

*Using Direct Sub-Level Entity Access  
to Improve Nuclear Stockpile  
Simulation Modeling*

**Los Alamos**  
NATIONAL LABORATORY

*Los Alamos National Laboratory is operated by the University of California  
for the United States Department of Energy under contract W-7405-ENG-36.*

*This thesis was accepted by the Department of Mechanical Engineering, Brigham Young University, Provo, Utah, in partial fulfillment of the requirements for the degree of Master of Science. The text and illustrations are the independent work of the author and only the front matter has been edited by the CIC-1 Writing and Editing Staff to conform with Department of Energy and Los Alamos National Laboratory publication policies.*

*An Affirmative Action/Equal Opportunity Employer*

*This report was prepared as an account of work sponsored by an agency of the United States Government. Neither The Regents of the University of California, the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by The Regents of the University of California, the United States Government, or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of The Regents of the University of California, the United States Government, or any agency thereof. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.*

*Using Direct Sub-Level Entity Access  
to Improve Nuclear Stockpile  
Simulation Modeling*

*Robert Y. Parker*

## TABLE OF CONTENTS

<b>LIST OF FIGURES .....</b>	<b>vii</b>
<b>LIST OF TABLES .....</b>	<b>viii</b>
<b>ABSTRACT .....</b>	<b>ix</b>
<b>CHAPTER 1 - INTRODUCTION .....</b>	<b>1</b>
1.1 STATEMENT OF PROBLEM .....	2
1.2 THESIS STATEMENT .....	3
1.3 APPROACH.....	3
1.4 CATEGORIES OF SIMULATION MODELING.....	4
1.5 NUCLEAR STOCKPILE, LIFE-EXTENSION MODELING .....	6
1.6 HIERARCHICAL ENTITY STRUCTURE.....	8
1.7 DIRECT SUB-LEVEL ENTITY ACCESS .....	10
1.8 DELIMITATIONS .....	12
1.9 DEFINITION OF TERMS.....	13
<b>CHAPTER 2 - REVIEW OF RELATED WORK .....</b>	<b>17</b>
2.1 NUCLEAR STOCKPILE MODELING .....	18
2.1.1 Nuclear Stockpile Simulation Modeling .....	18
2.1.2 Nuclear Stockpile, Life-Extension Modeling .....	19
2.2 METHODS OF SUB-LEVEL ENTITY INFORMATION ACCESS.....	19
2.2.1 Common Methods .....	20
2.2.2 Database Concept.....	22
2.2.3 Direct Methods .....	24
2.3 SURVEY OF VENDORS .....	25
<b>CHAPTER 3 - METHODS.....</b>	<b>27</b>
3.1 STEP 1 – EXISTING NUCLEAR STOCKPILE MODEL (2X2 MODEL) .....	28
3.1.1 Weapons.....	29
3.1.2 Maintenance and Surveillance Requests .....	29
3.1.3 Stockpile.....	30

3.1.4	<i>Assembly/Disassembly Facility</i> .....	31
3.2	STEP 2 – DEFICIENCIES IDENTIFIED IN THE 2X2 MODEL .....	31
3.2.1	<i>True Defect Analysis</i> .....	32
3.2.2	<i>Multiple Storage Sites</i> .....	32
3.2.3	<i>Additional Weapon and Subsystem Types</i> .....	33
3.3	STEP 3 – CONCEPT MODELS OBTAINED .....	33
3.3.1	<i>True Defect Analysis Concept Model (Baseline-1)</i> .....	34
3.3.2	<i>Multiple Storage Sites Concept Model (Baseline-2)</i> .....	36
3.4	STEP 4 – CONCEPT MODELS MODIFIED .....	37
3.4.1	<i>True Defect Analysis Concept Model (Mod-1A)</i> .....	38
3.4.2	<i>True Defect Analysis Concept Model (Mod-1B)</i> .....	39
3.4.3	<i>Multiple Storage Sites Concept Model (Mod-2)</i> .....	40
3.5	STEP 5 – CONCEPT MODELS ANALYZED AND COMPARED .....	42
3.5.1	<i>Comparative Analysis Methods</i> .....	42
3.5.2	<i>Analysis Metrics</i> .....	43
3.6	STEP 6 – 2X2 MODEL IMPROVED (7X7 MODEL).....	46
<b>CHAPTER 4 - RESULTS AND ANALYSIS</b> .....		<b>48</b>
4.1	TRUE DEFECT ANALYSIS MODELS .....	48
4.1.1	<i>Quantitative Analysis</i> .....	49
4.1.2	<i>Qualitative Analysis</i> .....	52
4.2	MULTIPLE STORAGE SITES MODELS.....	54
4.2.1	<i>Quantitative Analysis</i> .....	55
4.2.2	<i>Qualitative Analysis</i> .....	57
4.3	ORIGINAL 2X2 MODEL VS. IMPROVED 7X7 MODEL .....	59
4.4	DISADVANTAGES OF DIRECT SUB-LEVEL ENTITY ACCESS.....	62
<b>CHAPTER 5 - CONCLUSIONS AND RECOMMENDATIONS</b> .....		<b>64</b>
5.1	CONCLUSIONS .....	64
5.2	RECOMMENDATIONS .....	66
<b>APPENDIX A - VENDOR QUESTIONNAIRE</b> .....		<b>68</b>
<b>APPENDIX B - DIALOG OF GET ATTRIBUTE (SUB-LEVEL) BLOCK</b> .....		<b>72</b>
<b>APPENDIX C - DIALOG OF DEFECT ANALYZER BLOCK</b> .....		<b>73</b>
<b>APPENDIX D - DIALOG OF REMOTE CHOOSER BLOCK</b> .....		<b>74</b>
<b>REFERENCES</b> .....		<b>78</b>

## LIST OF FIGURES

<b>Figure 1.1</b> Entity Batching Operation .....	9
<b>Figure 1.2</b> Three-Level Entity Structure .....	10
<b>Figure 2.1</b> Unbatch/Batch Technique used to Access Sub-Level Entities .....	20
<b>Figure 2.2</b> Attribute-Copying Technique .....	21
<b>Figure 3.1</b> 2x2 Model – Basic Entity Flow Diagram .....	28
<b>Figure 3.2</b> Simplified Stockpile Representation in 2x2 Model – Entity Segregation .....	31
<b>Figure 3.3</b> True Defect Analysis “Baseline-1” Model – Basic Flow Diagram .....	34
<b>Figure 3.4</b> Accessing Sub-Level Attributes for Defect Analysis (“Baseline-1” Model) .....	35
<b>Figure 3.5</b> Multiple Storage Sites “Baseline-2” Model – Basic Flow Diagram.....	37
<b>Figure 3.6</b> Accessing Sub-Level Attributes for Defect Analysis (“Mod-1A” Model).....	38
<b>Figure 3.7</b> True Defect Analysis “Mod-1B” Model – Basic Flow Diagram.....	40
<b>Figure 3.8</b> Multiple Storage Sites “Mod-2” Model – Basic Flow Diagram.....	41

## LIST OF TABLES

<b>Table 3.1</b> Analysis Metrics .....	43
<b>Table 4.1</b> Scenario Parameter Values for the True Defect Analysis Models .....	49
<b>Table 4.2</b> Run Time of the True Defect Analysis Models.....	50
<b>Table 4.3</b> Size Contributed to Model by each True Defect Analysis Implementation.....	51
<b>Table 4.4</b> Qualitative Metric Comparison of Defect Analysis Models .....	53
<b>Table 4.5</b> Scenario Parameter Values for the Multiple Storage Sites Models.....	55
<b>Table 4.6</b> Run Time of the Multiple Storage Sites Models .....	56
<b>Table 4.7</b> Size Contributed to Model by each Sort and Select Implementation.....	57
<b>Table 4.8</b> Qualitative Metric Comparison of Multiple Storage Sites Models .....	58
<b>Table 4.9</b> Qualitative Metric Comparison of the 2x2 versus the 7x7 Model .....	60

USING DIRECT SUB-LEVEL ENTITY ACCESS TO IMPROVE  
NUCLEAR STOCKPILE SIMULATION MODELING

Robert Y. Parker

**ABSTRACT**

*Direct sub-level entity access* is a seldom-used technique in discrete-event simulation modeling that addresses the accessibility of sub-level entity information. The technique has significant advantages over more common, alternative modeling methods – especially where hierarchical entity structures are modeled. As such, *direct sub-level entity access* is often preferable in modeling nuclear stockpile, life-extension issues, an area to which it has not been previously applied.

Current nuclear stockpile, life-extension models were demonstrated to benefit greatly from the advantages of *direct sub-level entity access*. In specific cases, the application of the technique resulted in models that were up to 10 times faster than functionally equivalent models where alternative techniques were applied. Furthermore, specific implementations of *direct sub-level entity access* were observed to be more flexible, efficient, functional, and scalable than corresponding implementations using common modeling techniques.

Common modeling techniques (“unbatch/batch” and “attribute-copying”) proved inefficient and cumbersome in handling many nuclear stockpile modeling complexities, including multiple weapon sites, true defect analysis, and large numbers of weapon and subsystem types. While significant effort was required to enable *direct sub-level entity access* in the nuclear stockpile simulation models, the enhancements were worth the effort – resulting in more efficient, more capable, and more informative models that effectively addressed the complexities of the nuclear stockpile.

# CHAPTER 1

## INTRODUCTION

*Simulation* is the act of imitating a real-world process or system. The representation of the system, including the assumptions and relationships between system elements, is called a *model*. A model can also be defined as “a representation of a system for the purpose of studying the system” (Banks, Carson, and Nelson 11). Hence, a *simulation model* is used to evaluate and investigate the behavior of the system over time. The usefulness of such models depends on the desired goals and objectives of the simulation study, and whether the information extracted from the model appropriately addresses those objectives. The key, then, is having the ability to set up the model in such a way that the necessary behavior is exhibited and the needed information is accessible.

Within the framework in which simulation models are built, there exist various methods to access and use important model information. Simulation software packages have been developed specifically to make these modeling tasks easier. Such software typically contains the underlying structure for automatic model information management and provides techniques and links for obtaining and using that information during a simulation. While some methods for accessing information in a simulation model are quite common, other techniques are less often implemented, but may significantly enhance the usefulness of a model.

This thesis will focus on applying an important, though seldom-used, information access technique, called *direct sub-level entity access*, in discrete-event simulation modeling. The concept will be applied specifically to improve the usefulness of models in a domain where it has not been applied before – nuclear stockpile, life-extension modeling.

## **1.1 STATEMENT OF PROBLEM**

Modeling the United States (U.S.) nuclear weapons complex presents several challenges to account for the complexities of the nuclear stockpile. Multiple weapon sites, large numbers of different weapon types and critical subsystems, and surveillance/maintenance coordination are some of the complexities that need to be dealt with to effectively model stockpile life-extension issues. Common simulation modeling techniques have been used to model the stockpile in a limited fashion. However, they are limited in their ability to effectively access important model information and do not provide the flexibility to adequately address many required modeling objectives. Specifically, these common methods employ relatively convoluted and inefficient means to access sub-level entity information in nuclear stockpile simulation models. They require the user to implement modeling constructs that are bulky, slow, and inflexible. These constructs are often only able provide limited functionality in terms of the amount and type of entity information that can be handled. When model complexity and size is increased (by adding more weapon types, weapons sites, etc.), common methods fail by making the model unreasonably large and slow and by lacking the capability to conveniently access and use specific model information for analysis needs. Consequently,

common modeling implementations result in stockpile models that are limited in usefulness. More effective modeling techniques must be applied to enhance the usefulness of nuclear stockpile, life-extension models.

