Optimizing Costs of Web-based Modeling and Simulation*

Dhananjai M. Rao,
Harold W. Carter, and *Philip A. Wilsey*
Experimental Computing Laboratory
Dept. of ECECS, PO Box 210030, Cincinnati, OH 45221–0030.
(email: dmadhava@ececs.uc.edu, hal.carter@uc.edu, phil.wilsey@uc.edu)

Abstract

Web-based simulations are performed by utilizing the resources of the Word Wide Web (WWW) such as proprietary components*/models developed by third party modelers/manufacturers and web-based computational infrastructures (or compute servers). Access to such web-based resources, third party resources in particular, is usually circumscribed by a variety of pricing schemes. Therefore, optimal use of resources plays a critical role in minimizing the overall costs of web-based modeling and simulation which is directly dependent on the size of the model* i.e., *the total number of components constituting the model. Consequently, component aggregation and de-aggregation techniques that can be used to* statically *(before simulation) as well as* dynamically *(during simulation) vary the number of components constituting a model, have been developed. The techniques enable a range of tradeoffs between several modeling and simulation related parameters – thereby optimizing the resource consumption and overall costs. This paper presents a detailed discussion of the component aggregation and de-aggregation techniques along with the issues involved in implementing them in a Web-based Environment for Systems Engineering (*WESE*). Our studies indicate that these techniques provide an effective means to optimize the overall costs of web-based modeling and simulation.*

1 Introduction

Web-based simulation is an effective solution to address a number of issues exacerbating modeling, simulation, and analysis [1, 5, 8, 9, 13]. Due to its effectiveness, web-based simulations are steadily growing in importance. In web-based simulation environments, the models for simulations are usually developed using component based modeling techniques because of their advantages [6, 8, 9]. In a component based model, a system is represented as a set of interconnected components [9, 8]. A *component* is a well defined software entity which is viewed as a "black box", *i.e.,* only its interface is of interest and not its implementation. A component may also be specified as a set of sub-components and is called a *module*. During simulation, each *atomic* component is associated with a specific software module called a simulation object (or logical process (LP)) that implements its behavior and functionality. The simulation objects (or LPs) could be those implemented by the modeler or those obtained via the WWW from other third party model developers or vendors [8, 9]. The models may be simulated using a set of third party workstations (or *simulation servers*).

Given the accelerated growth in need and use of web-based simulations, soon, if not already, the third party resources will be offered as value added services based on different pricing schemes. Hence, optimal use of the resources plays a critical role in minimizing the overall costs of web-based modeling and simulation. The resource consumption of a model is directly related to its size; *i.e.,* the total number of components involved in the model [8, 9, 10]. Therefore, by decreasing (or increasing) the total number of components involved in a model, the resource requirements for modeling and simulation can be reduced (or increased).

The total number of components participating in a simulation can be decreased (or increased) by substituting a set of related components with a smaller or larger, but *functionally equivalent* set of components such that the desired characteristics of the model are not altered [11]. The model transformation wherein a set of components is replaced by *a functionally equivalent* component is called *Component Aggregation* (CA) and its inverse transformation is called *Component De-Aggregation* (CDA). These transformations are synonymous to varying the resolution or the level of abstraction of a model. CA and CDA can be used to vary the total number of components constituting a given model – thereby changing the resources requirements and the over-

*Support for this work was provided in part by the Advanced Research Projects Agency under contracts J–FBI–93–116 and DABT63–96–C–0055.

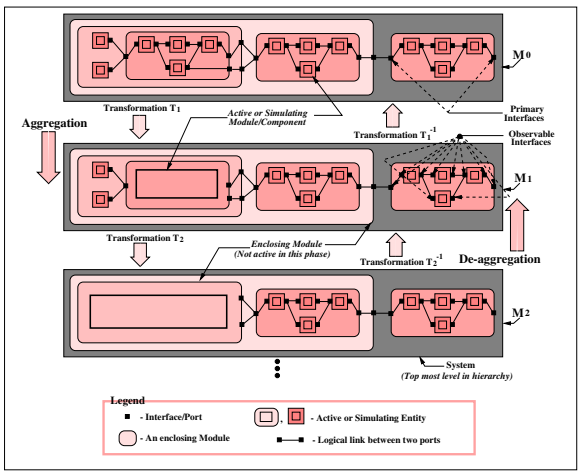


Figure 1. Overview of CA and CDA

all costs for modeling and simulation. The CA and CDA transformations achieve optimizations in costs by enabling a range of tradeoffs between several modeling and simulation related parameters such as modeling overheads, model observability, accuracy of simulation results, and simulation performance.

