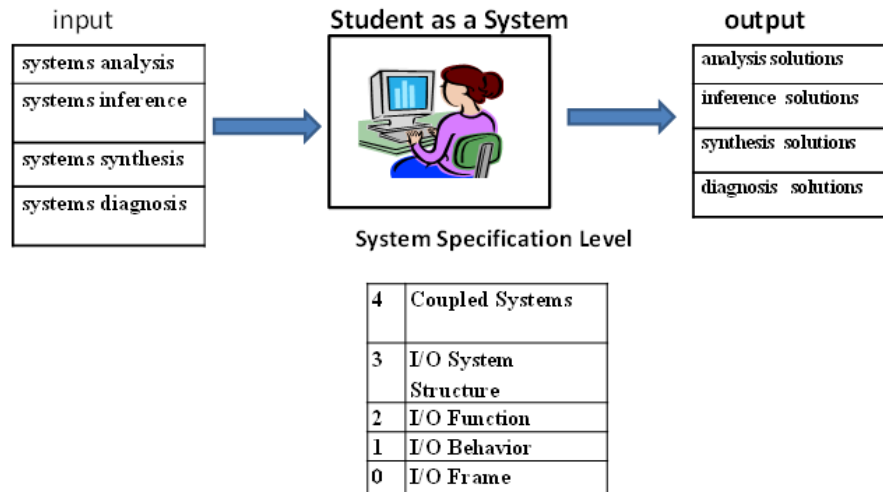


Figure 15 : Student as a system

It should be again reiterated that the theoretical concepts and notions in this sub-section were conceived with the help of Dr. Bernard Zeigler, and some figures are from his content library.

4.2 Research Plan Task 1 : Formulating Background Insight for a Pedagogical Approach

As mentioned in section 3.6, the Task 1 of the research plan was to research, identify and formulate some background insights for the pedagogical approach. Section 4.1 of this chapter would have covered the theoretical requirements part.

4.3 Research Plan Task 2 : GUI Development

With reference to task 2 as mentioned under section 3.6 plan, a user interface incorporating concepts as shown in figure 16 would be developed to allow students to define an FDDEVS atomic model using a succinct natural language. This input will be parsed and analyzed with results displayed for student inspection and correction. The validated input will then be used to automatically generate a DEVSJAVA implementation of the model with its visualized simulation execution. A key innovation will be the accompanying auto-generation of a unique model set called “inverse models” which can be used for testing, verification, and experimentation with a target model.

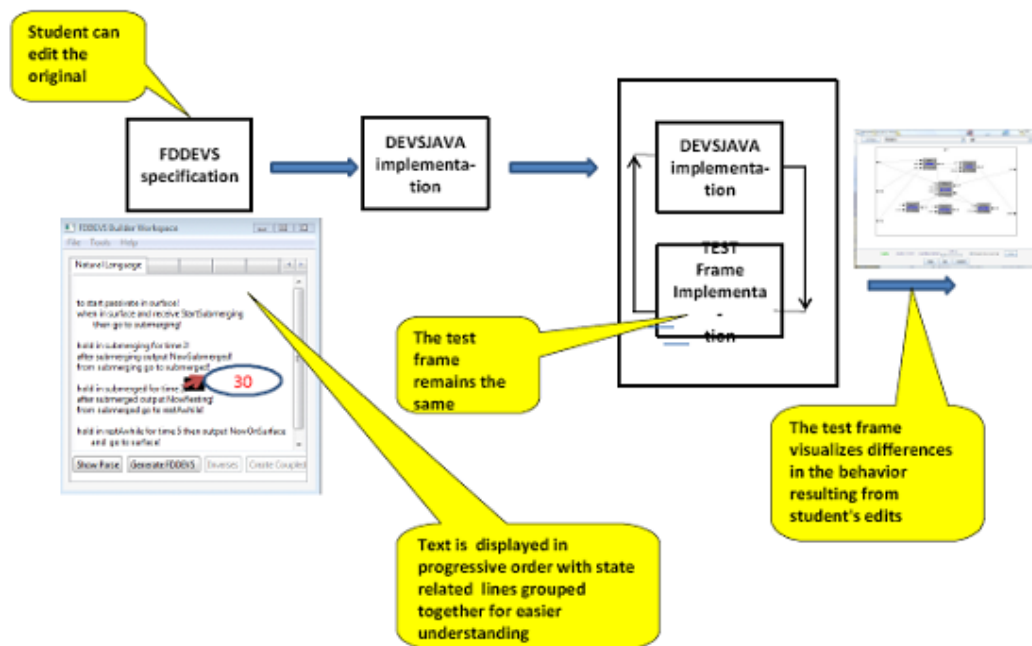


Figure 16 : GUI interface

4.4 Research Plan Task 3 : Guidelines for Active learning Content Development

In this section, we will formulate and discuss the plan for Task 3 as mentioned in section 3.6 to develop instructional support to utilize the GUI/toolset. Table 10 below shows the potential topics, student learning objectives, and instruction methodology along with the possible tests of understanding, skill, and creativity.

It should be noted the topics mentioned may or may not have been isolated into one lesson or would be in the exact order of sequence when content material was developed. The “Tests of Understanding, Skill, Creativity” column shows possible connections to the traditional system problem types and was added only for extra insight for any future needs.

Step Sequence	in	Student Learning Objective	Instruction Delivery Support	Test of Understanding, Skill, Creativity
DEVS Atomic models		Understand <ul style="list-style-type: none"> Ports: input, output States, including starting state Functions: time advance, internal transition, external transition, output 	<ul style="list-style-type: none"> Explanations using an easy-to-understand approach with animations to cover each of the aspects in clear detail 	<ul style="list-style-type: none"> Traditional exam Evaluating student response to systems analysis problem instances No test of creativity
Behavior of Finite DEVS		Understand and be able to: <ul style="list-style-type: none"> Inject inputs Observe state transitions, and outputs 	<ul style="list-style-type: none"> Use the inbuilt SimView tool in the FDDEVS Builder interface 	<ul style="list-style-type: none"> Evaluating student response to systems analysis and system synthesis problem instances No test of creativity
Behavior of Coupled Models		Understand <ul style="list-style-type: none"> Message exchange State trajectory Output trajectory 	<ul style="list-style-type: none"> Explanations using an easy-to-understand approach with animations to cover each of the aspects 	<ul style="list-style-type: none"> Traditional exam Evaluating student response to systems analysis problem instances No test of creativity
Coupled models (Compositions of Finite DEVS)		Understand <ul style="list-style-type: none"> Components Couplings (internal, external) automated port-matching coupling Be able to create coupled models via port-matching 	<ul style="list-style-type: none"> Tutorial with guidelines, demos & exercises 	<ul style="list-style-type: none"> Traditional exam Evaluating student response to systems synthesis problem instances test of creativity
Coupling of Finite DEVS and their Inverses		<ul style="list-style-type: none"> Understand and be able to Produce inverses Be able to couple inverses to models 	<ul style="list-style-type: none"> Tutorial with guidelines, demos & experimentation 	<ul style="list-style-type: none"> Traditional exam Evaluating student response to systems analysis problem instances Some tests of creativity
Construction of Atomic FDDEVS		<ul style="list-style-type: none"> Learn FDDEVS Natural Language format actively via sequential modeling problems Actively learn via experimentation Attain higher Nat language modeling skills 	<ul style="list-style-type: none"> Tutorial presentation with step-by-step guidance on learning the NL with exercises 	<ul style="list-style-type: none"> Evaluating student response to systems synthesis and systems inverse problem instances test of creativity
Construction of Coupled DEVS using the System Entity Structure		<ul style="list-style-type: none"> *Learn to do couplings using System Entity Structures * Modify the SES descriptions to experiment and learn further 	<ul style="list-style-type: none"> Tutorial presentation with step-by-step guidance with exercises 	<ul style="list-style-type: none"> Evaluating student response to systems synthesis problem instances test of creativity

Table 10 : Sample topics and learning objectives for DEVS/FDDEVS instruction.

CHAPTER 5

FDDEVS BUILDER GUI AND INSTRUCTIONAL TUTORIAL INTRODUCTION

5.1 Chapter Introduction

This chapter will provide insight into 2 core topics, namely, tasks 2 and 3 as mentioned in subsection 3.6. The 2 main topics of discussion will be

1. the innovative GUI/ toolset
2. instructional support/tutorials developed to utilize the GUI/toolset to teach basic concepts of DEVS and SESs

5.1.1 Pedagogical Material and Software

All software and material discussed here can be found on the below website specially created for the purpose

Weblink: <http://u.arizona.edu/~lahiru/>

5.2 Supporting Software

As mentioned in subsections 1.3 and 2.3.3, to encourage students who are minimally prepared in programming and math but have an interest to learn M&S through DEVS, a lesser stringent software approach was needed. Hence in this subsection, to serve task 2, we introduce an innovative software called FDDEVS Builder which was specifically developed for such purposes. Utilizing new pedagogical methodology and this software

5.2.1 FDDEVS Builder Software

FDDEVS builder is a software that was developed for simplified DEVS Model specification and generation without the need for prior programming or math skills which was a prerequisite if one was to follow the traditional DEVS learning approach. The software would let a student/user specify a DEVS model using a restricted but easy-to-learn natural language which would allow to then auto-generate DEVS Java code with a click of a button. Revisiting figures 11 and 13 would allow a reader/student to compare the significantly lesser complexity of the natural language model definition vs coding the DEVS Java model outright. It should be noted that this software is rather large-scale and was developed by a team of people. The author has contributed to it in different ways during the latter stages but wishes to assert the credit for the software development should go to a team of people.

5.2.2 FDDEVS Builder GUI and Software Features

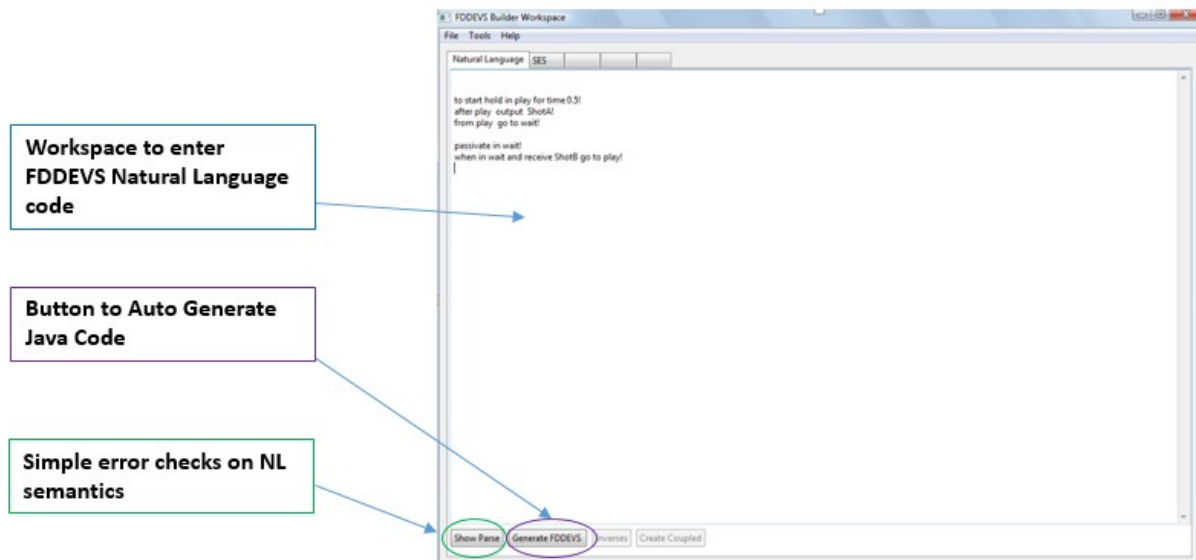


Figure 17 : FDDEVS Builder “Natural Language Tab.

Figure 17 above, shows the FDDEVS Builder Graphical User Interface. The Builder shown here has 2 clickable tabs at the top, “Natural Language” and “SES”.

The “Natural Language” tab is designed with features for novices. Under the “Natural Language” tab, a user can input a textual description of a DEVS model within the workspace provided by utilizing a specialized restricted natural language (basic semantics of the NL are covered under a module/tutorial developed for this research project). Under this tab, there are 4

buttons at the bottom right. The 2 in-focus buttons, namely. “Parse”, and “Generate FDDEVS”, would respectively do some basic error checking/reporting of the user-specified textual description of the DEVS model and allow auto-generation of DEVS Java code. The 2 other buttons named “Inverses” and “Create Coupled” are shown greyed out and would be activated once a specific methodology is followed.

The “SES” tab is designed with features for more advanced users. Figure 18 below shows the tab and its corresponding features/buttons (which are greyed-out in this instance).

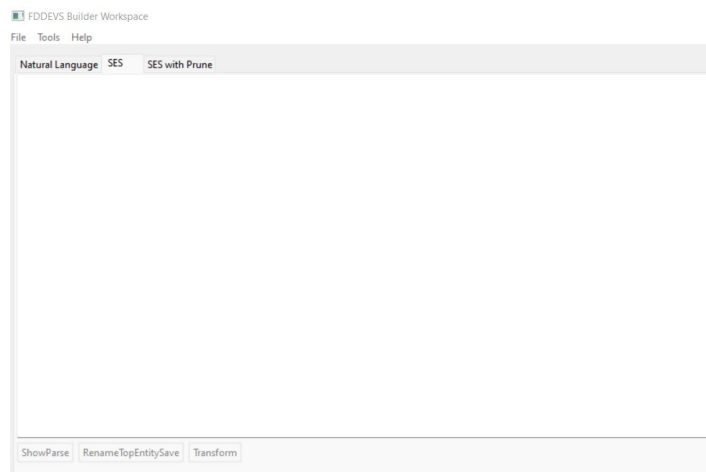


Figure 18 : FDDEVS Builder “SES” Tab.

It should be also noted that in a later updated version of the software, a new tab named “SES with Prune” was added. Figure 19 below shows the tab and its feature buttons.

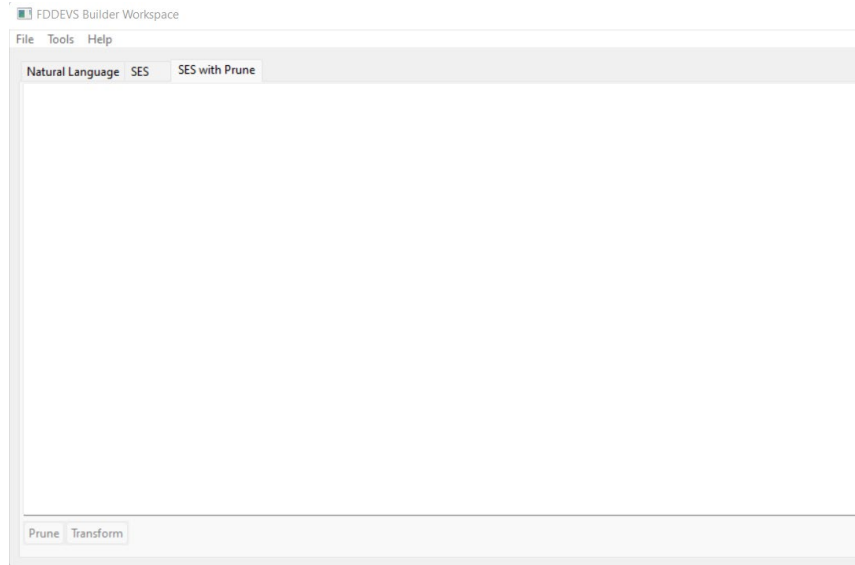


Figure 19 : FDDEVS Builder “SES with Prune” Tab

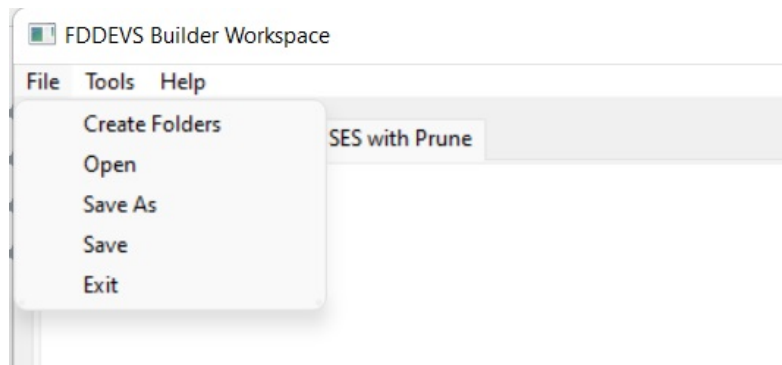


Figure 20 : FDDEVS Builder “Files” Menu option

Figure 20 above, shows the options under the “File” menu item of the GUI, and figure 21 below shows the “SimView” visualization option under the “Tool” menu item.

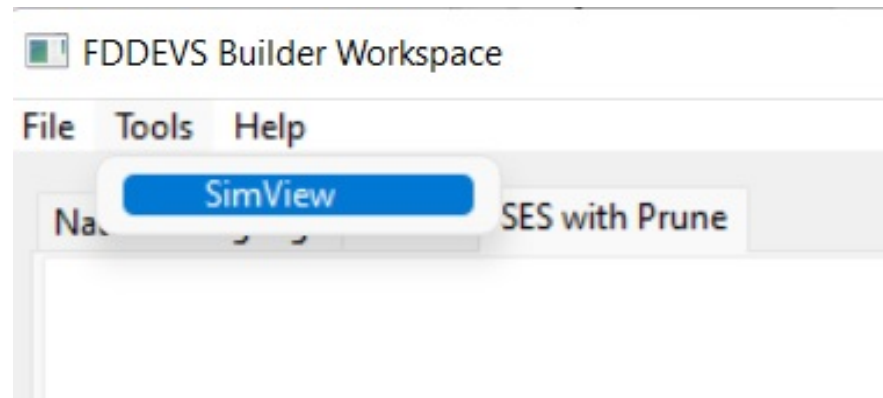


Figure 21 : FDDEVS Builder “Tools” Menu option

It should be noted that detailed pedagogical material and tutorials were developed covering each Tab, Menu item, and the use of their relevant features which would be visited in the next section.

5.2.3 Software Installation and Notes

To install and execute the FDDEVS Builder, the free open-source Eclipse IDE is utilized. All needed software, updates, links, and step-by-step installation guidelines via clear presentation or tutorial formats are hosted on the aforementioned website. Figure 22, and figure 23 below respectively show sample screenshots from an interactive presentation and a non-interactive installation tutorial.

It should be also noted that though the tutorials/content may have links or mentions of previous versions of software (ex : Eclipse IDE versions), the most recent versions will work seamlessly with the FDDEVS Builder. The most important issue to note is to install a 32-bit version of a JRE (Java Runtime Environment) to run in Eclipse IDE. For example, even if you have installed a 64-bit version of Eclipse IDE make sure you run a 32-bit version of a JRE.

Note

The below task can also be performed by

- Copying the contents of C:\installFDDEVSep\Examples
 - Can use Ctrl+C
- To C:\installFDDEVSep\FDDEVS\src\FDDEVSExamples\Models\txt folder
 - Can use Ctrl+V

Within Eclipse IDE,
right click on the txt subfolder under (/FDDEVSExamples/Models)
and choose **Import**.
Then choose **File System** and under General ,click Next

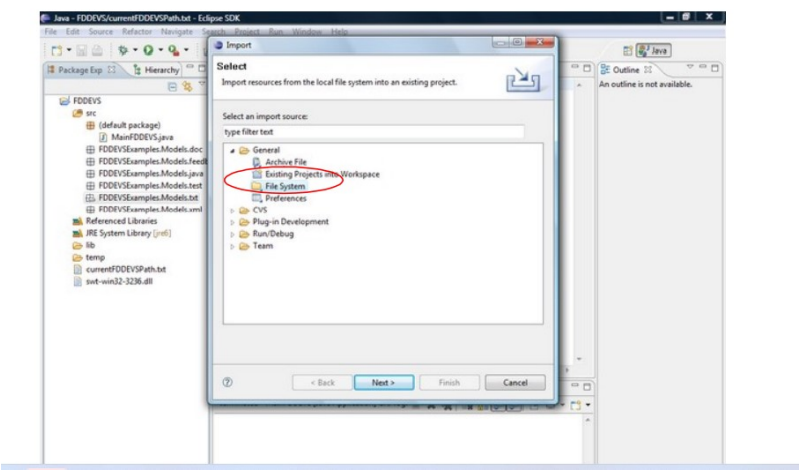


Figure 22 : Sample content page from an installation tutorial

Step 2: Run FDDEVS Builder GUI from Eclipse

- To run FDDEVS
 - Right click on MainFDDEVS.java (found under the src folder) in Eclipse
 - Run As a Java Application.

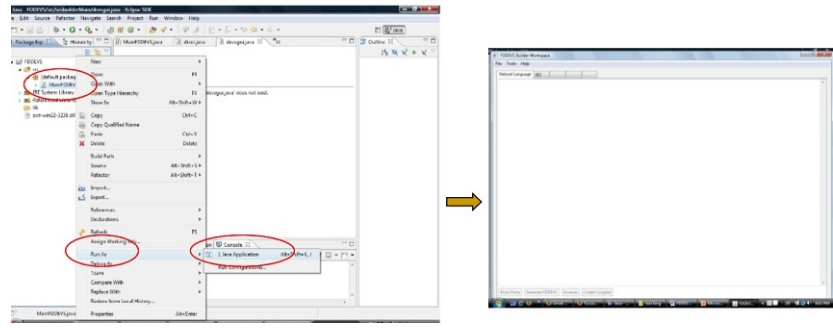


Figure 23 : Static screenshot from an interactive installation guide presentation

5.3 Supporting FDDEVS Pedagogical Material

A comprehensive set of material/tutorials were developed for this research/study to cover the FDDEVS Builder software features, impart basic DEVS M&S knowledge, and involve students in active learning and deeper thinking wherever possible.

5.3.1 Material Organization

The set of pedagogical material developed for this study targets 2 main groups: Beginners, and more Advanced students/users. The organization of the material created on the previously mentioned site <http://u.arizona.edu/~lahiru/> could be categorized as below.

- 1) Introductory material on M&S, DEVS/FDDEVS concepts, and FDDEVS builder usage for beginners
- 2) Material with more advanced concepts for more advanced user

5.3.1.1 Introductory Material

On the aforementioned website, the basic concept coverage of DEVS-based M&S, and features and usage of FDDEVS software along with supporting learning material are categorized under the “Introductory” tab as shown in figure 24 below.

