System Engineering and System Dynamics Models

Dr. Warren W. Tignor Ph.D.

Science Applications International Corporation (SAIC)

International Conference of the System Dynamics Society
Oxford, England
July 25 - 29, 2004

Abstract

System Dynamics (SD) set its roots in servomechanism systems that were a combination of the earliest hardware and software systems known. Over time, SD grew and evolved into a multiplicity of domains; coevolving was the domain of System Engineering. Today, System Engineering has sub-domains such as architecture, design, performance and modeling. The System Engineering topic of modeling has reached the stage of development where structure and behavior are of high-interest; and they are the very cornerstones of SD. In fact, there is enough System Engineering interest in system structure and behavior that entirely new procurements are being considered for a simulation capability based on the Unified Modeling Language™ (UML), particularly Unified Modeling Language 2™, UML2™.

This paper looks at the possibility of applying SD to the problem of modeling the system-engineered structure and behavior of information system architectures, designs and performance. What is proposed is the possibility to reconnect with the roots of SD from the servomechanisms systems to 21st century information systems. Reconnecting with SD roots in servomechanism systems offers the opportunity to bridge SD and its structure and behavior capability with information systems as described using UML2™. SD offers the System Engineering domain an opportunity to leverage a compatible field of interest, and its modeling and simulation tools without delay or costly new development.

Table of Contents

List of Tables

List of Figures

1 Introduction

Forrester (1961) discusses the evolution of industrial dynamics and its relationship to engineering systems with regards to World War II weapon fire-control systems. He notes that before 1950 there was little or no acceptance of automated decision-making. However, in the time from 1935 to 1945 information systems of 20 variables were easily modeled using analog computers; from 1945 to 1955, information systems of 200 variables were modeled with digital computers (Forrester 1961). Perhaps we are at the full circle point in time when it is essential to reexamine the relationship between system dynamic models and engineering systems. With the advent of the Unified Modeling Language2™ (UML2™) to model the structure and behavior of information systems, this paper examines system dynamics in relationship to information systems modeling and simulating structure and behavior.

2 Statement of the Problem

The system engineering domain has a clear need to model the structure and behavior of information systems prior to embarking on a long and expensive implementation process. The information system world has concentrated on the development of a modeling language, UML™, that provides multiple views of a system, to include structure but little or no support to the aspect of behavior, particularly with regard to simulation. Over time, UML2™ was developed to help close the gap between UML™'s capability to simulate the behavior of information system. Studies to date indicate that UML2™ has not closed the gap from a simulation capabilities perspective and neither have several other languages/systems, e.g., SimML, Rational Rose, TauGeneration2. This paper questions whether any new language is required at all Since System Dynamics was invented to model system structure and simulate behavior, this paper examines whether it will apply to information systems. If an information system's structure and behavior can be described using UML™, will it translate readily into the System Dynamics paradigm?

3 Literature Review

Towill wrote that system dynamics is a methodology for modeling and redesigning manufacturing, business, and similar systems that are part man and part machine (1993a). He went on to discuss the roots of system dynamics set both in servo theory and cybernetics with emphasis on "input-output analyses which are the system dynamics equivalent of writing down balanced equations for hardware systems", (Towill, 1993a, p. 201). Towill explains that system dynamics is a tool that can model, design and improve "hybrid" hard/soft systems.

System dynamics (SD) addresses building computer models of complex problem situations and then experimenting with and studying the behavior of the models over time (Caulfield & Maj, 2001). SD models demonstrate how unappreciated causal relationships, dynamic complexity, and structural delays can lead to counter-intuitive outcomes. Additionally, SD accommodates “soft factors” such as motivation and perceptions so that management can better understand problem spaces. In Caulfield’s and Maj’s opinions, “…system dynamics in particular represents a choice of first resort for the broadest range of problem spaces” (2001, p. 2793).

Caulfield and Maj (2001) see SD as part of general systems theory (GST) that they attribute to the work of organismic biologist Ludwig von Bertalanffy (p. 2795). Bertalanffy was concerned that modern science was over specialized, necessitated by enormous amounts of data, the complexity of techniques, and the theoretical structures within each field.

While others were extolling the virtues of system dynamics to model complex systems, UML™ was appearing on the seen as a third generation object-oriented information system modeling language. Kortright (1997) recognized that there was a great deal to be gained from the use of UML™ for simulation modeling of complex software systems. He noted that an important problem in simulation is that many models are described in a large variety of notations or directly in a programming language (Kortright, 1997). To the contrary, UML™ provided a set of proven notations for model description and permitted the visualization of alternative designs.

Despite the power of UML™ to visualize a model, the diagram descriptions themselves remained static. To perform an executable simulation, UML™ required translation into another language, an executable language. Kortright said, “There is a close correspondence between the Java programming language and UML™” (1997, p. 45).

Kobryn (1998) describes the UML™ constructs and techniques that support the modeling of distributed enterprise information system architectures. He shows how UML™ can be used for behavioral “(or dynamic)” architectural modeling. He makes clear that the information system architecture field is in its infancy and that the emphasis is on the structural properties of software systems rather than their behavioral properties.

Miller (2002) wrote that the Object Management Group™ (OMG™) knew that UML™ was a compromise among several competing schools of modeling and that all the available “good ideas” were not included in the language. Hence, the OMG™ initiated a request for proposal in 2000 for the revision of UML™ as UML2™. Five teams submitted proposals in 2001 and the OMG™ is expected to release a UML2™ standard in April 2004.

Selic, Ramakers, and Kobryn (2002) document that practitioners and researchers have adopted UML™ since its introduction in 1997 at a rate exceeding OMG™'s expectation. Its popularity confirms the need for a communication medium for both humans and software tools. At the same time, the rapid acceptance and usage has generated pressure for improvements and new features.

Before considering changes, Selic et al. (2002) advocate an understanding of the original purpose of UML™, i.e. a general purpose language for modeling object-oriented information system applications. The purpose of the model is to understand the suitability of a proposed solution before expenditure of resources to build it. They say that the need for modeling becomes more apparent as the complexity of information system software increases; the complexity will reach to the point of challenging our ability to comprehend it. So the challenge is to create a model that allows the concise expression of the essential aspects of information system software being designed while omitting irrelevant detail.

For practical reasons, Selic et al. (2002) want to see UML™ evolve rather than be replaced with something new. They see a primary requirement for UML2™ as a precise definition of semantics, including "its dynamic (runtime) semantics", (Selic et al., 2002, p. 71). They see the extended modeling of complex behavior as needed. Behavior modeling needs to include the ability to hierarchically compose and combine individual information system behavior specifications. However, they recognize that in some cases, cramming all useful concepts from diverse domains into a single language is not always practical. "It is crucial to avoid the infamous 'language bloat' syndrome while retaining all the advantages of a standard" (Selic et al., 2002, p. 71).

In contrast to Selic, et al. (2002), Dori (2002) thinks that UML™ reform may be "too little too late" to become the information system architect's tool of choice for modeling complex software systems. To him, UML2™ must integrate structure and behavior in a single user friendly manner; the scope required is significant and can only be achieved in a revolutionary way. He sees UML™ problems in three categories: "model multiplicity resulting from excess diagram types and symbols; confused behavior modeling; and the obscuring influences of programming languages" (Dori, 2002, p. 82). He declares that none of the nine UML™ models clearly shows an integrated view of the most prominent and useful system aspects: structure and behavior. According to Dori (2002), the interdependence of structure and behavior mandates that these two major system aspects be addressed concurrently.

UML™'s segregation of structure and behavior unnecessarily strains system engineers' cognitive ability by requiring they mentally integrate the various system models into a coherent, holistic view. “This cognitive load severely hinders use of UML™ as a system-modeling tool” (Dori, 2002, p. 83). System engineers have to struggle with complexity and inconsistency regarding the modeling of information system dynamics. “UML™’s inherent lack of a unifying system dynamics concept calls for another comprehensive revision”, according to Dori (2002, p. 84).

Wiklund (2003) looked at UML™ to see whether it was possible to model the dynamic change of behavior. In particular, he was evaluating how on-line replacement of JavaFrames CompositeStates could be used to change parts of the state space during run-time; he wanted to model the problem first with UML™. He found that it was possible to model run-time introduction of composite states, using interaction diagrams in UML2™. He found “…no solution to modeling such dynamic change of behavior” neither with UML™ 1.4, 1.5, nor the proposed UML2™ standard (Wiklund, 2003, p. 2).

At the time that Winlund did his research, the UML2™ superstructure standard had not been adopted by the OMG™ – May 2003. On June 12, 2003, OMG™ announced at their technical meeting in Paris, France, that the Analysis and Design Task Force voted to recommend adoption of the UML2™ Superstructure specification, completing the definition of this major upgrade to the industry's main software modeling notation Lenehan (2003).

De Wit (2003) has examined UML2™ as a step towards a universally accepted information system software performance engineering (SPE) tool. As his test case, he used the work of Bütow, Mestern, Schapiro and Kritzinger (1996) that examined the means of predicting the performance of a communication protocol. According to de Wit (2003), Bütow et al. (1996) took an approach that did not affect the syntax of the formal description technique and did not depend on it either. Since Bütow et al. (1996) were able to use “SDL Performance Evaluation of Concurrent Systems” as a way of mapping a system specification to its target environment without changing the syntax of the formal description technique, de Wit (2003) investigated applying the same approach using UML2™ (pp. 4-5).

What de Wit (2003) investigated is the feasibility of moving SPE from the traditional “start” of the development life-cycle, where requirements are determined and the “end” where they are verified by testing, to include a model-based approach during information system development that validates performance throughout the development cycle. He believes that “…software specification languages (notably UML™) should incorporate the ability to specify performance requirements, thus bridging the gap between information system software design and performance analysis” (de Wit, 2003, p. 7).

Arief (2001) states that building a new information system requires careful planning and investigation in order to avoid problems in later stages of development. Generally, using UML™ in the system specification eliminates or minimizes ambiguities. The proposed system's performance needs to be investigated before the implementation stage can be commenced (Arief, 2001). Understanding information system performance is necessary to decide the following: 1. whether a particular design will meet the requirements and 2. whether it is worth implementing the system or not.

One way to obtain performance estimations is by simulating to mimic the execution of the system. Arief (2001) points out that currently available UML™ tools do not provide any facilities for generating simulation programs from UML™ specifications. His research involved investigating a framework that could capture the information system simulation model from UML™ design notation.

Arief (2001) built a framework called the Simulation Modeling Language (SimML) as part of his Ph.D. dissertation. Arief (2001) constructed tools that enable an automated transformation of a UML™ design notation into a simulation program, Figure 3-1.

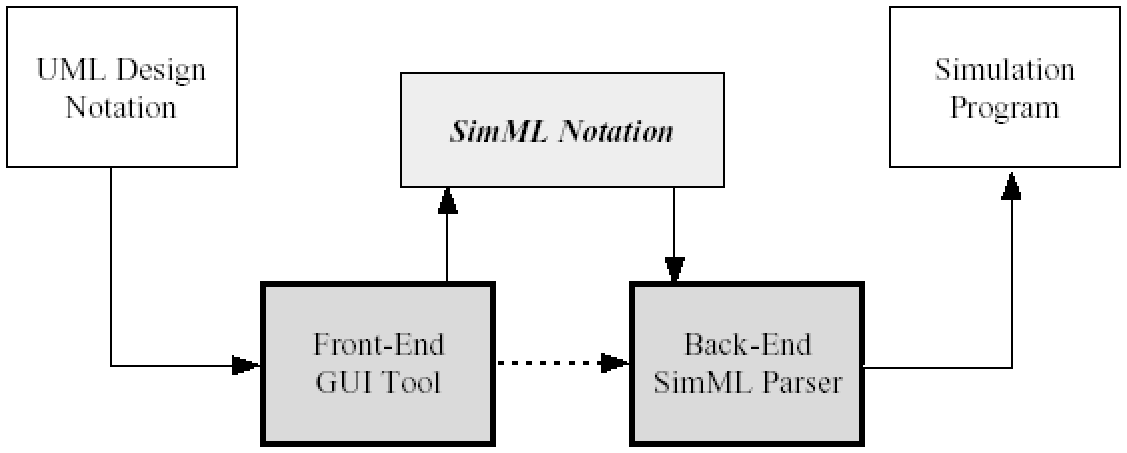


Figure 3-1 UML™ to Simulation Path

According to Arief (2001), the SimML framework can be used for generating simulation programs in other simulation languages (such as *SIMULA*) by modifying the program generator part of the SimML parser, Figure 3-2.