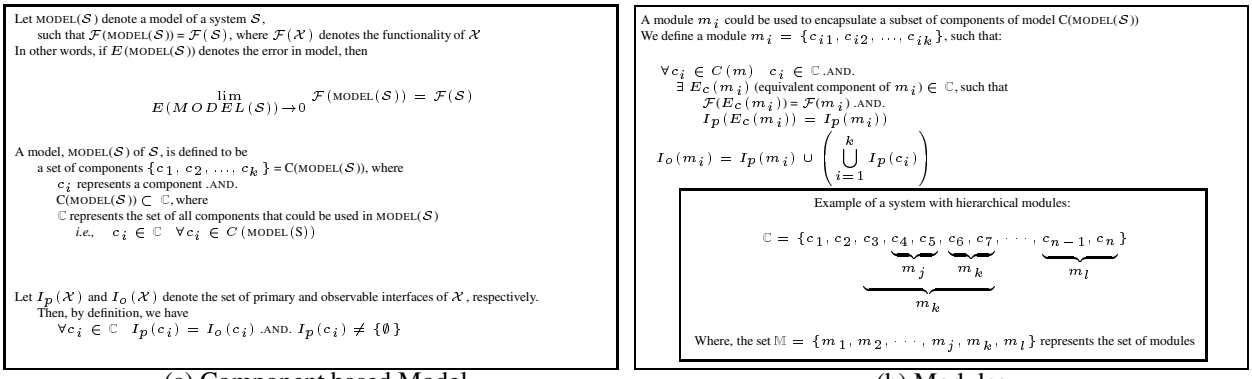
This paper presents a discussion of the CA and CDA techniques along with the issues involved in implementing them in a Web-based Environment for Systems Engineering (WESE). Section 2 presents a discussion of the CA and CDA techniques along with an analytical model of the transforms. Section 3 presents a brief overview of WESE along with the issues involved in the design and implementation of CA and CDA techniques in it. The experiments conducted to evaluate the effectiveness of CA and CDA along with the results obtained from the experiments are presented in Section 4. Section 5 presents some concluding remarks along with pointers to future work.

2 CA and CDA

In component based modeling techniques a model is specified as a set of interconnected *modules*. A *module* in turn consists of a set of sub-modules or *atomic* components. Figure 1 presents an example of a component based model. In such models, a set of related components can be replaced with a functionally equivalent component or vice versa, without altering the basic functionality of the model (as shown in Figure 1). The transformation in which the number of components reduces is called *Component Aggregation* (CA). The transformation in which the number of components increases is called *Component De-aggregation* (CDA). Figure 1 presents an overview of the CA and the CDA transformations applied to a hierarchical, component based model. Component aggregation and de-aggregation maybe performed *statically* or *dynamically*. *Static* transformations occur prior to commencement of simulation while *dynamic* transformations occur during the course of simulation. Earlier research activities related to static CA technique have been reported in the literature [2, 4, 14], to address issues related to performance of large scale simulations. On the other hand, dynamic CA and CDA not only encompass the utility of their static counterparts but also provide a number of additional advantages [11]. Further details about CA and CDA are available in the literature [11].

The study and analysis of the CA and CDA transformations is a complex task because it involves characterizing and analyzing the changes to a number of model related parameters, simulation related parameters, and overall cost of modeling and simulation. Some of the model related parameters of interest are cost of the model, observability, accuracy of the model, and level of abstraction. The simulation (parallel and distributed simulation in particular) related parameters include concurrency, number of discrete events that need to be processed, and overheads of the transformations. Consequently, numerous combinations of transforms need to be analyzed to identify the optimal configuration for a given simulation study. For example, if n unique transforms are possible, then 2^n combinations of the transforms may have to be analyzed to identify an optimal solution. Consequently, to ease effective use of the CA and CDA techniques a mathematical model of the transforms has been developed. It characterizes the changes to the modeling and simulation related parameters induced by each, unique transform and provides an algebra for studying combinations of transforms. The math model can be used to reduce the time complexity for analysis and exploration of optimal solutions from $\mathcal{O}(2^n)$ to $\mathcal{O}(n)$. The remainder of this section presents an overview of the analytical model for the CA and CDA transformations.