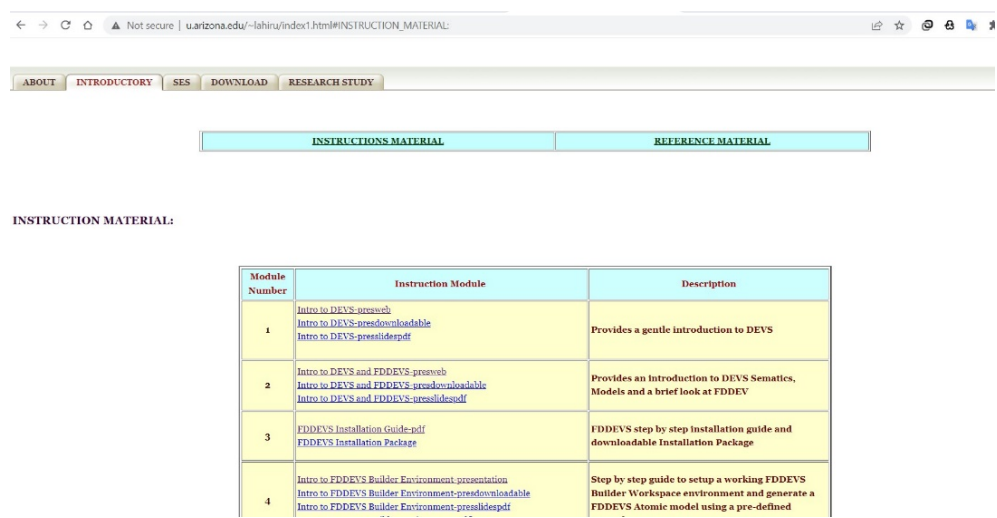


Figure 24 : Introductory material listed on the website

Figure 25 below shows a “static” screenshot from a dynamic and interactive presentation to teach the basics of M&S.

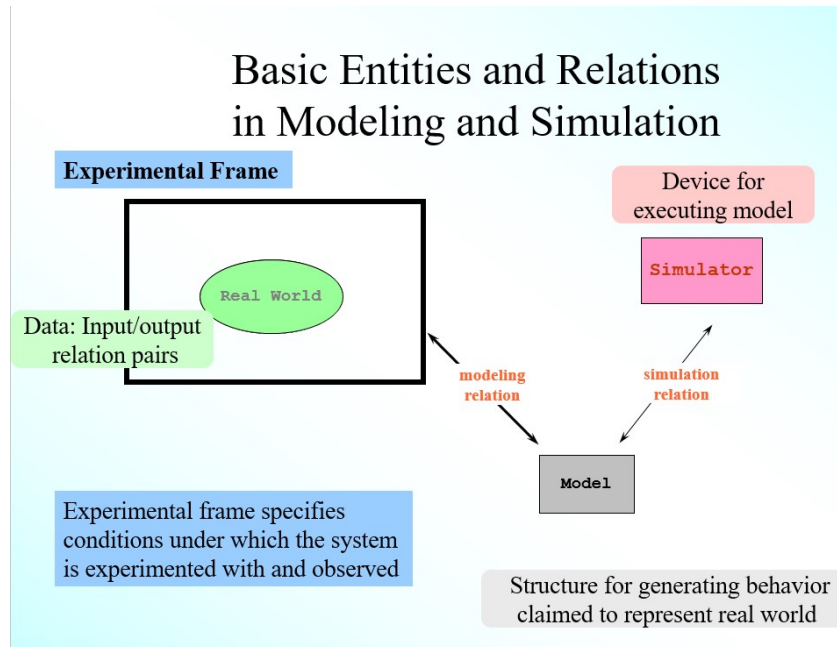


Figure 25 : Static screenshot from a dynamic presentation content slide

Whenever suitable, learning by example is used as the preferred method of knowledge imparting to reach higher levels of student comprehension and active learning. Figure 26 and figure 27 show such sample slides from 2 separate interactive presentations.

Learning by Example

- Let us start with a simple example
- **Problem** : In a singles **Tennis match** a **PlayerA** plays a shot and await a return shot from the opponent **PlayerB**. Assume it takes **0.5 seconds** to play a shot. Develop a model to describe **PlayerA** and Simulate it.

Figure 26 : Learning by example 1

Problem 3 (Interruptible Processor) :

- **Problem 3 (Interruptible Processor)** : **Modify the above Simple Processor model, to represent a processor which behaves the same as the above as far as accepting a Job when idling. However, if another Job arrives while processing a Job, the arriving Job takes higher priority and replaces the Job in process, i.e., processing the original Job is aborted and the new Job will complete after the given processing time (20) has expired.**

Same as Before

modify

Figure 27 : Learning by example 2

Reminders were employed much as possible during material development to reassert concepts and improve student learning. Figures 28 and 29 show such an example.

Important to Remember

“!” must to be included at the end of every NL sentence.

- This is analogous to using a period (ie “.”) when we end sentences in English.

Figure 28 : Reminders drive home concepts better!

Important to Remember

For the user defined input message value

- “ShotB”
 - FDDEVS **will auto create an input port named inShotB**
 - **during DEVSJAVA mapping!**
- It is important to remember this fact.

Figure 29 : Reminders to make people remember!

5.3.1.2 Advanced/Intermediate Material

On the website, the more advanced topics are covered under the “SES” tab as shown in Figure 30 below.

ABOUT INTRODUCTORY **SES** DOWNLOAD RESEARCH STUDY

INSTRUCTION MATERIAL:

Module Number	Instruction Module	Description
1	Download Software Update Updating FDDEVS Software-Presentation Updating FDDEVS Software.pdf	<p>Note :</p> <p>It is Mandatory that you download and do this software update before you can follow the lessons in SES.</p> <ul style="list-style-type: none"> It is also assumed that you have already Installed the FDDEVS software before you do this update <p>The lesson here provides a step by step guide to updating your existing FDDEVS software.</p>
2	Editing/Creating Couplings with SES-Presentation Editing/Creating Couplings with SES.pdf	Provides a step by step guide to introduce an advanced feature of FDDEVS involving System Entity Structures to modify/create model couplings
3	Exercise FDDEVS with SES-Presentation Exercise FDDEVS with SES.pdf FDDEVSNL.zip	Exercise involving/reviewing DEVS concepts FDDEVSNL and SESNL. Also need to download the FDDEVSNL.zip which has incomplete FDDEVSNL code.
4	Atomic Model Selection with PES-Presentation Atomic Model Selection with PES.pdf	Provides a step by step guide on an advanced feature of FDDEVS involving Atomic Model Selection using Pruned Entity Structures

Figure 30 : Advanced topics found under the “SES” tab

During the advanced topic material development process, attention was given to better the student learning experience by providing modeling tips (figure 31), revisiting prior concepts before expanding to new ones (figure 32), and revisiting prior examples with to-do refresher student tasks (figure 33) whenever possible.

Step 5 : Inheritance

Modeling Tip

As a practice, always give the same name for the saved file and the SES top entity contained in that file.

Figure 31 : Advanced topics found under the “SES” tab

Step 1 : Creating Atomic and Coupled Models revisited

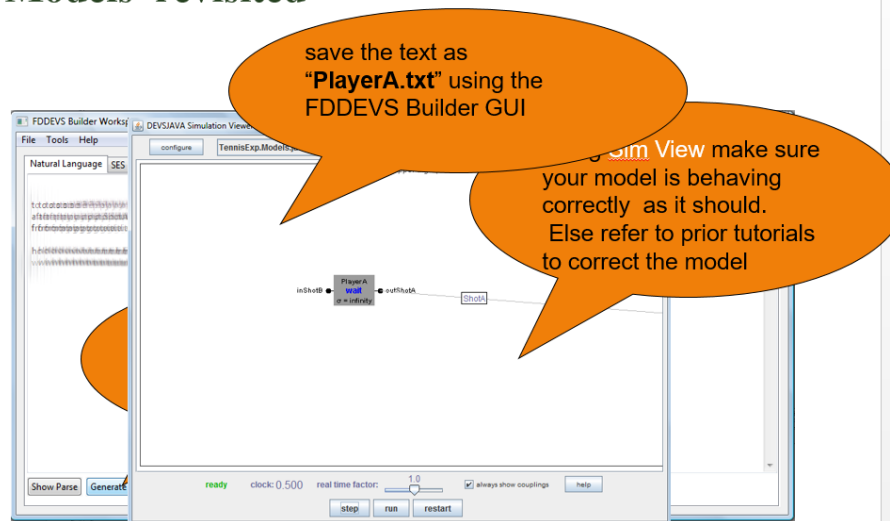


Figure 32 : Revisiting past concepts to refresh memories

Step 1 : Creating Atomic and Coupled Models revisited ...

As a refresher

- can you fill in the blanks for the FDDEVS natural language description below of TennisPlayerA?

```

_____ in play for time 0.5!
_____ play _____ ShotA!
from play go to _____!

hold in _____ for time Infinity!
when in _____ ShotB go to _____!

```

Figure 33 : Revisiting past examples to refresh memories

5.3.2 Content Distribution Formats

It should be noted that each topic of learning is supplemented whenever possible by up to four formats of content presentation delivery variations. 2 examples of tutorials using 2 different formats are shown in Appendix B.

- i) Interactive Presentation format. The author recommends this format as the most effective way. These interactive presentations were carefully designed (whenever possible), to shorten learning time, and to impart knowledge more quickly than other pedagogical methods.

- ii) Downloadable and independently executable (without PowerPoint installation, browsers, or other software) .exe file. Same as number i).
- iii) Traditional, non-interactive tutorials (mainly done as software installation and configuration tutorials) in pdf format.
- iv) Inanimate presentation slides as a pdf document.

CHAPTER 6

FDDEVS ANIMATOR

6.1 FDDEVS Animator Introduction

FDDEVS Animator was developed as a sister software to FDDEVS Builder with the intention that it would supplement the overall educational experience of learning DEVS Modeling basics via dynamic visualization, animation, and analysis. The software would generate learning-friendly analytical breakdown information of FDDEVS models, and simulate the behavioral aspects of a model via an auto-generated presentation with dynamic illustrations/animations via user inputs when necessary. It should be noted that the software is not perfected as of date, and is somewhere between the pre-alpha and alpha stages of its life cycle. However, as the software has a rich set of working features that would supplement the goals of this research/project, it is hence included here in some greater detail.

As shown in figure 34, the FDDEVS Animator has been integrated into a version of the FDDEVS Builder as a clickable extension.

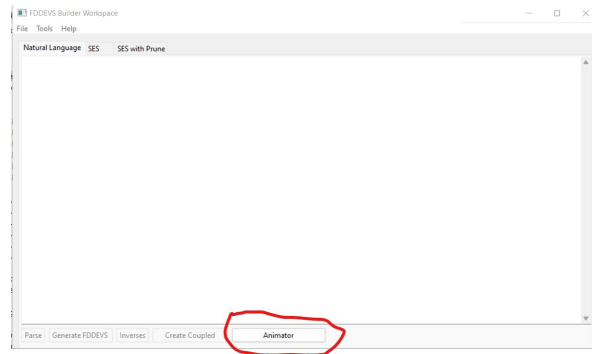


Figure 34 : FDDEVS Animator added as an extension to FDDEVS Builder

Once clicked/executed, as shown in figure 35, the below FDDEVS Animator Graphical User Interface would be displayed!

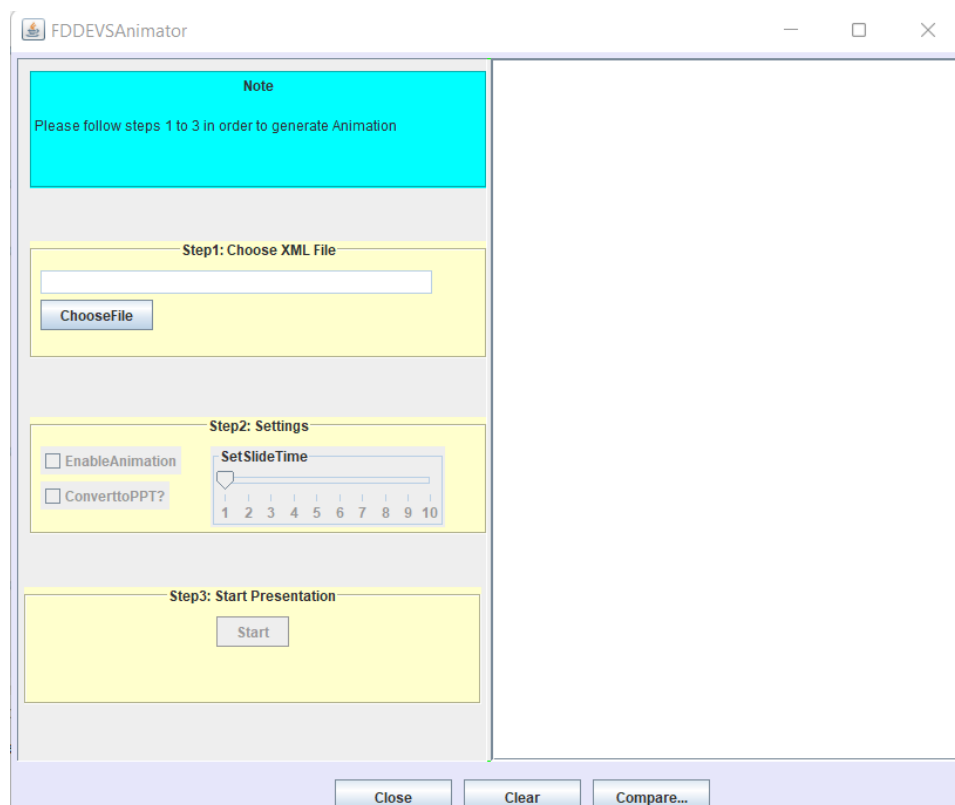


Figure 35 : FDDEVS Animator GUI

6.2 FDDEVS Animator Dependencies, Work-Folders and Inputs

The FDDEVS Animator takes as input, an XML file/s of FDDEVS model/s which would be auto-generated by the FDDEVS Builder as discussed under Chapter 5/tutorials. In the pre-alpha unified version of the FDDEVS Builder and FDDEVS Animator software, the XML model file will be auto-generated and placed into 2 locations,

- i) FDDEVSExamples\Models\xml folder
- ii) FDDEVSAnimator\input folder

which are both found under the workspace of Eclipse.

6.3 FDDEVS Animator Outputs

The Animator generates 4 kinds of outputs

- i) analytical info/content breakdown of a single model
- ii) FDDEVS model animation via auto-executed OpenOffice presentation format
- iii) .odp file (open office presentation file) and a converted .ppt (PowerPoint) file
- iv) comparison of two separate FDDEVS models

Each output format and the corresponding features will be discussed in detail below.

6.3.1 Analytical Info/Content Breakdown of an FDDEVS Input Model

As shown in figure 36, once the “Choose File” option is clicked it will open up a selectable list of FDDEVS XML models to pick from as shown in figure 37.

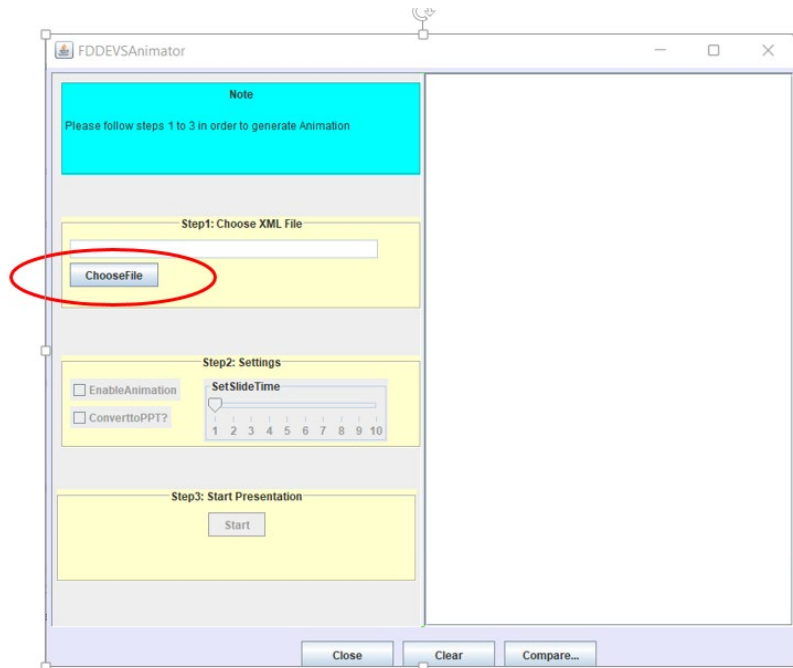


Figure 36 : FDDEVS Animator “Choose File” option

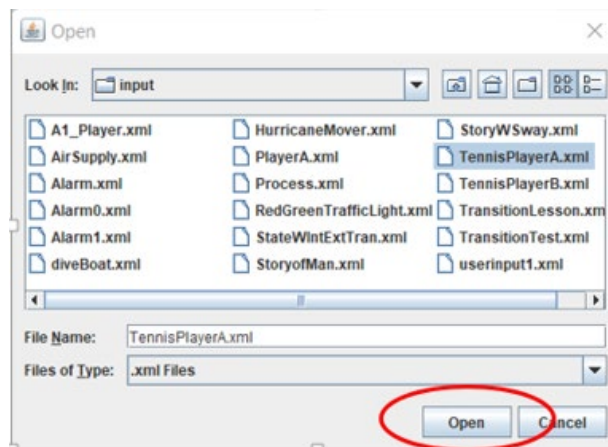


Figure 37 : Selectable FDDEVS XML model list

Once the file is chosen and “Open” is clicked, FDDEVS Animator software will analyze and populate the right-hand pane with a breakdown of DEVS model-based info as shown in figure 38.

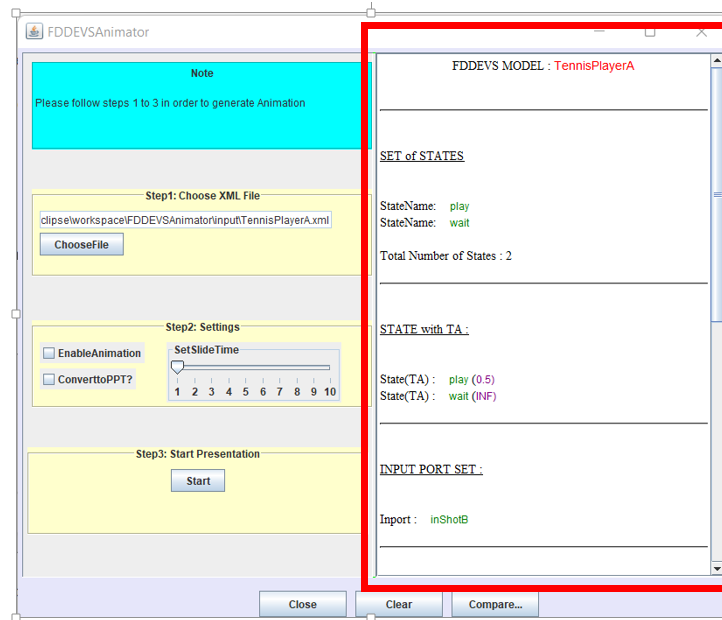


Figure 38 : Analytical breakdown of FDDEVS model

On close inspection, you will note the model-based analytical output generated will provide a clear-cut breakdown of useful information to the student, such as the number of states and their corresponding names, state transition times, input ports and their names, output ports and their names, internal transition mappings, external transitions mappings, and the output message mappings. Figure 39 below shows the auto-generated analytical output of a Tennis Player A model.

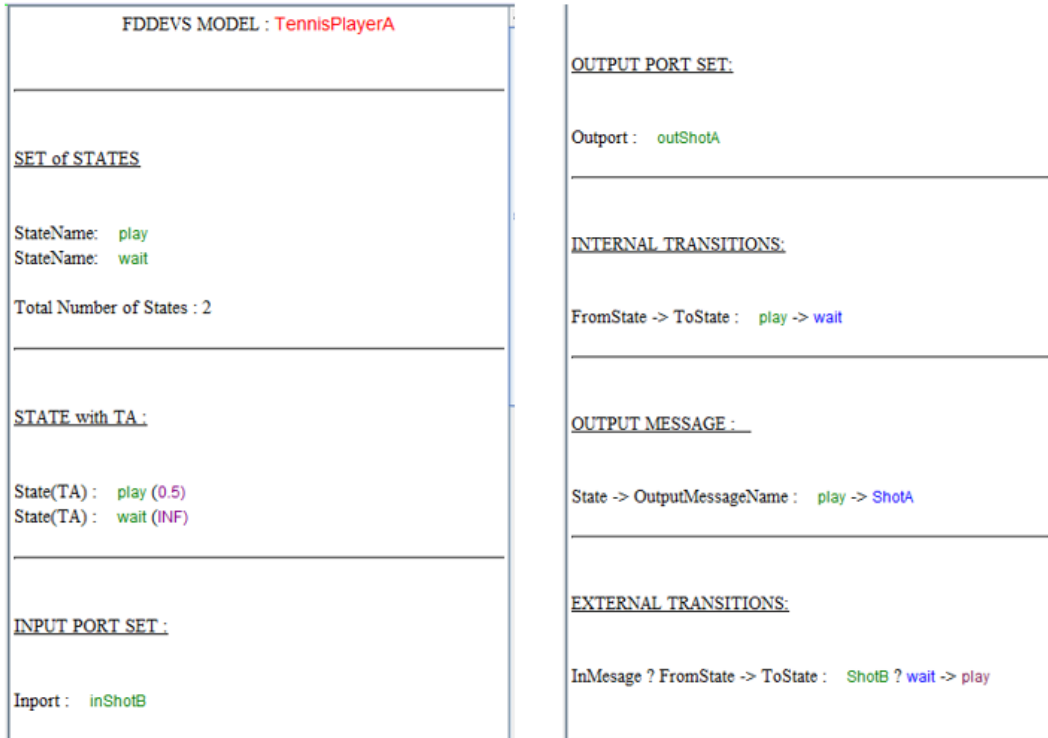


Figure 39 : Detailed analytical info generated by FDDEVS Animator

6.3.2 Model Behavior Simulation via Auto-Executed OpenOffice Presentation Format

The most unique and useful feature of the FDDEVS Animator is its ability to auto-formulate and generate an Open-Office based presentation (.odp format) to demonstrate the dynamic behavioral working of the FDDEVS model under consideration.

6.3.2.1 Model Behavior Simulation Without Selectable Options

Figure 40 below shows the method to execute the presentation without any optional bells and whistles.