## **1.2 THESIS STATEMENT**

*Direct sub-level entity access* is an important concept that offers valuable benefits over more common modeling methods and provides significant advantages – including enhanced functionality, capability, flexibility, and efficiency – that are critical to effectively model and analyze nuclear stockpile complexities.

## **1.3 APPROACH**

The thesis will emphasize the application of *direct sub-level entity access* to discrete-event simulation models that are used to evaluate the potential response of the nuclear weapons complex and subsequent impacts on nuclear stockpile characteristics associated with specific stockpile life-extension plans. Key deficiencies from a current discrete-event, nuclear stockpile, life-extension model will be identified. Other existing models that use common modeling techniques to help address the deficiencies will be obtained and tested. These models will be modified and tested again after implementing *direct sub-level entity access*. The results and performance of both the original and modified versions of the models will be evaluated and quantified/qualified as appropriate. The information will be analyzed to show the advantages, benefits, and added functionality that *direct sub-level entity access* offers over alternative methods –

ultimately resulting in more flexible, useful, and informative models capable of addressing many complexities presented by the nuclear stockpile.

To more fully understand the focus of the thesis, some additional background information is addressed in the following sections. Specific topics covered are categories of simulation modeling, nuclear stockpile, life-extension modeling, hierarchical entity structures in discrete-event simulation models, and direct sub-level entity access.

#### **1.4 CATEGORIES OF SIMULATION MODELING**

The vast number of processes and systems gives rise to many areas of application for simulation modeling. However, the general area of application addressed in this thesis will be simulation modeling of manufacturing/production systems within which are the aforementioned nuclear stockpile models. To further differentiate between this general area of application and other related areas, three categories of simulation modeling will be briefly discussed: mechanical modeling, process modeling, and systems modeling.

*Mechanical modeling* relates to the simulation of the mechanics and/or kinematics of a machine or workcell. It may include spatial/geometrical and motion control modeling of the physical system. It is frequently employed in machine automation exercises and in the design and operation of workcells, robots, etc. (This type of simulation is not included in the general area of application addressed in this thesis.)

*Process modeling* relates to the simulation of a collection of activities forming a specific process. The focus is on “what” is performed by the process itself and its interrelationship with other processes, as opposed to “how, when, and where” the process is performed (Harrell and Tumay 18). Process modeling is used to help analyze the

practices and procedures involved in a process. These procedures are the types of things frequently characterized by flow charts. (Process modeling itself is not the general area of application addressed in this thesis.)

*Systems modeling* relates not only to the simulation of processes (as in process modeling), but also of the elements used to perform the processes (resources, controls, activities, etc.). The focus is on “how, when, and where” the processes occur. Systems modeling encompasses the dynamic interrelationships between, and the effects on, the system elements (including the entities being processed), as well as the effects on the system as a whole. Systems simulation modeling is frequently employed in analyzing the causes and effects of behaviors in manufacturing/production settings – such as part production factories and job-shops.

System simulation models are generally categorized as discrete or continuous. A *continuous* model is one whose state changes continuously over time. A chemical delivery system, whose liquid volume changes continuously based on the input/output flows, is an example of a system modeled in a continuous manner. Values and calculations in continuous models are updated at evenly spaced time intervals.

A *discrete* model (or a discrete-event model) is one whose state changes only at discrete, separate points in time. A manual assembly line at a manufacturing plant is an example of a system modeled in a discrete manner. Values and calculations in a discrete-event model are updated only when certain events occur. An event, in this case, may be the arrival of a part at an inspection station, or the end of a machine’s processing cycle. A discrete-event simulation essentially produces a series of snapshots representing the state of the system as the simulation clock steps through the process events.

The general area of application addressed in this thesis is computer-based, discrete-event system simulation modeling of manufacturing-related systems. This general area of application is assumed by any reference to simulation or modeling from this point on (unless otherwise noted). The nuclear stockpile, life-extension models used in this thesis certainly fall under this general manufacturing-related category.

## **1.5 NUCLEAR STOCKPILE, LIFE-EXTENSION MODELING**

The United States nuclear stockpile is the collection of nuclear weapons maintained by and for the U.S. government. The Department of Energy (DOE) nuclear weapons complex embodies those facilities and organizations charged with the design, production, maintenance, certification, and disposition of the nuclear weapons in the stockpile.

The end of the cold war and the increasing number of international treaties and arms limitation agreements have resulted in dramatic changes in the way the DOE must carry out its nuclear stockpile responsibilities (Lawrence Livermore, *Keeping*). The size of the nuclear stockpile is being reduced, underground nuclear testing has stopped, and much of the weapons production complex has been shut down. Because of these recent developments, and the fact that no new nuclear weapons are being produced to replace and modernize existing systems, there is a significant challenge to maintain the weapons in the stockpile – which in many cases are rapidly approaching their designed life-expectancy (Lawrence Livermore, *Through 8-9*).

To help address the requirements of assuring the safety and reliability of the existing nuclear stockpile without nuclear testing, new weapons development, or a large

production complex, life-extension programs are being developed and evaluated. Various life-extension options are being explored so that the appropriate organizations can anticipate and plan for future maintenance and refurbishment requirements (Lawrence Livermore, *Through* 13).

Simulation, with its inherent abilities to capture complex interdependencies and to do comparative analyses, is being leveraged as a planning, evaluation, analysis, and communication tool in the nuclear stockpile, life-extension arena. Many questions are being asked, such as the following:

- What weapons are nearing their life expectancy?
- Where are they located?
- How can they be serviced without diminishing the number of active weapons available?
- What is the reliability of a specific group of weapons?
- How well can the reliability be predicted with limited testing options?

Models representing the nuclear stockpile and weapons complex have been developed to begin addressing some of these questions in the context of specific life-extension options being considered. While promising and somewhat useful, these preliminary models are still limited in capability and usefulness.

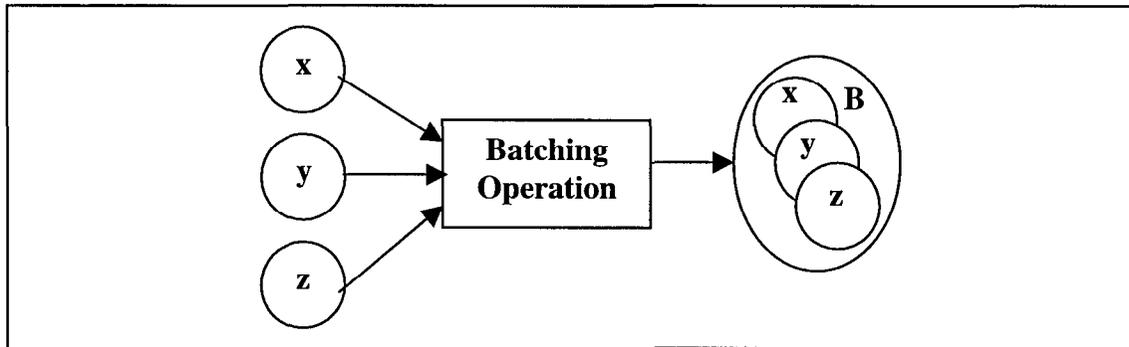
The desire and need is to enhance and improve the stockpile life-extension models so that they can better address more of the life-extension questions being posed by decision-makers. Information needs to be more effectively used and extracted from the models. One way of doing this will be demonstrated in this thesis. An advanced modeling concept will be applied to enhance nuclear stockpile, life-extension models. The concept,

*direct sub-level entity access*, is especially applicable to models where the desired entities of interest are structured hierarchically and where the sub-level entities in the hierarchy are often just as important as the enclosing top-level entity. Such is the case of the stockpile life-extension models, where weapons are modeled in the same way they exist in real life – as hierarchical assemblies of critical sub-component parts.

## **1.6 HIERARCHICAL ENTITY STRUCTURE**

*Entities* are the dynamic objects that move around within discrete-event simulation models. They typically represent real-world objects (parts, people, etc.) from the system being simulated. Entities are assigned characteristics and attributes that identify and individualize them (for example “type,” “color,” etc.). These attributes are carried by the entities themselves. Model behavior and functionality often rely on this important entity-specific information.

*Batching* entities together in discrete-event simulation models is a common technique for representing manufacturing scenarios, including assembly operations. It is the primary modeling mechanism for establishing *hierarchical entity structures* that represent real-life situations. To illustrate this, consider a simulation model representing three parts or entities (x, y, and z) each with unique characteristics or attributes. The three entities are batched together in an assembly-type operation. The resulting assembly is a new entity ‘B’. Because the assembly may at some future time be separated into its original parts, entities x, y, and z must maintain their unique characteristics while batched together. The process being modeled is illustrated in Figure 1.1.

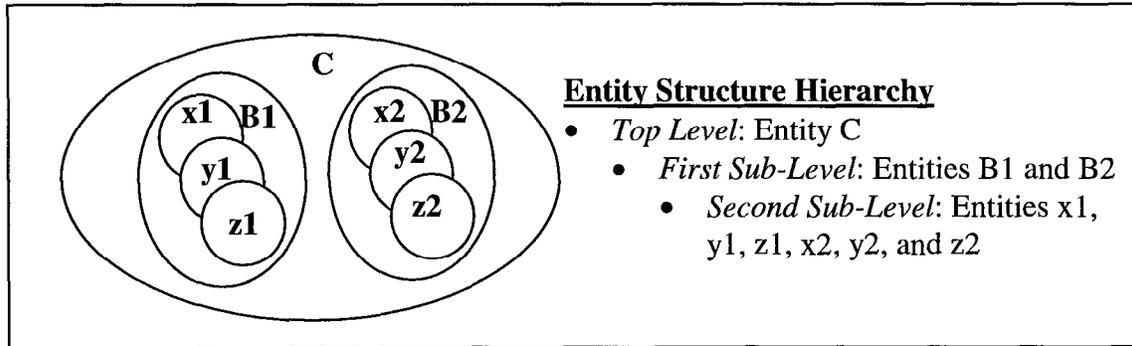


**Figure 1.1** Entity Batching Operation

Prior to the batching operation, entities *x*, *y*, and *z* are considered *top-level* entities. After batching, new entity *B* (the assembly) is considered a *top-level* entity, while entities *x*, *y*, and *z* are now considered *sub-level* entities (constituent entities of a batch). In this particular case, a two-level entity structure exists with *B* being at the top level, and *x*, *y*, and *z* being at the first sub-level in the entity structure.