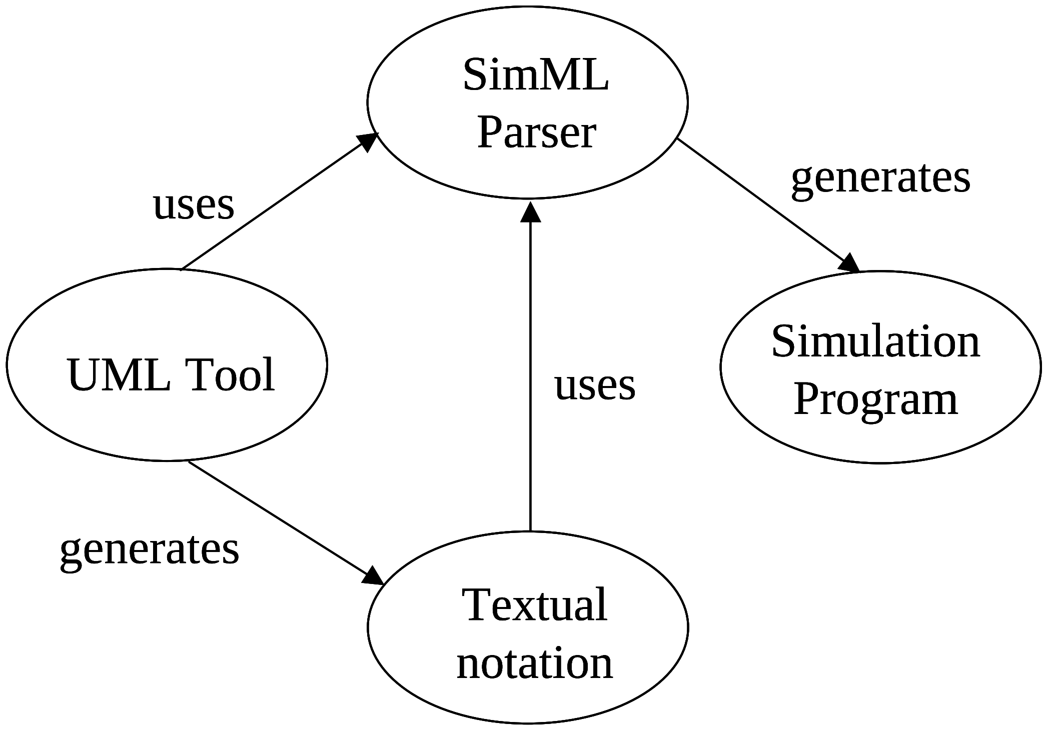


Figure 3-2 Paths for Generating Simulation from UML™

With the tools, he performed case studies to demonstrate their ability to create a simulation framework in general. The case study of interest focuses on the `makeCall` operation of the British Telecom's Intelligent Network (IN) application (Arief, 2001).

4 Research Method and Design

The research method uses a case study of a real-life problem described in UML™ and a non-system dynamics simulation language, SimML, and recasts the UML™ into a System Dynamics model using Vensim. The work by Arief (2001) documents the real-world problem identified by British Telecom (BT) Intelligent Network (IN) to size the server capacity needed to make and receive calls within an information system that featured the capability to perform caller id, call blocking, billing and more.

Arief (2001) modeled the system structure using UML™ diagrams. For behavior modeling, Arief (2001) created a Java language based simulation language framework he called SimML. The research design reuses the UML diagrams created by Arief and applies them to the creation of a Stock and Flow system dynamics model. From the stock and flow diagram, a Vensim model is constructed for structure and behavior performance comparison to the SimML model created by Arief.

The intent of the research method and design is to show that UML™ diagrams support the construction of a System Dynamic model, Stock and Flow diagram, and that the simulation results produced by Vensim are comparable to Arief's (2001) results.

If the results of the System Dynamic model and Vensim simulation are comparable to the case study results (Arief, 2001), the hypothesis that System Dynamic models are derivable from UML™ is supported and the relevance of System Dynamics models to UML™ is substantiated, making room for Vensim and similar languages to become part of the information system structure and behavior modeling toolset. Although there is limited comparison of model results, the focus is not to produce a comparison of SimML to Vensim, or the specific results of the simulations.

5 Data Analysis

The data analysis begins with the BT problem statement, block diagram, UML™ diagrams, and SimML simulation results generated by Arief (2001). Following these artifacts are the System Dynamic Stock and Flow models and simulation results used for comparison to Arief (2001). The focus of the comparison is the ability to produce a comparable System Dynamic stock and flow model, and simulation results from the same UML™ diagrams and requirements that were used by Arief (2001). Although there is limited comparison of model results, the focus is not to produce a comparison of SimML to Vensim, or the specific results of the simulations; the emphasis is the ability of a system dynamics approach to produce a useful stock and flow model and simulation results from UML™ diagrams.

5.1 Description of the BT IN Requirements

The BT IN system added new call handling features such as credit card charges and call barring (Arief, 2001, p. 125). According to Arief (2001) a call is initiated at a switch that passes the call request to a local computer for processing made of two operations: 1.

makeCall, and 2. receiveCall. MakeCall maintains information relating to outgoing calls such as should the call be barred, call barring. ReceiveCall maintains information concerning incoming calls such as whether the receiver wishes to receive the call from the caller, call blacklisting. The processing performed needs to be completed within a second for customer satisfaction.

5.1.1 BT Physical Block Diagram

A nonUML™ physical block diagram is presented below in Figure 5-3 (Arief 2001, p. 134). Parenthetical annotations of parameters are added based on analysis of the text.

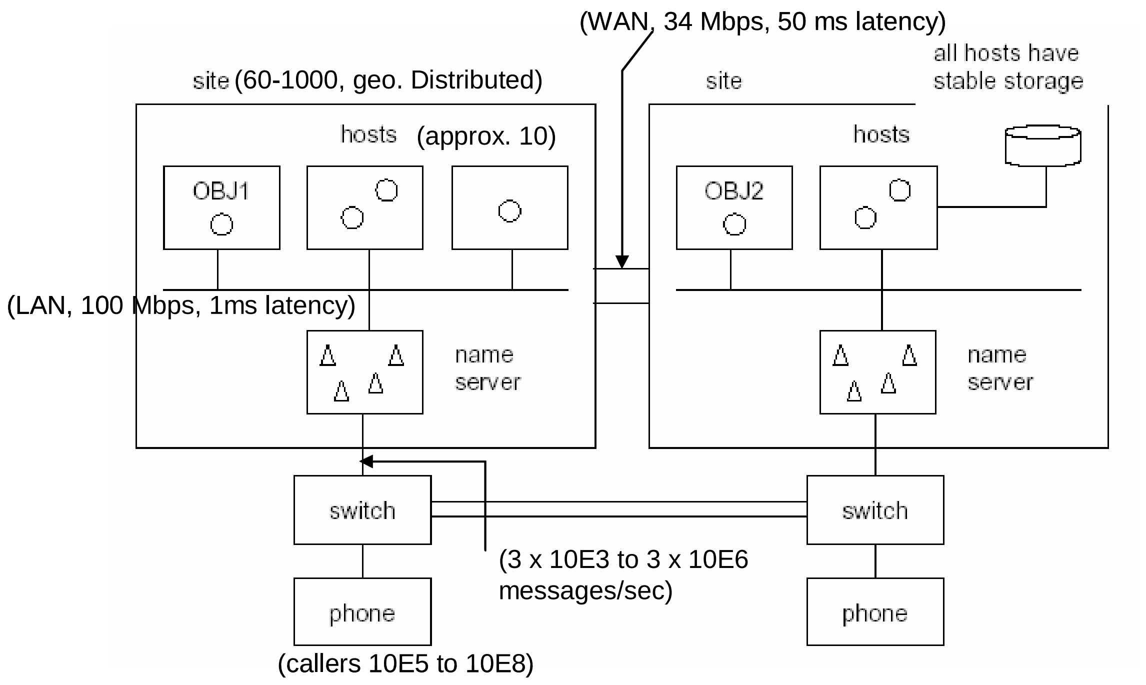

(WAN, 34 Mbps, 50 ms latency)
all hosts have stable storage
site (60-1000, geo. Distributed)
site
hosts (approx. 10)
hosts
OBJ1
OBJ2
(LAN, 100 Mbps, 1ms latency)
name server
name server
switch
switch
(3 x 10E3 to 3 x 10E6 messages/sec)
phone
phone
(callers 10E5 to 10E8)

Figure 5-3 Physical Architecture of the BT IN Application

5.1.2 BT Specification using UML™

The BT physical and applications architecture are described in the UML™ diagrams below (Arief, 2001, pp. 135-137).

5.1.2.1 UML™ Physical Architecture

An UML™ class diagram depicts the architecture and notes many of the physical parameters involved in the simulation model, Figure 5-4, (Arief, 2001). This diagram is particularly interesting given its identification of host memory and speed; values for these parameters are neither provided nor are they considered in the SimML model. Message size is not considered. Bandwidth is given but in the model it does not appear to be considered.