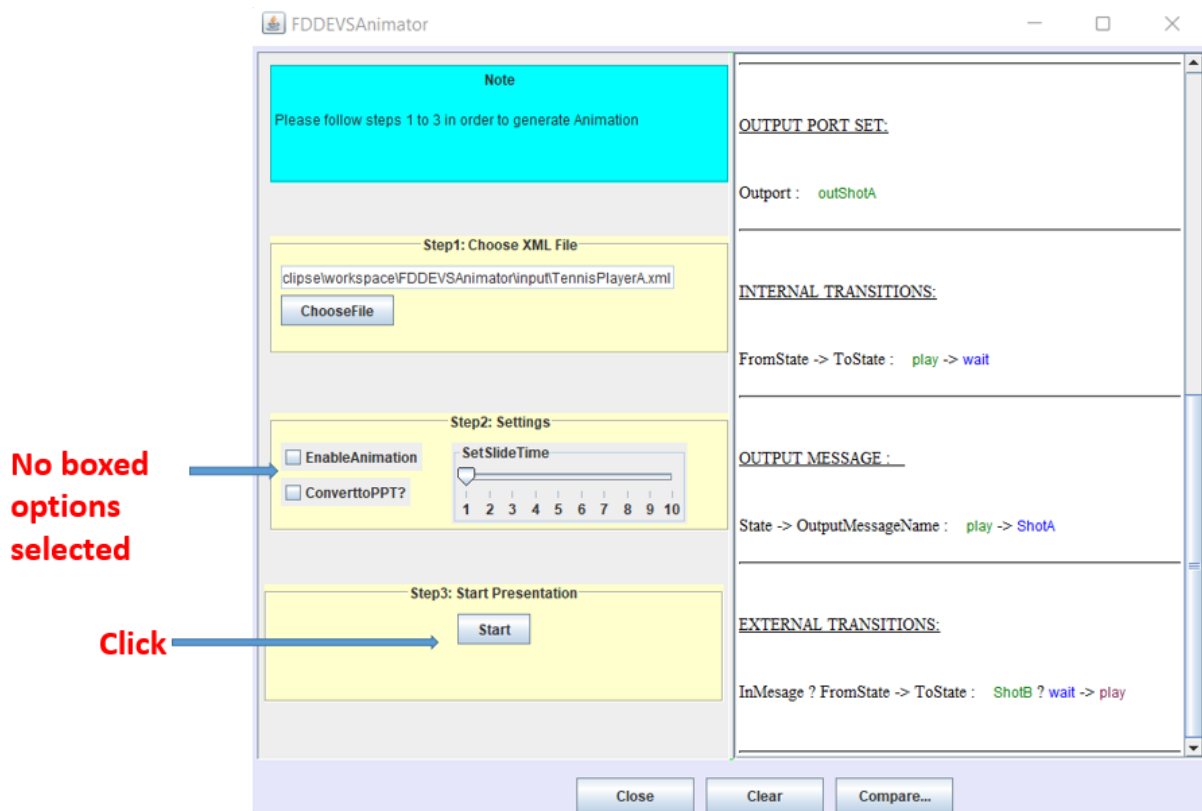


Figure 40 : FDDEVS Animator presentation execution sans selectable options

Once the “Start” button is clicked, it will generate and auto-execute an Open-Office-based presentation. As shown in figures 41 and 42, the auto-generated dynamic presentation below is for the Tennis Player A model.

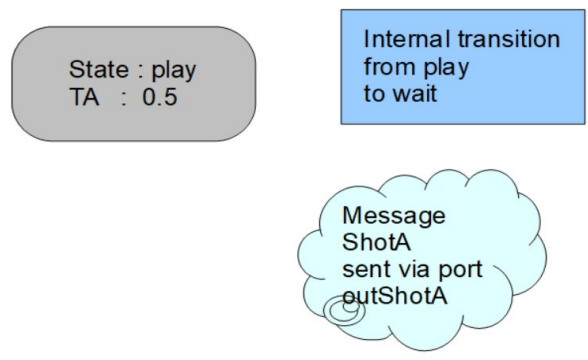


Figure 41 : A static screenshot from the Tennis PlayerA presentation animation

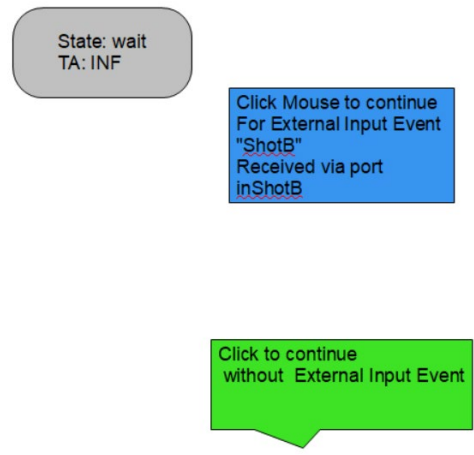


Figure 42 : Another static screenshot from the Tennis Player A presentation animation

It should be noted that the FDDEVS Animator presentation will attempt to illustrate all the dynamic behaviors of the model under consideration like transition-time-advancing, internal transitions, auto-generated output messages, and state changes in real-time.

To illustrate

- i) internal transitions, the software will generate a “light-blue rectangle shape with descriptive info”, and will “auto-click” (without user intervention) on the shape to simulate the internal transition to the next state.
- ii) external transitions when necessary, the presentation would wait for the user input on a clickable area to proceed to the next state/stage.

For example, as shown in figure 43, the software will generate an auto-mouse-click on the light-blue descriptive box to simulate an internal transition.

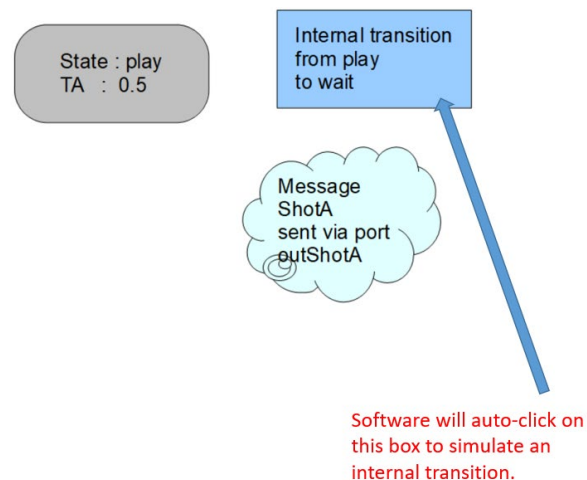


Figure 43: Software will generate an auto-mouse-click to demo an Internal transition

To simulate an external event/transition, as seen in figure 42, the user needs to click the darker-blue-shape which will trigger the external event “ShotB” and will result in the slide transition back to the “play” state.

If the user chose to click the green shape, in figure 42, he/she would observe that the slide would remain in the same state (wait) to simulate the passing of TA without any external event trigger.

6.3.2.2 Model Behavior Simulation With Selectable Options Enabled

There are 3 selectable options affecting the presentation content/dynamics as shown in figure 44. The options are, namely

- i) Enable Animation
- ii) Convert to PPT
- iii) Set Slide time

- i) Option “Enable Animation”: Enabling the option would add some extra rudimentary stick figures or objects to the presentation to further enhance the visual experience/learning. Such animation can be static in nature as shown in model presentation animation examples of figures 45 and 46, or dynamic as in figures 47 and 48 model presentation/animation examples.

It should be noted that extra objects/images are loaded to the presentation depending on the model selected as an input, and such animation/images are limited in nature at this stage and will not be active for any/all models. However, even if the option is selected and no extra animation is available, the basic presentation with other features described in section 6.3.2.1 will work. Such an option was included to demonstrate the versatility and expandable capacity of the software.

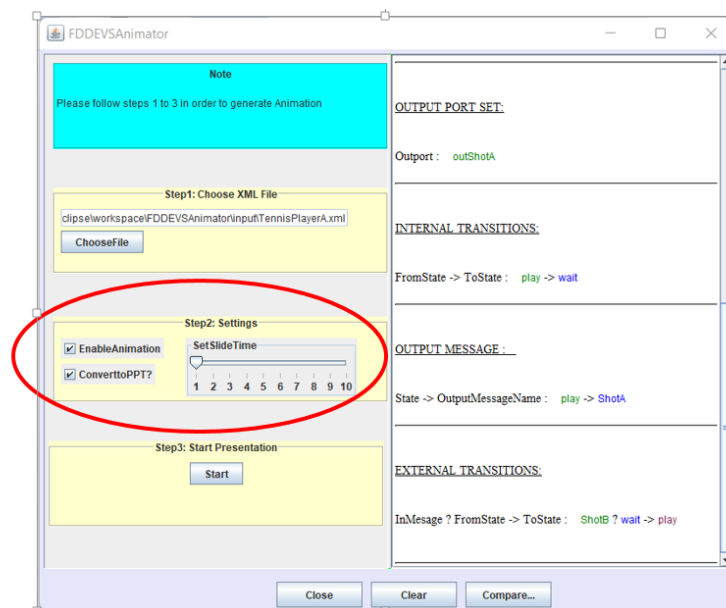


Figure 44: Presentation presets/selectable options

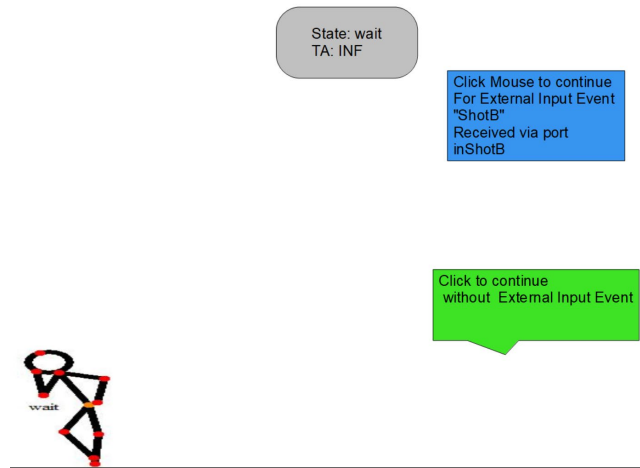


Figure 45: A stick figure is added to denote the “wait” state in “TennisPlayerA” model

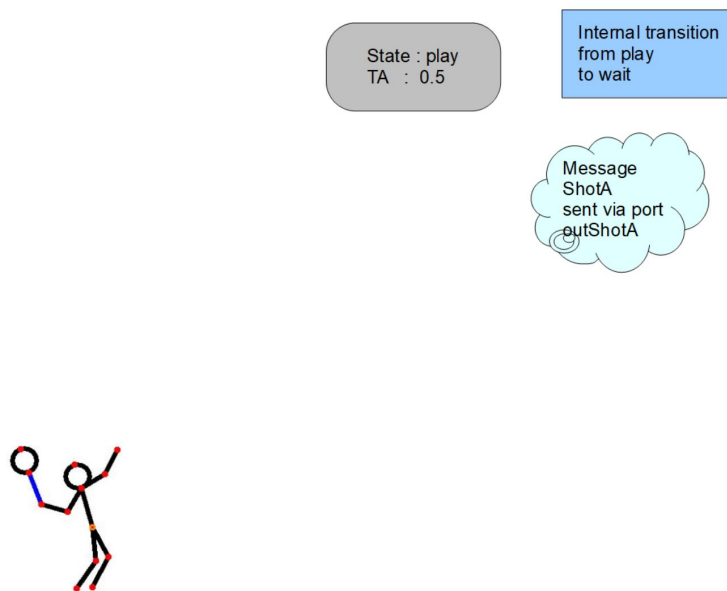


Figure 46: A stick figure is added to denote the “play” state in “TennisPlayerA” model.

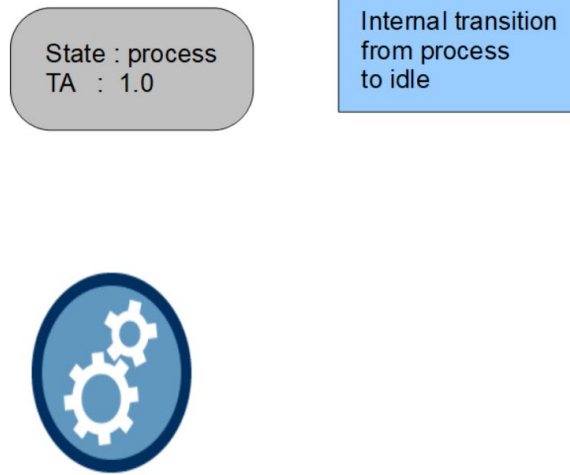


Figure 47: Dynamically rotating gears were added to denote the “process(ing)” state

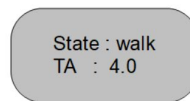


Figure 48: A dynamic stick figure was added to denote the “walk” state in “StoryofMan”

- ii) Convert to PPT: This option will convert the Open-Office presentation (.odp) to Powerpoint (.ppt) and save it in the same output folder. This feature was introduced to allow someone who prefers to view/analyze the auto-formulated presentation/animation in Powerpoint format. Though the execution will be similar to the Open-office presentation, please note that the auto-mouse-click internal transitions will not work with this option. Such will only work with the original OpenOffice format.

- iii) Set Slide time: This feature will allow the user to scale/slow down time by 1 to 10 times. For example, if you select 5 on the slider scale, it will slow down each time tick 5 times instead of 1 (which would be in real-time). In some models which will have states that have TA timings of less than 1 second (for example Tennis Player A “play" state), it will be very hard for the learner to observe the presentation transitions/animations as things happen too fast. Hence enabling this feature will be of use depending on model timing or complexity.

6.3.3 Comparison of Two FDDEVS Models

As shown in figure 49, clicking the “Compare” button will enable another learning tool useful to students which could analytically break down, compare, and contrast 2 FDDEVS models.

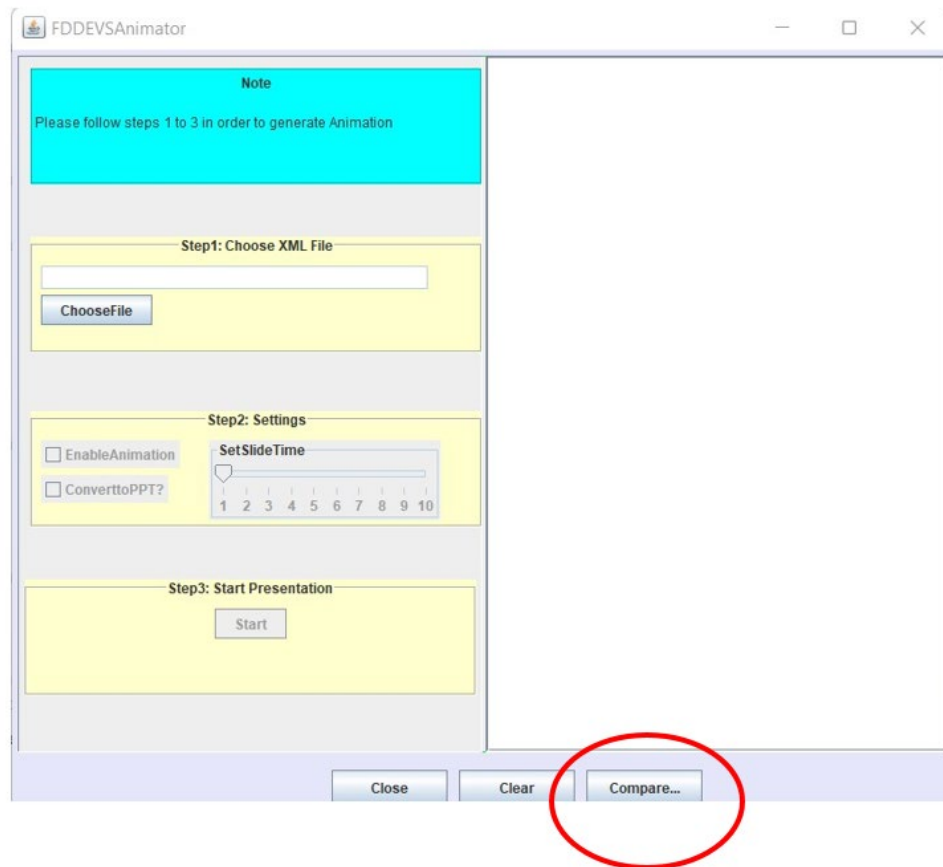


Figure 49 : Feature to compare 2 FDDEVS models

Once the “Compare” button is clicked, a new interface will open as shown in figure 50.

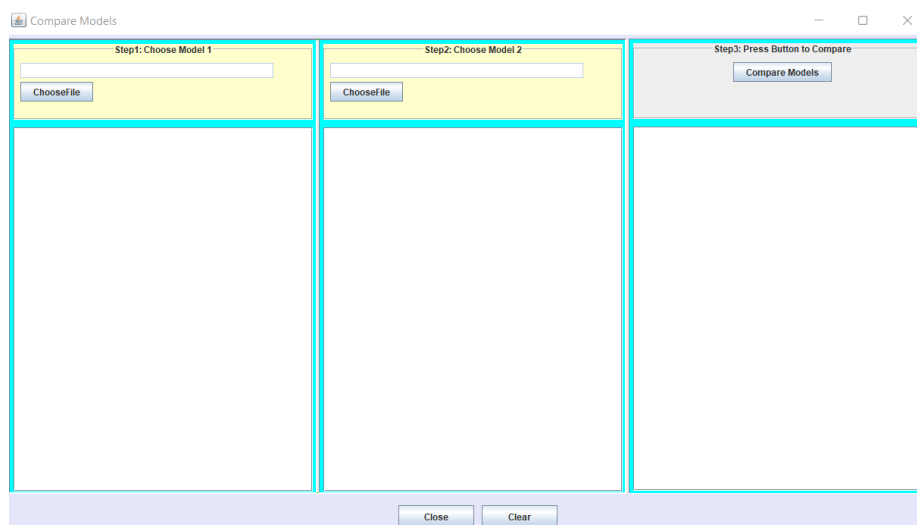


Figure 50 : Model selection and comparison interface

Using the interface, one can select 2 models (different or identical-which would be rather redundant) as shown in figure 51.

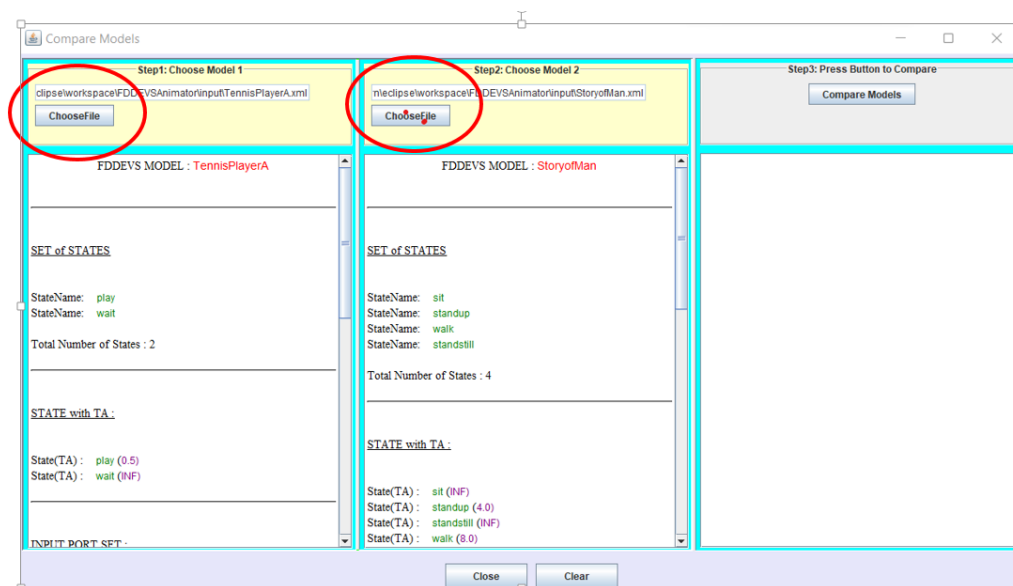


Figure 51 : 2 FDDEVS models are selected for comparison

The right-side and middle panes will each display the relevant analytical information of the 2 chosen FDDEVS models which is similar to the discussion in section 6.3.1. However, the key feature of this tool would be the option to analytically compare and contrast the 2 models which is executed by clicking the “Compare Models” button on the left-side pane as shown in figure 52.

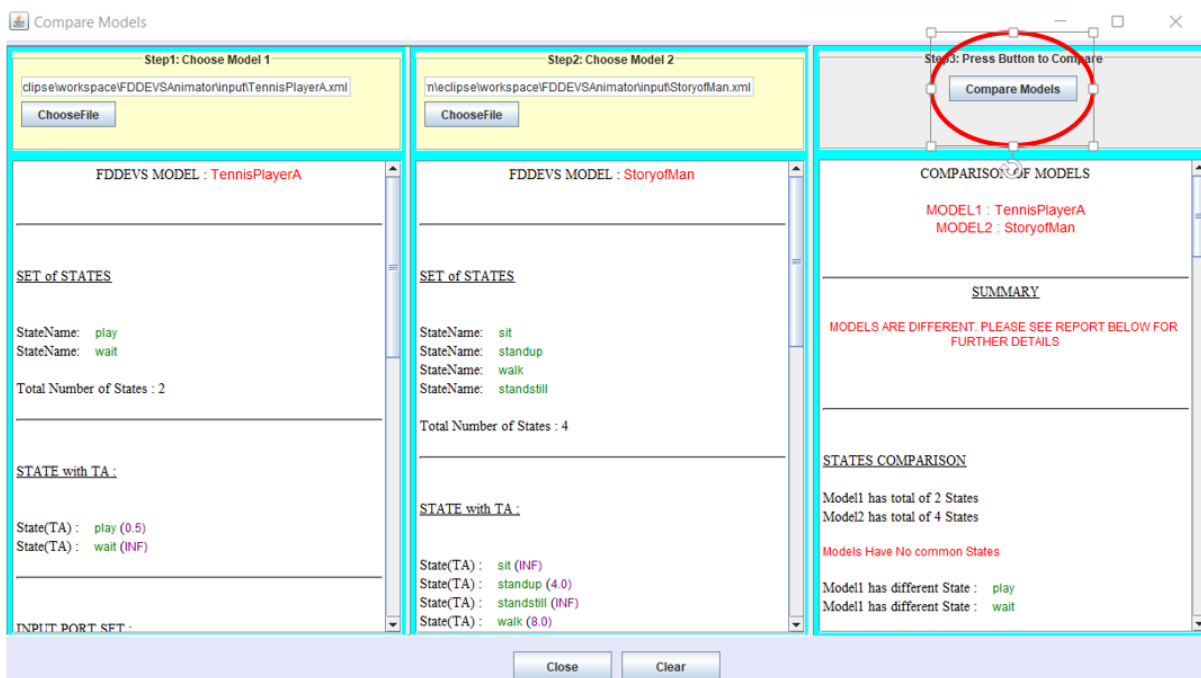


Figure 52 : Clicking the “Compare Models” button to enable the model comparison feature

As you will note, the 2 models chosen here are “TennisPlayerA” and “StoryofMan” for demonstration purposes and the left-side pane is now populated with analytical information comparing and contrasting the 2 models.