Deeper-level entity structures are the result of multiple batching operations. For example, if a couple of assembly *B*'s were subsequently batched together into another assembly '*C*', then the new assembly *C* would be at the top of the resultant three-level entity structure. The *B* entities would then exist at the first sub-level while the *x*, *y*, and *z* entities would exist at the second sub-level in the entity structure. Figure 1.2 illustrates this three-level case.

Whether there are one or more levels of batching, the result is a hierarchical entity structure in the model that represents the real-world assembly of parts. This is one way weapons in the United States nuclear stockpile have been modeled. (Note that none of the models addressed in this thesis deal with three-level or deeper entity structures.)



**Figure 1.2** Three-Level Entity Structure

### 1.7 DIRECT SUB-LEVEL ENTITY ACCESS

Consistent with the illustration in Figure 1.1, a nuclear weapon is an assembly of subsystems or sub-components (often represented in models by sub-level entities). The reliability of the weapon depends on the condition of its critical subsystems. So in a model where the objective relates to nuclear stockpile life-extension, the subsystem characteristics and attributes (which may only exist on the sub-level entities themselves) are of particular interest. For example, consider a system of weapons, each with several vital, limited-life subsystems. In a model of such a system, the subsystems are unique sub-level entities with attributes attached to them. The attributes may indicate “part type,” “date of manufacture,” “serial number,” etc. As part of the modeling exercise, suppose all of the subsystems of a particular type that are older than ten years need maintenance. The age of a given part is based on its date of manufacture; an attribute that is carried by the sub-level entity itself. So the ability to access this information – independent of where the entities are located in the model – is critical.

In most discrete-event simulation models, interaction with and access to entity attributes occurs at the top-level. That is, entity information is obtained directly from the

model entities themselves at designated points in the model while the entities are in a top-level state. In the case of a weapon, access to attributes on a subsystem is accomplished by unbatching the weapon assembly entity – bringing the subsystem entities back up to the top-level – and then re-batching them when done accessing the desired information. This is one common method that has been used to access and manage necessary subsystem information in models of the nuclear stockpile and the nuclear weapons complex. This method also requires that all entities to be accessed be physically moved or directed to designated points in the model (i.e. *local* entity access) where the unbatching takes place.

One technique that has not been implemented before in such models is that of *direct, sub-level* entity access. This concept involves accessing and manipulating entity-specific information directly on entities while they are in a sub-level state – independent of where the entities are located in the model. This implies that the sub-level entity information can be accessed *globally* from anywhere in the model. Such direct, sub-level entity access is a very useful, though uncommon, technique in discrete-event simulation modeling, and has not been applied in nuclear stockpile, life-extension models. The ability to extract sub-level entity information from the model during a simulation is vital to addressing many of the complexities of the nuclear stockpile. The advantages and benefits of using this technique to improve nuclear stockpile simulation modeling will become apparent through this thesis.

## 1.8 DELIMITATIONS

*Direct access* of sub-level entities implies the ability to view, change, and set entity-specific information (primarily attributes) on entities while they are in a sub-level state – independent of where the entities are located in the model. This concept, as presented in this thesis, applies only to discrete-event system simulation modeling as defined previously in section 1.2. This means that continuous simulation modeling, process modeling, static modeling, symbolic modeling, mathematical modeling, mechanical simulation modeling, motion control modeling, automation/control modeling, and other forms of modeling will not be addressed.

The implementation and demonstration of direct sub-level entity access in this thesis will be conducted on nuclear stockpile, life-extension models using the same simulation environment in which the selected original models are found.

Although it is not an oft-used concept, direct sub-level entity access can be applied in a variety of models of different systems. The direction of this thesis, however, will be toward nuclear stockpile, life-extension issues in the nuclear weapons complex. Specifically, the focus will be on manufacturing- or production-type models that are used to help demonstrate and evaluate the nuclear weapons complex response to certain stockpile life-extension plans and the subsequent effect on nuclear stockpile characteristics. Other specific areas of potential applicability may be mentioned but will not be demonstrated.

The nuclear stockpile models in which the direct sub-level entity access concept is applied only have weapons represented as either one- or two-level entity structures. That is, weapons are represented either by a top-level weapon entity only, or by a top-level

weapon entity with one layer of sub-level component entities (as in Figure 1.1). Direct access of entity information at deeper sub-levels will not be demonstrated.

## 1.9 DEFINITION OF TERMS

**Attributes:** User-defined characteristics of entities in a discrete-event simulation model.

Individual entities can carry different values for a given attribute. For example, an entity representing a part may have attributes called *Type* and *Color*, with certain values assigned to differentiate between this and other entities.

**Batch:** A group of entities that have been assembled (batched) together. This is the same as an assembly entity.

**Batching:** The act of assembling model entities together in a discrete-event simulation model. The result of a batching operation is a top-level entity representing the assembly, with contained sub-level entities representing the constituent sub-components used to make the assembly.

**Discrete-Event Simulation:** A type of simulation in which the state of the model changes at discrete points in time based on the occurrence of events. A discrete-event simulation essentially proceeds by producing a series of snapshots representing the state of the system as the simulation clock steps through each specific event time during a simulation run. An example of an event is the arrival of a part at a processing station or the scheduled shutdown of a processing cycle.

**Entity:** A dynamic object of interest that moves around within a discrete-event simulation model. For example, in a nuclear stockpile, life-extension model, weapons and associated sub-component parts would be the entities of interest.

**Global Entity Access:** The ability to access entity information globally from anywhere in the model, independent of where a given entity resides. Entities are not required to pass through certain points in the model in order to have their attributes read. *Direct sub-level entity access* is capable of such global entity access. However, the common modeling techniques for accessing sub-level entity information that are addressed in this thesis are incapable of such global entity access.

**Hierarchical Entity Structure:** The resulting entity construct after an entity batching operation. In such a structure, a top-level entity exists with sub-level entities contained within it. The sub-level entities may, in turn, contain other constituent entities, and so on, as in Figure 1.2. (Note: This thesis only deals with entities down to the first sub-level, as in Figure 1.1.)

**Local Entity Access:** The ability to access entity information locally – when an entity is at a specific, designated point in the model. Access to entity-specific information takes place only at certain points in the model through which an entity must pass in order to have its attributes read. The common modeling methods for accessing sub-level entity information that are addressed in this thesis are instances of local entity access. *Direct sub-level entity access* can also be applied as a local access implementation.

**Model:** A representation of a system for the purpose of studying the system. A *simulation model* is used to evaluate and investigate the behavior of the modeled system over time.

**Nuclear Stockpile:** The collection of existing nuclear weapons in the U.S. nuclear arsenal.

**Nuclear Stockpile Life-Extension:** The act of extending the life of existing nuclear weapons in the stockpile through maintenance, refurbishment, etc. This can entail policies, programs, schedules, or options for executing a desired life-extension plan.

**Nuclear Weapons Complex (Or the DOE weapons complex):** The collection of facilities and organizations whose responsibilities include the design, production, maintenance, safety, certification, and disposition of the weapons in the nuclear stockpile.

**Simulation:** The act of imitating a real-world process or system with the intent of studying the system's behavior over time.

**Sub-Level Entity:** An entity in a discrete-event simulation model that has been batched together with other entities to form an assembly entity. It is an entity that is currently a constituent sub-component of a top-level assembly.

**Surveillance/Surveillance Analysis:** As used in this thesis, surveillance refers to performing quality assurance tests on a small sub-set of a given weapon population which is essentially analogous to statistical sampling. Surveillance analysis (as performed by simulation models discussed herein) involves determining age-related weapon defect rates based on the attributes carried by the weapon entities and their subsystems that are actually included in the sample set.

**Systems Modeling:** An approach to modeling whose design is to capture the behaviors and interdependencies of the system as a whole.

**Top-Level Entity:** An entity in a discrete-event simulation model that exists at the top level of the entity hierarchy. It is an entity that is not currently a constituent part of any other batch of entities.

**True Defect Analysis:** The process (as performed by simulation models discussed herein) by which age-related weapon defect rates are determined based on the attributes carried by *all* the weapon entities and/or their subsystems in a given weapon population. That is, the entire population (as opposed to a limited sample set) is used in the analysis.

**Unbatching:** The act of disassembling model entities that had previously been batched together. Unbatching operations can bring constituent sub-level entities back up to the top level.

**User/Modeler:** A person who builds and/or uses a simulation model, or who uses a simulation software product to construct simulation models.

## CHAPTER 2

### REVIEW OF RELATED WORK

Complex-wide, systems simulation modeling of the entire nuclear stockpile has only recently been undertaken. Simulation is being leveraged as a way to capture the complex interdependencies encountered in the nationwide network of nuclear weapons-related facilities, and the functions they contribute to the nuclear stockpile effort. However, most current simulation models rely on rudimentary techniques for using model entity information. More advanced, less common capabilities exist in discrete-event simulation modeling, however these capabilities have not been extensively applied to develop more useful methods for using entity information, especially sub-level entity information. Furthermore, no techniques for directly accessing sub-level entity information have been used to improve current nuclear stockpile modeling efforts. This thesis will address this lack by implementing an advanced, seldom-used modeling concept (*direct sub-level entity access*), and showing that it can be of great value in enhancing nuclear stockpile simulation models.

For the purpose of reviewing related work, the thesis topic can effectively be divided into two separate areas:

- 1) Nuclear stockpile simulation modeling
- 2) Sub-level entity access in discrete-event simulation modeling

The following sections in this chapter address these two areas by briefly discussing current nuclear stockpile modeling efforts, methods of entity information access in discrete-event simulation modeling, and capabilities existing in current simulation software products.

## **2.1 NUCLEAR STOCKPILE MODELING**

Behaviors and characteristics of individual nuclear weapon components have been, and continue to be, modeled extensively using complex physics and mathematical models. However, when it comes to modeling the entire stockpile as a whole, less extensive relevant work is encountered – none of which was found in documented open literature.

### **2.1.1 Nuclear Stockpile Simulation Modeling**

Most of the discrete-event simulation modeling relating to the nuclear stockpile has not been focused on the stockpile itself, but rather on specific weapon-related facilities in the DOE weapons complex. In these facility-specific models, the objectives relate to things like optimal glovebox placement (Hench, Olivas, and Finch), localized production planning (Kjeldgaard et al), machine utilization, and other issues addressed by more traditional manufacturing process simulations.

The simulation models that have been developed for taking a system-wide look at the nuclear stockpile itself are relatively immature compared to other facility-specific models and consist largely of concept and developmental models that are typically constrained to a limited sub-set of the entire stockpile. The desire, of course, is to expand

the scope of these simple models to incorporate greater portions of the nuclear stockpile and to address the related issues more effectively (Helm, Boerigter, and Eisenhower). However, many modeling techniques being currently implemented restrict the expandability of such models. This applies to current simulation models built to address stockpile life-extension issues.

### **2.1.2 Nuclear Stockpile, Life-Extension Modeling**

Nuclear stockpile, life-extension issues include scheduling weapon maintenance and component production, determining weapon surveillance and disposition policies, and estimating the reliability of the weapons. Most existing stockpile life-extension models that address some of these issues are static in nature – primarily spreadsheet models (Boerigter). As such, they do not adequately address the dynamic interdependencies, nor take into account a system-wide view of the stockpile that simulation models have the potential to do. However, the existing stockpile life-extension simulation models are fairly limited in breadth and scope. This thesis will show some of the enhancements that can be made to these kinds of models by applying *direct sub-level entity access*. By so doing, the work contributed by this thesis significantly broadens the capability and expandability of stockpile life-extension models.