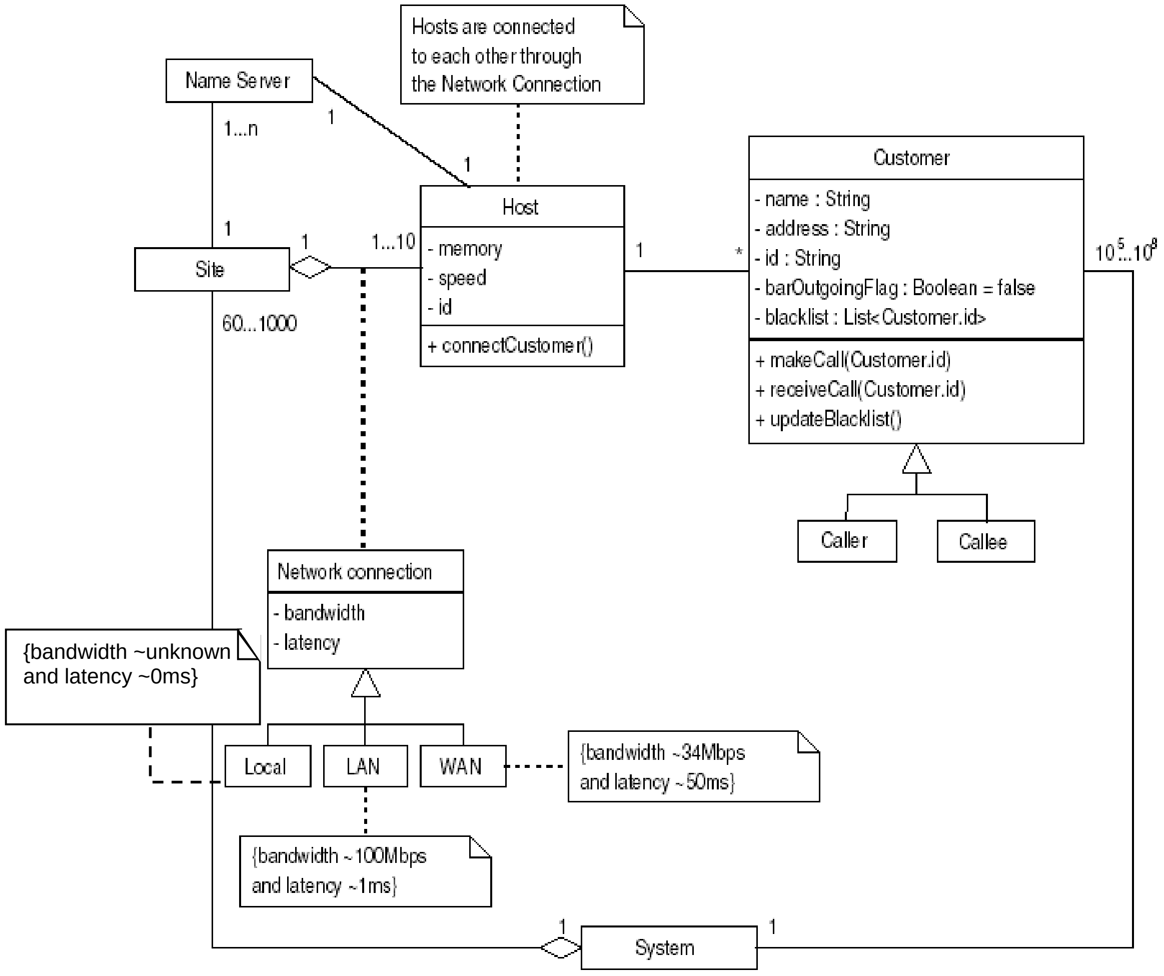

Hosts are connected to each other through the Network Connection
Name Server
1...n
1
1
Host
- memory
- speed
- id
+ connectCustomer()
Customer
- name : String
- address : String
- id : String
- barOutgoingFlag : Boolean = false
- blacklist : List<Customer.id>
+ makeCall(Customer.id)
+ receiveCall(Customer.id)
+ updateBlacklist()
1
*
10^5..10^8
Site
1
1
1...10
60...1000
Caller
Callee
Network connection
- bandwidth
- latency
{bandwidth ~unknown and latency ~0ms}
Local
LAN
WAN
{bandwidth ~34Mbps and latency ~50ms}
{bandwidth ~100Mbps and latency ~1ms}
1
System
1

Figure 5-4 UML™ Class Diagram of BT IN Physical and Application Architecture

5.1.2.2 UML™ Application Logic

An UML™ Sequence Diagram, Figure 5-5, represents the application logic sequence (Arief, 2001). The overall concept of the flow of the logic sequence is depicted in the Activity Diagram, Figure 5-6. It is particularly important to note the performance requirements given textually on the lower left hand corner of the Sequence Diagram. When this model was built, UML™ did not provide any mechanism other than comments to capture performance parameters; the current plan for UML2™ does not appear to change that capability.

The Sequence Diagram presents the steps of the processing sequence in a left to right manner, with looping and feedback indicated by arrows. Time is presented on the vertical axis.

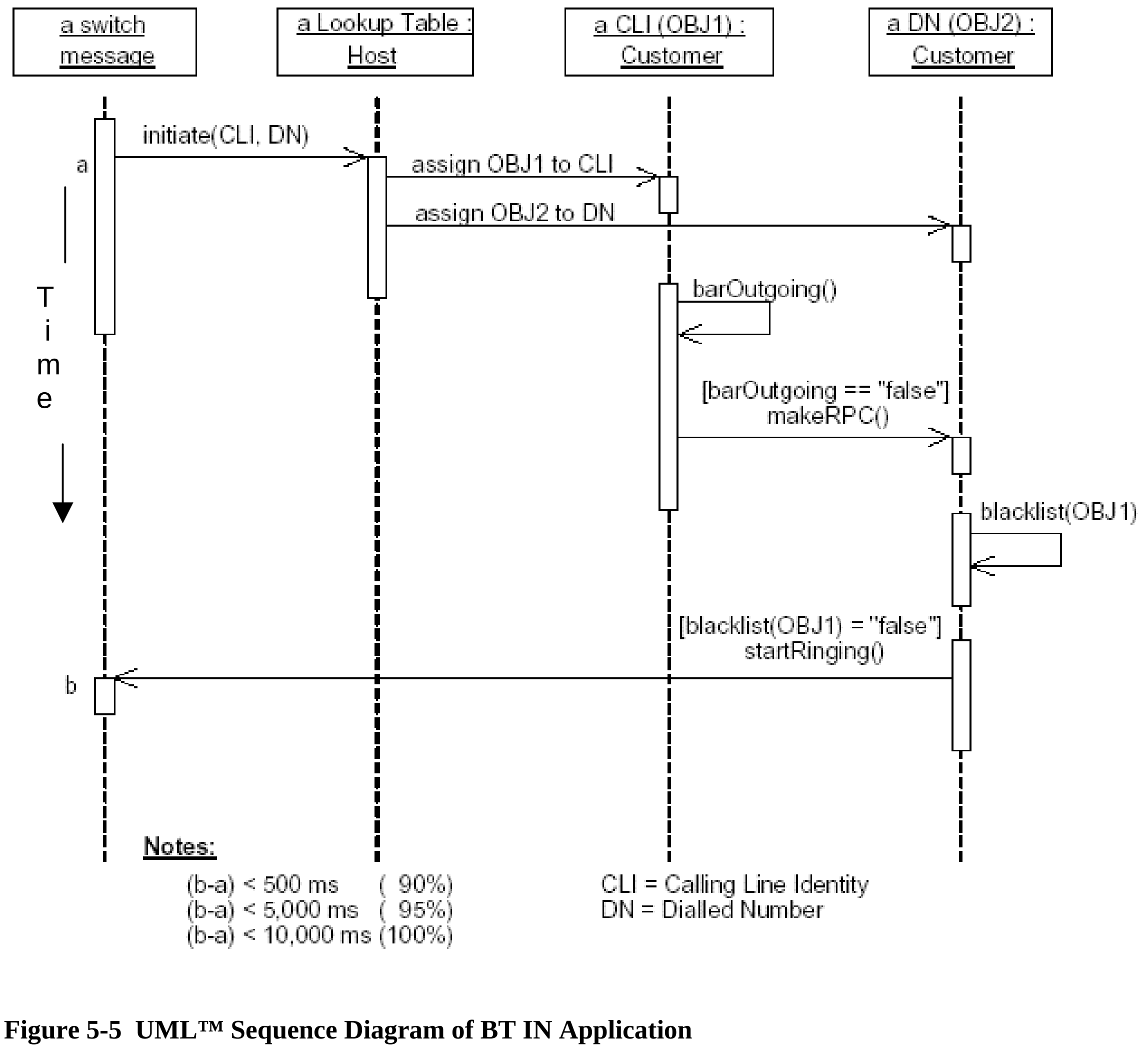


Figure 5-5 UML™ Sequence Diagram of BT IN Application

The activity diagram, Figure 5-6, presents a good overall conceptual flow of the BT IN system activities (Arief, 2001). The activity diagram is a graphic concept of operation of the BT IN system. There are several parts of the overall concept as presented below that are not modeled in SimML. For example, the error handling activity and the activities that occur after the start of ringing are neither modeled nor considered within the scope of the performance parameters. Performance is measured only to the point of ringing the dialed number phone.

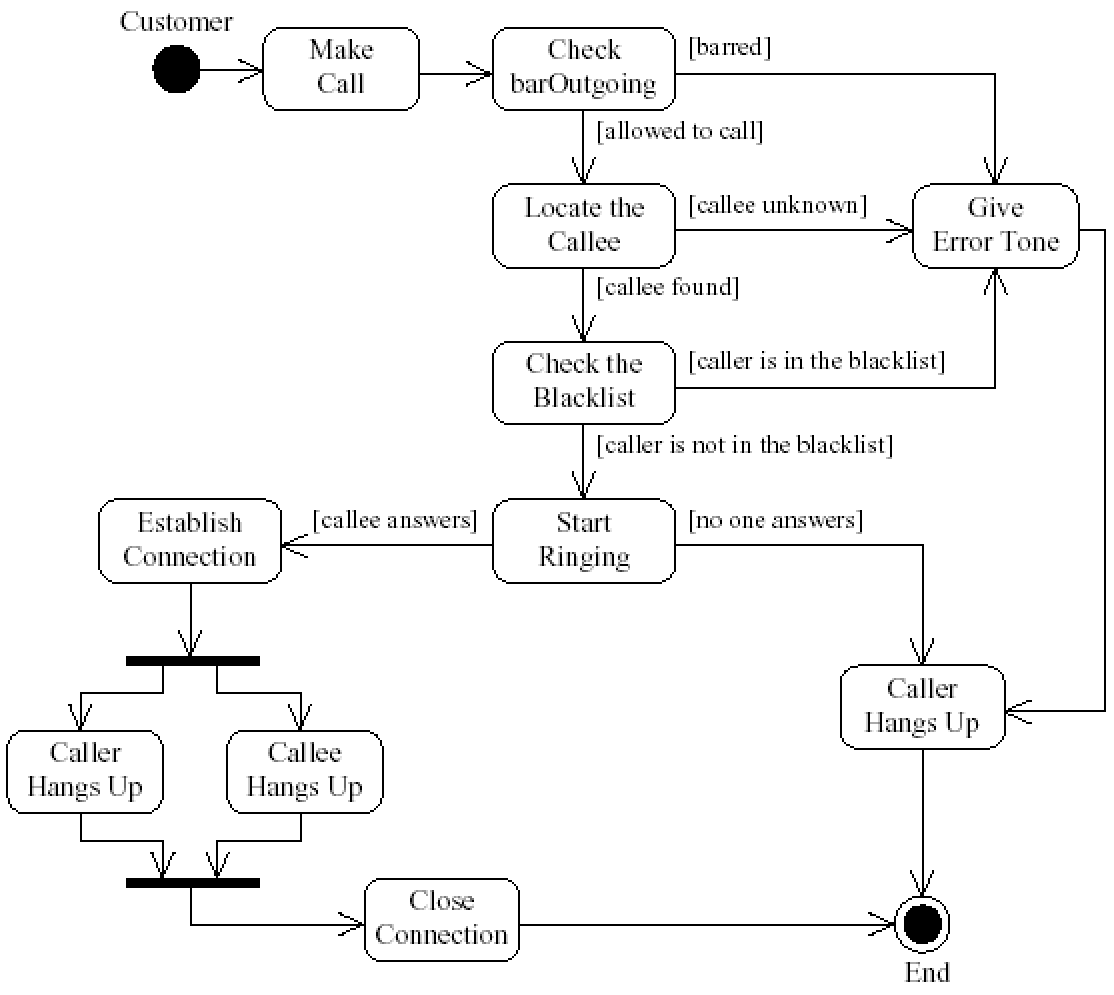


Figure 5-6- UML™ Activity Diagram for makeCall

5.1.3 Given BT Simulation Variables

The SimML tool used the random number variables shown below. The approximate values are culled from the textual reporting of the model, Table 5-1. The lookupTime is identified as the dependent variable. In many cases the units were not provided with the data and some reverse engineering was necessary to deduce the units.