Figure 2 illustrates the basic definitions of a hierarchical, component based model. As show in Figure 2, a valid model of system $\mathcal{S}$, represented by MODEL($\mathcal{S}$), consists of a set of components ($\mathbb{C}$(MODEL($\mathcal{S}$))) such that the desired functionality of $\mathcal{S}$ (represented by $\mathbb{F}(\mathcal{S})$) is modeled by MODEL($\mathcal{S}$). The components constituting the model C(MODEL($\mathcal{S}$)) (*i.e.,* the *active* components) are a subset of all the components used to model the system ($C(\mathcal{S})$). The definition for a *module* is shown in Figure 2(b). As shown in Figure 2(b), a module is used to group a set of related components such that for each module (m_i) there exists a component (represented by $E_c(m_i)$) such that the components constituting a module can be substituted with $E_c(m_i)$ (equivalent component of m_i), without altering the functionality of the module. In other words, the functionality and the primary interface of a component (shown in Figure 1) must be equivalent to that of the module (*i.e.,* $\mathbb{F}(m_i) = \mathbb{F}(E_c(m_i))$ .AND. $I_p(m_i) = I_p(E_c(m_i))$). The



(a) Component based Model (b) Modules

Figure 2. The basic definitions

restriction that each module has an equivalent component is made to clearly define and restrict our domain of interest.

Having defined the set of modules such that each module has an equivalent component, we now proceed to define a set of transforms (T) (or operators) that map modules to their equivalent components and vice versa. The definition for a transform is shown in Figure 3. As shown in Figure 4(a), for each module (in the set $\mathbb{M}$, shown in Figure 2(b)) two distinct transforms can be defined. One transform (namely τ_i) substitutes the set of components associated with a given module ($C(m_i)$) with its equivalent component ($E_c(m_i)$). If the number of components encapsulated by the module is greater than one ($|C(m_i) > 1$) then this transforms results in CA (*i.e.,* two or more components is substituted by a single component). The other transform (namely τ_j) substitutes the equivalent component of a module ($E_c(m_i)$) with the corresponding set of components ($C(m_i)$). If the number of components contained by a module is greater than one ($|C(m_i)| > 1$) then this transform results in CDA (*i.e.,* one component is substituted by two or more components). It must be noted that the two transforms are inverse of each other (Figure 4(a)) and for every transform there exists a corresponding inverse transform. As shown in Figure 4(a), the transforms may result in NULL transforms (τ_ϕ). Such scenarios, wherein the transforms result in NULL transformations, could arise due to the hierarchical structure of modules. As illustrated by the definition (Figure 4(a)), these transforms are functionality preserving transforms *i.e.,* the overall functionality of the system does not change (*i.e.,* $\mathcal{F}$(MODEL($\mathcal{S}$)) = $\mathcal{F}(\tau_i(\text{MODEL}(\mathcal{S})))$). The set of valid transforms on a model is represented by the set T.

Having defined a CA and CDA transformation, we now proceed to define a formal model for a sequence of transformations on a model. Figure 4(b) defines an algebra for a sequence of transformations on a model. As shown in Figure 4(b), a sequence of transformations forms a well understood algebraic system called a *Group* [12] that has its foundations in discrete mathematics. Having shown that the sequence of transforms constitute a *Group*, we can further utilize operations and tools based on discrete mathematics to reason and verify different characteristics and attributes of the transformations. The algebraic model of the changes induced by the transforms on the modeling and simulation parameters are shown in Figure 5. As shown in Figure 5, the modeling parameters such as modeling costs, observability of simulation results, and level of abstraction are directly or indirectly dependent on the number of components constituting the model. The simulation time (or cost) is a complex function and is directly dependent on the number of discrete events exchanged in the simulation. It is also dependent on the partitioning of the components of a model, which influences the concurrency, communication, and synchronization costs in a parallel simulation (these costs are