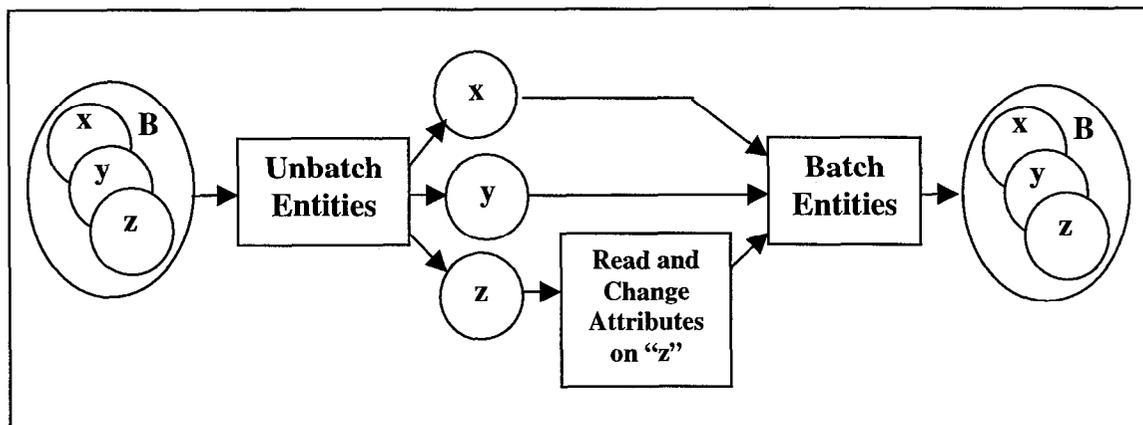
## **2.2 METHODS OF SUB-LEVEL ENTITY INFORMATION ACCESS**

There are various methods for accessing and using entity-specific model information in discrete-event simulation models. This section will outline a few of the most common methods for accessing sub-level entity information, as well as a couple of

less common techniques. Included in the discussion will be how/where the direct sub-level entity access concept (applied in this thesis) fits in relative to the other methods.

### 2.2.1 Common Methods

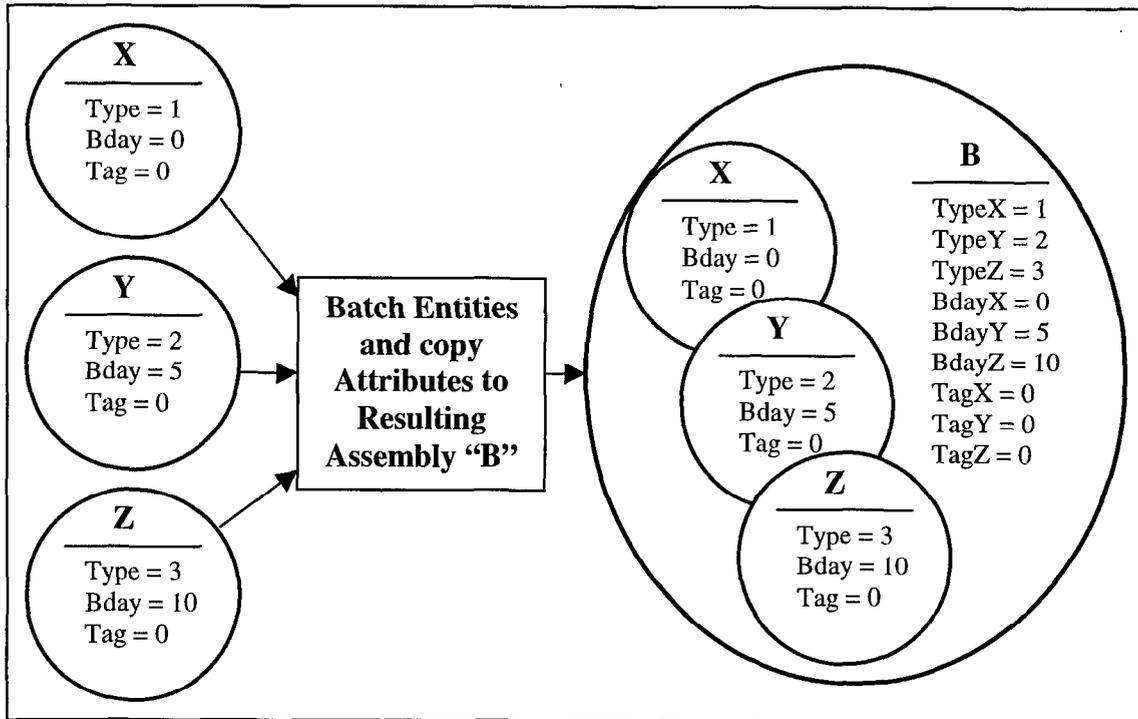
The most common methods for accessing sub-level entity information (i.e. the attributes carried on sub-level entities) involve interacting with or manipulating the information at the top entity level. This is because it is usually assumed that when several entities are batched together into an assembly, the assembly is the primary entity of interest (as opposed to the sub-component entities making up the assembly). If the sub-level entity information within the assembly's sub-components is needed, there are common ways to make it available. This is typically accomplished through an unbatching/batching technique or through some type of attribute-copying technique.



**Figure 2.1** Unbatch/Batch Technique used to Access Sub-Level Entities

In the unbatch/batch technique for accessing sub-level entity information (see Figure 2.1), sub-level entities are brought up to the top entity level by unbatching the assembly. That is, the assembled entity is taken apart, effectively bringing the constituent

entities back up to the top-level (Imagine That, *Extend+MFG* 146). Here the entities and entity information can be accessed like any other top-level entity. After the entity information has been used or changed as needed, the original constituent entities are re-batched into the assembly.



**Figure 2.2** Attribute-Copying Technique

The attribute-copying technique for accessing sub-level entity information (see Figure 2.2) does not directly access the information from the sub-level entities, but it copies the sub-level entity attributes to a different top-level entity (the created batch entity) where the information is more readily accessible. The top-level batch entity essentially contains merged properties of its constituent sub-level entities (Imagine That, *Extend+MFG* 141). For example, when several entities are batched together into an assembly, all the relevant attributes from the constituent entities are copied onto the top-

level entity that represents the assembly (see Figure 2.2). The redundant attributes on the top-level can be viewed and used in the model without digging down into the sub-levels of the batched entity. However, if the attributes on the top-level are modified, the changes are not automatically reflected on the sub-level entities.

Neither the unbatch/batch nor attribute-copying techniques involve *directly* accessing the information from the sub-level entities themselves *while* they are in the sub-level state. They both require more complicated procedures just to get the entity information in an accessible, top-level state. This is where the *direct sub-level entity access* concept applied in the thesis differs from these other common methods. *Direct sub-level entity access* bypasses all the extra steps and accesses the entity information directly without any extraneous entity manipulation.

### **2.2.2 Database Concept**

Implementing a separate custom database or database-type structure in a discrete-event simulation model is not nearly as common a technique for managing and using sub-level entity information as the previous methods discussed. Although most simulation software packages automatically manage entity information in built-in data structures, this database concept essentially entails the user setting up and managing a separate set of entity information (including all sub-level entity information) in some type of database form. The user-defined database would then be the means whereby access to sub-level entity information takes place. This concept is somewhat similar to the attribute-copying technique mentioned in section 2.2.1, in that the information in the database is a top-level copy of information that otherwise exists at the various entity levels. The database, in this

case, would be the interface to this model information, essentially replacing the data management interface already existing in the simulation software.

Some simulation software packages have links to commercial database or spreadsheet packages (Gobel and Wood; Siprelle, Phelps, and Barnes), although this functionality is mostly used for holding model input and output data; not real-time management of entity information. Some simulation software also allows the user to program their own data structures (essentially a user-defined, pseudo-database within the software itself), which can be used to manage whatever information the modeler chooses (Imagine That, *Extend*). However, in either case, the modeler has to do a lot of extra programming and setup to establish the necessary links between the model and the database to effectively manage entity-specific information.

Like the more common techniques of section 2.2.1, the database concept is in some ways an *indirect* way of accessing sub-level entity information. It is “indirect” because it involves working with a set of information that exists independent of the built-in information management structures already provided in simulation software to track and maintain entity data. Assuming the entity information already exists and is maintained in the simulation environment, the database would be a separate, perhaps redundant, source that would have to be maintained independent of the software’s existing model information manager. This, again, is a primary difference between the database concept and the modeling concept applied in this thesis (*direct sub-level entity access*). The direct sub-level entity access technique involves accessing entity-specific information *directly* from the sub-level entity itself in real-time, from where the software

already manages the information, while in the sub-level state. The other methods take more inefficient, roundabout approaches to accessing sub-level entity information.

The database concept may be advantageous in some discrete-event simulation situations, but is not present in any simulation models used in the thesis. It will not, therefore, be compared against *direct sub-level entity access* nor addressed further by this thesis.

### **2.2.3 Direct Methods**

The ability to access sub-level entity information *directly* from the sub-level entities themselves while they are in a sub-level state is the focus of the concept applied to nuclear stockpile, life-extension models in this thesis. This concept is not very common and is sometimes considered an esoteric function in discrete-event simulation modeling (Sadowski). The one instance identified in the literature where a similar concept is frequently applied is in the case of the Visual Simulation Environment (VSE), a simulation software product from Orca Computer, Inc. The similar concept is called *dynamic object decomposition*.

Dynamic object decomposition in the Visual Simulation Environment is an architecture for creating dynamic, hierarchical object (entity) structures “in order to achieve the modeling paradigm *What You See is What You Represent*” (Balci et al, *Dynamic 70*). It has been applied in various simulation models primarily to *visually* represent a hierarchically based, real-world system more accurately. To illustrate the concept, Balci states the following:

In VSE, a dynamic object X can move into another dynamic object Y, move within the hierarchical structure of Y, while it is moving together with Y. For example, a cellular phone dynamic object enters into pocket P component of a passenger dynamic object who enters into a metro train and moves within it while the train is in motion. Such capability provided by VSE enables the user to create direct and natural model representations without any twisting of logic. (*Dynamic 70*)

To achieve movement of objects within other moving objects, a message passing capability in VSE exists that allows access to anything in the object hierarchy (Balci, *Questionnaire*). This is where the framework in VSE is essentially analogous to the direct sub-level entity access concept applied in this thesis. Both enable direct access to the sub-level entities in a hierarchical entity structure. While the concepts are similar, no current application of the VSE technique to nuclear stockpile simulation models was found (the area to which *direct sub-level entity access* will be applied in this thesis).