Table 5-1 Variables used in the makeCall simulation

Name	**Type**	**Explanation**	**Approx. Values**
interArr	Exponential	The inter arrival time of the calls	1000 calls/sec
lookupTime	Exponential	The time taken by the Name Server to lookup for call identities	**Dependent Variable** ~ .0002 ms avg
readTime	Exponential	The time taken to perform the `barOutgoing` flag evaluation	.00023 ms avg
searchTime	Exponential	The time taken to check the `blacklist`	.0057 ms avg
localDelay	Exponential	The network latency for local call objects	~ 0
lanDelay	Exponential	The network latency for LAN call objects	1 ms
wanDelay	Exponential	The network latency for WAN call objects	50 ms
rndCallGen	Uniform	Used for randomly generating the local/LAN/WAN call types	
Run time	integer	Duration of simulation	100,000 ms

5.1.4 Given UML™ SimML Results

The simulation results presented in the Table 5-2 below are from Arief (2001, p. 135) who found that in order to satisfy the performance requirements for the WAN calls on average in 500 ms, the upper limit of the lookupTime had to be 19.91 ms; in the system dynamics model and simulation, this compares to 20 ms and 455 ms respectively.

Table 5-2 SimML results of the makeCall operation

lookup Time	average times			all calls		
	local	LAN	WAN	done	time	avgTime
20.00	523.80	526.02	719.90	299603	1.77E8	589.20
19.99	499.13	501.38	695.36	299748	1.69E8	564.58
19.98	474.39	476.76	670.77	299889	1.62E8	539.93
19.97	449.67	452.12	646.22	300053	1.55E8	515.29
19.96	424.88	427.49	621.68	300223	1.47E8	490.65
19.95	400.05	402.79	597.00	300353	1.40E8	465.91
19.94	375.25	378.10	572.33	300511	1.33E8	441.19
19.93	350.44	353.33	547.61	300650	1.25E8	416.42
19.92	325.60	328.58	522.87	300804	1.18E8	391.64
19.91	300.71	303.76	***498.13***	300945	1.10E8	366.83
19.90	275.83	278.94	473.33	301091	1.03E8	341.99

5.1.5 System Dynamic Stock and Flow Model

The System Dynamics Vensim stock and flow model, Figure 5-7, is based on the preceding UML™ diagrams that were used for the SimML model.

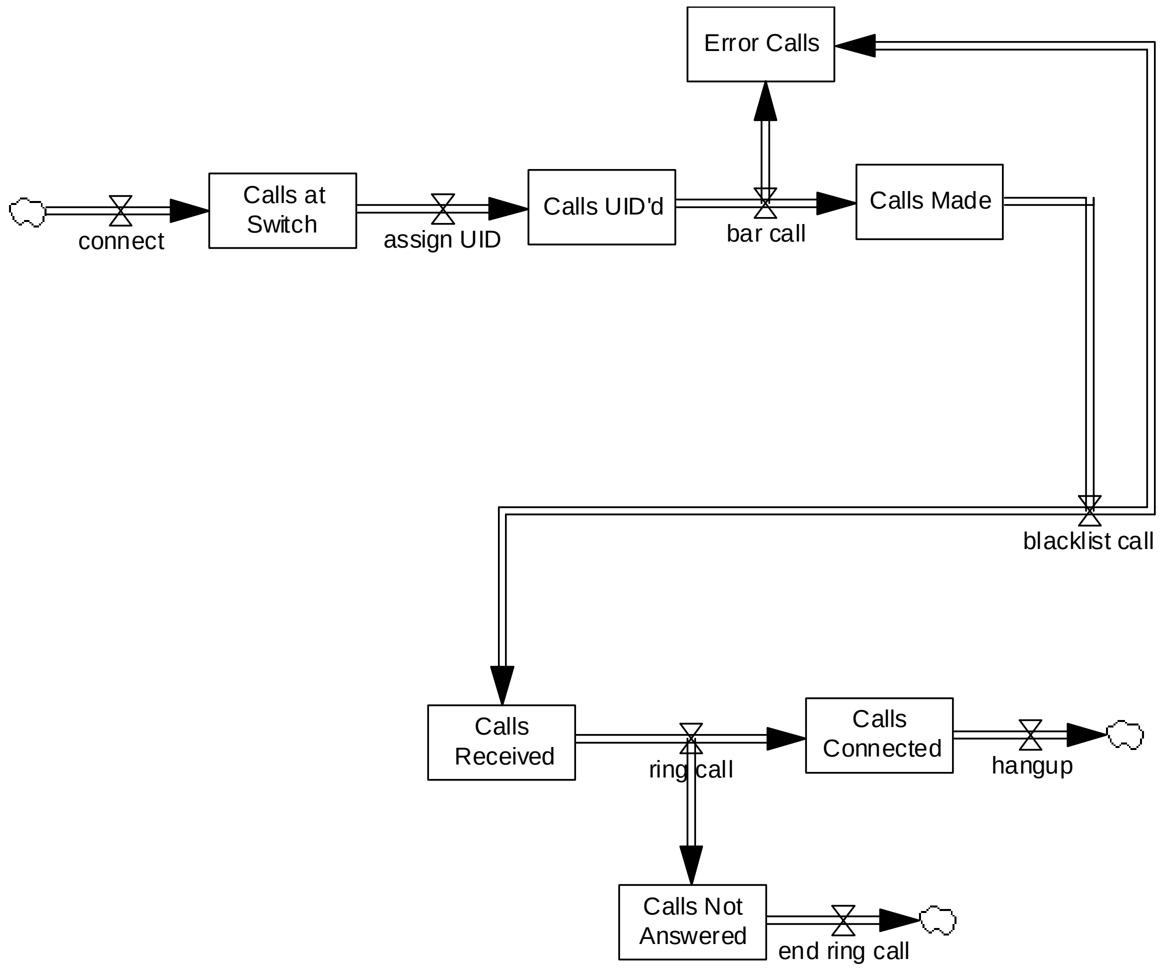


Figure 5-7 SD Stock and Flow Diagram based on UML™ Diagrams

For simplicity, the generalized stock and flow model above is shown without its auxiliary parameters. It is the foundation structure for the System Dynamics simulation model.

In order to complete the model, the auxiliary parameters were added as shown below, Figure 5-8. The auxiliary parameters add the behavior to the structure of the stock and flow model presented above. Additionally, the structure follows the behavior of a typical first-order material delay pipeline (Sterman 2000).

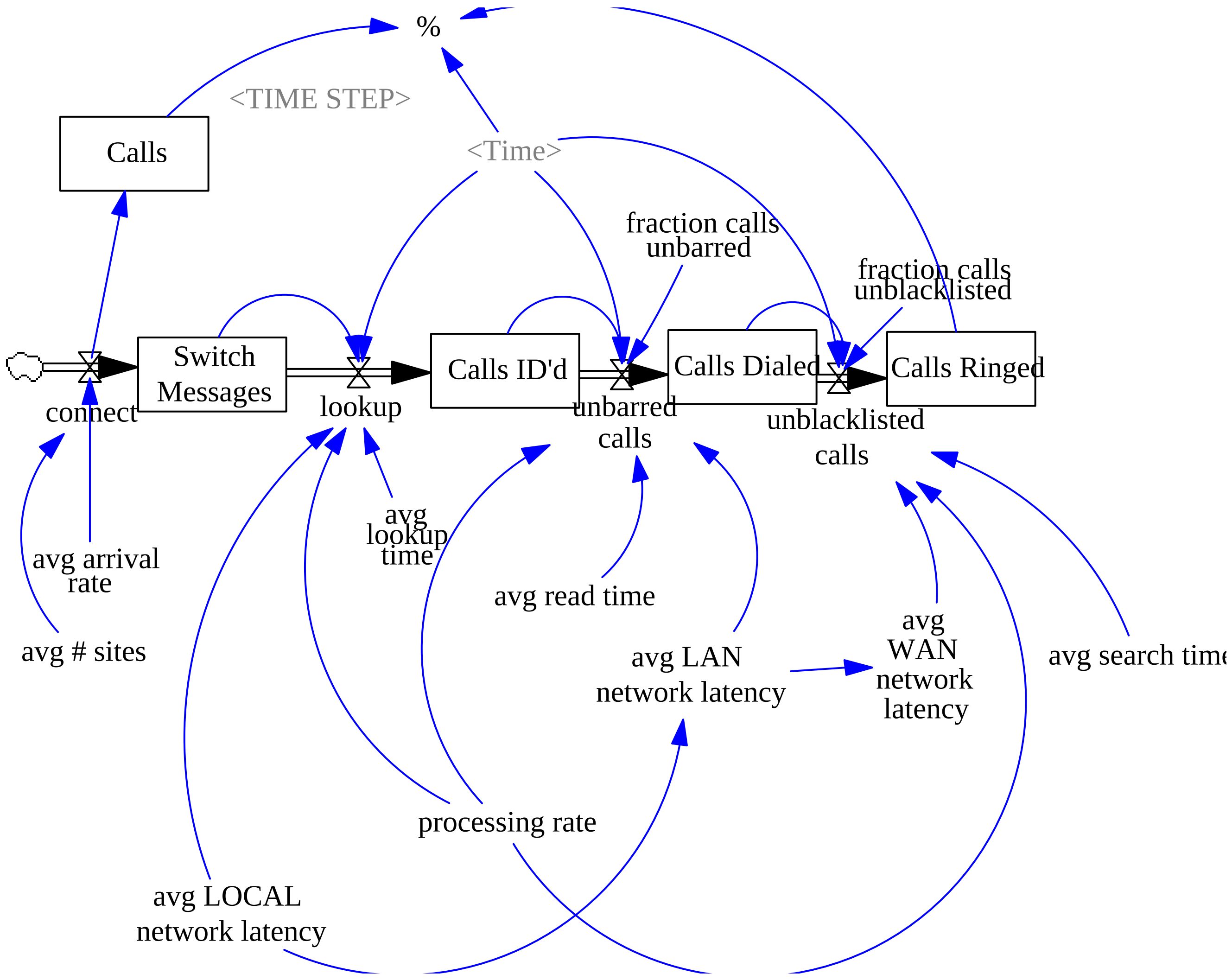


Figure 5-8 SD Stock and Flow Diagram with Auxiliary Parameters based on UML™ Diagrams

The "analytic knife" was applied to the generalized model based on reverse engineering of the Arief (2001) model material. For instance, the model above neither processes error conditions nor considers post "ring" activities such as calls connected. Additionally, even though the discussion of the Arief (2001) model includes processes for barring and blacklisting calls, no evidence of their inclusion in the model results was found. Lastly, although the Arief (2001) model simulated Local, LAN and WAN activities, the Arief

(2001) decision as to performance satisfaction was based on the WAN result alone. Therefore, the System Dynamics model concentrates only on the WAN activities.

Three sets of data were generated using the System Dynamics model. The first attempts to duplicate, roughly, the results of the SimML model as a reference set. Within this set of simulations the sensitivity of the performance requirements are examined as a function of the average lookup times.

The graphs that follow below show the curves labeled by the rate of calls and the average lookup rate (scientific notation). The graph scale is set to display the performance at 500ms, 5000ms (vertical line at between unit point) and 10,000ms over the course of the 100,000ms data run.

5.1.5.1 System Dynamic Model Comparable Results with Various Average Lookups

The following results are for the system dynamics model where the average lookup rate is run at three points: .002, .0002, and .00002 ms/call. Very little sensitivity to average lookup time is shown with regard to meeting the performance requirements, Figure 5-9. Each of the simulation curves generally met the performance requirements at the 5000ms and 10,000ms points for the nominal value of 20ms (average value lookup .0002ms/call). None of the simulation runs in this set meet the requirement for 90% processing at the 500ms point; the network latencies seem to prevent that occurrence. With the slowest average lookup, there is an accumulation of calls at the switch that may problematic depending on memory capacity, Figure 5-10. Calls identified, dialed, and ringed are nominal, Figure 5-11, Figure 5-12 and Figure 5-13, respectively.