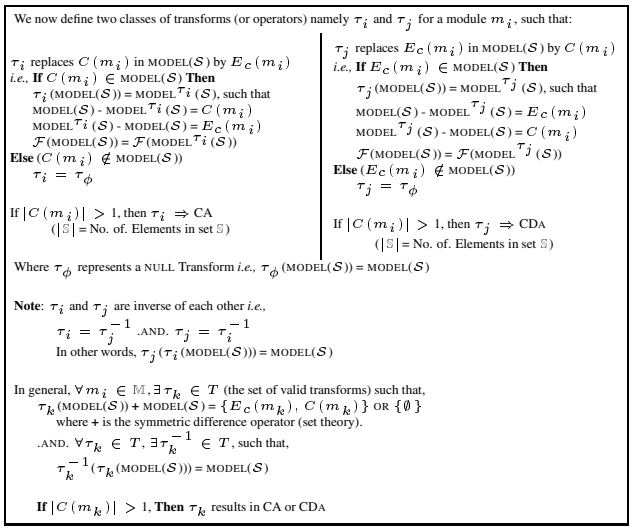


Figure 3. Definition of CA and CDA

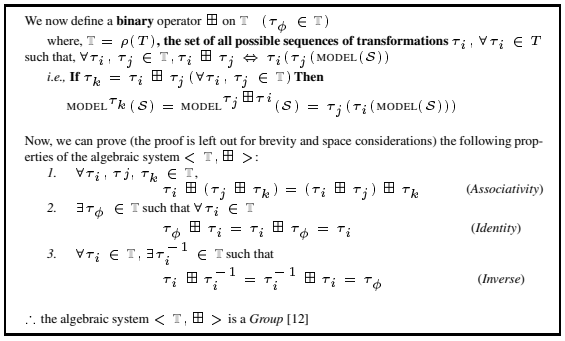


Figure 4. Algebra for CA and CDA

not present in sequential simulations). It must be noted that the modeling and simulation costs are independent of each other. In other words, although the number of components may decrease (and consequently the cost of modeling may decrease) the time taken to simulate the model may increase or vice versa.

Figure 5 also illustrates the changes induced by a sequence of transforms on the modeling and simulation related parameters. As shown in the figure, given the changes induced by a *basic* transform ($\forall \tau_i \in T$), the changes caused due to a sequence of transforms ($\forall \tau_k \in \mathbb{T}$) can be estimated using the algebra. Estimation of the changes induced by combinations or sequences of transforms is possible because of the way the transformations and the algebra is defined. On the other hand, it also places a few restrictions – for instance, the transforms are not necessarily commutative (*i.e.,* $\forall \tau_i, \tau_j \in \mathbb{T}, \tau_i \boxplus \tau_j \neq \tau_j \boxplus \tau_i$). The algebra has been utilized to implemented support for estimating changes induced by a sequence of transforms, in WESE. The issues involved in the design and implementation of support for estimating the changes induced by CA and CDA are presented in the following section.

3 WESE

The Web-based Environment for Systems Engineering (WESE) was developed to ease modeling and simulation of systems via the WWW [8]. WESE provides a component based modeling language, a framework for developing a web-based repository of components, and the infrastructure for distributed simulation. An overview of WESE is shown in Figure 6. As shown in Figure 6, WESE provides a Hyper Text Markup Language (HTML) interface and a text based frontend that can be used to interact with the WESE Server. The Server controls and coordinates the various parallel and distributed activities of the system. The primary input to WESE is the model of the system described using the System Specification Language (SSL). The specification of a model or a SSL design file consists of a set of interconnected *modules*. Each module consists of three main sections, namely; *(i)* the *component definition section* that contains the details of the components to be used to specify a module (such as the Universal Resource Locator (URL) of a factory and name of the source object along with initial parameters); *(ii)* the *component instantiation section* that defines the various components constituting the module; and *(iii)* the *netlist section* that defines the interconnectivity between the various instantiated components. SSL permits a *label* to be associated with each module. The *label* may be used as a component definition in subsequent module specifications to nest a module within another. In other words, the *labels*, when used to instantiate a component, results in the complete module associated with the label to be embedded within the instantiating module. This technique can be employed to reuse module descriptions and develop hierarchical specifications. As shown in Figure 6, the input SSL source is parsed into an object oriented (OO) in-memory *intermediate form* (SSL-IF) using the SSL parser. Hierarchical SSL models are elaborated or "flattened" at the end of parsing by the elaborator [9]. Elaboration is a recursive process that flattens a hierarchical model by substituting each module reference (made through the use of *label*s) with an unique instance of the module. As shown in Figure 6, the elaborated model, which is also represented using SSL-IF, forms the primary input to all the other modules of WESE.

The WESE Server also performs the task of collaborating with the distributed factories and coordinating the simulations. As shown in Figure 6, the *simulation manager* performs the activities associated with coordinating with the object factories (via the *factory manager*) to setup a distributed simulation. The *factory manager* performs the tasks of interacting with the distributed factories using a predefined protocol. It not only provides an uniform interface to communicate with different object factories but also insulates the other modules of the server from the intricacies of the underlying protocols. To ease design, development, and use of components WESE provides a framework for constructing web-based *object factories*. An object factory can be viewed as a web-based repository of components with an added capability for simulating them. A WESE factory is built from sub-factories and *object stubs*. The object stubs are the atomic components of a factory. Object stubs contain attributes of the physical component (such as cost and size). The stubs also compute the average event granularity (as specified by the user through suitable API methods) of the component associated with them. This value utilized to estimate changes in simulation time induced by CA and CDA. The distributed simulation capabilities of the WESE factories have been enabled using WARPED [7]. WARPED is an Application Program Interface (API) for a general purpose discrete event simulation kernel with dif-