### **2.3 SURVEY OF VENDORS**

As part of the exercise of reviewing work associated with the thesis topic, various discrete-event simulation software vendors were surveyed. This was done to identify existing capabilities relating to the direct sub-level entity access concept. Specific software vendors were selected by consulting experts in the nuclear stockpile simulation modeling arena, visiting vendor web sites, and reviewing published simulation software surveys. The final group of vendors ultimately surveyed was determined by identifying those discrete-event simulation software products typically used for building

manufacturing-oriented systems simulations (like the existing nuclear stockpile, life-extension models). A few other general-use simulation packages that can be applied in a similar way were also included.

A questionnaire (included in Appendix A) was developed to gather specific information relating to direct sub-level entity access capabilities. This, along with an explanatory cover letter, was sent to specific individuals (who had been pre-identified and pre-contacted) at the vendor companies. Of the 16 vendors surveyed, 14 responded with completed questionnaires addressing the capabilities of 16 different simulation software products. Based on the questionnaire responses and subsequent follow-ups with vendors, some inferences about currently available capabilities were made.

In most of the software packages, some capabilities can be developed by the user to allow some degree of direct sub-level entity access. Limited forms of the capability come built-in to only a few of the 16 products, and only one product (discussed previously in section 2.2.3) applies the capability with any regularity. Overall, direct sub-level entity access is a concept seldom exploited in discrete-event simulation modeling and not very conveniently available. In fact, one vendor did not even think that modelers would benefit from, or see as useful, capabilities allowing direct sub-level entity access. However this thesis will demonstrate that direct sub-level entity access can be very useful and effectively applied to enhance nuclear stockpile simulation models (and, hence, worth the effort).

## CHAPTER 3

### METHODS

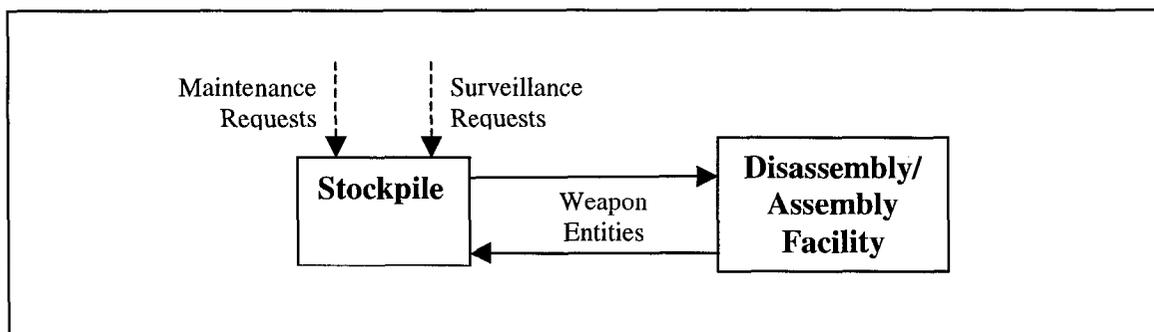
The primary purpose of this thesis is to show that using direct sub-level entity access in discrete-event simulation modeling is useful and often preferable in nuclear stockpile, life-extension models. To do this, the following general steps were taken – each of which is addressed in detail in subsequent sections of the chapter:

- 1) An existing nuclear stockpile, life-extension model was obtained.
- 2) Specific, relevant deficiencies were identified in the model where *direct sub-level entity access* could potentially be applied.
- 3) Existing concept models that attempted to address the deficiencies using common modeling methods were obtained.
- 4) *Direct sub-level entity access* was implemented in the concept models replacing the more common modeling techniques.
- 5) The implementations were analyzed and compared to the original concept models.
- 6) Using information established from the analysis of the concept models, *direct sub-level entity access* was applied to help expand and address the deficiencies in the original nuclear stockpile, life-extension model (step 1).

### 3.1 STEP 1 – EXISTING NUCLEAR STOCKPILE MODEL (2X2 MODEL)

An existing discrete-event simulation model, herein referred to as the “2x2 model,” was built at Los Alamos National Laboratory using the Extend™ simulation software package (Imagine That, *Extend*). The model was designed to demonstrate and evaluate the effect of specific surveillance and maintenance schedules on weapons in the nuclear stockpile. A brief description of relevant portions of the model follows.

The 2x2 model represents the US DOE nuclear weapons complex at a high level, including representations of several facilities, organizations, and locations involved in the design, production, maintenance, storage, and disposition of the weapons in the nuclear stockpile. The model essentially simulates the movement of two weapon types around the complex and tracks vital information (attributes) on the weapons as they move and are acted upon during the simulation. For the purposes of this thesis, the primary elements of interest in the system are associated with the movement of weapons in response to maintenance and quality assurance (surveillance) requests. Figure 3.1 illustrates the basic flow diagram of this portion of the model.



**Figure 3.1** 2x2 Model – Basic Entity Flow Diagram

As shown in Figure 3.1, scheduled maintenance and surveillance requests are made to the stockpile for a specific set of weapons. Those weapons leave the stockpile and go to the assembly/disassembly facility. The required weapon disassembly, maintenance, and re-assembly are performed. Surveillance testing (quality assurance testing performed on a limited set of the weapon population) is also conducted at the facility when appropriate. The serviced weapons then return to the stockpile. More specific descriptions and assumptions relating to the basic elements of Figure 3.1 follow.

### **3.1.1 Weapons**

The primary entities of interest in the model are the weapons. In the 2x2 model there are two weapon types included, each made up of two subsystems (hence the 2x2 model designation). The weapons are represented by top-level entities with attributes identifying the characteristics of both the assembled weapon as well as the subsystems making up the weapon assembly. Hence, the entity structure representing a weapon in the model is consistent with Figure 2.2, where all relevant information is copied to and maintained on the top-level entity. Some of the most important attributes carried by the weapon entities are related to the “age” of the weapon and its associated subsystems.

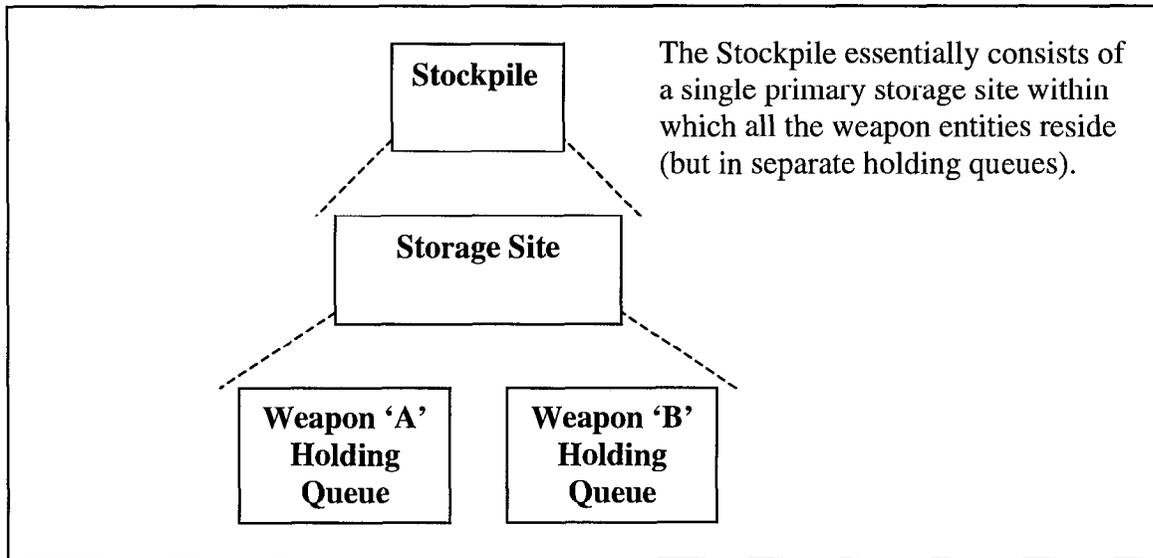
### **3.1.2 Maintenance and Surveillance Requests**

Maintenance and surveillance requests are sent independently to the stockpile to initiate the movement of certain weapons from the stockpile to the assembly/disassembly facility. The requests are sent according to a schedule. Each request generally specifies the number of weapons of a particular type that are requested, the type of work that is to

be performed (i.e. changing a subsystem or just doing surveillance testing), and the particular weapons to be sent (i.e. the oldest or a random selection). For example, a given maintenance request might be to “send the 20 type ‘A’ weapons that have the oldest ‘X’ subsystems to the maintenance facility to have the ‘X’ subsystem replaced.” Note that the request does not necessarily contain unique information about individual weapons or exactly where a particular weapon should come from. The request just asks for weapons meeting certain characteristics – regardless of where the weapons are actually located in the model.

### **3.1.3 Stockpile**

In the “stockpile” section of the 2x2 model (refer to Figure 3.1), the weapon entities are actually kept segregated, according to type, while they are waiting for use. That is, the two weapon entity types are housed in separate holding queues within a given storage site in the stockpile. Furthermore, another simplifying assumption is that the stockpile is not spread out among many storage sites, but consolidated into a single storage site instead (see Figure 3.2). This was all done so that the maintenance and surveillance requests for a particular weapon type could be handled more easily. If both weapon entity types were held in the same holding queue and/or distributed among several different storage sites, it would be much more difficult (using the common modeling constructs applied in the 2x2 model) to properly sort out the appropriate weapon entities when a given request arrived.



**Figure 3.2** Simplified Stockpile Representation in 2x2 Model – Entity Segregation

### 3.1.4 Assembly/Disassembly Facility

In the 2x2 model, when a weapon arrives at the assembly/disassembly facility, it is serviced and/or surveillance tested according to the original request. When maintenance is performed, an old subsystem may be replaced with a newer one. When surveillance testing is performed, the age of the weapon or an associated subsystem is calculated based on an attribute carried by the weapon entity, and this information is used to determine whether an age-related defect exists. This data is compiled during the simulation from the relatively small sample set of weapon entities that are actually surveyed to estimate the defect rates in the entire weapon entity population.

## 3.2 STEP 2 – DEFICIENCIES IDENTIFIED IN THE 2X2 MODEL

The primary deficiencies identified in the 2x2 model include the limited form of doing defect analysis (surveillance only), the oversimplified representation of the stockpile (single storage site with entities segregated by type), and only having two

weapons and two subsystems addressed in the model. These deficiencies highlighted the need to expand the capabilities of the model so that it could address three new required objectives:

- 1) Performing true defect analysis
- 2) Modeling multiple storage site locations
- 3) Including more weapon and subsystem types

These objectives are briefly discussed in the following subsections.

### **3.2.1 True Defect Analysis**

“True” defect analysis involves checking *every* weapon and/or weapon subsystem for age-related defects (as opposed to checking just a few weapons as in surveillance testing). While resource and other constraints prevent this in real life, having the model perform true defect analysis – independent of the surveillance defect analysis done at the disassembly/assembly facility – provides very useful insights. It can be used, for example, to help determine how well a particular surveillance program reflects the actual condition of the stockpile.

### **3.2.2 Multiple Storage Sites**

The 2x2 model essentially models the stockpile as a single storage site where all the weapons reside. This makes the task of modeling maintenance and surveillance requests much easier. However, this oversimplifying assumption limits the usefulness of the model. In reality, the stockpile weapons are distributed among many sites. To account for the complexities involved with coordinating between sites, the model must represent

multiple sites and be able to schedule maintenance/surveillance effectively – independent of which site holds a particular weapon.