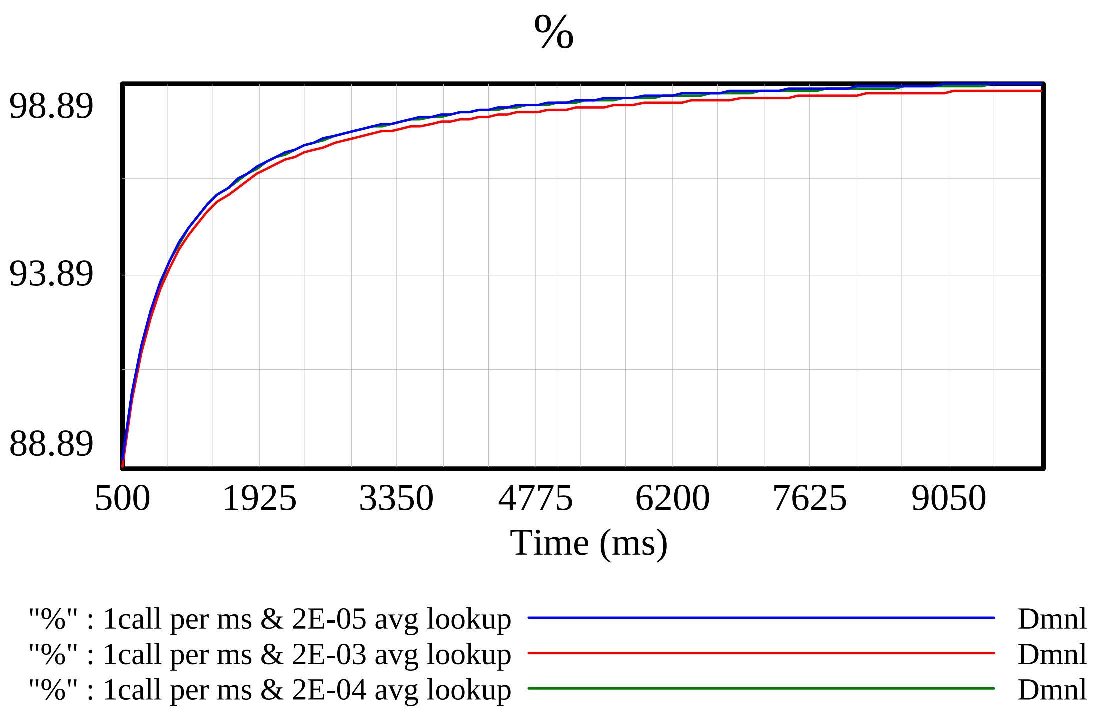


Figure 5-9 SD Model Percentage Performance with various average Lookups

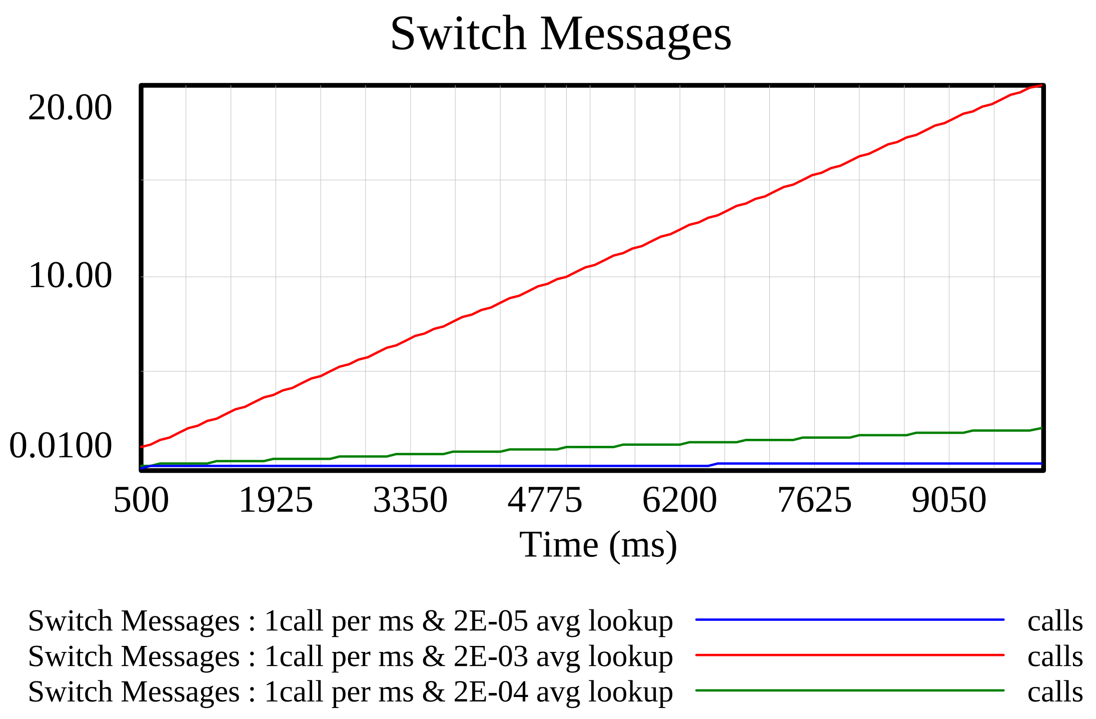
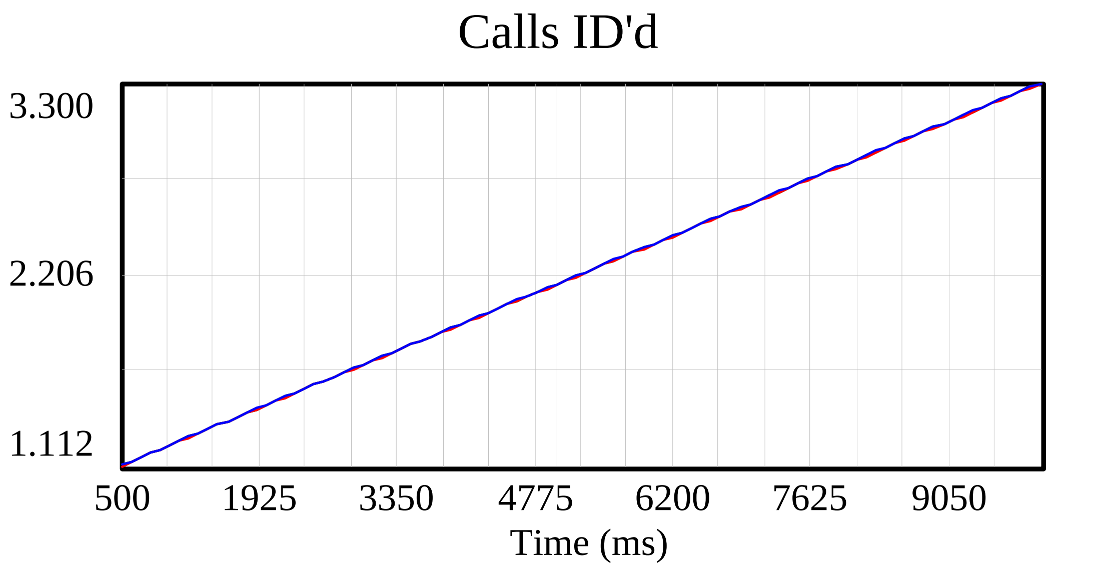
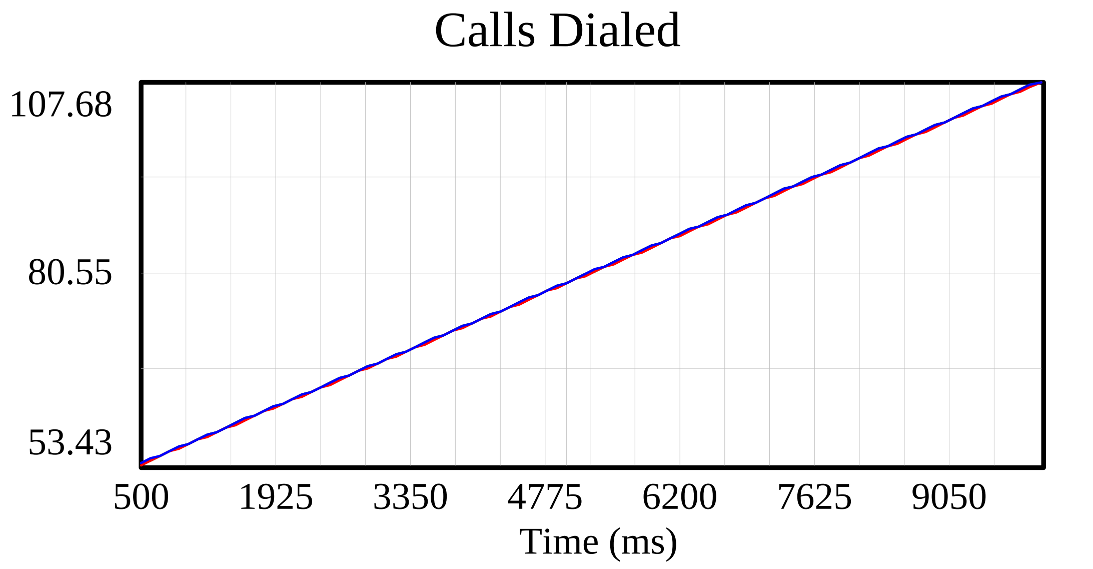


Figure 5-10 SD Model Switch Messages accumulations with various average Lookups



Calls ID'd : 1call per ms & 2E-05 avg lookup	🟦	calls
Calls ID'd : 1call per ms & 2E-03 avg lookup	🟥	calls
Calls ID'd : 1call per ms & 2E-04 avg lookup	🟩	calls

Figure 5-11 SD Model Calls ID’d accumulations with various average Lookup



Calls Dialed : 1call per ms & 2E-05 avg lookup	🟦	calls
Calls Dialed : 1call per ms & 2E-03 avg lookup	🟥	calls
Calls Dialed : 1call per ms & 2E-04 avg lookup	🟩	calls