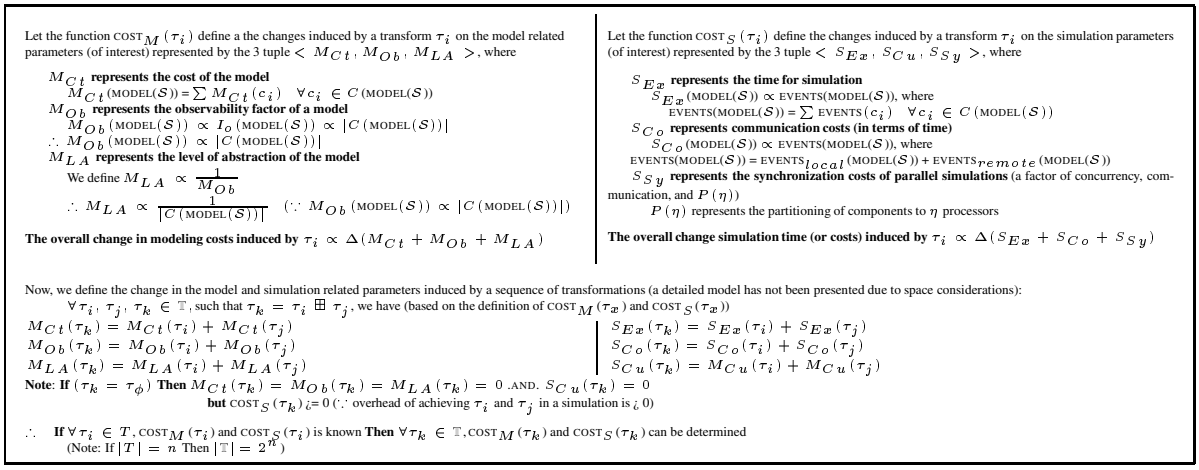


Figure 5. Effect of CA and CDA on Model and Simulation related parameters

ferent implementations [7]. WESE utilizes the Time Warp based simulation kernel of WARPED. Time Warp is an optimistic synchronization strategy [3]. A detailed description of WARPED is available in the literature [7]. The current implementation of WESE and all associated components is in C++. The object-oriented features of the language have been utilized to provide a simple and yet robust API. Further details on the factory API and WESE are available in the literature [8].

3.1 Support for CA and CDA

The current version of WESE provides support for *static* and *dynamic* CA and CDA. In WESE CA and CDA is performed at a module level. SSL permits an auxiliary (or equivalent) *component definition* to be associated with a model. When aggregation or de-aggregation for a module is requested, the set of components contained by the module are substituted using the auxiliary *component definition* and vice versa, as the case may be. The SSL Parser, SSL-IF, and the elaborator provide necessary frontend support required to enable static and dynamic CA and CDA. The process of dynamically substituting components during simulation involves the following steps: triggering CA or CDA in the simulation, creation of necessary LPs that model the components, updation of states and events of the LPs, and updation of kernel information. In Time Warp synchronized simulations,additional care must be exercised to implement these phases due to the presence of rollbacks [3].

In WESE, an event driven approach has been adopted for sequencing the various phases involved in achieving dynamic CA and CDA. The event driven algorithm utilizes the regular event scheduling mechanisms provided by the underlying simulation kernel; thereby abstracting away any overheads (and complications) introduced by the synchronization protocol. This solution also enables the algorithm to exploit any optimizations provided by the simulation kernel and is independent of the underlying synchronization mechanism. A detailed description of the event driven algorithm implemented in WESE along with the issues involved in the design and implementation of support for CA and CDA is available in the literature [11].