### **3.2.3 Additional Weapon and Subsystem Types**

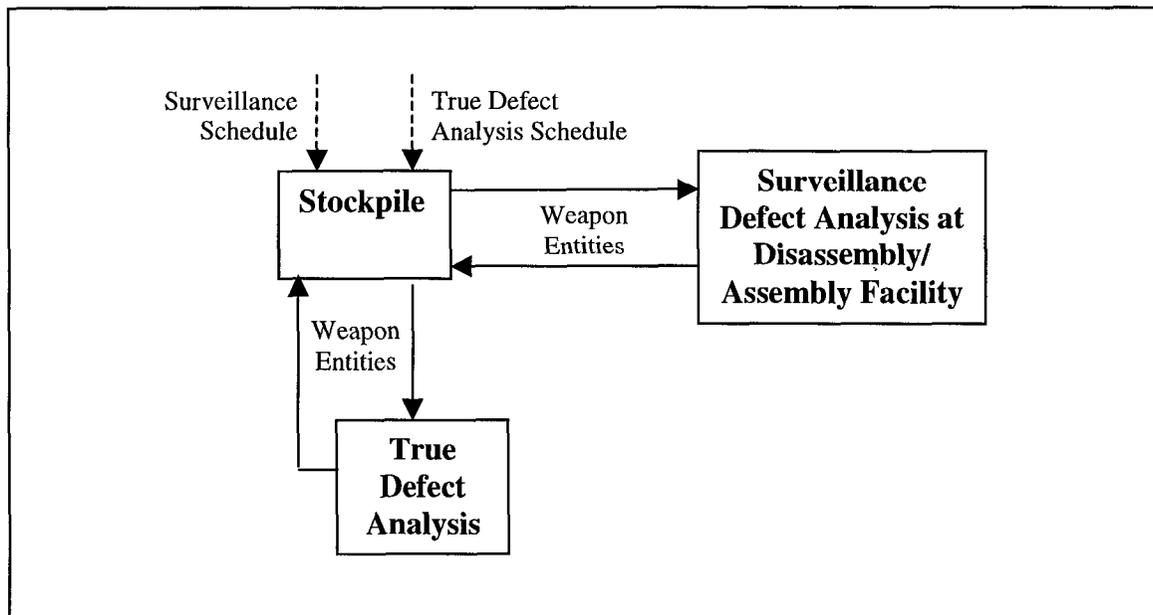
Adding more weapon and subsystem types further complicates the previous two objectives. The 2x2 model only tracks two weapon types (each with two constituent subsystems). More weapon types require more holding queues at each storage site just to keep all the types segregated. This undesirable situation can be alleviated if all the weapons (independent of type) at a given storage site reside in a single holding queue. More subsystem types require tracking more attributes. In the 2x2 model all relevant attributes are maintained on the top-level weapon entity. However, modeling many additional subsystems makes maintaining attributes for all of them on the top-level entity cumbersome and undesirable. Consequently, a true hierarchical entity structure, where the attributes for sub-level entities (the subsystems) are actually maintained and accessed on the sub-level entities themselves, is more efficient.

### **3.3 STEP 3 – CONCEPT MODELS OBTAINED**

Two different existing concept models were obtained. The purpose of these two models was to propose ways to address, in part, the objectives discussed previously – namely enabling true defect analysis, accounting for multiple weapon sites, and including more weapon and subsystem types. Both concept models employ only common modeling techniques and are described below. (These two models are hereafter considered the baseline cases for comparative analysis purposes.)

### 3.3.1 True Defect Analysis Concept Model (Baseline-1)

The baseline case of the “True Defect Analysis” concept model (denoted “Baseline-1”) demonstrates a method for performing true defect analysis on weapons in the stockpile (separate from continuing surveillance defect analysis) using common modeling techniques. Figure 3.3 illustrates the general flow of the model.

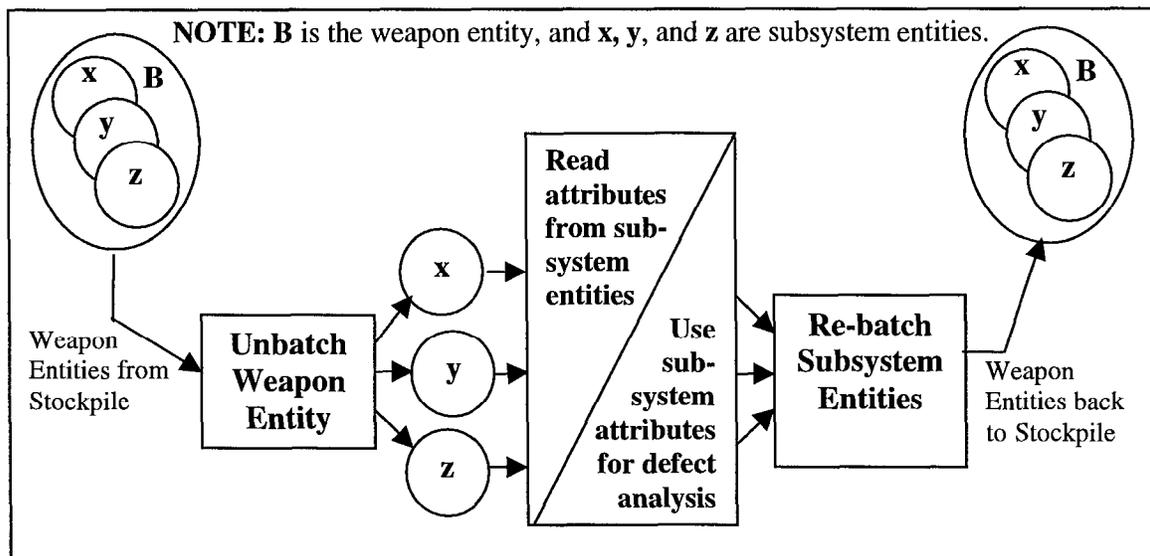


**Figure 3.3** True Defect Analysis “Baseline-1” Model – Basic Flow Diagram

In the Baseline-1 model, a small sample of select weapon entities move from the stockpile to the disassembly/assembly facility according to the schedule for surveillance defect analysis (similar to the 2x2 model). After the weapons are tested, they return to the stockpile. Periodically, a true defect analysis is performed by triggering the release of all weapon entities from the stockpile. The common modeling methods employed for doing true defect analysis require that all of the weapons entities be moved out of their stockpile holding queues and routed to a designated point in the model. This stockpile movement is

required so that the appropriate attributes can be accessed on all the weapon entities to determine the “true” defect profile of the entire weapon population.

There are three weapon subsystem types represented in the Baseline-1 model. Also, the weapon entities are hierarchically structured such that the weapon assembly is represented by a top-level entity, and the subsystems within the weapon are represented by different sub-level entities. The attributes on the sub-level entities need to be accessed whenever a defect analysis is performed. To do this, the common unbatch/batch technique (discussed in section 2.2.1) is used. For every weapon entity that is analyzed (by either surveillance or true analysis), Figure 3.4 illustrates the process required to obtain the necessary information. Each time a weapon is checked for defects, it must be unbatched to reveal its constituent subsystem entities, and then re-batched after the appropriate subsystem information has been read from the sub-level attributes.



**Figure 3.4** Accessing Sub-Level Attributes for Defect Analysis (“Baseline-1” Model)

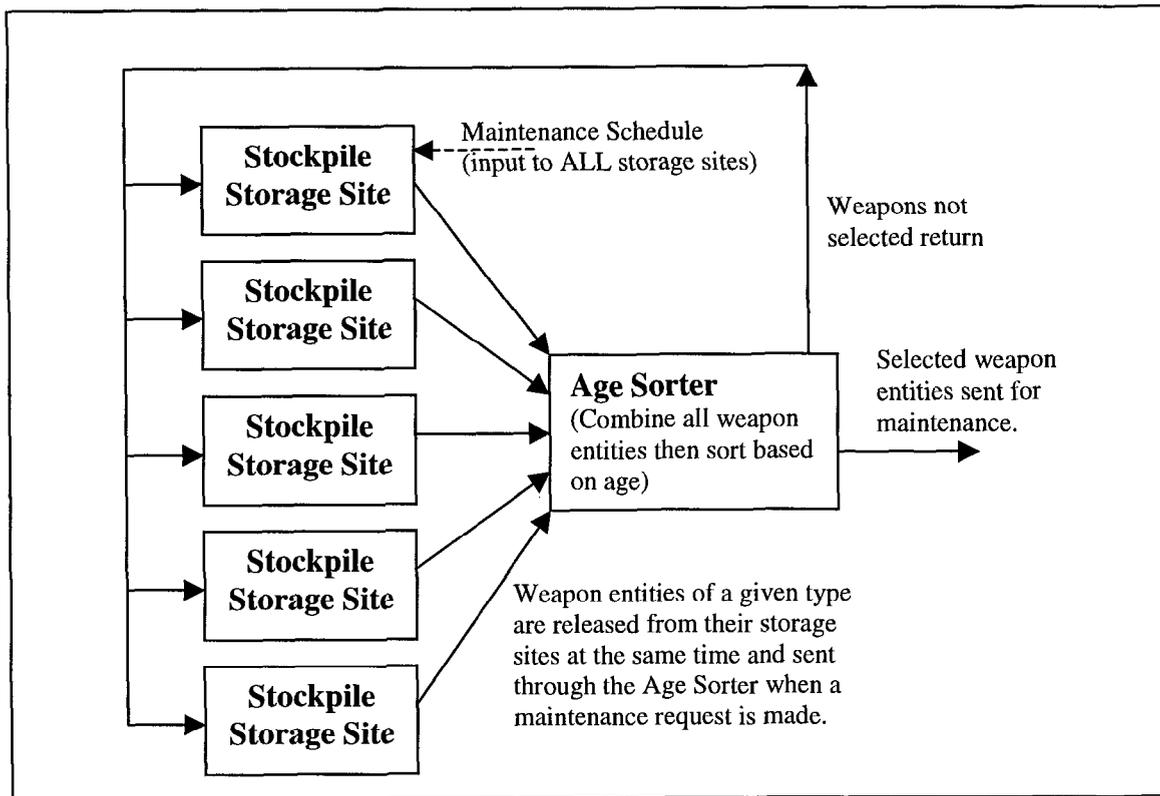
The surveillance defect analysis generates defect profiles for each weapon subsystem based on the small sample of weapons routed to the assembly/disassembly facility. The true defect analysis generates defect profiles for each subsystem based on the entire weapon population.

### **3.3.2 Multiple Storage Sites Concept Model (Baseline-2)**

The baseline case of the “Multiple Storage Sites” concept model (denoted “Baseline-2”) demonstrates a method for having the stockpile divided among multiple storage sites while still being able to select weapons according to a given maintenance or surveillance schedule using common modeling techniques.