Figure 5-12 SD Model Calls Dialed accumulations with various average Lookups

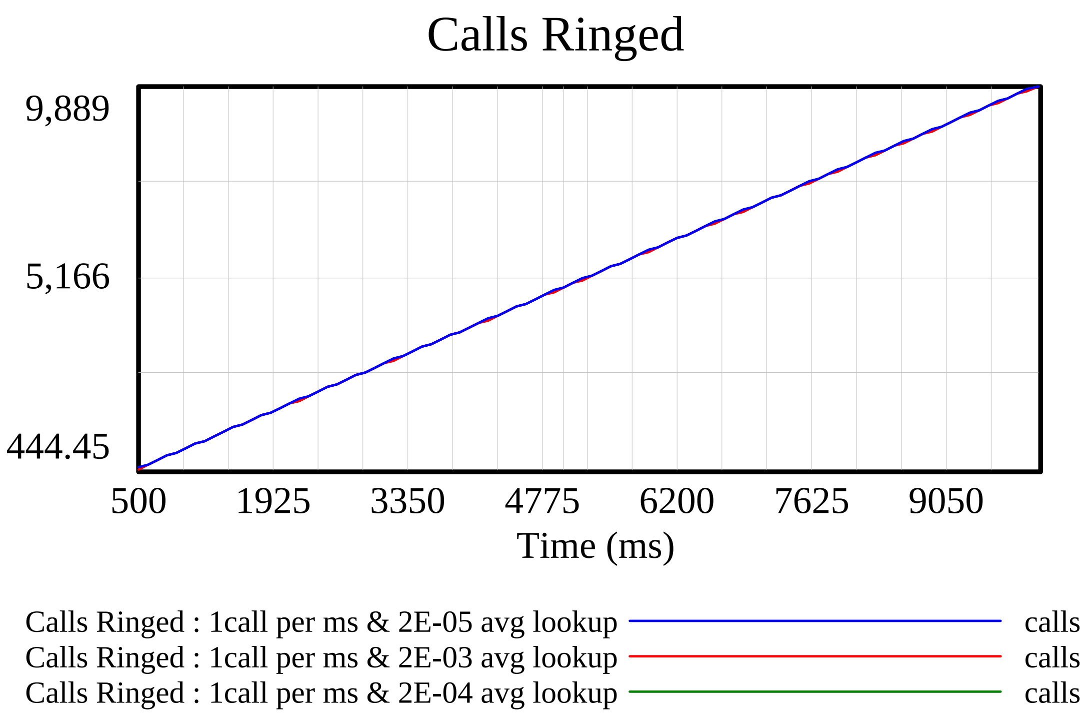


Figure 5-13 SD Model Calls Ringed accumulations with various average Lookups

5.1.5.2 System Dynamic Model Comparable Results with Nominal Average Lookup and 3X Input Rate

The following results are for the system dynamics model where the nominal average lookup rate (.0002 ms/call) is run with a 3x input rate (3 calls/ms). Very little sensitivity to the 3x input rate is shown with regard to meeting the performance requirements, Figure 5-14. As with the previous run, none of the simulation runs in this set meet the requirement for 90% processing at the 500ms point; the network latencies seem to prevent that occurrence. With a 3x input rate and the slowest average lookup, there is an accumulation of calls at the switch that may problematic depending on memory capacity, Figure 5-15. The 3x input rate also causes an accumulation at the Calls Identified stock; this is probably a result of the LAN latency and read rate, Figure 5-16. Similarly, there is an accumulation with the 3x input rate at the Calls Dialed stock, probably as a result of the WAN latency and search rate, Figure 5-17. Calls ringed increase assuming the system does not break, Figure 5-18.

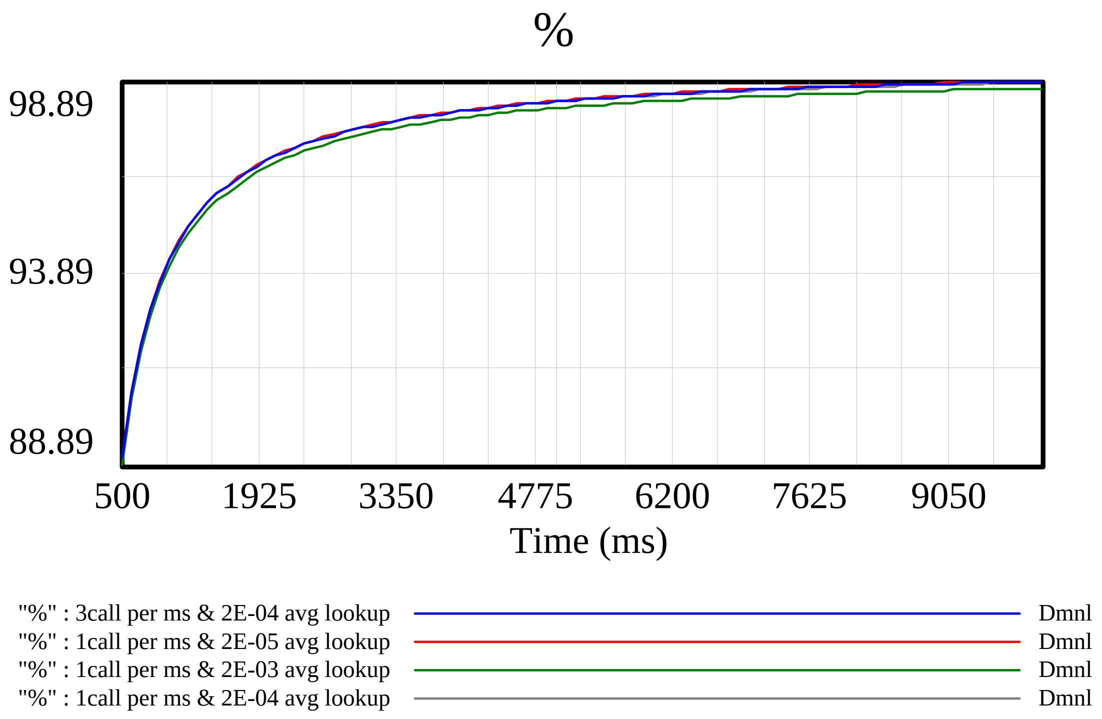


Figure 5-14 SD Model Percentage 3x Performance with various average Lookups

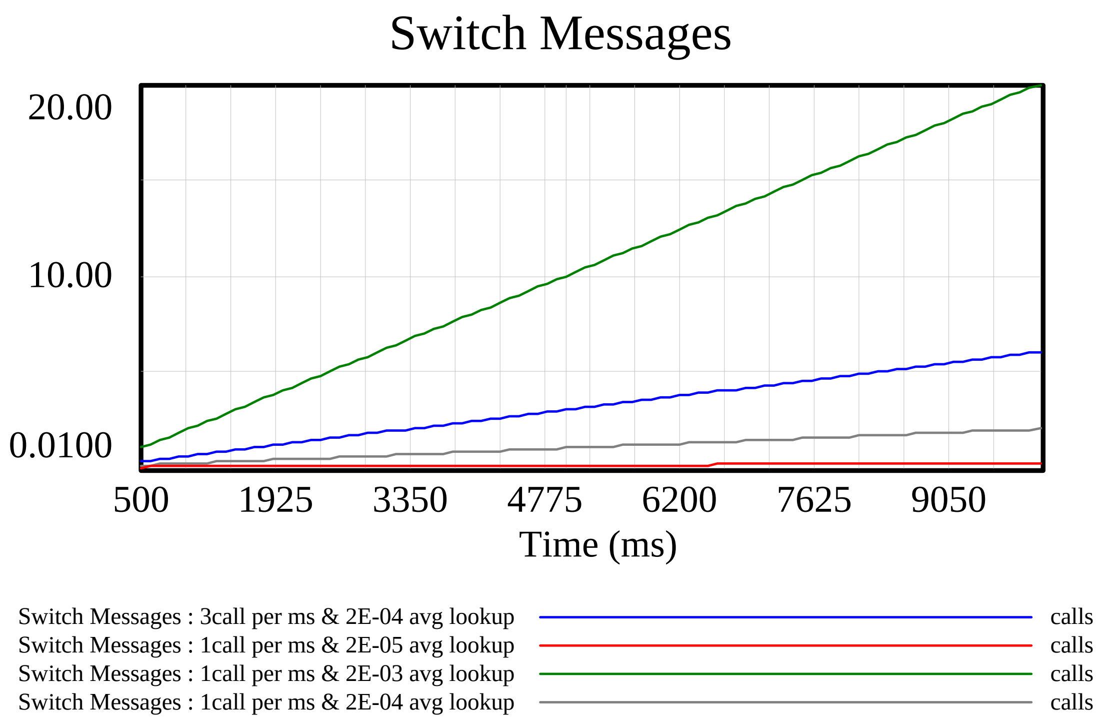


Figure 5-15 SD Model 3x Switch Messages accumulations with various average Lookups