Step3: Press Button to Compare

Compare Models

<p style="text-align: center;">COMPARISON OF MODELS</p> <p style="text-align: center; color: red;">MODEL1 : TennisPlayerA MODEL2 : StoryofMan</p> <hr/> <p style="text-align: center;"><u>SUMMARY</u></p> <p style="color: red; text-align: center;">MODELS ARE DIFFERENT. PLEASE SEE REPORT BELOW FOR FURTHER DETAILS</p>	<p><u>STATE TA COMPARISON:</u></p> <p style="color: red;">Models Have No common States with matching TAs</p> <p>Model1 have State and TA: play 0.5 Model1 have State and TA: wait INF</p> <p>Model2 have State and TA: sit INF Model2 have State and TA: standup 4.0 Model2 have State and TA: standstill INF Model2 have State and TA: walk 8.0</p>	<p><u>INTERNAL TRANSITIONS:</u></p> <p>Model1 has total of 1 Internal Transitions Model2 has total of 3 Internal Transitions</p> <p style="color: red;">Models Have No common Internal Transitions</p> <p>Model1 have Internal Transitions FromState to NextState: play to wait</p> <p>Model2 have Internal Transitions FromState to NextState: sit to sit standup to walk Model2 have Internal Transitions FromState to NextState: walk to standstill</p>
<p><u>STATES COMPARISON</u></p> <p>Model1 has total of 2 States Model2 has total of 4 States</p> <p style="color: red;">Models Have No common States</p> <p>Model1 has different State : play Model1 has different State : wait</p> <p>Model2 has different State : sit Model2 has different State : standup Model2 has different State : standstill Model2 has different State : walk</p>	<p><u>INPUT PORT SET :</u></p> <p>Model1 has total of 1 Input Ports Model2 has total of 1 Input Ports</p> <p style="color: red;">Models Have No common Input Ports</p> <p>Model1 has different Input Ports : inShotB</p>	<p><u>OUTPUT MESSAGE :</u></p> <p>Model1 has total of 1 Output Messages Model2 has total of 0 Output Messages</p> <p style="color: red;">Models Have No common Output Messages</p> <p>Model1 has below Output Messages</p> <p>State -> OutputMessageName : play -> ShotA</p> <p>Model2 has No Output Messages</p>

Figure 53 : Example analytical information generated by the model comparing tool

Figure 53 above shows some selected outputs generated by the model comparison tool. As can be seen, such info can be very useful for a student to compare, contrast, and learn in-depth about DEVS/FDDEVS models/modeling.

Though it is unlikely unless intentionally done, if you chose 2 models with the same number of states, and all other DEVS/FDDEVS info matching, then the tool will spit out a note saying “Models are Identical”. Figure 54 is just a feeble attempt to pick the same model 2 times to compare so the results will say the models are identical.

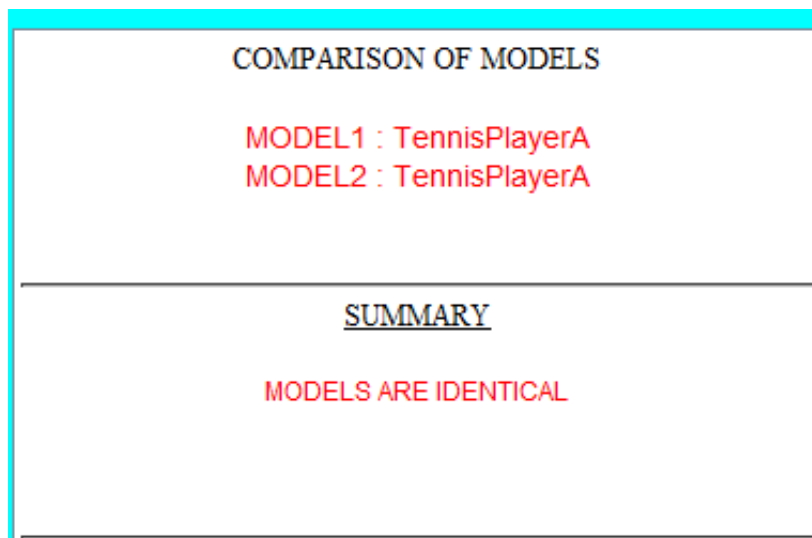


Figure 54 : Picking the same model 2 times demo they are identical

6.4 More Features, Caveats, and Challenges Faced

6.4.1 Additional Features

The FDDEVS Animator also has a “Clear” button feature located on the main interface and the model comparison interface as shown in figure 55.

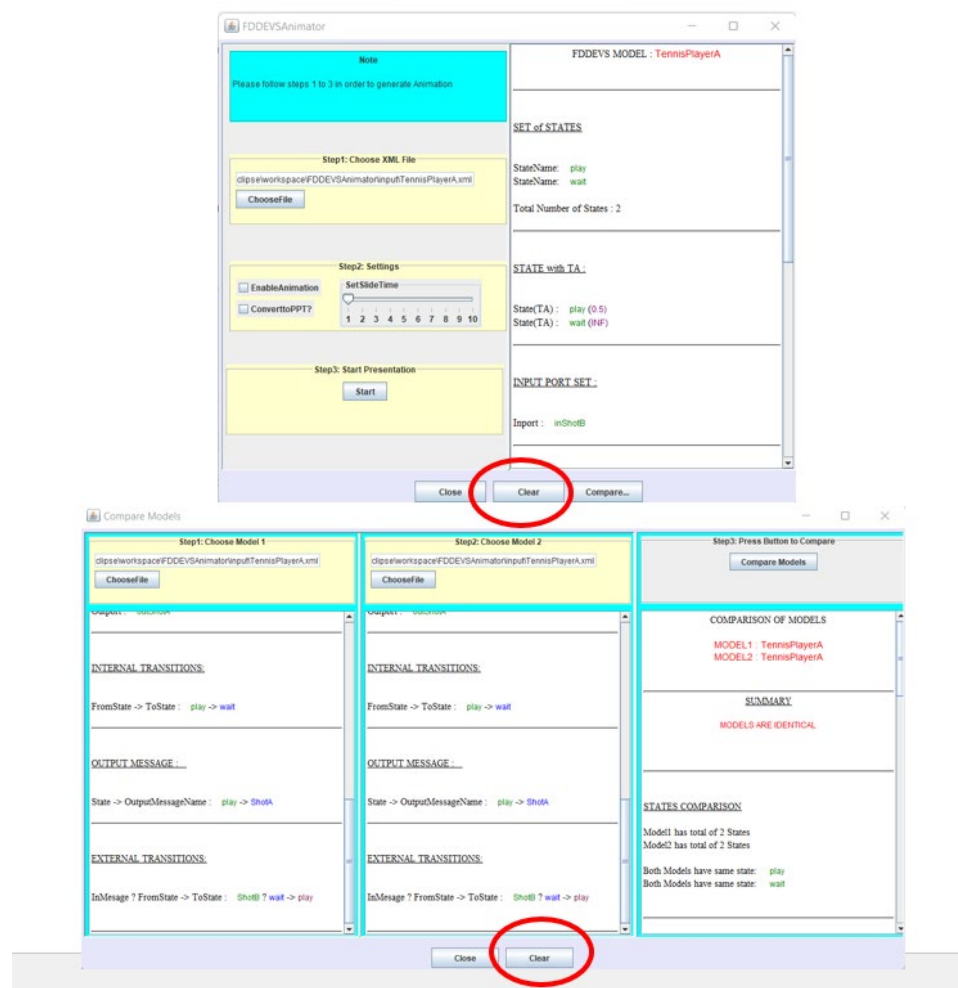


Figure 55 : Clear button feature

The “Clear” feature, as the name will suggest, will simply erase/clear all auto-populated panes/windows to default status.

The FDDEVS Animator software is designed to handle rather sophisticated models. Figure 56 shows a model which has multiple states, multiple external-input options, multiple output messages, and complex internal and external transitions.

```

SET of STATES
StateName: waitForCategory
StateName: moving3
StateName: moving4
StateName: sendNotice3
StateName: sendNotice4
Total Number of States : 5

STATE with TA :
State(TA) : waitForCategory (INF)
State(TA) : sendNotice4 (0.0)
State(TA) : moving4 (5.0)
State(TA) : moving3 (10.0)
State(TA) : sendNotice3 (0.0)

INPUT PORT SET :
Inport : inFinished
Inport : inCategory3
Inport : inCategory4

OUTPUT PORT SET :
Output : outNotifyOthers
Output : outMarkCoveredArea3WS
Output : outMarkCoveredArea4WS

INTERNAL TRANSITIONS:
FromState -> ToState : waitForCategory -> waitForCategory
FromState -> ToState : sendNotice4 -> moving4
FromState -> ToState : moving4 -> sendNotice4
FromState -> ToState : moving3 -> sendNotice3
FromState -> ToState : sendNotice3 -> moving3

OUTPUT MESSAGE :
State -> OutputMessageName : sendNotice4 -> NotifyOthers
State -> OutputMessageName : moving4 ->
MarkCoveredArea4WS
State -> OutputMessageName : moving3 ->
MarkCoveredArea3WS
State -> OutputMessageName : sendNotice3 -> NotifyOthers

EXTERNAL TRANSITIONS:
InMessage ? FromState -> ToState : Category3 ? moving4 ->
moving3
InMessage ? FromState -> ToState : Category4 ? moving3 ->
moving4
InMessage ? FromState -> ToState : Category3 ?
waitForCategory -> moving3
InMessage ? FromState -> ToState : Finished ? moving4 ->
waitForCategory
InMessage ? FromState -> ToState : Category4 ?
waitForCategory -> moving4
InMessage ? FromState -> ToState : Finished ? moving3 ->
waitForCategory

```

Figure 56 : A complex FDDEVS model

Figure 57 shows a screenshot at one stage of the simulation presentation awaiting user inputs for multiple external events and figure 58 shows a screenshot where extra educational information is displayed to the student/observer!

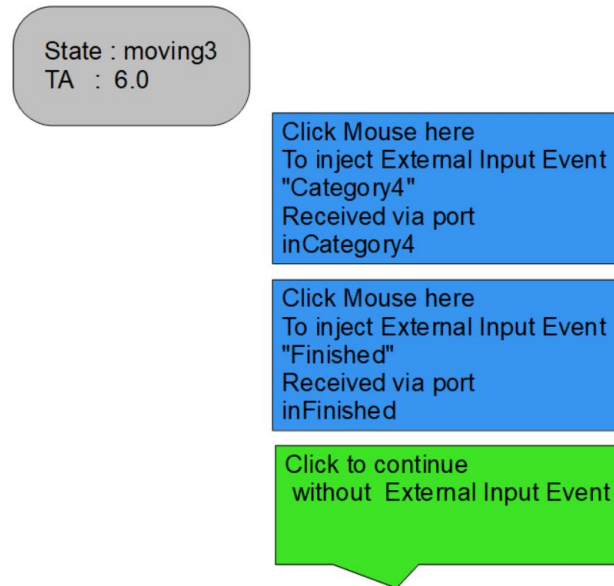


Figure 57 : A screen capture from the presentation of fig 56 model

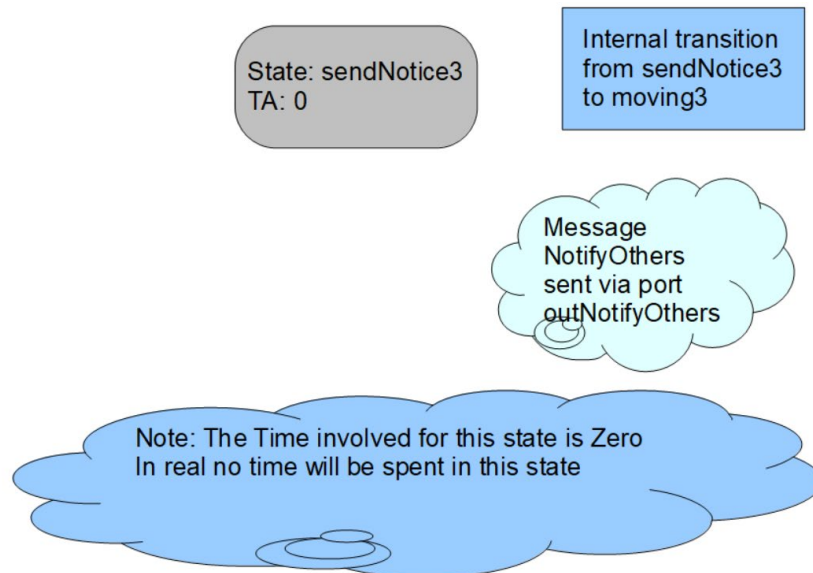


Figure 58 : A screenshot with extra educational information is provided

Figure 59 below demonstrates some additional error-checking features the FDDEVS Animator is incorporated with. The figure shows a message generated during the presentation when the user forgot to define a TA time for the respective model.

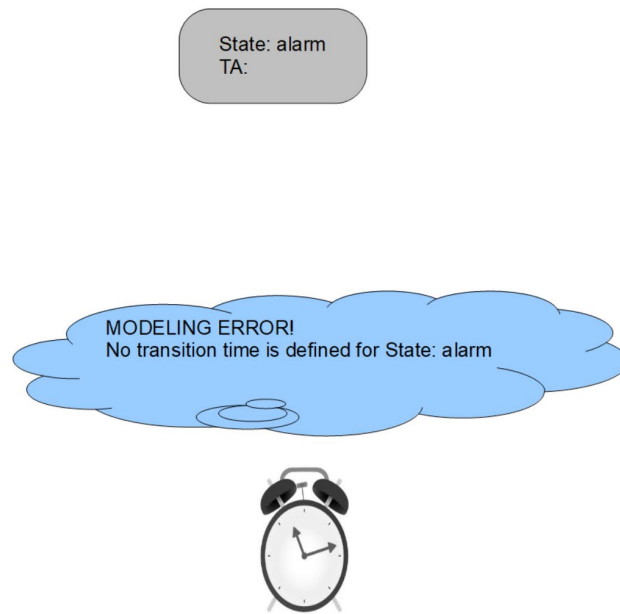


Figure 59 : Error generated when no TA is defined

6.4.2 Caveats

Below are a few caveats of the FFDEVS Animators that are either as designed or identified after some testing.

The auto-mouse-click function depicting internal transitions will only work when both the FDDEVS Animator and the auto-executed Open-Office presentation are running. The feature will not work if you just run the generated .odp or .ppt files individually/separately.

As previously mentioned, the additional static/ dynamic objects that are added to certain model presentations by enabling “Enable Animation” will not be available for all models.

As also initially mentioned, since the software has not fully completed the development stage, there could be unidentified errors and hence further testing needs to be done.

6.4.3 Challenges Faced

The first challenge was to identify what format the model animations would be generated in. Would it best to write software that can produce the simulation animations maybe within an integrated window of the FDDEVS Animator itself, or would be better to interface the animations to some non-proprietary software (to keep within the goals of the overall project)? After some thought, it was decided best to have the flexibility for the animations to be written to some file that can be later reproduced/executed without the need for the FDDEVS animator.

The main reason a presentation slide format was chosen for the purpose was due to the notion that “states” of an FDDEVS model would naturally follow the passage of time along with the option for some event to occur at a known discrete unit of time which can correspondingly be

mapped to a series of slides representing each time unit and then any event could be displayed as contents of a slide. Open Office was the obvious choice due to it being free and non-proprietary.

Needless to say, so much planning and visionary logic went into the design of this software. At the initial stages of software development, sadly the Open-Office API documentation was at a very rudimentary stage, and hence a lot of author-motivated self-research, effort, and processes involving trial and error had to be gone through before a working model was created.

6.5 Utilizing the Tool for Effective Pedagogy/Learning and Future Work

Presented below are some thoughts and ideas which could be adopted to make the FDDEVS Animator's usage more productive in DEVS/FDDEVS M&S pedagogy and learning.

As a more obvious fact, FDDEVS Animator's built-in basic error disclosure features can be used by students to self-evaluate if the model he/she wrote is lacking important modeling information.

An educator can provide XML model files to students with strategically missing or important modeling information so that a novice student can run the model through the various FDDEVS Animator features to identify the issue/s. The educator can also strategically use the model comparison features of this software to demonstrate subtle differences between models or to highlight modeling concepts. For example, state names are the same but TAs are different, or

all internal transitions match but timing or external transitions are different. It is wide open for the educator to identify, and make use of the features of this software, to write exercises to test and improve student comprehension and learning on DEVS/FDEVES M&S.

CHAPTER 7

EXPERIMENTAL DESIGN, STUDY RESULTS, AND ANALYTICAL DISCUSSION

7.1 The Study Background

This chapter will cover the goals, prerequisites, methodology, content, analysis, and results of a study that was conducted to assess the basic efficacy of the new pedagogical approach presented in chapters 3-5. The final section discusses the challenges faced before and during the study execution.

7.1.1 Study Goals

The main objective to undertake this study was to quantify the basic effectiveness of the innovative pedagogical approach vs the existing traditional approach. The study was designed with the notion some participants may be minimally prepared, and hence the scope would be limited to allow for a basic survey/evaluation which could be conducted within a limited time-frame.

7.1.2 Study Prerequisites

To carry out this study within a recognized and formal framework, seeking the university's Institutional Review Board (IRB) approval was needed. The Institutional Review Board (IRB) is an administrative body of the institution it is affiliated with to protect the rights and welfare of human research subjects recruited to participate in a research activity conducted.

After due diligence, an IRB approval for the project was obtained and additional info on the process and challenges faced are discussed under section 7.5.

7.2 Study Methodology

7.2.1 Participant Eligibility

Participant eligibility for the study was straightforward. It was open to any consenting undergraduate or graduate student from the Department of Electrical and Computer Engineering at the University of Arizona.

7.2.2 Participant Recruitment

Potential participants were emailed by ECE Faculty with details of the study. The participant then would reply to the Informed Consent to volunteer for the study. A sample of the recruitment email sent along with the Informed Consent is included in the Appendix.

7.2.3 Participant Allocation

Participants were randomly assigned to one of two groups. Group 1 was provided with a PDF on specific topics on traditional learning material from a leading textbook. Group 2 was provided a new approach to teaching the same topics. Both were asked to complete a series of multiple-choice questions assessing their general knowledge of the topics. Participants were also asked for feedback (e.g., perceived efficacy of the material) and their background (e.g., degree status). All collected data were anonymized and no identifiable information was collected.

7.2.4 Study Administration

The study was designed to be conducted online, and hence the participants could choose a location and time convenient to them. The study material was accessible as a google form which the student needed to read, answer and complete within the expected time guideline. Learning materials were posted on a website which was accessed through a link in the google form. A participant could go over the time limit if he/she chooses to and would report the estimated time taken to complete the study as an answer to a question at the end. For added accuracy, a third-party online tool was incorporated into the google form to keep track of the time engaged and a unique invitation link was sent to each volunteer participant to access the study. For data collection and reporting purposes, the participant response time was noted but the time stamp tool was kept in case there was a big disparity in the time reported.

7.2.5 Study Sample Size and Expected Participant Time Commitment

The study count was kept to 10 participants, 5 each for the 2 groups, A and B. A larger sample size was not necessary for the objective to be achieved. Participants were expected to commit up to 75 minutes to the study. Challenges faced when recruiting participants and conducting this study are further discussed in section 7.5.

7.3 Assessment Material

The study content was formulated to facilitate the data collection of 3 pre-determined objectives. Namely, to determine the participant's basic academic background, general knowledge assessment of the studied topics, and feedback.

7.3.1 Study Introduction and Participant Academic Background

7.3.1.1 Study Group A - Following the Traditional Approach

Figure 60 below shows the “welcome screen” of the google form for a Group A participant that includes the study background and goals.

Assessing effectiveness of a new pedagogical methodology for active learning of DEVS concepts

PLEASE READ BEFORE PROCEEDING!

(not shared) [Switch account](#)

Background :
 DEVS (Discrete Event System Specification) is a formalism that was introduced in the mid 1970s for modeling and analysis of discrete event systems. In present day, it serves as a robust and popular engine for simulation-based design in areas such as manufacturing, transportation, communication, military etc. DEVS, and Modeling and Simulation in general are multi-disciplinary in nature. Anyone interested in learning DEVS concepts had 2 options : 1) Take a semester long course usually offered at Graduate level at some Universities, 2) Self-study with the limited available resources.

Goal of the Study :
 The goal of this study/survey is to analyze the effectiveness of an innovative Pedagogical approach of disseminating DEVS basics/skills compared to the Traditional learning methods. You are in GROUP A which is following the TRADITIONAL approach

[Next](#) Page 1 of 4 [Clear form](#)

Figure 60 : Study Group A introduction and goal specification

Figures 61 and 62, show the 3 questions asked about a participant's academic background

Student Background

Please choose the relevant answer

Question 1) : Are you *

An Undergraduate

Masters Student

Doctoral Student

Other

If other please specify

Your answer

Figure 61 : Question 1 on participant's academic background

Question 2) : What is your familiarity/skill level in Object Orientation /Java Programming ? *

None

Minimal

Proficient

Superior

Question 3) : Have you taken any prior Simulation Course ? *

No

Yes at Undergraduate Level

Yes at Graduate Level

Yes but not at University level

If Yes, please name the Course Name and Institution

Your answer _____

[Back](#) [Next](#) Page 2 of 4 [Clear form](#)



Figure 62 : Questions 2-3 on participant's academic background

7.3.1.2 Study Group B - Following the New Pedagogical Approach

Figure 63 below shows the “welcome screen” that includes the study background and goals for a Group B participant.

Assessing effectiveness of a new pedagogical methodology for active learning of DEVS concepts

PLEASE READ BEFORE PROCEEDING!


[REDACTED] (not shared) [Switch account](#)


Untitled Title

Untitled Title

Background :
 DEVS (Discrete Event System Specification) is a formalism that was introduced in the mid 1970s for modeling and analysis of discrete event systems. In present day, it serves as a robust and popular engine for simulation-based design in areas such as manufacturing, transportation, communication, military etc. DEVS, and Modeling and Simulation in general are multi-disciplinary in nature. Anyone interested in learning DEVS concepts had 2 options : 1) Take a semester long course usually offered at Graduate level at some Universities, 2) Self-study with the limited available resources.

Goal of the Study :
 The goal of this study/survey is to analyze the effectiveness of an innovative Pedagogical approach of disseminating DEVS basics/skills compared to the Traditional learning methods. You are in GROUP B which is following the NEW Innovative learning approach.

Figure 63 : Group B Study introduction and goal specification

It should be noted that Group B participant background questions remain identical to Group A questions as shown in Figures 61 & 62.

7.3.2 Participant Knowledge and Skill Assessment

As previously mentioned, Study Group A was following the traditional learning approach while Group B was following the new pedagogical approach. Hence the learning material provided for the 2 groups would be different. However, the set of knowledge and skill assessment questions posed to both groups remained identical to make a fair assessment after data analysis.

7.3.2.1 Study Group A

Figure 64 below shows the “guidance screen” presented to Group A participants that provided a link to access relevant material from the “Theory of Modeling and Simulation” [3]. The material provided from the book was carefully selected and scanned to optimize the reading time and allow the participant to answer the posed questions.