In this model a typical maintenance request asks for a specific number of weapons that are the oldest of a specified type (determined by an age attribute carried by each weapon entity). However, since like weapon entity types are distributed among multiple storage sites, something must be done to sort through and select from the collective population of the given weapon type. This is not easy to do using common modeling techniques unless all the weapons are brought to a single point and considered together. The flow diagram of how the Baseline-2 model does this is illustrated in Figure 3.5.

When a maintenance request is made, all the weapons of a particular type are released from the various weapon sites at the same time. They are routed to a single point in the model so an age-related attribute on each weapon can be read. The oldest are identified and sent for maintenance, and the others are returned to their original storage site locations.



**Figure 3.5** Multiple Storage Sites “Baseline-2” Model – Basic Flow Diagram

While there are three weapon subsystem types represented, the Baseline-2 model regresses by having all relevant subsystem attributes maintained on the top-level weapon entity (similar to the attribute-copying technique discussed in section 2.2.1). This was done to prevent the model from being further burdened by digging down (unbatching) to entity sub-levels every time sorting/selection was done.

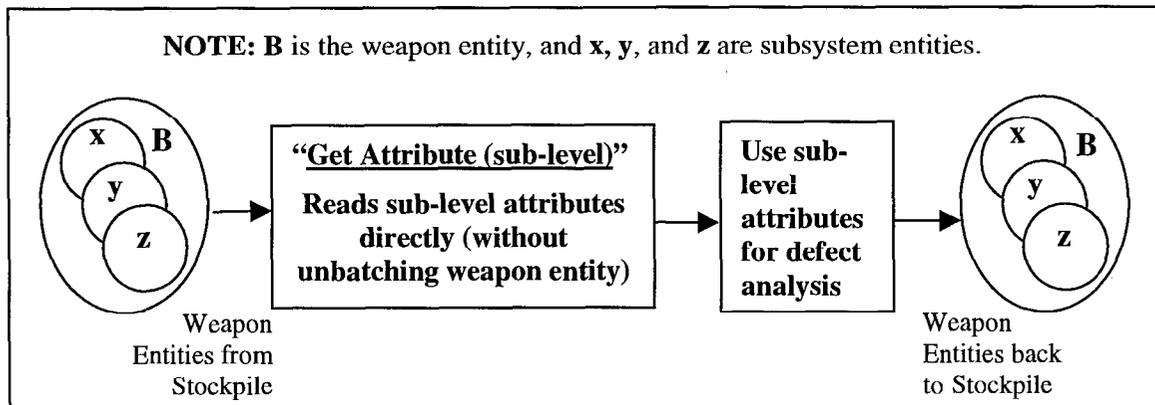
### **3.4 STEP 4 – CONCEPT MODELS MODIFIED**

The two concept models (Baseline-1 and Basline-2) described in the previous section employed common modeling techniques to address specific deficiencies identified in the 2x2 model. These baseline models were then modified to incorporate

*direct sub-level entity access* as an alternative to addressing the deficiencies. The following sections describe the modified versions of the baseline models.

### 3.4.1 True Defect Analysis Concept Model (Mod-1A)

The first modified case of the True Defect Analysis Baseline-1 model is denoted “Mod-1A.” The Mod-1A model is the same as the Baseline-1 model except where the weapons are analyzed for defects in the “True” and “Surveillance” analysis sections (refer to Figure 3.3). Instead of unbatching the weapon entities to obtain the sub-level information, (as in the Baseline-1 case shown in Figure 3.4), a special functional block was created to get the sub-level information without bringing the sub-level entities to the top level. Figure 3.6 illustrates this modified process.



**Figure 3.6** Accessing Sub-Level Attributes for Defect Analysis (“Mod-1A” Model)

As a weapon entity passes through the special block, the appropriate sub-level attributes are read from the appropriate sub-level entities (x, y, z) without unbatching the weapon entity. The special block that does this (called *Get Attribute (sub-level)*) represents a *local* implementation of the direct sub-level entity access concept. It is *local*

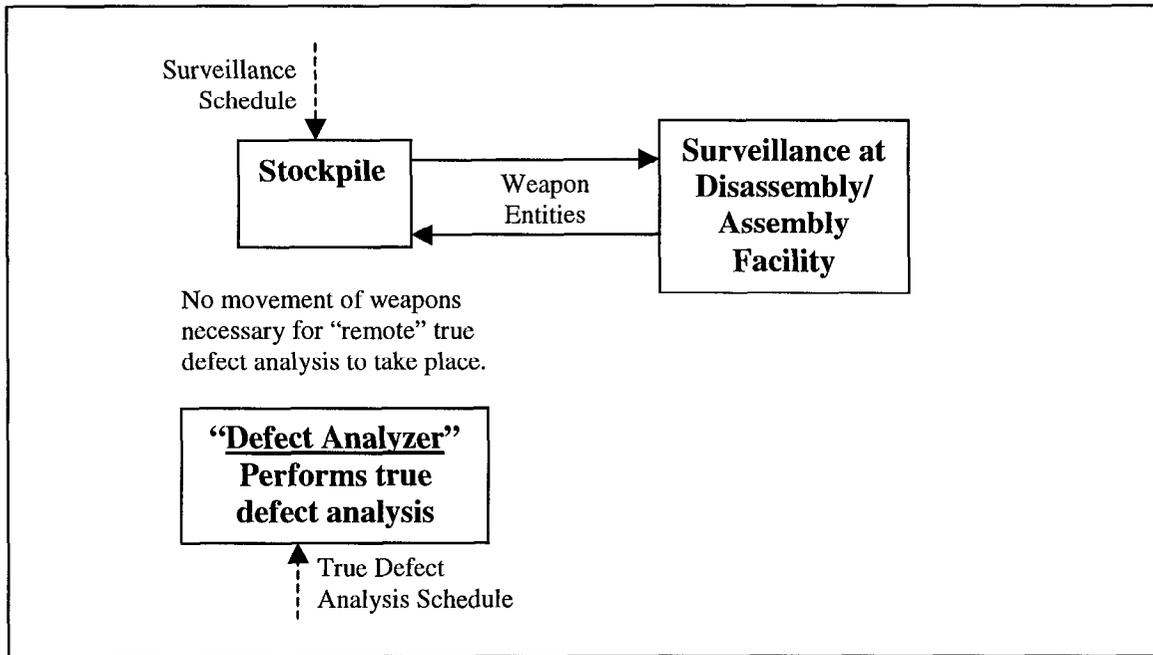
in the sense that sub-level entity attributes can only be accessed on weapon entities that actually pass through the block. The *Get Attribute (sub-level)* block dialog (i.e. the block's basic user interface) is displayed in Appendix B.

### 3.4.2 True Defect Analysis Concept Model (Mod-1B)

The second modified case of the True Defect Analysis Baseline-1 model is denoted "Mod-1B." The Mod-1B model is the same as the Mod-1A model except in the "True" defect analysis section (refer to Figure 3.3). The "Surveillance" defect analysis section remains as a *local* implementation of *direct sub-level entity access* (Figure 3.6). This is because the surveillance section models a real-world occurrence where weapons have to actually move to a designated place (the disassembly/assembly facility) to be analyzed. However, the "True" defect analysis section does not necessarily model a real-world, physical transfer of all the weapons to a physical location for analysis. Instead, it is intended to be a "virtual" analysis of the entire stockpile where no weapons actually move. Consequently, a *global* implementation of the direct sub-level entity access concept was used for the "True" defect analysis section of the model. The basic Mod-1B flow diagram is illustrated in Figure 3.7.

As shown in the figure, weapon entities do not have to leave the stockpile to be analyzed. The true defect analysis section that was in the baseline model is replaced by a special functional block (called *Defect Analyzer*) that was created to handle *global* direct sub-level entity access. This block directly accesses the sub-level attributes of the weapon entities *remotely*, independent of where the *Defect Analyzer* block is or where the weapon entities are in the model. Hence, it can globally access the necessary information from the

sub-level entities themselves without physically disturbing or moving the weapon entities. The *Defect Analyzer* block dialog (i.e. the block’s primary user interface) is displayed in Appendix C.



**Figure 3.7** True Defect Analysis “Mod-1B” Model – Basic Flow Diagram

### 3.4.3 Multiple Storage Sites Concept Model (Mod-2)

The only modified case of the Multiple Storage Sites Baseline-2 model is denoted “Mod-2.” The Mod-2 model is the same as the Baseline-2 model except in the way that weapon entities are sorted and selected for surveillance or maintenance. The Mod-2 model does not rely on attribute-copying techniques to access sub-level information. The model takes advantage of the model’s hierarchical entity structure and accesses information directly from sub-level entities as needed. Instead of routing all weapons of the requested type from all storage sites to a central place just to sort out the oldest (as in

Figure 3.5), it sorts and selects without moving any weapon entities at all. To achieve this enhanced functionality, a special functional block was created to select the appropriate weapons *remotely* based on the surveillance/maintenance request information. The block uses a *global* implementation of the direct sub-level entity access concept to access the sub-level entity information remotely without having to move any weapon entities from their holding queues. The Mod-2 flow diagram is illustrated in Figure 3.8.

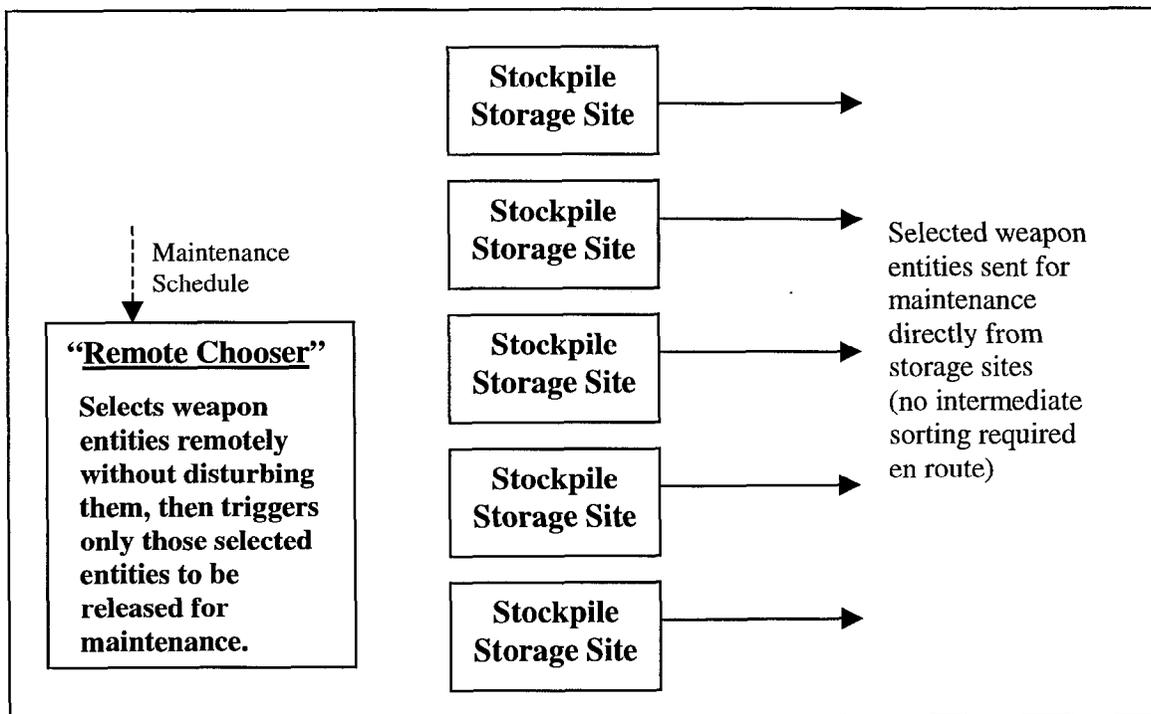


Figure 3.8 Multiple Storage Sites "Mod-2" Model – Basic Flow Diagram

In this case, weapon entities do not have to leave the stockpile storage sites to be sorted and selected by age. The new special functional block (called *Remote Chooser*) directly accesses the sub-level attributes of the weapon entities independent of where the *Remote Chooser* block is or where the weapon entities are in the model. Hence, it can globally access the necessary information from the sub-level entities themselves without

physically disturbing or moving the weapon entities. The *Remote Chooser* block has a global view of all entities in the model and selects the ones that meet the requested criteria. It can then trigger the release of the appropriate weapon entities. The *Remote Chooser* block dialog (i.e. the block's primary user interface) is displayed in Appendix D.