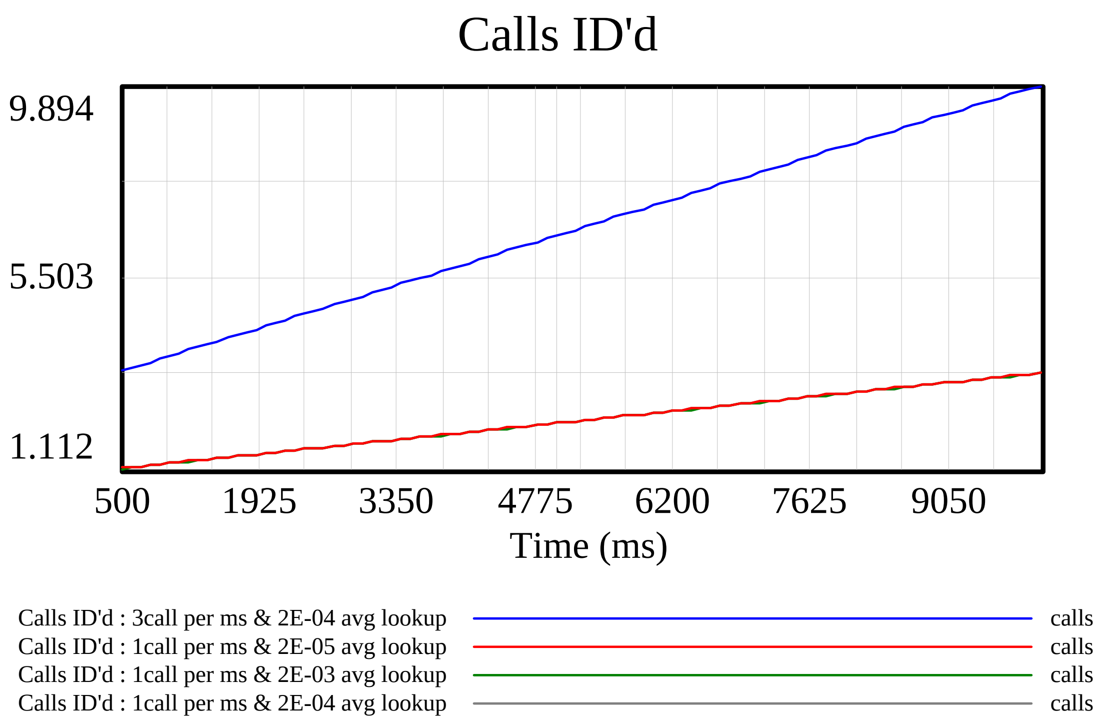


Figure 5-16 SD Model 3x Calls ID'd accumulations with various average Lookups

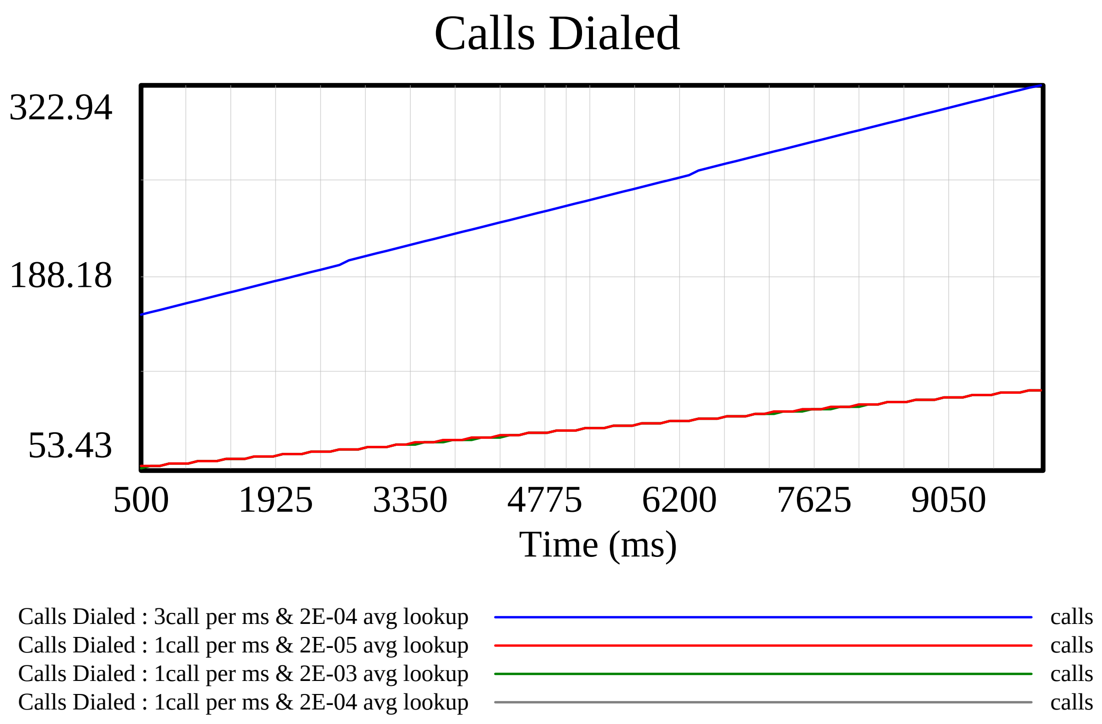


Figure 5-17 SD Model 3x Calls Dialed accumulations with various average Lookups

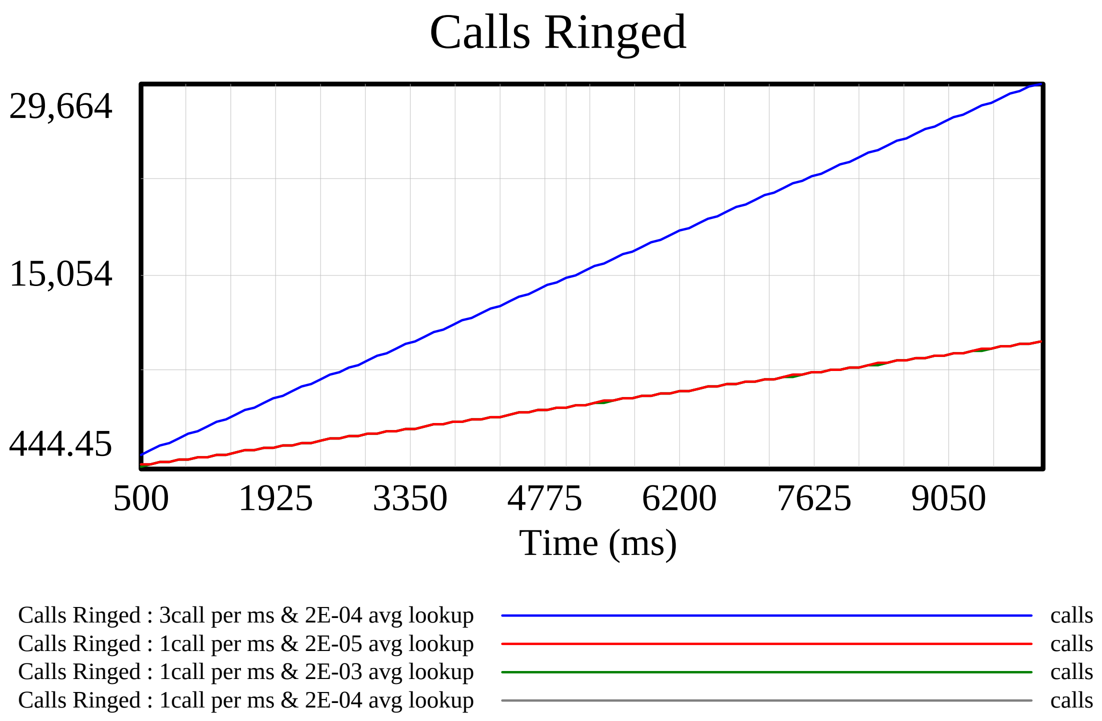


Figure 5-18 SD Model 3x Calls Ringed accumulations with various average Lookups

5.1.5.3 System Dynamic Model Comparable Results with Nominal Average Lookup and 30X Input Rate

The following results are for the system dynamics model where the nominal average lookup rate (.0002 ms/call) is run with a 30x input rate (30 calls/ms), Figure 5-19. Very little sensitivity to the 30x input rate is shown with regard to meeting the performance requirements. As with the previous run, none of the simulation runs in this set meet the requirement for 90% processing at the 500ms point; the network latencies seem to prevent that occurrence. With the a 30x input rate and the slowest average lookup, there is an accumulation of calls at the switch that may problematic depending on memory capacity, Figure 5-20. The 30x input rate also causes an accumulation at the Calls Identified stock; this is probably a result of the LAN latency and read rate, Figure 5-21. Similarly, there is an accumulation with the 30x input rate at the Calls Dialed stock, probably as a result of the WAN latency and search rate, Figure 5-22. Calls ringed increases as expected, assuming the system does not break, Figure 5-23.

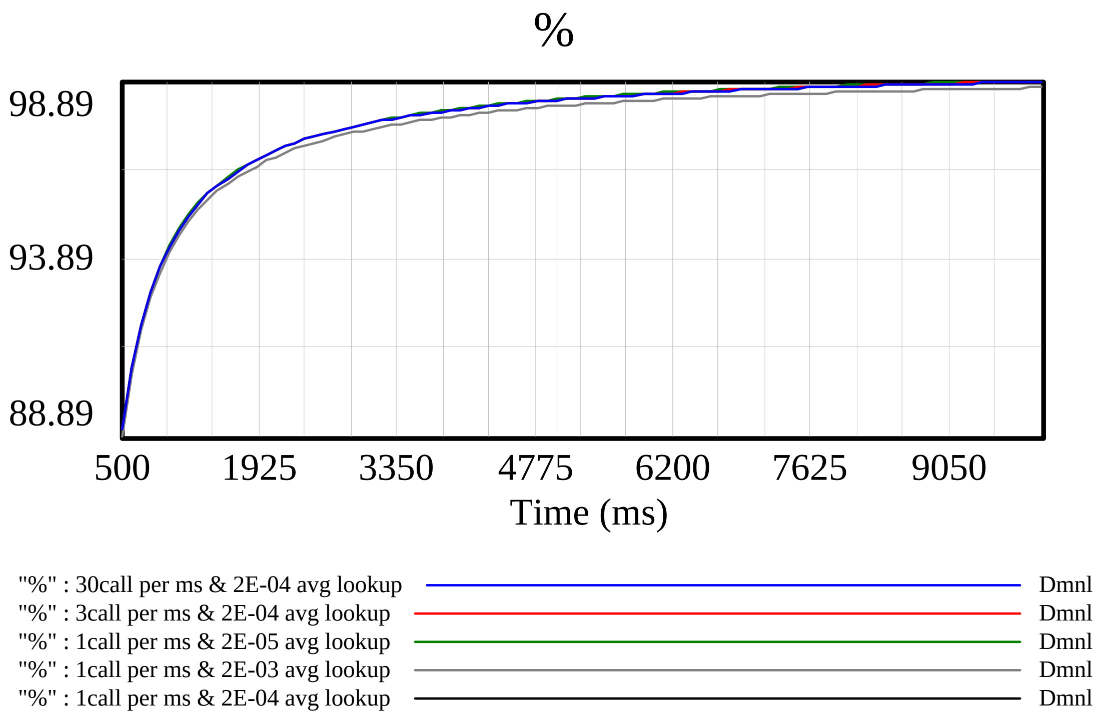


Figure 5-19 SD Model Percentage 30x Performance with various average Lookups

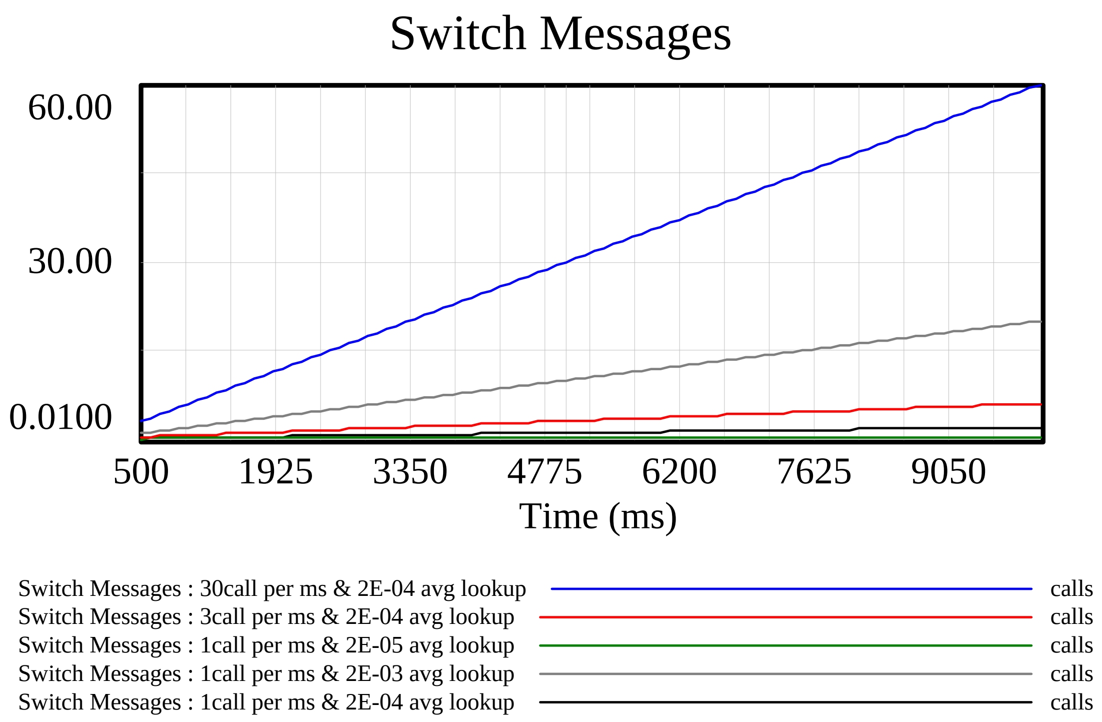


Figure 5-20 SD Model 30x Switch Messages accumulations with various average Lookups