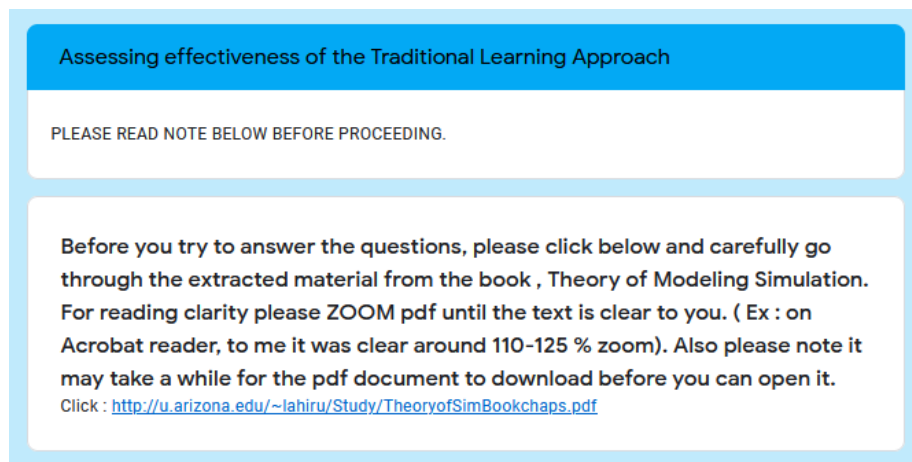


Figure 64 : Scanned introductory material from the Theory of Simulation

7.3.2.2 Study Group B

Figure 65 below shows the guidance screen presented to Group B participants to access relevant learning material from the new pedagogical approach.

Assessing effectiveness of the New Learning Approach

PLEASE READ BEFORE

Before you try to answer the questions, please click below and carefully go through the learning material in the 2 Presentations given at the links below. Please HAVE ADOBE FLASH INSTALLED and ENABLED in your web browser.

IT'S EXTREMELY IMPORTANT to Click SLOWLY, One Click at a Time . Each mouse click maybe a sequential part of an animation on each slide which you don't want to miss. (Within the presentation you can navigate the interface to go back and forth to previous slides and start again if you need).

Please go through

1) Presentation 1 : Slides 4 -17 ONLY. Click link : <http://u.arizona.edu/~lahiru /1Intro-DEVSpert1/index.html>

2) Presentation 2 : Slides 4 -15 ONLY : Click link : <http://u.arizona.edu/~lahiru /2Intro-DEVSFDDEVSpert2/index.html>

Figure 65 : Introductory material from the new pedagogical approach

7.3.2.3 Knowledge Assessment Questions 1 -14 for Both Group A and B

Figure 66 below shows knowledge assessment questions 1-3 and figures 67 to 72 show the rest.

Question 1) : Which below Real World Entity (Source System) can be considered a good candidate for Discrete event modeling and simulation ? *

- A Toothpick
- An Eye Glass
- A Ping Pong Match
- A Chess Piece

Question 2): Which best describes a Model ? *

- a structure to produce information to represent a real world system
- a structure to produce behavior claimed to represent a real world system
- conditions under which a real world system is observed
- conditions under which specific instructions are observed

Question 3): A Simulator *

- executes a Real world system to generate its structure
- executes an Experimental Frame to generate its structure
- executes an Experimental Frame to generate its behavior
- executes a Model to generate its behavior

Figure 66 : Knowledge assessment questions 1-3

Question 4): An Experimental Frame defines *

- conditions under which a Simulator is observed
- conditions under which a System is observed
- a structure to produce behavior claimed to represent a Real world system
- a structure to produce instructions to represent a Virtual system

Question 5) : In Modeling and Simulation context, a logical "State" can represent *

- part of a structure of a Model
- part of a structure of an Experimental Frame
- part of a structure to produce Instructions
- part of a structure to produce Simulation

Figure 67 : Knowledge assessment questions 4-5.

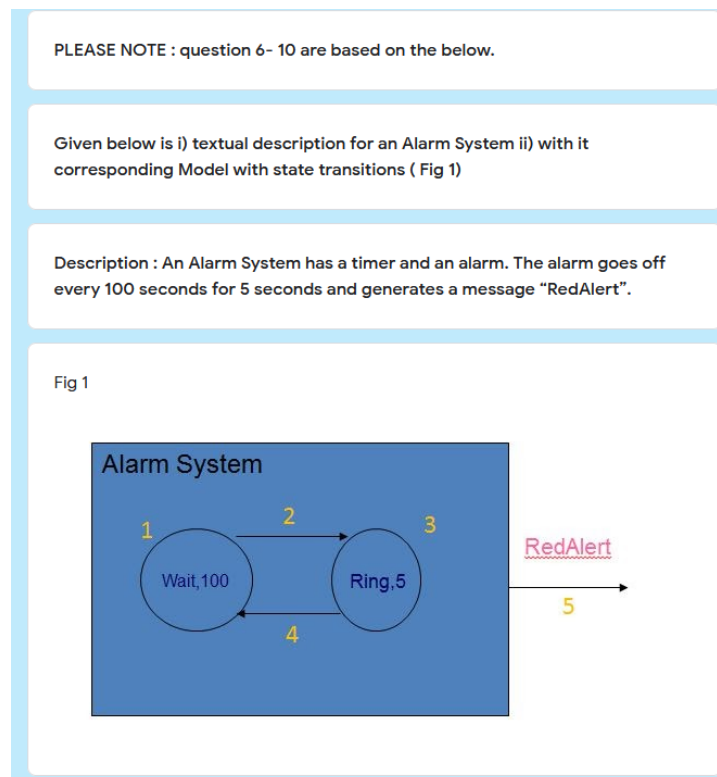


Figure 68 : Reference screen for questions 6-10

Question 6) : In the Model above, # 4 (arrow) is *

- an Internal Transition
- an External Transition
- an Output
- an Input

Question 7) : In the Model above "RedAlert" is *

- an Input
- an Output
- a State
- an External Transition

Question 8) : Which below is a correct DEVS Mapping on the above Model *

- $S = \{ \text{Wait, Ring} \}, ta(\text{Wait}) = 5, Y = \{ \text{RedAlert} \}$
- $S = \{ \text{Wait} \}, ta(\text{Wait}) = 100, X = \{ \text{RedAlert} \}$
- $S = \{ \text{Wait, Ring} \}, ta(\text{Wait}) = 100, Y = \{ \text{RedAlert} \}$
- $S = \{ \text{Ring} \}, ta(\text{Wait}) = 5, X = \{ \text{RedAlert} \}$

Figure 69 : Knowledge assessment questions 6-8

Question 9) : In the above model there are No *

- Outputs and External Transitions
- Outputs and Internal Transitions
- Inputs and External Transitions
- Inputs and Internal Transitions

Question 10) : In Classical DEVS there are ____ Tuples *

- 8
- 6
- 5
- 7

Question 11) : Pick the correct set of answers for the blanks. After waiting for _____, only _____ can generate an Output (a message) . *

- ta , Internal Transitions
- ta , External Transitions
- input , Internal Transitions
- input , External Transitions

Figure 70 : Knowledge assessment questions 9-11

For questions 12 – 14 , please refer to the black box diagram (Fig 2) below and choose the best answer

Fig 2

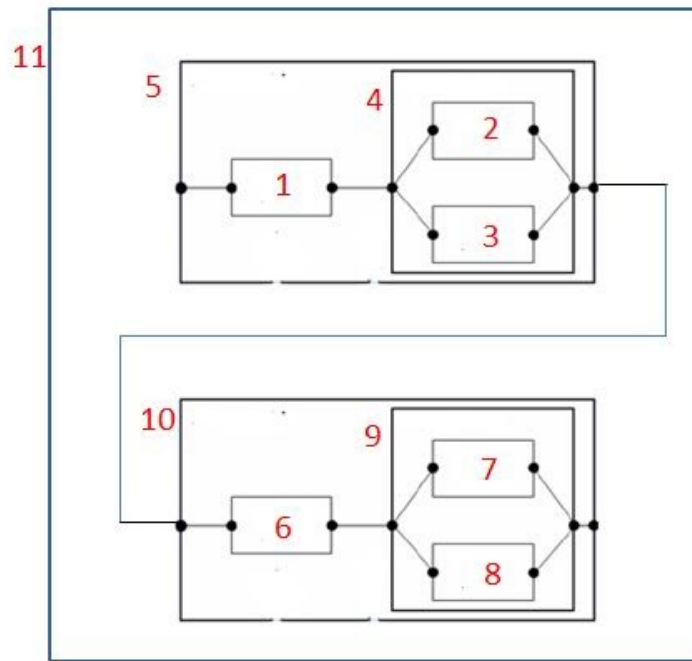


Figure 71 : Reference screen for questions 12-14

Question 12) : In the above diagram, Atomic models are defined by black boxes numbered *

- 1, 5, 6, 10
- 5, 10, 11
- 1, 2, 3
- 1, 4, 5

Question 13) : In the above diagram, Coupled models are defined by black boxes numbered *

- 6, 7, 8
- 6, 9, 10
- 2, 3, 5
- 4, 9

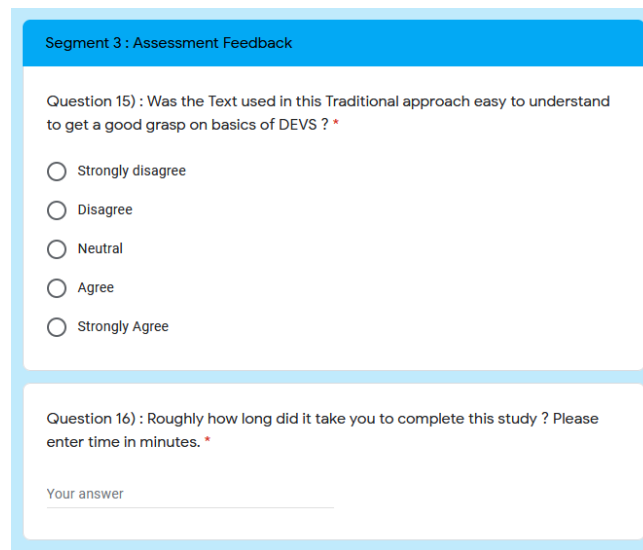
Question 14) : Refer to Fig 2 and choose the best answer *

- Black box # 4 is Coupled model and 5 is Hierarchical Coupled Model
- Black box # 4 is Coupled model and 5 is an Atomic Model
- Black box # 4 is a Hierarchical Coupled Model and 5 is an Atomic Model
- Black box # 4 is a Hierarchical coupled Model and 5 is an Coupled Model

Figure 72 : Knowledge assessment questions 12-14

7.3.3 Participant Feedback

The feedback questions posed to Group A and Group B essentially remained the same except for slight wording differences relative to the learning method followed. Figures 73 and 74 show the feedback question posed to Group A and B respectively.



Segment 3 : Assessment Feedback

Question 15) : Was the Text used in this Traditional approach easy to understand to get a good grasp on basics of DEVS ? *

Strongly disagree

Disagree

Neutral

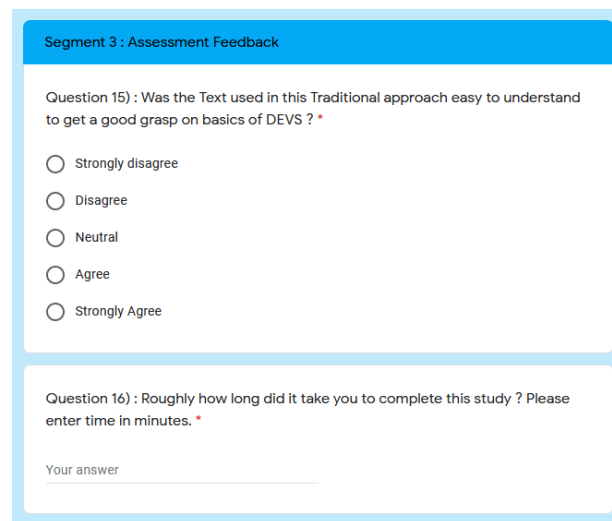
Agree

Strongly Agree

Question 16) : Roughly how long did it take you to complete this study ? Please enter time in minutes. *

Your answer _____

Figure 73 : Group A participant feedback questions



Segment 3 : Assessment Feedback

Question 15) : Was the Text used in this Traditional approach easy to understand to get a good grasp on basics of DEVS ? *

Strongly disagree

Disagree

Neutral

Agree

Strongly Agree

Question 16) : Roughly how long did it take you to complete this study ? Please enter time in minutes. *

Your answer _____

Figure 74 : Group B participant feedback questions

7.4 Assessment Results and Analytical Discussion

In this section, study results are presented in graphical format (based on participant responses) along with some analytical discussions. In section 7.4.4, a conclusion is arrived at after the objective analysis of data.

7.4.1 Participant Background Assessment

7.4.1.1 Graduate vs Undergrad Participant Composition

Group A participants consisted of 4 graduate (doctoral) students and 1 undergraduate, while Group B participants consisted of 4 graduate (3 doctoral and 1 masters) students and 1 undergrad. Figures 75 and 76 show the stats in pie chart format. Hence it can be seen the academic background of the 2 groups is essentially at the same level.

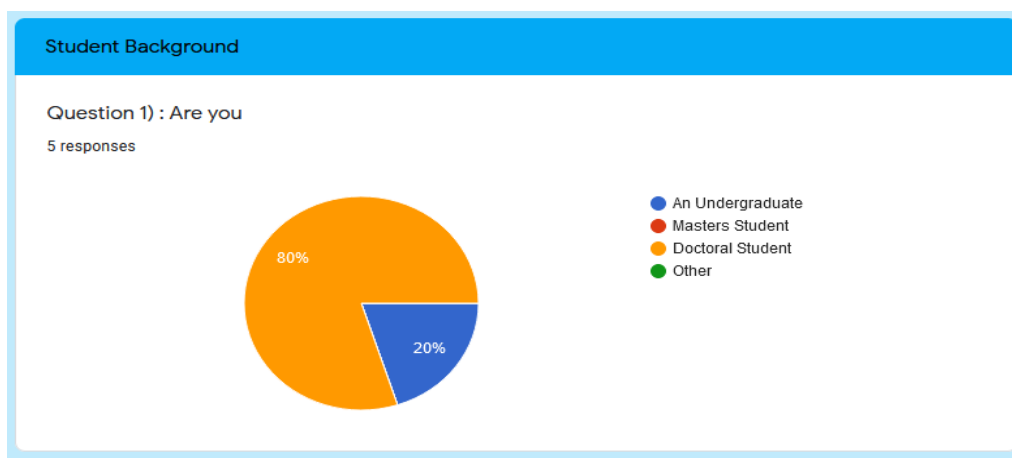


Figure 75 : Group A academic standing

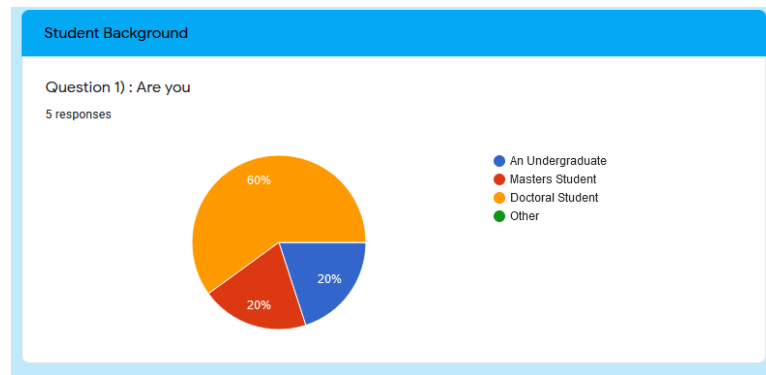


Figure 76 : Group B academic standing

7.4.1.2 Participant Familiarity with Object-Oriented and Java Programming

As can be seen in Figure 77, 60% of Group A stated they possess a “proficient level” of programming skills while 40% stated they had minimal. On the other hand as seen in figure 78, 80% of Group B stated they possess a “proficient level” of programming skills while 20% stated they had minimal. The disparity between the 2 groups is not significant.

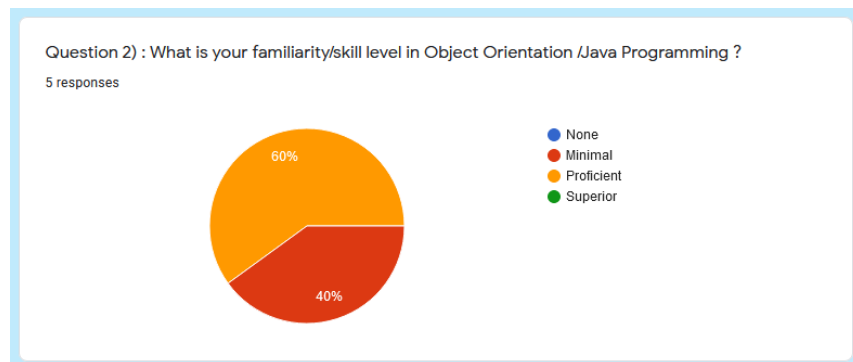


Figure 77 : Group A computer programming background

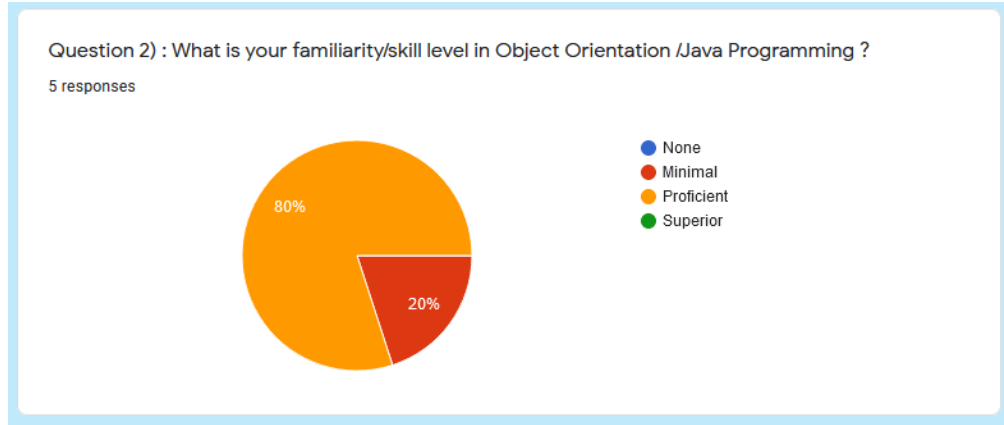


Figure 78 : Group B computer programming background

7.4.1.3 Participant Exposure to a Prior Simulation Course

Figure 79, shows that 60% of Group A had no prior exposure to any simulation course, while 40% had taken a graduate-level simulation course. On the other hand, Figure 80 shows that 80% of Group B had no prior exposure and only 20% had taken a prior simulation course.

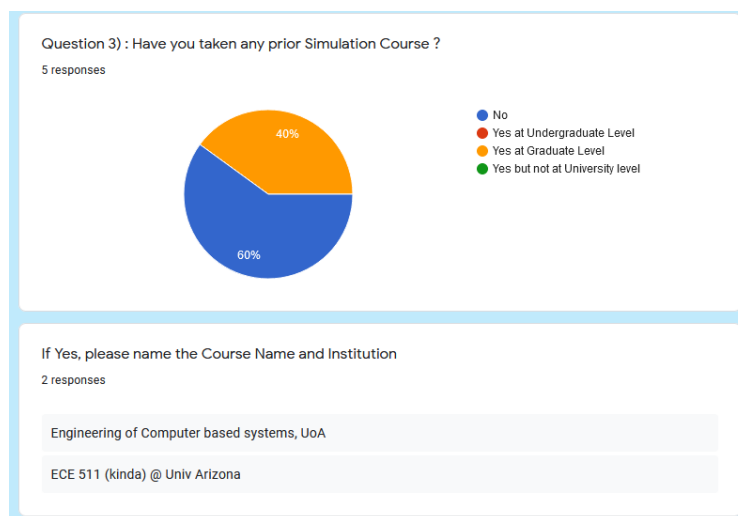


Figure 79 : Group A exposure to a prior Simulation course

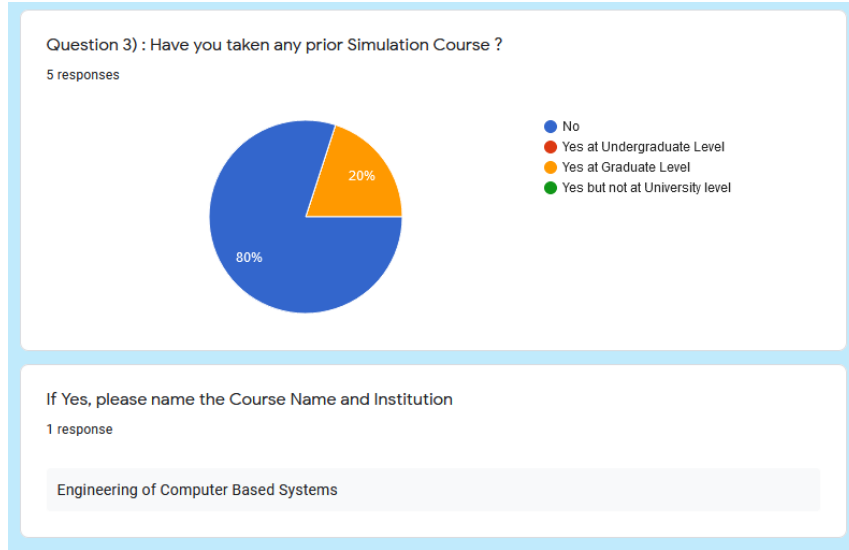


Figure 80 : Group B exposure to a prior Simulation course

7.4.2 Participant Knowledge Assessment

This section evaluates the basic knowledge/skill testing of participants based on the accuracy of the answers provided to the 14 multiple-choice questions described in subsection 7.3.2.3.

Tables 11 and 12 show respectively how the 10 participants of Group A and B answered the 14 questions. As shown in table 13, “C” corresponds to a Correct answer while “IC” corresponds to an “In Correct” answer. Analysis of results will be provided in section 7.4.4.

Study Group A	Ques 1	Ques 2	Ques 3	Ques 4	Ques 5	Ques 6	Ques 7	Ques 8	Ques 9	Ques 10	Ques 11	Ques 12	Ques 13	Ques 14
Student 1	C	IC	C	C	C	C	C	C	C	C	C	C	C	C
Student 2	IC	IC	C	C	C	C	C	C	C	IC	IC	IC	C	C
Student 3	IC	C	C	C	IC	IC	C	IC	C	IC	IC	IC	C	IC
Student 4	C	C	C	C	C	C	C	C	C	C	C	IC	IC	IC
Student 5	C	C	C	C	IC	IC	C	IC	C	C	IC	IC	IC	C

Table 11 : Group A participant responses

Study Goup B	Ques 1	Ques 2	Ques 3	Ques 4	Ques 5	Ques 6	Ques 7	Ques 8	Ques 9	Ques 10	Ques 11	Ques 12	Ques 13	Ques 14
Student 1	C	C	C	C	C	C	C	C	C	C	C	C	C	C
Student 2	C	C	C	IC	C	C	C	IC	C	C	IC	C	C	C
Student 3	C	C	C	C	C	C	C	C	C	C	C	C	C	C
Student 4	C	C	C	C	IC	C	C	C	C	C	C	C	C	C
Student 5	C	C	C	C	C	C	C	C	C	C	IC	C	C	C

Table 12 : Group B participant responses

Legend	
C	CORRECT
IC	INCORRECT

Table 13 : Legend for Tables 11 and 12

7.4.3 Participant Feedback

2 feedback questions were posed to the participants of the 2 groups. Figures 81 and 82 respectively show the response of Group A and B participants to the query of whether their respective learning approach was easy to grasp.