A *cost predictor* (shown in Figure 6) based on the analytical model for CA and CDA was also added to WESE. The cost predictor utilizes SSL-IF to collate necessary information about the components used in the model from the web-based factories, and predicts the changes in the modeling and simulation costs when a module is replaced with an auxiliary component. The modeling costs are computed as the sum of the costs of each *atomic* component (obtained from the appropriate factory) constituting the model. The observability of a model is computed the number of visible ports (as shown in Figure 5) in the model. The level of abstraction reported by the cost predictor, is a complex term and is a combination of the number of hierarchies in the model and the number of components at each hierarchi-

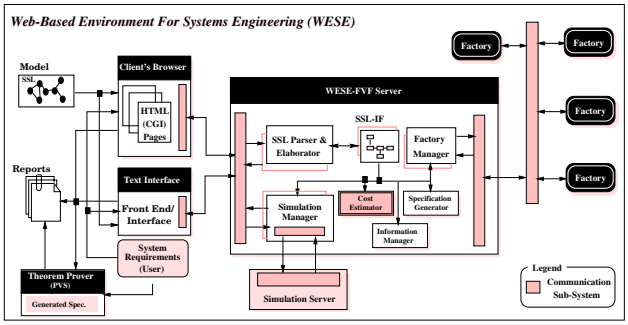
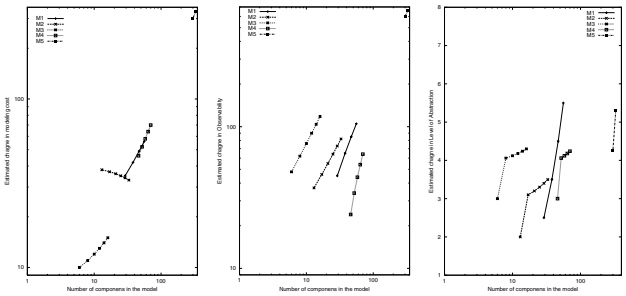


Figure 6. Overview of WESE

Model Name	Description	Number of Components		
		Regular	Aux.	Replaced by Aux.
M1	4-bit adder	56	4	9
M2	5-bit Mux.	33	6	4
M3	Cascaded half-adders	16	6	6
M4	Chain of not gates	70	5	6
M5	Chain of not gates	330	1	30

Table 1. Models used in experiments



(a) Costs **(b) Observability** **(c) Abstraction**

Figure 7. Change in Modeling parameters

cal level. The interfaces have been designed such that other parameter estimators can be easily plugged into WESE.

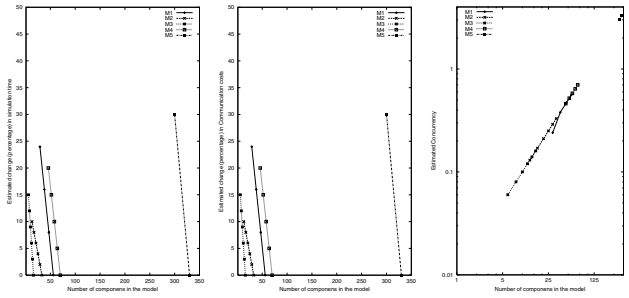
The change in simulation time and communication costs reported by WESE is proportional to the change in the size of the netlist of a module (which is proportional to the change in number of components). The average event granularity of each component (as reported by the corresponding factories) is utilized to estimate changes in simulation time. Currently, a straightforward technique has been employed to estimate changes in simulation performance, when WESE is utilized for distributed simulation. In distributed simulation scenarios, the sum of the event granularities of the components allocated to each factory is computed and the maximum of these values is utilized as the overall granularity of the simulation. This heuristic is based on the fact that, in WESE parallelism occurs at the factory level and processing of events within a factory proceeds in a serial fashion. It is also assumed that a given percentage (50% by default) of events get communicated over the network. The network latency between the factories is measured and the average latency is added on to the overall granularity (computed earlier). This value is used as the representative simulation time for comparisons against other configurations of the model. Currently, WESE reports the concurrency of a model as a factor of the number of components constituting the model. As shown in Figure 5, the predicted changes in costs, in conjunction with the algebra for CA and CDA is used to predict the changes in the costs induced by a sequence of transforms (as specified by the user) on the given model. The modeler can use the information and perform suitable transformations to the model either at startup or during the simulation and optimize the costs of modeling and simulation.

4 Experiments

The experiments conducted to evaluate the support for CA and CDA in WESE consisted of two phases. During the first phase an object factory consisting of a collection of logic gates was developed. The domain of logic simulation was selected because the systems inherently lend themselves for CA and CDA. The factory contained logic gates such as two input `and` gate, two input `or` gate, two input `exclusive-or` gate, and `not` gate. More complex components, such as a `HalfAdder` and a `FullAdder`, were included in the hardware factory. The factory also contained a bit pattern generation component and a bit display component. These components were used to estimate the average event granularity (*i.e.,* time taken to execute one event) which is used by the *cost estimator* of WESE to predict changes in simulation time induced by CA and CDA. The factory also contained a *controller* component that provides a convenient interface to trigger CA and CDA. The object stubs for the various components included the costs related to the various modeling and simulation related parameters. Further details on the object factory used in the experiments is available in the literature [11].