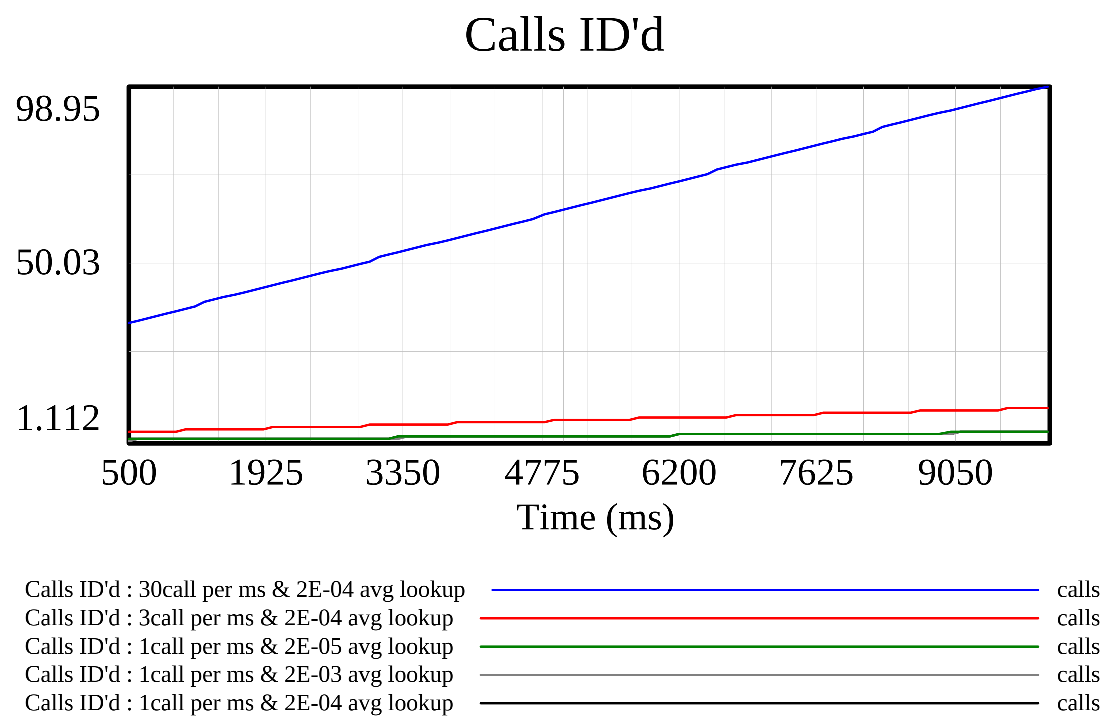


Figure 5-21 SD Model 30x Calls ID'd accumulations with various average Lookups

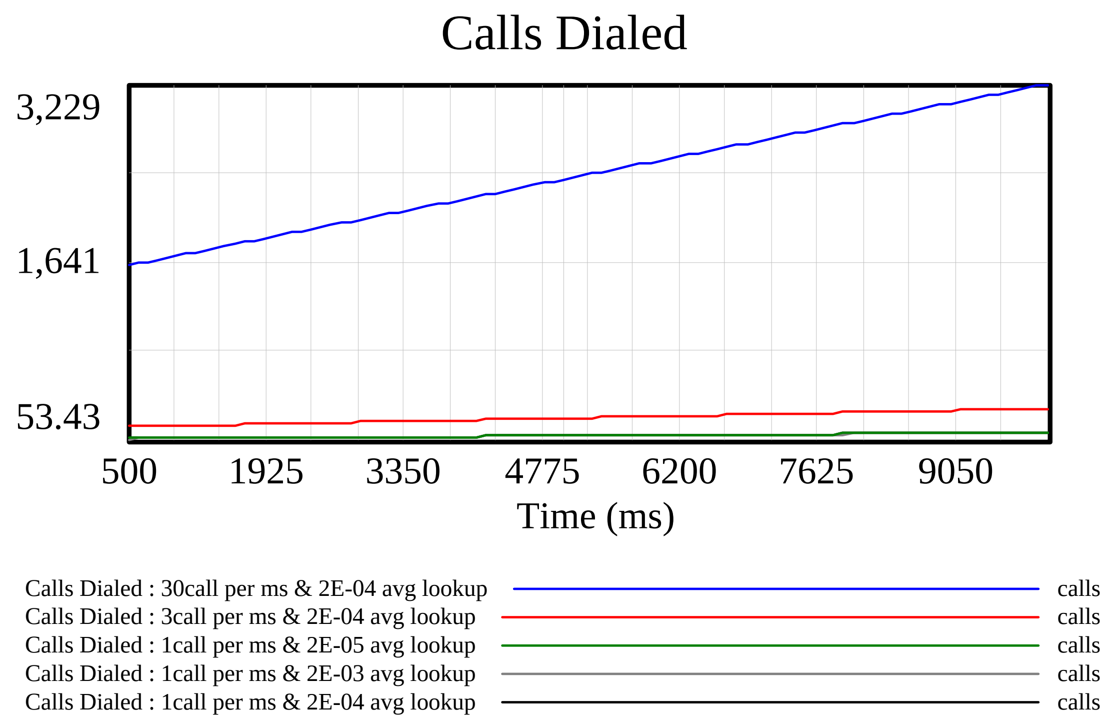


Figure 5-22 SD Model 30x Calls Dialed accumulations with various average Lookups

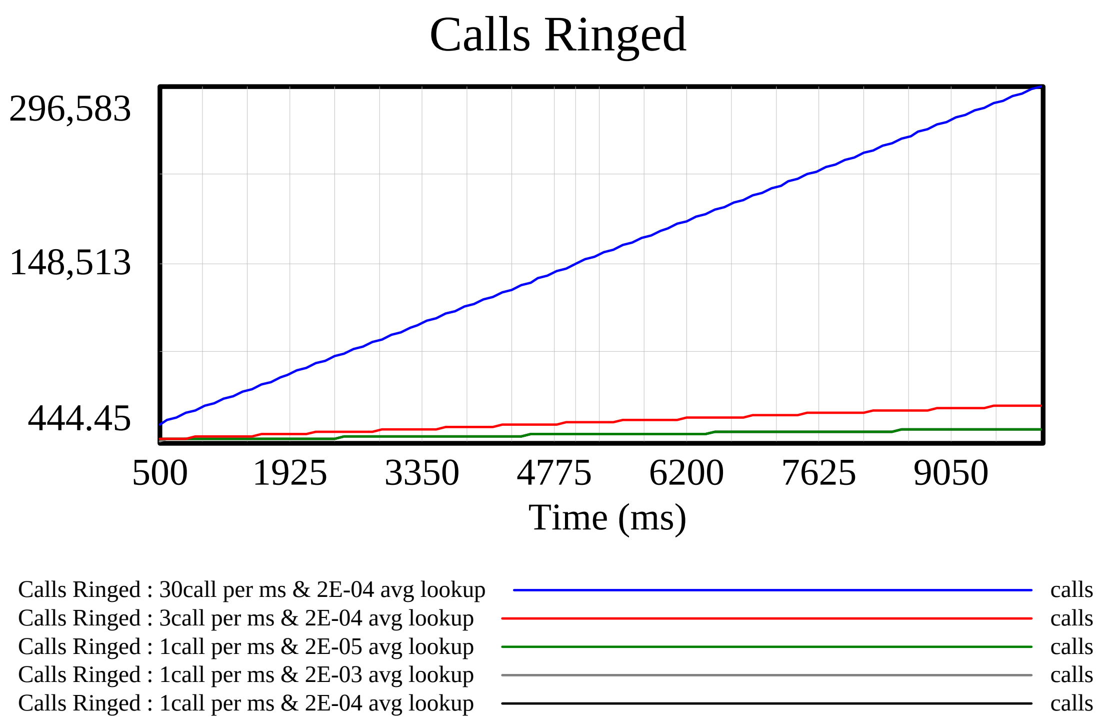


Figure 5-23 SD Model 30x Calls Ringed accumulations with various average Lookups

6 Major Findings and Significance

The major findings and significance below resulted from the data analysis. Each major finding and significance presented has a header topic followed by the statement of the finding and its significance.

6.1 UML and System Dynamics Models

System Dynamics models expressed in VENSIM do not solve the problem of having to manually translate static UML™ diagrams into executable language as SimML automated this step. However, the analysis does show that System Dynamics models using VENSIM have the power to support the concept of simulating information system behavior based on UML descriptions of the system. Additionally, leveraging SD's VENSIM innate capability to support the creation of simulation models from static UML™ diagrams was apparent and added benefits that were not apparent in SimML, e.g., a picture of the structure of the model and its organic flow.

There is a significant opportunity for System Dynamics using VENSIM to support the information system domain if a seamless interface is made between UML™ and VENSIM models, possibly using XML, XMI, or a new SimML parser. Many man-years of effort would be saved using System Dynamics VENSIM models as opposed to the currently charted course of a request for proposal to create a “new” language to make static UML™ diagrams into dynamic models. This would also prevent the tendency toward “language bloat” for UML™ by keeping its focus on the artifacts it produces so well today and not adding a new simulation language set of semantics and syntax.

6.2 Integrated View of Structure and Behavior

The study finds that System Dynamics using VENSIM shows an integrated view of the structure and behavior of the subject BT IN model that neither the given UML™ models nor the SimML language presented. It is significant that SD and VENSIM have the capability to show the interdependence of structure and behavior concurrently using stock and flow diagrams and simulation results.

6.3 Boundaries of the Reference Model

Several parameters given in the customer problem statement such as host memory, message, and processing speed were not found in the SimML model. Similarly, call barring, call blacklisting, and error handling were not modeled.

Host memory, message size, and processing speed may have a significant impact on system throughput. The dynamics of these parameters may have a significant influence on the structure and behavior of the implemented information system and should be part of the simulation to ensure they will not adversely affect the decision process. Limiting the modeling simulation boundary to exclude what happens post “ring” may be reasonable, but handling the consequences of call-barring and call-blacklisting could have significant impact on the performance.

6.4 Sensitivity Analysis

The SimML model included a very limited sensitivity analysis. The System Dynamics VENSIM model readily lent itself to sensitivity analysis.

Without looking at the model structure and behavior with regard to sensitivity to stress, a significant finding may be overlooked. In the case of this study, not looking at the sensitivity of the model to increased input volume may be a fatal flaw. The input volume was given as 3x10E3 to 3X10E6. The simulation was confined to the 1X10E3 magnitude case. When increasing the input volume using the System Dynamics VENSIM model, an accumulation of messages was observed at the switch waiting for the “lookup ID” to be completed. It is very likely that the switch will not have the memory capacity to sustain an accumulation of this nature; further investigation is warranted.

7 Conclusions

The data analysis and findings support the problem statement that an information system described using static UML™ is translatable into a System Dynamics VENSIM simulation model. The rough comparison of the System Dynamics VENSIM model simulation results to those from the SimML case study show comparable results. Using System Dynamics to model information system architectures and software design before implementation has potentially significant cost benefits to an industry-wide information system problem of cost and schedule overruns.

For credibility with the system engineering information system domain, System Dynamics will need to build interfaces into its simulation languages that will accept UML™ artifacts, perhaps documented as XML or other standards. A seamless interface between UML™ and SD simulation languages is essential to be responsive to the system engineering information system domain; otherwise, new and specialized languages will be procured that will be duplicative and competitive to the System Dynamics simulation languages.

8 References

Arief, L. B. (2001). A Framework for Supporting Automatic Simulation Generation from Design. Unpublished Ph.D. dissertation, University of Newcastle Upon Tyne, from http://homepages.cs.ncl.ac.uk/l.b.arief/home.formal/Papers/PhDThesis.pdf

Bütow, M., Mestern, M., Schapiro, C., & Kritzinger, P. S. (1996). Performance Modelling with Formal Specification Language SDL. Retrieved December 10, 2003, from http://people.cs.uct.ac.za/~psk/

Caulfield, C.W., & Maj, S.P. (2001). A case for systems thinking and system dynamics. IEEE, 2793-2798.

De Wit, N.D. (2003). Software Performance Engineering with UML-RT and UML 2.0. Retrieved December 10, 2003, from http://people.cs.uct.ac.za/~ndewet/Performance_with_UML-RT.pdf

Dori, D. (2002). Why significant uml change is unlikely. Communications of the ACM, 45, No. 11, 82-85.

Forrester, J. W. (1961). Industrial Dynamics. Portland: Productivity Press.

Kobryn, C. (1998). Modeling enterprise software architectures using uml. IEEE, 25-34.

Kortright, E. V. (1997). Modeling and simulation with uml and java. IEEE, 43-48.

Lenehan, N. (2003). UML 2.0 Standard Officially Adopted at OMG Technical Meeting in Paris. Retrieved December 9, 2003, from http://www.omg.org/news/releases/pr2003/6-12-032.htm

Miller, J., (2002). What uml should be. Communications of the ACM, 45, No. 11, 67-69.

Selic, B., Ramackers, G., & Kobryn, C. (2002). Evolution, not revolution. Communications of the ACM, 45, No. 11, 70-72.

Sterman, J.D., (2000). Business Dynamics. Boston: Irwin McGRaw-Hill.

Towill, D. R. (1993a). System dynamics-background, methodology, and applications, part 1. Computing & Control Engineering Journal, October, 201-208

Towill, D. R. (1993b). System dynamics-background, methodology, and applications, part 2. Computing & Control Engineering Journal, December, 261-268

Wiklund, A. (2003). Using Ericsson NorArc's frameworks as test bed for dynamic change behavior based on CompositeStates. Unpublished M.S. thesis, Agder University College.

Back to the Top