As we can see from figure 81, 80% of the participants from Group A said they disagree that the traditional approach was easy to grasp. 20% said they agree. On the other hand, from figure 82 we see, 40% of the participants from Group B said they strongly agree that the new pedagogical approach was easy to learn from while another 40% said they agree while 20% were neutral.

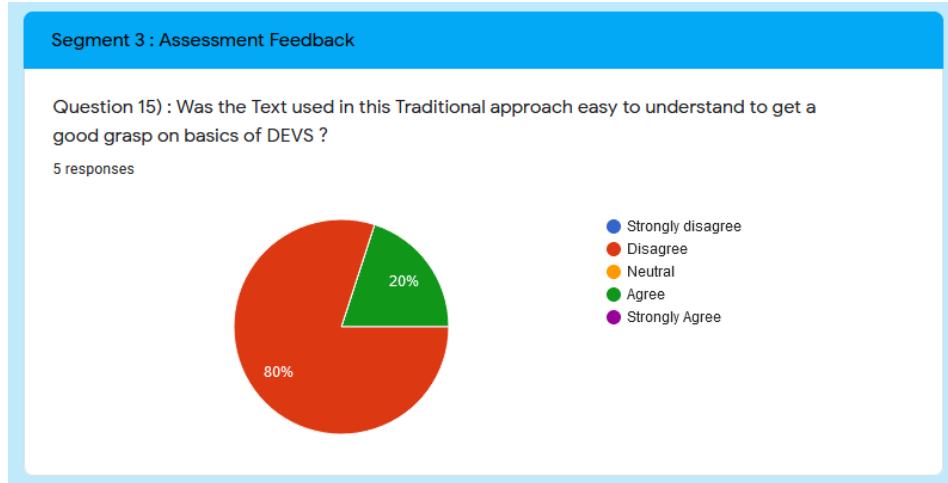


Figure 81 : Group A response to the question “was the traditional approach easy to grasp?”

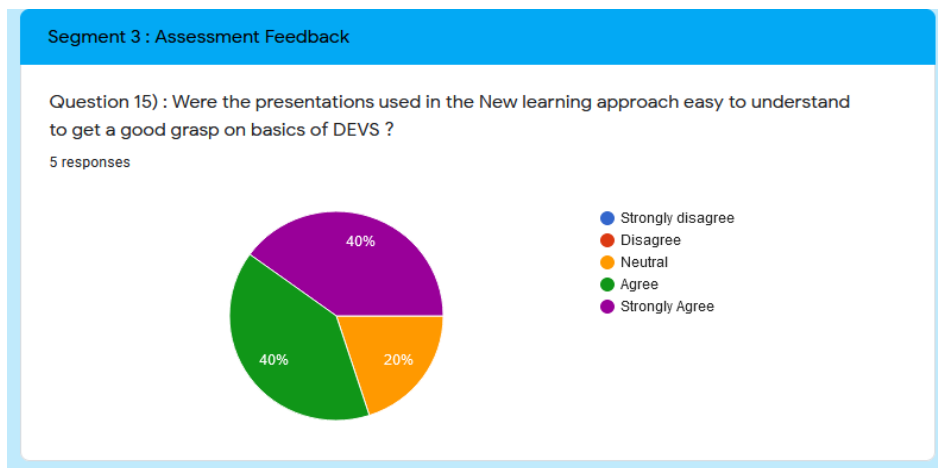


Figure 82 : Group B response to the question “was the new approach easy to grasp?”

The second feedback question was asking the participants to report the time it took them to complete the study. Such can be seen in subsection 7.3.3. These numbers were compared with the previously mentioned timing widget reporting to see whether there was any big disparity which fortunately did not happen. The timing report is included in the next section (7.4.4).

7.4.4 Study Results Analysis and Conclusion

To assess the effectiveness of one pedagogical methodology over the other, 2 basic but reflective yardsticks were determined. 1) The participant's ability to correctly answer the questions posed in the 14 multiple-choice knowledge assessment segment. 2) The time to complete the study.

Each row of tables 14 and 15 shows respectively each participant's score sheet and the reported time it took them to complete the study. Table 14 is for Group A and table 15 is for Group B.

	Correct	Incorrect	Time to Complete- mins
	13	1	79
	9	5	60
	6	8	30
	11	3	70
	8	6	30
Average	9.4	4.6	53.8

Table 14 : Group A participant scoring table with study completion times

	Correct	Incorrect	Time to Complete- mins
	14	0	45
	11	3	20
	14	0	45
	13	1	45
	13	1	40
Average	13	1	39

Table 15 : Group B participant scoring table with study completion times

From table 14, we can gather the average score of correct answers for a Group A participant is 9.4 out of 14. That is roughly 71.4% accuracy. From table 15 we see the number is 13 out of 14 for Group B. Percentage accuracy rate for Group B is 92.8%.

From table 14, we can see the average time a participant in Group A took to complete the study was 53.8 minutes. In table 15, we see the average number is 39 minutes.

In conclusion, we can deduce that the Group B participants who were following the new pedagogical approach performed at a much higher rate than Group A (who was following the traditional pedagogical approach) in both the knowledge testing segment and the time it took them to complete the study. The feedback segment discussed in subsection 7.4.3, provided good insight that the new approach was easier to grasp.

7.5 Challenges Faced Concerning the Study

Initially, it was under the opinion of the author and dissertation advisor that this project/study would be exempted under IRB minimal risk definitions as it was pedagogical and had no direct or in-person contact to carry out the study. Thus, an application for an exemption was submitted to IRB to expedite and simplify the process. However, after several months of communication, IRB determined the project will fall under their “Human Subjects Research” category and a full IRB application for approval was needed.

Before applying for the full application process, the author had to take and pass 2 courses under the Collaborative Institutional Training Initiative (CITI), namely, a basic course for “Human Research Social & Behavioral Research Investigators” and a mandatory basic course on “Native American Research” along with other things as part pre-requisite requirements.

The full IRB application process was rigorous and strenuous, to say the least, as it involved multi-document submissions, and multiple updates/ revisions over a several-month period before the final approval was given.

The next challenge was even more daunting to find suitable volunteers who would be willing to undertake the study and more importantly focus to complete it under the guidelines. It was a multi-month task to find suitable study volunteer participants to complete this project successfully especially as no monetary incentive was provided to them as a motivation to participate. On the positive side, this in essence is a blessing in disguise as the study results presented would be completely impartial and unbiased due to the non-incentive-based volunteer participation.

CHAPTER 8

CONCLUSIONS

Modeling and Simulation is a field that has so much further potential to revolutionize the way engineering and science will be conducted in the 21st century. Though much literature and field-specific practical applications can be found to support this premise, in this dissertation, we attempted to address a much lesser-discussed subtopic of M&S: the pedagogical side at the grass-root level. We recognize the broad and multi-disciplinary nature of M&S and want to highlight the imminent need to introduce the discipline to minimally prepared students to get more people interested in the field without the usual academic complexities or costs of learning.

It is said that Little drops of Water make the Mighty Ocean! Hence our humble effort here to introduce the new software and the pertaining pedagogical approach should not be constituted as an infallible optimal attempt but as a unique effort to fill a needed void in teaching the basic M&S to novices in an innovative way.

Though our approach/software will have its merits, and the experimental study presented in Chapter 7 will prove as a form of proof for the basic efficacy of such, there is much more room for it to be improved upon and perfected as discussed in chapter 9 under “Future Work”.

The author also wants to reiterate especially that the feature-rich FDDEVS Animator software is still in its elementary stages though introduced as a supplementary tool and more testing and documentation needs to be done.

As a final note, the author hopes that the readers will recognize the potential of this research and use such as a motivation for further research into the lesser-charted territory of the pedagogical needs/aspects of M&S.

CHAPTER 9

CHALLENGES FACED AND FUTURE WORK

9.1 Challenges Faced During this Research/Study

A few key academic/technical obstacles faced are mentioned below.

- i) As some initial software coding was conducted/developed years ago, it was a task over the years to check/update/port the software to be compatible with the current environments.

- ii) The key set of pedagogical tutorials that were developed was distributed in multiple formats for effective dissemination as mentioned in sub-section 5.3.2. For example, PowerPoint presentations with unique/strategic timing/animations were found a proven key delivery format to better impart DEVS & M&S concepts to students. In the past, to effectively post such presentations on the web, the author found a way to convert PPT slides to a web-friendly SWF (Flash) format which would essentially preserve 95% of timings /animations. This proved to be the best option at that time. But since December 2020, Flash is discontinued and no browser would support such. Hence the old web-based method is obsolete now. Also, the standalone power-point presentation player is no more as Microsoft has discontinued such (it would beat the purpose of the dissertation goals to request students to install a costly proprietary software, MS office on a student machine). Hence the author had to spend

time exploring different ways including converting slides to HTML5, splitting a single slide with multiple animations into multiple PDF slides for "the cartoon effect" etc, but found all these methods were complex and not very effective. Hence, in the end, the most simplified and effective option was chosen.

iii) When developing the FDDEVS Animator software, a key goal was to generate/distribute any basic animations to execute in a non-propriety presentation software, and hence Open-Office was picked to preserve a dissertation goal. But unfortunately, at the time of software development, the Open-Office API documentation was at a very rudimentary stage, and hence a lot of research, trial and error, and effort had to be put to create a working model.

iv) As elaborated in sub-section 6.5, the IRB study approval process was time-consuming and rigorous. It involved obtaining prior certifications, multi-document submissions, and multiple updates/ revisions over a several-month period before the final approval was given. Also, as previously mentioned, finding suitable volunteers who would be willing to undertake the study and more importantly focus to complete it under the guidelines was a big challenge as this study was done on a non-incentive basis to the participants. However, this could be a blessing in disguise as it would result in providing completely unbiased inputs/results.

9.2 Future Work

It should be highlighted again that our research effort here should not be constituted as an infallible optimal attempt but a unique attempt to fill a needed void in M&S to teach the basics to novices using an innovative way.

As discussed in sub-section 3.6, “The research plan and methodology”, this dissertation primarily focused on covering tasks 1 to 3. Hence the tasks below are wide open for future work.

- Introduce material to online and live engineering classroom instruction
- Formulate evaluation methods for online and live classroom environments
- Disseminate results and educate the workforce

The author recognizes that the methodology and framework introduced here have much more room for further expansion to make it even more useful and user-friendly.

The collection of pedagogical tutorials developed as part of this research effort can be improved upon to make them even more student-friendly and effective. Such a task will take more experimentation by utilizing them in an online or live classroom. Some longer tutorials can be split into 2, or a few intermediate tutorials can be developed as needed to fill in any missing gaps or make the student learning experience more enriching. More hands-on exercises and tasks requiring

active student engagement need to be developed. There is more room to expand upon or extend the features of FDDEVS Builder software while there is significantly more room for more testing, debugging, feature incorporation, and further development of the FDDEVS Animator software.

As another possible future research idea, it would be interesting to explore the possibility of developing an innovative pedagogical approach to cater to persons with impairments drawing further inspiration from trends emerging from the computer games space previously discussed, and the experience drawn from the new pedagogical methodology presented,

As a final note, the author hopes that the readers will recognize the potential of this study so that more future research and development in this area of focus will be encouraged.

APPENDIX A : IRB PROJECT RELATED DOCS

1) The IRB study recruiting email sent to potential participants

Hello,

We are seeking participants for a research study on "Assessing effectiveness of a new pedagogical methodology for active learning of DEVS concepts". DEVS (Discrete Event System Specification) is a formalism for modeling and analysis of discrete event systems. Examples of discrete event systems include obtaining an item at a vending machine or a traffic light system. Learning DEVS imposes challenges, and traditional approaches require a steep learning curve. This research projects seeks to evaluate the effectiveness of a simplified innovative teaching methodology to learn basics of DEVS.

Your participation in this research study is voluntary. For further information, please see the attached informed consent form.

If you are interested in participating in this study, please contact Lahiru Ariyananda at lahiru@ece.arizona.edu.

.....

(faculty name here)

|

An Institutional Review Board responsible for human subjects research at The University of Arizona reviewed this research project and found it to be acceptable, according to applicable state and federal regulations and University policies designed to protect the rights and welfare of participants in research.

Una Junta de Revisión Institucional responsable de la investigación con seres humanos en la Universidad de Arizona examinó este proyecto de investigación y encontró que era aceptable, de acuerdo con las disposiciones estatales y federales, así como las políticas universitarias destinadas a proteger los derechos y el bienestar de los participantes en la investigación.

2) Informed Consent document sent to potential participants



Study Title : Assessing effectiveness of a new pedagogical methodology for active learning of DEVS concepts

Principal Investigator : Lahiru Ariyananda

Supervisor : Dr Roman Lysecky

Note to Participants :

You are being requested to participate in a research study. Your participation in this research study is voluntary. This document contain background information about this study and what to expect if you decide to participate. Please consider the information and feel free if necessary to ask questions before making your decision. By agreeing to this email, you will give consent to participate in the study. No signatures will be collected or needed. There are no expected risks to you as a result of participating in this study. You will not benefit directly from participating in this study.

Study/Research Background:

DEVS (Discrete Event System Specification) is a formalism which was introduced in the mid 1970s for modeling and analysis of discrete event systems. Hence DEVS can be used for modeling and analyzing any general system that can be discrete event in nature as opposed to continuous time we experience in life. Some examples of Discrete event systems are the process of obtaining an item at a vending machine or a traffic light system.

However, learning DEVS imposes challenges. Presently, for anyone interested in learning DEVS concepts has 2 options : 1) Take a semester long course usually offered at Graduate level at some Universities, 2) Self-study with the limited available resources including referring to the foremost resource on DEVS titled "Theory of Modeling and Simulation" by B Zeigler et al.

In both cases, a steep learning curve is required with significant time dedication. A student will need as prerequisite a solid mathematical and computer programming background to get a good grasp. Hence this research introduces a simplified innovative teaching methodology/system to learn basics of DEVS and the study helps to assess the effectiveness of such. This study is conducted as part need for the PI PhD dissertation.

Study Procedure :

You as participants will be randomly assigned to one of two groups. Group 1 will be provided with a PDF on traditional learning material from a leading textbook for a specific topic. Group 2 will be provided a new approach for teaching the same topic. Both will be asked to complete a series of multiple choice questions assessing the general knowledge of the topic and perform some task related to the study. Participants will also be asked for feedback (e.g., perceived effectiveness of the material) and their background (e.g., degree status). All collected data will be anonymized and no identifiable information will be collected.

Risks to Participants : No such risks have been identified

Expected Time commitment : 60-90 mins

Confidentiality of information : All collected data will be anonymized and your name will not be used in any report or linked to your answers. Please note though information that you provide in the study will be handled confidentially, there may be exceptional circumstances where this information must be released or shared as required by law or by the need of University of Arizona policies.

For questions, concerns, or complaints about the study you may contact
lahiru@ece.arizona.edu

For questions about your rights as a participant in this study or to discuss other study-related concerns or complaints with someone who is not part of the research team, you may contact the Human Subjects Protection Program at 520-626-6721 or online at <http://rgw.arizona.edu/compliance/human-subjects-protection-program>.

APPENDIX B : TUTORIAL SAMPLES

1. Sample screenshots from the presentation “Introduction to DEVS and FDDEVS”

INTRODUCTION TO DEVS

Lahiru Ariyananda
UNIVERSITY OF ARIZONA

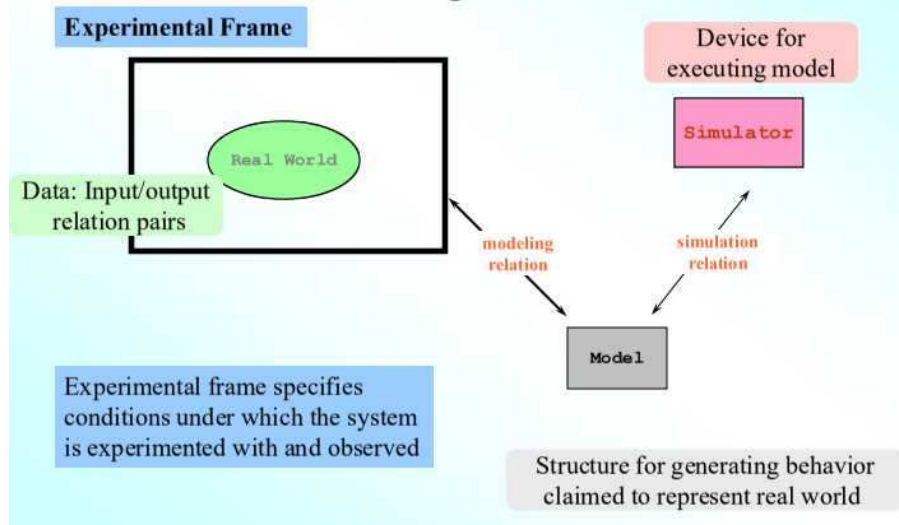
Outline

- Key Entities and Relations in Modeling and Simulation
- Building a model through example
- Gentle intro to DEVS
- DEVS semantics

Learning by Example

- Let us start with a simple example
- **Problem** : In a singles **Tennis match** a **PlayerA** plays a shot and await a return shot from the opponent **PlayerB**. Assume it takes **0.5 seconds** to play a shot. Develop a model to describe **PlayerA** and Simulate it.

Basic Entities and Relations in Modeling and Simulation



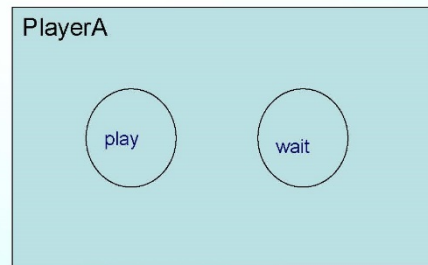
Keys things to identify

- *Real System*
 - a source of data with behavior.
 - Ex : Tennis Match
- *Model*
 - Set of instructions for generating data comparable to that in real system.
 - Ex : What we need to develop
- *Simulator*
 - Exercises the model's instruction to actually generate specified behavior.
 - Ex : Using Computer
- *Experimental frame*
 - determine the condition under which the system will be experimented with and observed
 - Ex: Duration of Tennis Match

Building a Simple Model

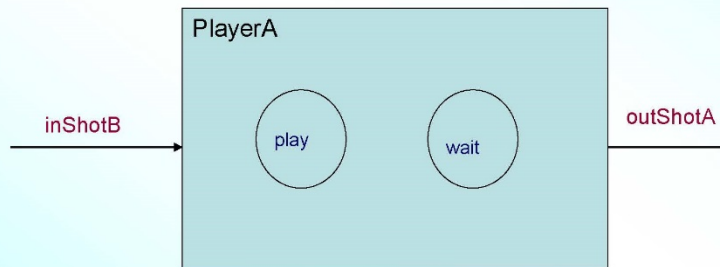
Lets build a simple model for PlayerA.

- We are now looking at **what concerns the behavior of PlayerA only** (from his point of view)
- For the notion of **playing a shot**
 - A logical **State** called **"play"**
 - note at this juncture we are not interested in details of How/What shot he plays but the idea that a shot is played
- For the notion of **waiting to receive a shot**
 - A logical **State** called **"wait"**
 - note we are not interested in PlayerB model behavior right now but the notion that PlayerA is waiting to receive a return shot from PlayerB
- The Player can be either in state "play" or state "wait"



Communication?

- What about communication?
- To send the shot played to PlayerB
 - we can make an output place available to PlayerA
 - Let us call it an **Output port and give it a name: outShotA**
 - we can make a message value to send : ShotA
- To receive a return shot from PlayerB while in state “wait”
 - we can make a input place available to PlayerA.
 - Let us call it an **Input port and give it a name : inShotB**
 - we can make a message value to receive : ShotB
- So to exchange shots to and from PlayerA to PlayerB , we need to define a “**message**”
 - a **message** has two things
 - a **port** to send or receive (output or input)
 - a **value** which is passed through the port from Player A to PlayerB or vice versa

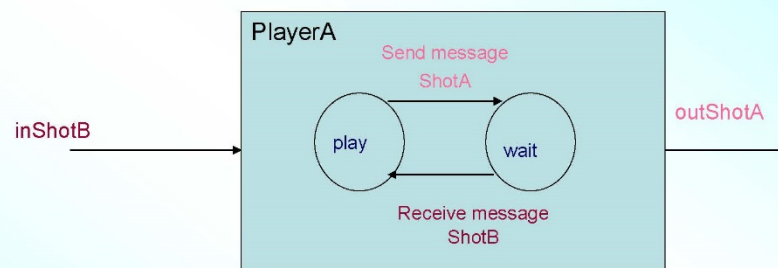


State Transition

- How do we go from state **play** to state **wait**?
- Lets define a **"Transition"** function
 - to automatically go from play to wait state without depending on an external influence
 - we can call it an **"Internal State Transition"**
 - Note before we go to "wait" state we need to communicate an output message to PlayerB that a shot was played
 - We can use the port and message value we defined before

State Transition Cont...

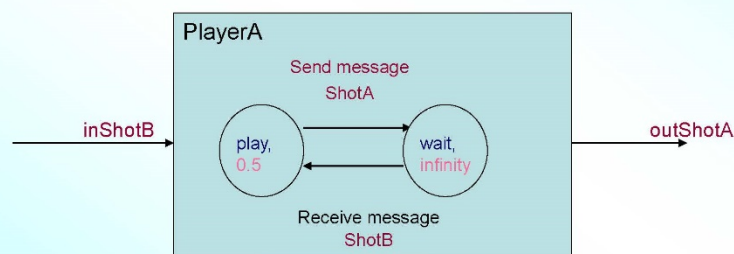
- Question : Under what premise do we go from state **wait** to state **play**?
- Lets define another “Transition” function
 - but this time the transition is dependant on an external influence : a receiving shot from PlayerB received via Input port
 - Hence we can call it an “**External State Transition**” from states Wait to Play



Note : PlayerA uses outShotA port to send to PlayerB's input port and vice versa

Anything Missing?

- Are we missing any information to complete the model??
- How long should you remain in a state before we do a transition?
 - For play state => 0.5 seconds as defined
 - For wait state => as long as it take for PlayerB to return which is **dependant on the received input**. For that we wait for **time infinity** in wait state and transit **only based on the external input**
 - **time infinity (for ever)** can be represented by having wait time of a state set to **infinity**.
 - we need a function to **keep track of time remaining in the current state** till its ready for a transition to the next state.