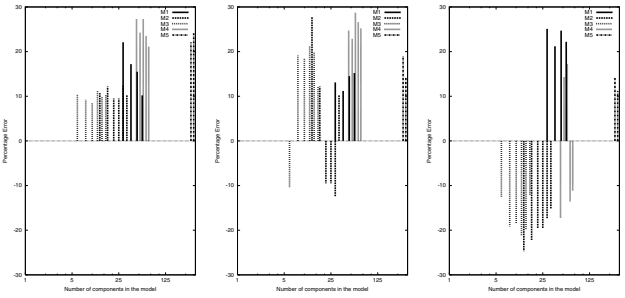
The second phase of the experiment consisted of developing logic models in SSL using the various components from the hardware factory. The characteristics of some of the models using the experiments is shown in Table 1. The models included auxiliary component specifications for the modules that had equivalent higher level abstractions. The number of components replaced by each auxiliary component in the models is also shown in the table (column `Replaced by Aux.`). For example, model M1 was implemented using structural models of full adders. The structural models of also included an auxiliary specification to use the `FullAdder` component available in the factory. The `FullAdder` component substitutes nine components constituting the structural model. The SSL descriptions also used the `controller` components to activate (or deactivate) the auxiliary modules (or trigger a transformation) at different time points during simulation.

All the experiments were conducted on a network of shared memory multi-processor (SMP) workstations. Each workstation consisted of two Pentium pro Processors (166 Mhz.) with 128 mega bytes (MB) or main memory (RAM). The workstations were inter-connected using fast Ethernet. The graphs in Figure 7 presents the change in the modeling related parameters (estimated by WESE) of the different



(a) Time (b) Communication (c) Concurrency

Figure 8. Change in Simulation parameters

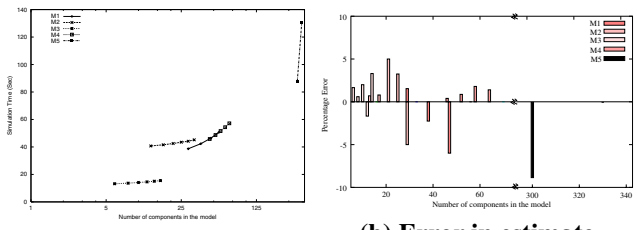


(a) 2 Clusters (b) 3 Clusters (c) 4 Clusters

Figure 10. Parallel Simulation Times

models when different CA and CDA transformations are applied to the models. Figure 8 presents the changes in simulation related parameters corresponding to the different transformations in the models. Figure 9(a) presents the actual time for simulating the models with various transformations (without initialization costs) using a single factory *i.e.,* the simulations are inherently sequential in nature and do not involve any rollbacks. The timing information shown in the graph is the average of 10 simulation runs. The error (in percentage) between the estimated change in simulation time and observed change in simulation time for the various transformations on the different models is shown in Figure 9(b). As illustrated by the graphs in Figure 9(b), the estimated change in simulation time closely reflects the measured change in simulation time even though the estimation model is straightforward. Several factors play a critical role in enabling close estimation of changes in simulation time. The primary factors being *(i)* The variance in the event granularities (time for executing an event) of the models is small; *(ii)* the transformations are linear with respect to the number of events; *(iii)* the underlying simulation kernel scales linearly with respect to the number of events in the simulation. We believe, but for these three important factors, the simple estimator (currently employed in WESE) would not provide a very accurate estimate.

The shortcomings of the simple estimator (as explained earlier) is evident when the simulations are performed in parallel. The graphs in Figure 10 present the error (in percentage) between the estimated change in simulation time and observed change in simulation time for the various transformations when simulating using 2, 3, and 4 factories. The parallel simulation experiments were conducted by deploying the object factory on different workstations and modifying the SSL descriptions to choose components from the different factories. The components were chosen from the different factories at random. As illustrated by the graphs, the performance of the simulations increases as the number of components decreases because the total number of events that need to be processed decreases. Although, the estimator correctly predicts the change in simulation time the estimate is not accurate. It must be noted that the goal of this study is to provide an estimate for the potential change (whether the costs increase or decrease) and not to precisely quantify it. As illustrated by the experiments, the CA and CDA transformations provide a range of tradeoffs between model related factors such as modeling overheads, model observability, & accuracy of simulation results and simulation related parameters such as concurrency, communication overheads, simulation overheads, and simulation performance versus resource consumption and overall costs.



(a) Simulation Time (b) Error in estimate

Figure 9. Simulation Time

5 Conclusions

The number of components constituting a model determines the overall costs of web-based modeling and simulation. Hence, by varying the number of components, the overall costs of modeling and simulation can be varied. Accordingly, two functionality preserving transformations, namely CA and CDA, that alter the number of components constituting a model without altering the basic functionality of the model, were presented. The transformations provide a range of tradeoffs between several modeling and simulation related parameters such as modeling overheads, model observability, accuracy of simulation results, communication overheads, simulation overheads, and simulation performance versus resource consumption and overall costs. The paper presented a mathematical model of the

CA and CDA transformations that can be used to estimate the changes in modeling and simulation related parameters. The math model can be used by the user (or by the simulation system) to optimally configure a model to meet the goals of the analysis. The effectiveness of the transformations and the math model in optimizing the costs of web-based simulations were demonstrated through experiments. Quantitative use of the math model is currently limited to sequential simulations. Studies are being conducted to extend the model to closely reflect the changes in parallel simulations. As illustrated by the experiments, the CA and CDA transformations provide an effective means to regulate and reduce the overall costs of web-based modeling and simulation.

References

[1] FISHWICK, P. A. Web-based simulation: Some personal observations. In *Proc. of the 1996 Winter Simulation Conference* (Dec. 1996), pp. 772–779.

[2] HUANG, P., ESTRIN, D., AND HEIDEMANN, J. Enabling large-scale simulations: Selective abstraction approach to the study of multicast protocols. In *In Proceedings of International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Networks* (Oct. 1998).

[3] JEFFERSON, D. Virtual time. *ACM Transactions on Programming Languages and Systems 7*, 3 (July 1985), 405–425.

[4] MCBRAYER, T., AND WILSEY, P. A. Process combination to increase event granularity in parallel logic simulation. In *9th International Parallel Processing Symposium* (Apr. 1995), pp. 572–578.

[5] PAGE, E. H., GRIFFIN, S. P., AND ROTHER, L. S. Providing conceptual framework support for distributed web-based simulation within the high level architecture. In *Proceedings of SPIE: Enabling Technologies for Simulation Scicence II* (Apr. 1998).

[6] PIDD, M., OSES, N., AND CASSEL, R. A. Component-based simulation on the web? In *In Proceedings of the 1999 Winter Simulation Conference (WSC'99)* (Dec. 1999).

[7] RADHAKRISHNAN, R., MARTIN, D. E., CHETLUR, M., RAO, D. M., AND WILSEY, P. A. An Object-Oriented Time Warp Simulation Kernel. In *Proceedings of the International Symposium on Computing in Object-Oriented Parallel Environments (ISCOPE'98)*, D. Caromel, R. R. Oldehoeft, and M. Tholburn, Eds., vol. LNCS 1505. Springer-Verlag, Dec. 1998, pp. 13–23.

[8] RAO, D. M., CHERNYAKHOVSKY, V., AND WILSEY, P. A. WESE: A Web-based Environment for Systems Engineering. In *2000 International Conference On Web-Based Modelling & Simulation (WebSim'2000)* (Jan. 2000), Society for Computer Simulation.

[9] RAO, D. M., RADHAKRISHNAN, R., AND WILSEY, P. A. FWNS: A Framework for Web-based Network Simulation. In *1999 International Conference On Web-Based Modelling & Simulation (WebSim'99)* (Jan. 1999), A. G. Bruzzone, A. Uhrmacher, and E. H. Page, Eds., vol. 31, Society for Computer Simulation, pp. 9–14.

[10] RAO, D. M., AND WILSEY, P. A. Simulation of ultra-large communication networks. In *Proceedings of the Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (Oct. 1999), pp. 112–119.

[11] RAO, D. M., AND WILSEY, P. A. Dynamic component substitution in web-based simulation. In *In Proceedings of the 2000 Winter Simulation Conference (WSC'00)* (Dec. 2000). (to appear).

[12] TREMBLAY, J. P., AND MANHOHAR, R. *Discrete Mathematical Structures With Applications to Computer Science*. McGraw-Hill Computer Science Series, USA, 1975.

[13] VINOSKI, S. CORBA: Integrating diverse applications within distributed heterogenous environments. *IEEE Communications Magazine 35*, 2 (Feb. 1997), 46–57.

[14] WANG, Z., AND MAURER, P. M. Lecsim: A levelized event driven compiled logic simulation. In *Proceedings of the 27th ACM/IEEE Design Automation Conference (DAC'90)* (June 1990), pp. 491–496.