Note: (play,0.5) and (wait,infinity) corresponds to state and waiting time in that state



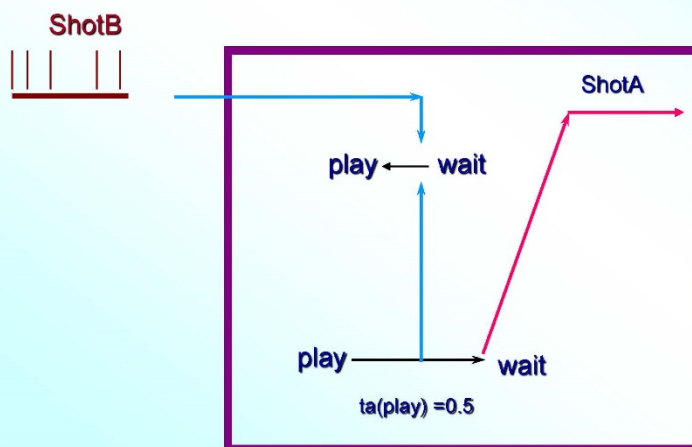
Do we know Discrete Event System Specification already?

- In essence we do now!!!!
- **Classical DEVS** is defined as a 7 tuple

X: set of Inputs
S: set of States.
Y: set of Outputs
 δ_{int} : Internal Transition function.
 δ_{ext} : External Transition function.
 λ : Output function.
 ta : Time advance function.

Theory of modeling and simulation, Zeigler B.P, John Wiley Editor, New York, 1976.

DEVS Semantics in a nutshell



2. Document format tutorial on FDDEVS Builder Environment & Atomic model generation and visualization

1

INTRODUCTION TO FDDEVS BUILDER ENVIRONMENT AND ATOMIC MODEL GENERATION AND VISUALIZATION

This step by step guide will help you to

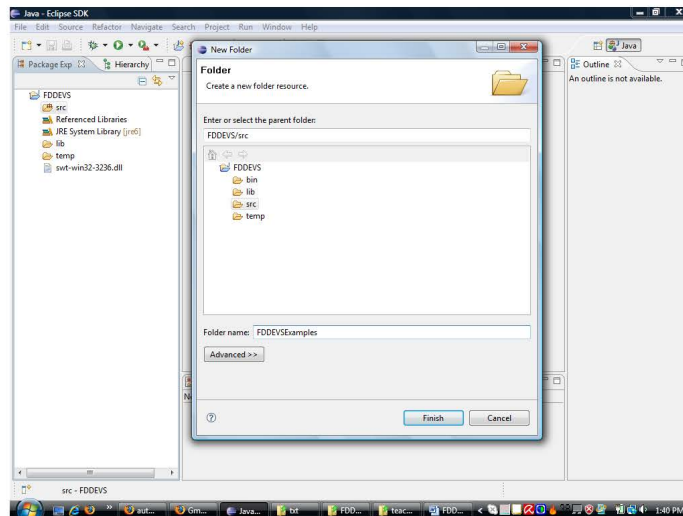
- setup a working FDDEVS Builder Workspace environment
- generate a FDDEVS Atomic model using a pre-defined example

Note : Before you read through this guide, you need to have installed FDDEVS on an Eclipse IDE. If you have not installed FDDEVS, please see the relevant guides “FDDEVS step by step installation guide for Eclipse IDE” or “Concise guide to installing FDDEVS on Eclipse IDE.”

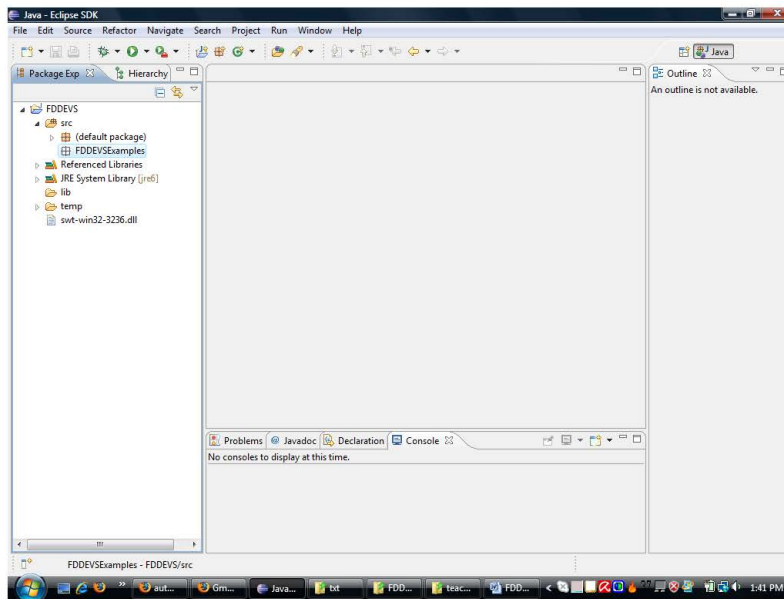
Step 1 : Setup a working examples folder in Eclipse

Let us start by creating a working folder under FDDEVS project for the examples we will be doing.

On Eclipse IDE, under the FDDEVS project ,
right click on the “src” folder
and go to NEW->Folder.
Give “FDDEVSExamples” as the folder name.



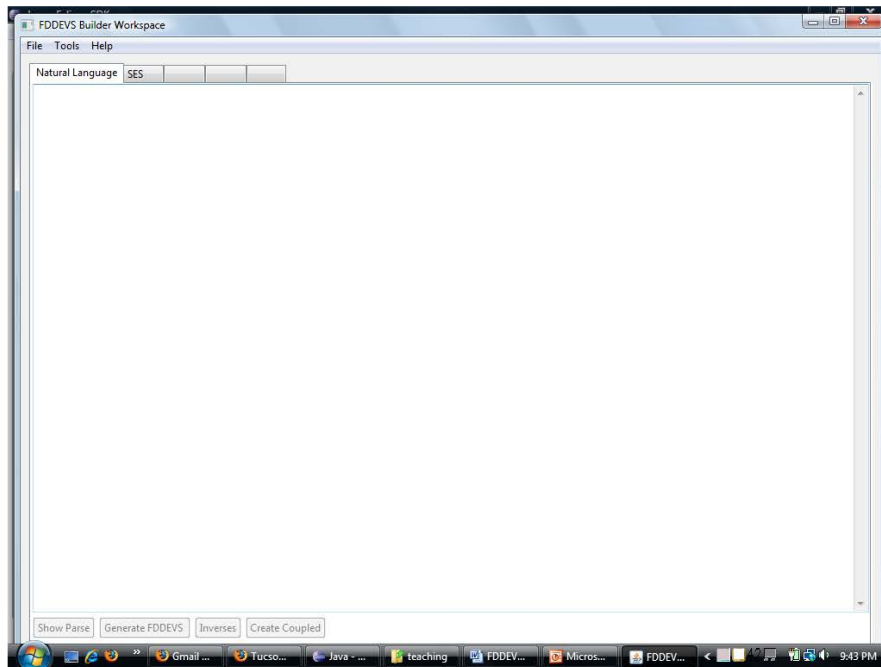
2



Created FDDEVSExamples folder within Eclipse IDE

Step 2: Run FDDEVS Builder GUI from Eclipse

Recall that you need to right click on `MainFDDEVS.java` (found under the `src` folder) and Run As a Java Application.

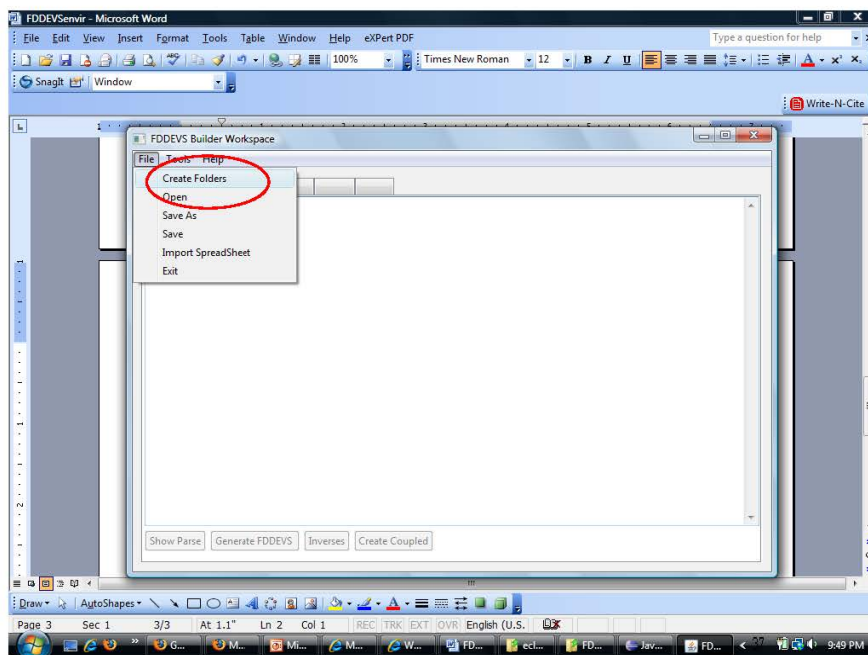


FDDEVS Builder GUI

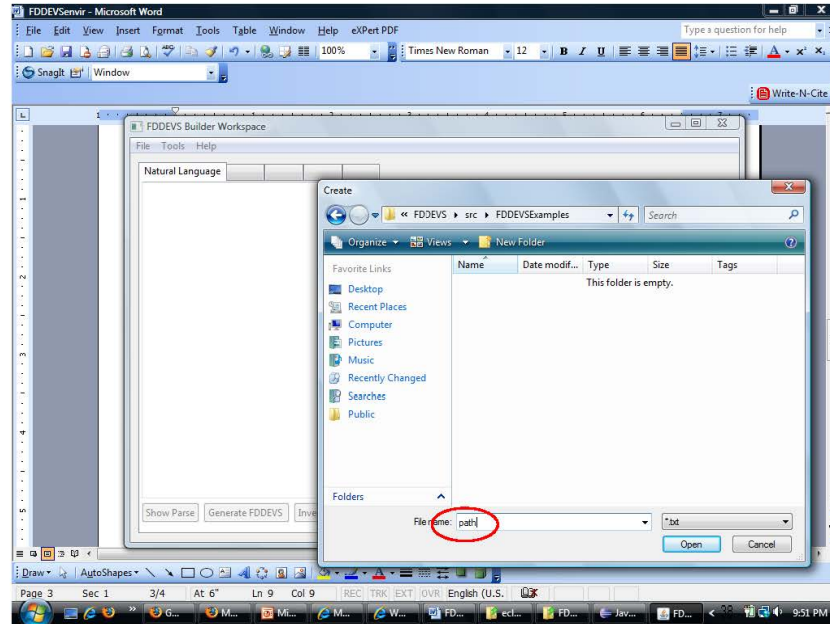
4

Step 3 : Create a working folder for FDDEVS Builder environment

On the FDDEVS Builder Workspace GUI
go to File-> Create Folder



Browse to the FDDEVSExamples folder and under “File Name” and type “path”



Note on FDDEVSExamples path:

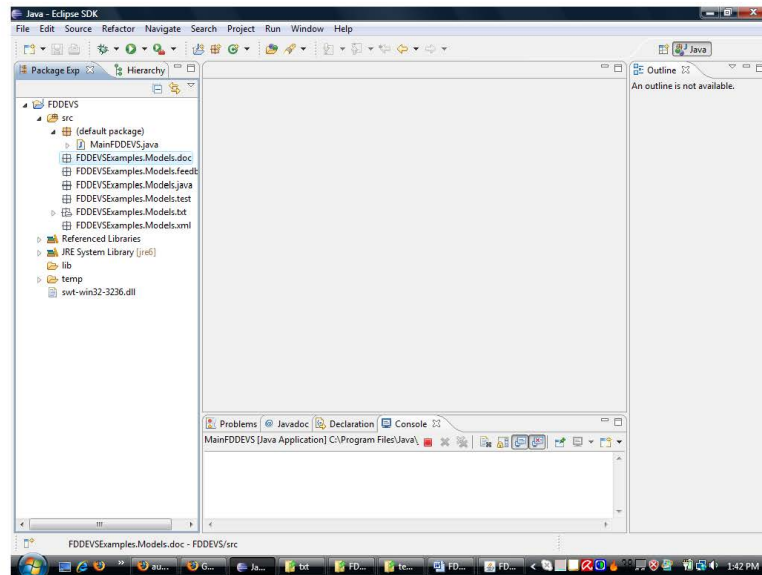
- a) if you used the “Concise guide to installing FDDEVS on Eclipse IDE” to install FDDEVS, the FDDEVSExamples folder will be under
C:\installFDDEVSEp\FDDEVS\src\FDDEVSExamples
- b) if you used the “FDDEVS step by step installation guide for Eclipse IDE” to install FDDEVS, the FDDEVSExamples folder will be under
C:\eclipse\workspace\FDDEVS\src\FDDEVSExamples

**For this guide, we will be using option a).
(ie C:\installFDDEVSEp\FDDEVS\src\FDDEVSExamples**

Important Note

Once you do step 3 you will see

- a) CurrentFDDEVSPATH.txt automatically created at C:\installFDDEVSep\FDDEVS
- file will contain the path C:\installFDDEVSep\FDDEVS\src\FDDEVSEExamples\path
- b) Models folder automatically created at
C:\installFDDEVSep\FDDEVS\src\FDDEVSEExamples\Models
- c) Under Models folder note the other auto created subfolder structure
i) doc ii) feedback iii)java iv)txt v)test vi)xml



Auto Generated folder structure under FDDEVSEExamples as seen within Eclipse IDE

Step 4 : Import examples to the txt folder**Note**

The below task can also be performed by

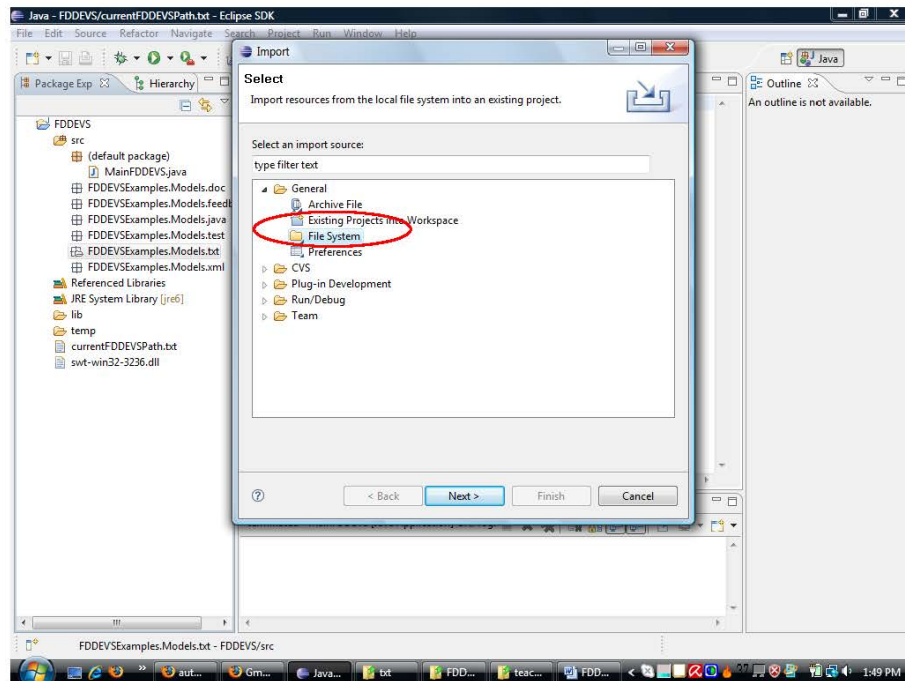
- Copying the contents of C:\installFDDEVSep\Examples
 - Can use Ctrl+C
- To C:\installFDDEVSep\FDDEVS\src\FDDEVSExamples\Models\txt folder
 - Can use Ctrl+V

Within Eclipse IDE,

right click on the **txt** subfolder under (/FDDEVSExamples/Models)

and choose **Import**.

Then choose “**File System**” and under General ,click Next

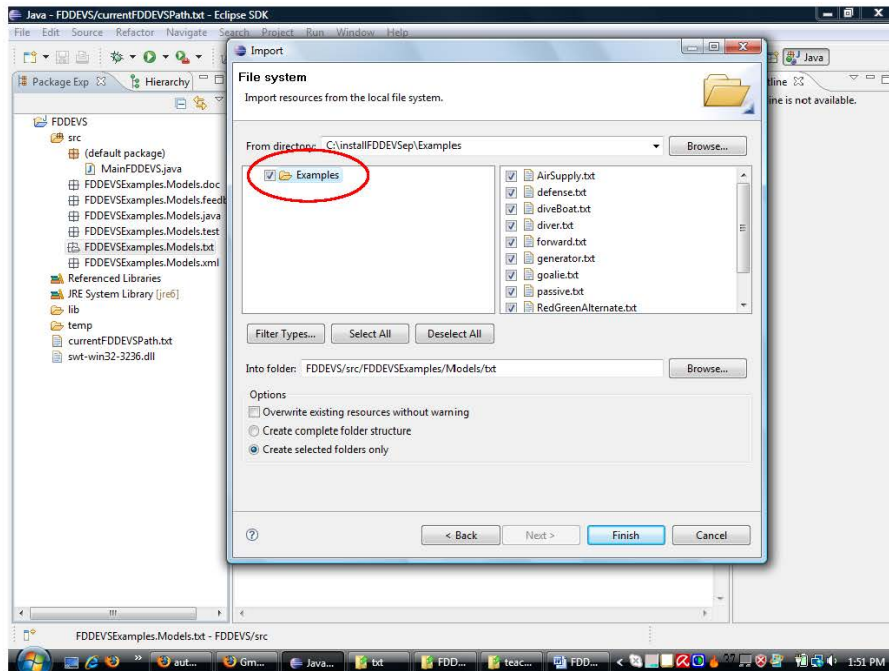


8

Under Import File System choose
“From Directory”

and browse to C:\installFDDEVSep\Examples as seen below.

Check the Box next to Examples folder to select all .txt files within.

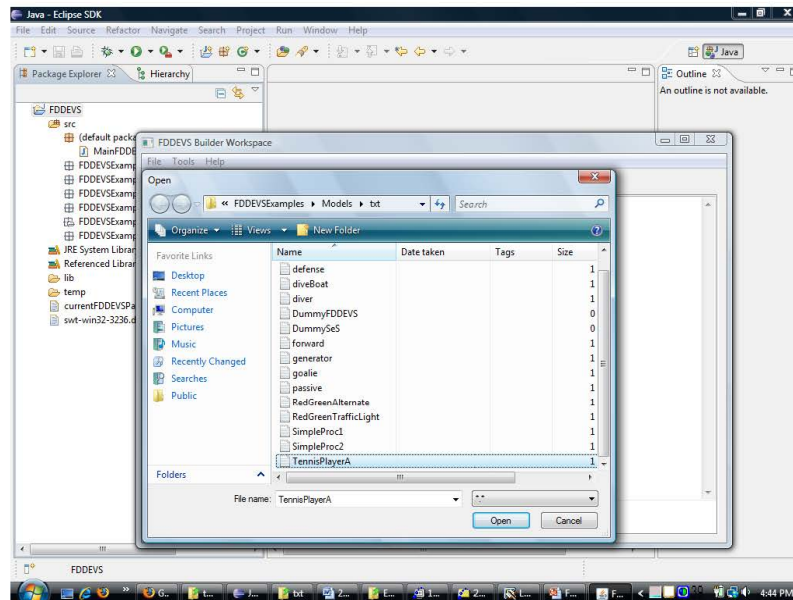


Click Finish. You should see several .txt files imported now to the txt folder.

9

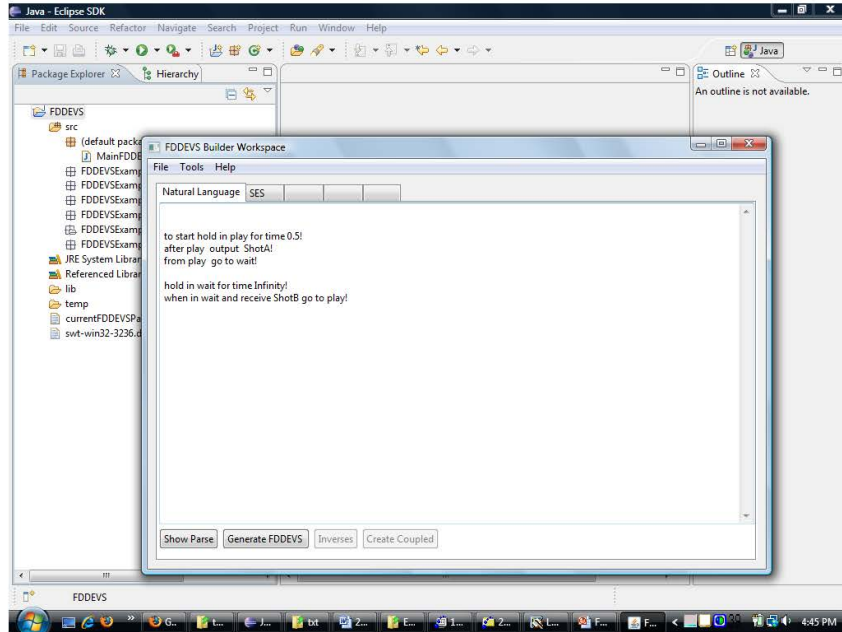
Step 5 : Open the predefined “TennisPlayerA” example.

Now Go to FDDEVS Builder GUI and choose File-> Open. Select “TennisPlayerA.txt”



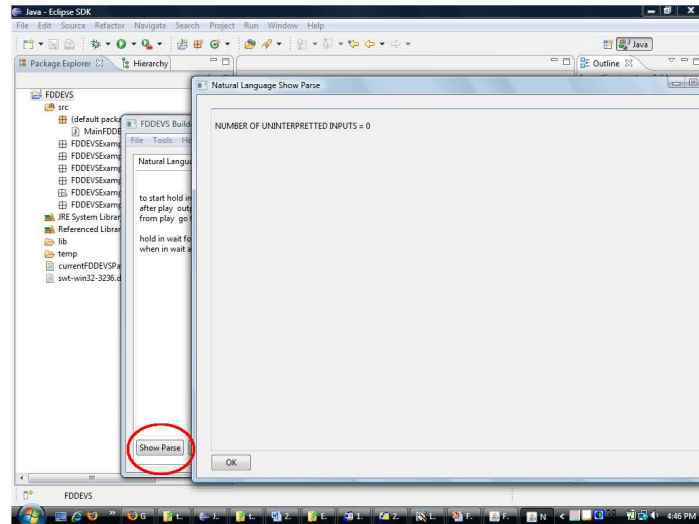
10

You should now see a similar screen as below. (Don't worry about the text content for now)



Step 6 : Parse the TennisPlayerA example.

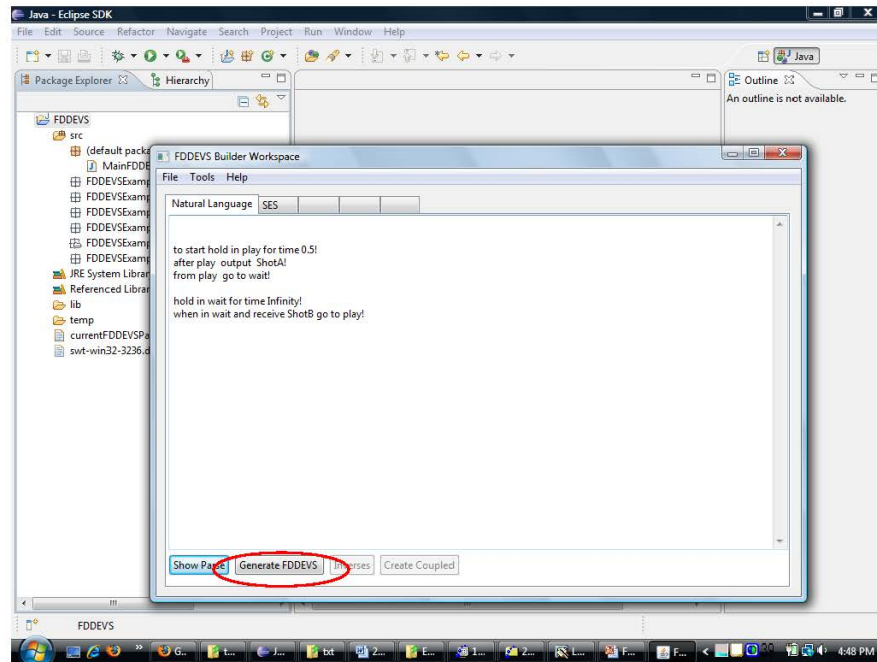
On the FDDEVS Builder GUI , click on the “**Show Parse**” button. If all is well, no errors will be shown as below.

**Note :**

- The pre defined “TennisPlayerA.txt” model we opened has been described using a Natural Language which we will look at in detail later. (not in this guide)
- The “Show Parse” function will scan through the text model and notify you if there are any syntax errors.
- In this particular case , you should find no errors (as the above figure) as the TennisPlayerA model has been correctly defined before.

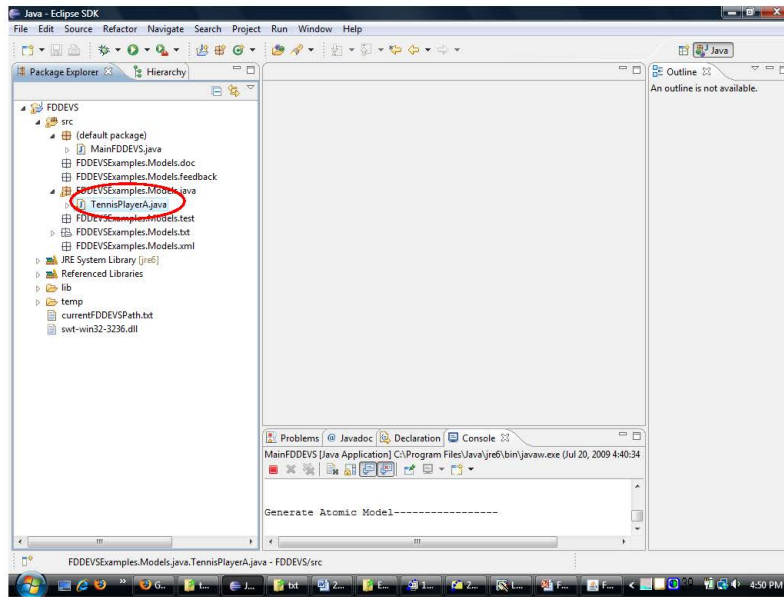
Step 7 : Generate a FDDEVS Atomic model from the TennisPlayerA.txt

On the FDDEVS Builder GUI, click the Generate FDDEVS button.



Important Note:

- Once Generate FDDEVS Button is clicked, go to the Eclipse IDE
 - click **F5 To refresh. Make sure you do this step**
- You should see an automatically generated **TennisPlayerA.java** under FDDEVSExamples\Models\Java folder
- TennisPlayerA.java is an auto generated DEVJSJAVA Atomic model which was created using the text based TennisPlayerA.txt we opened previously

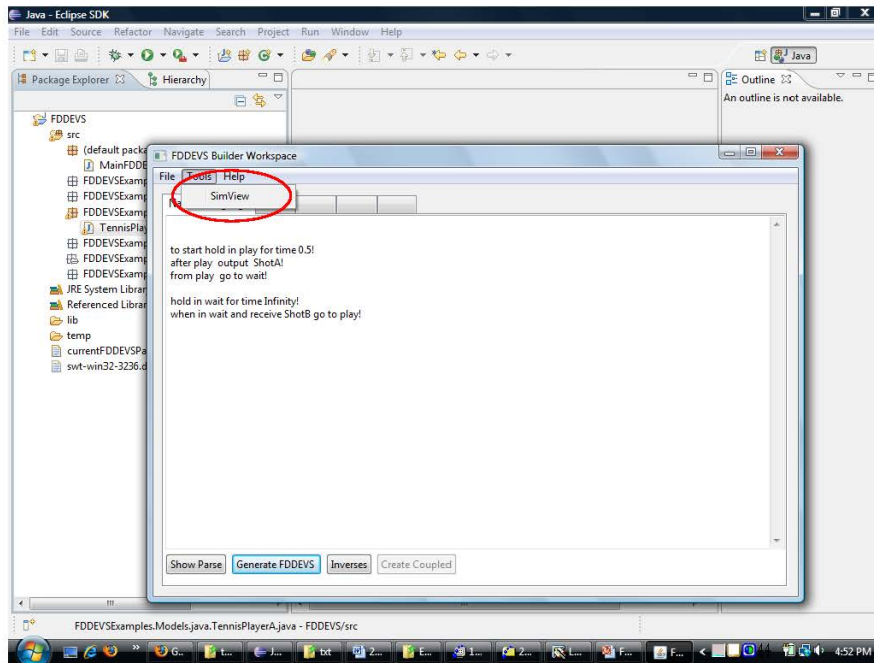


Auto generated TennisPlayerA.java as seen in Eclipse IDE

Step 8 : Visualize the TennisPlayerA Atomic model in SimView

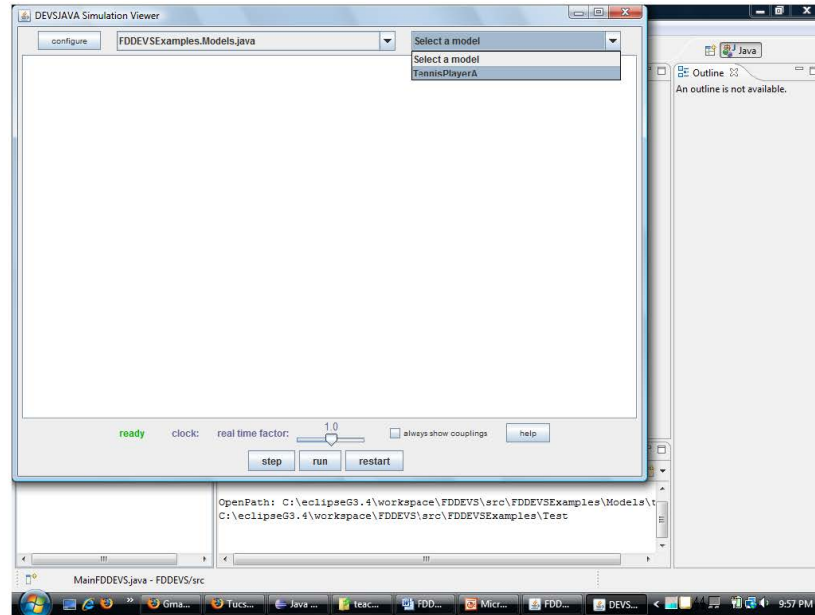
Go to the FDDEVS Builder GUI

Choose Tools-> **SimView**



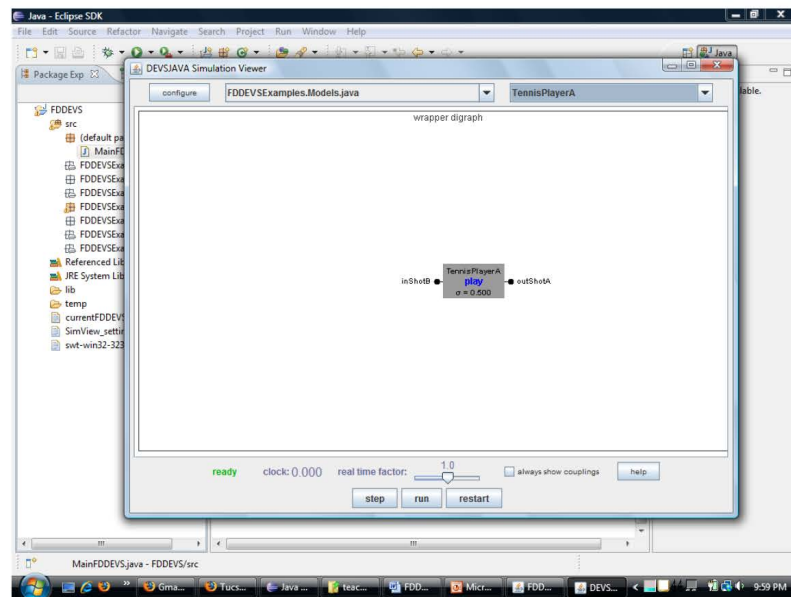
15

A new window for **SimView** will open as below.



In SimView

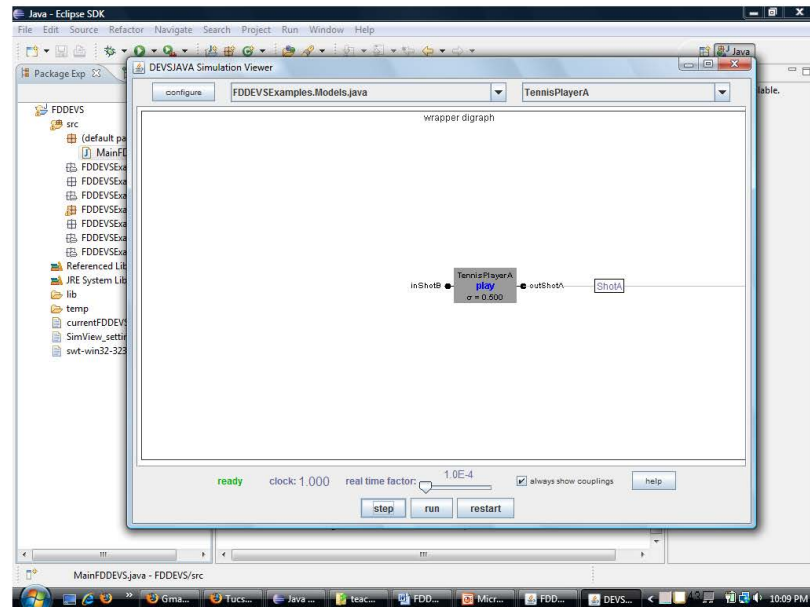
- 1) Click **Configure** Button once
- 2)
 - under “Select a Package”
 - choose FDDEVSExamples.Model.java
 - under “Select Model”
 - choose TennisPlayerA



Atomic Model of TennisPlayerA within SimView.

You should now see the Atomic Model view of TennisPlayer in SimView.

17



Above screenshot shows activity when “step” button is clicked. “Real time factor” is set to lowest and “always show couplings” checkbox is enabled.

Exercise:

- 1) Experiment in SimView
 - a) with the above Tennis PlayerA model.
 - b) with the other predefined examples.

Next Up : INVERSE MODEL GENERATION

This document was prepared by Lahiru Ariyananda.

REFERENCES

- [1] NSF Blue Ribbon Panel, "Revolutionizing Engineering Science through Simulation," 2006.
- [2] V. Singh, System Modeling and Simulation, New Age International Ltd, 2008.
- [3] B. P. Zeigler, H. Praehofer and T. G. Kim, Theory of Modeling and Simulation, Academic Press, 2000.
- [4] B. Wu, Manufacturing Systems Design and Analysis, Chapman & Hall , London, 1992.
- [5] P. Fritzson, "MathModelica—An Object-Oriented Mathematical Modeling and Simulation," *The Mathematica*, vol. 10, no. 1, p. 209, 2006.
- [6] R. M. Fujimoto, Parallel and Distributed Simulation Systems, John Wiley and Sons, 2000.
- [7] J. A. Sokolowski and C. M. Banks, Modeling and Simulation Fundamentals : Theoretical Underpinnings and Practical Domains, John Wiley and Sons, 2010.
- [8] P. A. Fishwick, Simulation Model Design and Execution, Prentice Hall, 1995.
- [9] M. L. Loper, Modeling & Simulation in the Systems Engineering Life Cycle : Core Concepts and Accompanying Lectures, Springer, 2015.
- [10] M. Jahangirian, T. Eldabi, A. Naseer and L. Stergioulas, "Simulation in manufacturing and business: A review," *European journal of operational research*, vol. 203, no. 1, pp. 1-13.
- [11] R. Kitamura and M. Kuwahara, Simulation approaches in transportation analysis recent advances and challenges, New York: Springer, 2005.

- [12] M. Sundberg, "The Everyday World of Simulation Modeling: The Development of Parameterizations in Meteorology," *Science, Technology, & Human Values*, vol. 34, no. 2, pp. 161-182, 2009.
- [13] M. Jeruchim, P. Balaban and S. Shanmugan, *Simulation of communication systems*, New York: Plenum, 1992.
- [14] M. Hong, J. Rozenblit and A. Hamilton, "Simulation-based surgical training systems in laparoscopic surgery: a current review," *Virtual reality : the journal of the Virtual Reality Society*, vol. 25, no. 2, pp. 491-510, 2021.
- [15] L. Gantt and M. Young, *Healthcare simulation : a guide for operations specialists*, Hoboken: John Wiley & Son, 2016.
- [16] P. Hsu and J. Rozenblit, "A computer-aided design framework for modeling and simulation of VLSI interconnections and packaging," *Integration*, vol. 17, no. 2, pp. 163-187, 1994.
- [17] N. R. Council, *Defense Modeling, Simulation, and Analysis: Meeting the Challenge*, Washinton DC: National Academies Press, 2006.
- [18] S. Diallo, K. Gupton, J. Padilla and C. Turnitsa, "Proceedings of the 2013 Spring Simulation Multiconference : Military Modeling & Simulation Symposium," in *SpringSim 2013*, San Diego, 2013.
- [19] J.-M. Réveillac, *Modeling and Simulation of Logistics Flows 1: Theory and Fundamentals*, Newark: John Wiley & Son, 2017.
- [20] J. J. Padilla, S. Y. Diallo and A. Tolk, "Do We Need M&S Science?," *SCS M&S Magazine*, pp. 161-166, Oct 2011.

- [21] B. Horowitz, "Defense Science Board recommendations:an examination of defense policy on the use of modeling and simulation," in *Proceedings of the 1990 Winter Simulation Conference*, NJ, 1990.
- [22] B. K. Choi and D. Kang, *Modeling and Simulation of Discrete Event Systems*, John Wiley & Sons , 2013.
- [23] T. T. Allen, *Introduction to Discrete Event Simulation and Agent-based Modeling*, 2011, Springer-Verlag.
- [24] S. Taylor, A. Khan, K. Morse, A. Tolk, L. Yilmaz, J. Zander and P. Mosterman, "Grand challenges for modeling and simulation: simulation everywhere—from cyberinfrastructure to clouds to citizens," *SIMULATION: Transactions of The Society for Modeling and Simulation International*, pp. 1-18, 2015.
- [25] R. Fujimoto, C. Bock, W. Chen, E. Page and J. Panchal, *Research Challenges in Modeling and Simulation for Engineering Complex Systems*, Springer, 2017.
- [26] R. Hawkins and K. Klukis, "Tools for the simulation profession : proceedings of the 1987 conferences, tools for the simulationist, the simulation profession," in *Proceedings of the 1987 conferences ,Society for Computer Simulation*, Orlando, 1987.
- [27] A. Goldman and C. S. Sripada, "Simulationist models of face-based emotion recognition," *Cognition*, vol. 93, no. 3, pp. 193-213, 2005.
- [28] S. Kardong-Edgren, "Is Simulationist a Word?," *Clinical Simulation in Nursing*, vol. 9, no. 12, p. 561, 2013.
- [29] "National Training & Simulation Association," [Online]. Available: <https://simprofessional.org/pdf/SimulationistCodeOfEthics.pdf>. [Accessed October 2021].

- [30] J. Rozenblit, "Experimental Frame Specification Methodology For Hierarchical Simulation Modeling," *International journal of general systems*, vol. 19, no. 3, pp. 317-336, 1991.
- [31] "AutoDesk," Autodesk, [Online]. Available: <https://www.autodesk.com/solutions/simulation/overview>. [Accessed October 2021].
- [32] "Arena Simulation Software," Rockwell Automation, [Online]. Available: <https://www.rockwellautomation.com/en-us/products/software/arena-simulation.html>. [Accessed October 2021].
- [33] "PSPICE," Cadence, [Online]. Available: <https://www.pspice.com/about>. [Accessed October 2021].
- [34] . H. Szczerbicka, J. Banks, R. Rogers, T. Oren, H. Sarjoughian and B. Zeigler, "Conceptions of Curriculum for Simulation Education," in *2000 Winter Simulation Conference Proceedings*, 2000.
- [35] W. V. Tucker, B. T. Fairchild and D. C. Gross, "Simulationists: What Does Industry Want?," in *Summer Computer Simulation Conference*, 2000.
- [36] R. Rogers, "Results of the Workshop "What Makes A Modeling & Simulation Professional"," in *SCSC*, Vancouver, 2000.
- [37] R. Rogers, "What makes a modeling and simulation professional ? : The consensus view from one workshop," in *Proceedings of the 1997 Winter Simulation Conference*, Piscataway , 1997.
- [38] W. Yurcik and R. Silverman, "1998 NSF Workshop on Teaching Simulation to Undergraduate Computer Science Majors," in *SCSC*, Vancouver, 2000.
- [39] R. Fujimoto, "Principles for M&S Education," *Simulation and Technology Magazine*, 2000.

- [40] A. Gonczi, "Competency-Based Approaches: Linking theory and practice in professional education with particular reference to health education," *Educational Philosophy and Theory*, vol. 45, no. 12, pp. 1290-1306, 2013.
- [41] SIGSIM, "Association for Computing Machinery," [Online]. Available: <https://www.acm-sigsim-mskr.org/msDegrees.htm>.
- [42] B. P. Zeigler, *Theory of Modeling and Simulation*, Wiley, 1976.
- [43] D. Fone, S. Hollinghurst, M. Temple, A. Round, N. Lester and A. Weightman, "Systematic review of the use and value of computer simulation modelling in population health and health care delivery," *Journal of Public Health*, vol. 25, no. 4, pp. 325-335, 2003.
- [44] K. Katsaliaki and N. Mustafee, "Applications of simulation within the healthcare context," *Journal of the Operational Research Society*, vol. 62, no. 8, p. 1431–51, 2011.
- [45] N. Mustafee, K. Katsaliaki and S. Taylor, "Profiling Literature in Healthcare Simulation," *SIMULATION*, vol. 86, no. 8-9, p. 543–58, 2010.
- [46] "Arizona Center for Integrative Modeling and Simulation," [Online]. Available: <https://acims.asu.edu/>. [Accessed October 2021].
- [47] "Advanced Real-Time Simulation Laboratory," [Online]. Available: <https://arslab.sce.carleton.ca/index.php/devs-tools/>. [Accessed October 2021].
- [48] B. Zeigler and A. Muzy, "From Discrete Event Simulation to Discrete Event Specified Systems (DEVS)," *International Federation of Automatic Control*, vol. 50, no. 1, pp. 3039-3044, 2017.
- [49] "DEVS Suite," [Online]. Available: <https://acims.asu.edu/devs-suite/>. [Accessed October 2021].

- [50] G. Wainer, "An Introduction to Modeling and Simulation with DEVS," [Online]. Available: <https://vdocuments.mx/an-introduction-to-modeling-and-simulation-with-devs-gabriel-a-wainer-department.html>. [Accessed October 2021].
- [51] "ADEVs," [Online]. Available: <https://web.ornl.gov/~nutarojj/adevs/>. [Accessed October 2021].
- [52] S. Mittal, J. Risco-Martín and B. Zeigler, "DEVS/SOA: A Cross-Platform Framework for Net-centric Modeling and Simulation in DEVS Unified Process," *Simulation*, vol. 85, no. 7, pp. 419-450, 2009.
- [53] G. Wainer, "Introduction to CD++," [Online]. Available: http://cell-devs.sce.carleton.ca/mediawiki/index.php/Main_Page. [Accessed 2021].
- [54] M. Bonaventur, G. Wainer and R. Castro, "Graphical modeling and simulation of discrete-event systems with CD++Builder," *Simulation*, vol. 89, no. 1, 2013.
- [55] . T. G. Kim and J. Kim , "DEVS Framework and Toolkits for Simulators Interoperation Using HLA/RTI," in *DEVS Framework and Toolkits for Simulators Interoperation Using HLA/RTI*, Beijing, 2005.
- [56] M. Hwang and B. Zeigler, "A reachable graph of finite and deterministic DEVS networks," in *DEVS Integrative M&S Symposium*, Huntsville , 2006 .
- [57] S. Mittal, B. Zeigler and M. Hwang, "XFD-DEVS," [Online]. Available: <http://www.saurabh-mittal.com/fddevs/>. [Accessed January 2022].
- [58] B. P. Zeigler and M. C. Salas, "AutoDEVS: A Methodology for Automating Modeling and Simulation Software Development," *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 6, no. 1, pp. 33-52, 2009.

- [59] T. G. Kim and B. P. Zeigler, "Hierarchical scheduling in an intelligent environmental control system," in *The Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Tullahoma, 1989.
- [60] L. Ariyananda, "Graphical User Interface Representation and Generation using System Entity Structures," 2007. [Online]. Available: <https://acims.asu.edu/wp-content/uploads/2012/02/LahiruThesis28.pdf>. [Accessed January 2022].
- [61] G. Klir and D. Elias, *Architecture of Systems Problem Solving*, New York: Springer, 2003.
- [62] B. Hollocks, "Assessing Simulation Learning in Higher Education," in *Proceedings of the 2005 Winter Simulation Conference*, 2005.