### **3.5 STEP 5 – CONCEPT MODELS ANALYZED AND COMPARED**

Three different scenarios representing different input conditions were run for each model. The different models were run using the same input scenarios, then compared and analyzed with each other as appropriate. The True Defect Analysis Baseline-1 model was compared against the True Defect Analysis Mod-1A and Mod-1B models. The Multiple Storage Sites Baseline-2 model was compared against the Multiple Storage Sites Mod-2 model. While the results and analysis will be discussed in detail in chapter 4, a brief discussion of some of the methods and metrics used in the analysis are given below.

#### **3.5.1 Comparative Analysis Methods**

Several actions were taken to ensure that the models were tested under the same operating conditions and yielded the same output (within a given input scenario). The same random seed values were used in each case to help verify that the output could be replicated and to assure that comparative differences were not the result of stochastic variations between the simulation runs. The models were all run for the same simulated elapsed time (5000 weeks). Three different input scenarios were run for each model with the same input scenarios being used between each compared case. The simulation output was compared and verified in each case to assure that each scenario of the baseline and

corresponding modified models were performing the same functionality and giving the exact same output results.

The reason for assuring that the output was the same for each compared case was because the desired comparison was between the modeling methods – not the model output data. Had the simulation results not been equivalent for each of the compared cases, the quantitative metrics used in the comparisons would not have been valid (i.e. a “level playing field” would not have been assured).

After the models were verified to be functionally equivalent and operating under the same conditions, the models were run again to gather data and inferences relating to the metrics listed in the next section. From the results, relative advantages, benefits, and disadvantages were established for using *direct sub-level entity access* (as in the modified models) compared with common modeling methods (as in the baseline cases).

### 3.5.2 Analysis Metrics

Table 3.1 lists the quantitative and qualitative metrics used in the model analyses.

**Table 3.1** Analysis Metrics

<b>Metric</b>	<b>Type</b>	<b>Description</b>
Run Time	Quantitative	Clock time in which the model runs. Simulation speed.
Size	Quantitative	Size (memory usage) that a given implementation contributes to the model.
Complexity	Qualitative	How convoluted/difficult an implementation/model is.
Scalability	Qualitative	How easily expandable a given implementation or model is in response to adding modeling components and constructs (like storage sites or weapon and subsystem types).
Flexibility/ Functionality	Qualitative	How easily adaptable to change or to different configurations an implementation or model is. The limitations/constraints (or alternatively strengths/ options) associated with an implementation or model.

The *Run Time* metric is a measure of how fast the simulation runs in terms of real clock time, as opposed to simulation time. The run time is a vital issue for analysts using a simulation model. The time it takes to run a model is an important factor that directly affects the ability to deliver timely results based on model analysis. Any improvement in run time (without adversely affecting the simulation behavior) is always desirable.

In the case of the concept models analyzed in this thesis, the run time is a time duration based on Equation 3.1.

$$\text{Run Time} = \text{End Time} - \text{Start Time} - \text{Initialization} \quad (3.1)$$

**End Time** is the time at which simulation execution ends, **Start Time** is the time at which simulation execution begins, and **Initialization** is the amount of time the simulation spends initializing the model and is characterized by Equation 3.2.

$$\text{Initialization} = \text{Last Entity Arrival Time} - \text{Start Time} \quad (3.2)$$

The models start empty with no entities present. During initialization, the appropriate entities are generated and placed in the proper holding queues in the model (i.e. the original weapon entities are created and loaded into their respective stockpile storage locations). This puts the model in the proper starting state. The arrival of the last initialized entity to its appropriate holding queue (**Last Entity Arrival Time**) signals the end of the initialization period. The initialization period is not included in the run time since initialization takes the same amount of time in each compared case, is very small

(less than 0.5 seconds), and has nothing to do with the impact that the various modeling techniques have on execution speed.

The run time calculation was implemented in code and integrated into the simulation models so that when a simulation run concluded, the run time would automatically be calculated and displayed. Each model case was run five times (five replications) with the run time being recorded for each replication. The sample mean ( $\bar{X}$ ) and standard deviation ( $s$ ) of the run times were calculated based on Equations 3.3 and 3.4 respectively:

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n} \quad (3.3)$$

$$s = \sqrt{\frac{\sum_{i=1}^n [X_i - \bar{X}]^2}{n-1}} \quad (3.4)$$

where  $X_i$  is the run time value observed for each of the individual simulation replications and  $n$  is the number of replications (sample size). The results are included in chapter 4.

The *Size* metric is a quantitative measure of how much a given modeling construct contributes to the size of the model (in terms of bytes). This value is provided automatically in the simulation software by selecting the desired portion of the model to be measured. The memory consumed (number of bytes) by that portion of the model is then displayed.

In the concept models, the sections specifically relating to accessing sub-level entity information for true defect analysis and weapon selection were selected and sized. This allowed the different modeling techniques to be compared according to size. This comparison was valuable in helping to establish and illustrate which methods were more size efficient. Since the sizes of the given modeling constructs are constant between simulation replications, no statistical quantification on the size metric was performed.

### **3.6 STEP 6 – 2X2 MODEL IMPROVED (7X7 MODEL)**

Taking what was learned from the comparative analyses of the concept models, the 2x2 model was improved and expanded using direct sub-level entity access techniques. The new, improved model is hereafter referred to as the “7x7 model.” The 7x7 model does everything the 2x2 model does and more. It performs true defect analysis on the stockpile weapon entities, models several different storage sites with all weapons distributed appropriately among them, and tracks seven weapon types, each with seven constituent subsystems (hence the 7x7 designation). Additionally, the weapon types are no longer segregated into different holding queues at each storage site. All weapons at each site reside in a single holding queue.

The entity structure in the 7x7 model is hierarchical, with all sub-level entities (the weapon subsystems) maintaining their own attributes. True defect analysis is performed in the model using the same *global* implementation of *direct sub-level entity access* applied in the True Defect Analysis Mod-1B model. Sorting and selection of weapons from the multiple storage sites is done remotely using the same *global*

implementation of *direct sub-level entity access* as applied in the Multiple Storage Sites Mod-2 model.

Since the 7x7 model is markedly different than the 2x2 model, the two were not quantitatively compared (since the quantitative metrics used to compare the other models would be meaningless). However, some observations relative to the qualitative metrics were made and will be discussed in chapter 4.

## CHAPTER 4

### RESULTS AND ANALYSIS

The results supporting the use of *direct sub-level entity access* to improve nuclear stockpile simulation modeling come primarily from the comparative analyses of the concept models described in the previous chapter. This current chapter presents the results and observations from the analyses and discusses additional implications relating to the use of *direct sub-level entity access* in nuclear stockpile models. The discussion will be divided into the following sections:

- 1) Comparison of the True Defect Analysis models
- 2) Comparison of the Multiple Storage Sites models
- 3) Comparison of the 2x2 and 7x7 models
- 4) Potential disadvantages of implementing *direct sub-level entity access*.

#### 4.1 TRUE DEFECT ANALYSIS MODELS

The True Defect Analysis Baseline-1 concept model was compared against the two corresponding modified cases (Mod-1A and Mod-1B). Both quantitative and qualitative metric comparisons were performed to identify the advantages of using *direct sub-level entity access*.

To provide an ample base for comparative observation and analysis, three scenarios were set up under which all the models were run to explore the behaviors under

different input conditions. All input parameters were the same in each of the three scenarios with the exception of how often true defect analysis was performed. Table 4.1 lists the parameter values that varied between the scenarios.

**Table 4.1** Scenario Parameter Values for the True Defect Analysis Models

	<b>Models Perform True Defect Analysis Every ...</b>
<b>Scenario 1</b>	13 weeks
<b>Scenario 2</b>	26 weeks
<b>Scenario 3</b>	52 weeks

The three scenarios represent situations where true defect analysis is conducted quarterly, semi-annually, and annually.

#### **4.1.1 Quantitative Analysis**

Table 4.2 lists the average run times (as well as the standard deviations) of the three True Defect Analysis models for each of the scenarios tested. The Baseline-1 model implements the common unbatch/batch technique for doing true defect analysis, while the Mod-1A and Mod-1B models represent *local* and *global* implementations of *direct sub-level entity access* used to accomplish true defect analysis. The values in the table are based on five simulation replications of each model/scenario. As mentioned previously in chapter 3, all the models produced identical model output data within a given scenario. This allowed the modeling methods to be compared as opposed to the output data.

**Table 4.2** Run Time of the True Defect Analysis Models

	<b>Baseline-1 Model</b>	<b>Mod-1A Model</b>	<b>Mod-1B Model</b>
<b>Scenario 1</b> (Avg. / Std Dev)	97.2 seconds <i>0.10 seconds</i>	56.9 seconds <i>0.07 seconds</i>	9.4 seconds <i>0.00 seconds</i>
<b>Scenario 2</b> (Avg. / Std Dev)	50.0 seconds <i>0.05 seconds</i>	30.5 seconds <i>0.05 seconds</i>	5.8 seconds <i>0.04 seconds</i>
<b>Scenario 3</b> (Avg. / Std Dev)	27.2 seconds <i>0.05 seconds</i>	16.1 seconds <i>0.05 seconds</i>	4.1 seconds <i>0.00 seconds</i>

One important thing to note about the run times of the models is that there is very little spread in the observed values for any given case (reflected by the standard deviations). These variations are statistically insignificant for the purposes of comparing run times.

The *local* implementation of *direct sub-level entity access* applied in the Mod-1A model resulted in a run time that was between 1.6 and 1.7 times faster than the baseline case (depending on which scenario was run). The *global* implementation of *direct sub-level entity access* applied in the Mod-1B model produced a run time that was between 6.6 and 10.3 times faster than the baseline case (again, depending on the scenario).

The various scenarios are presented to illustrate the point that different models and modeling conditions will affect – to different degrees – the run time improvements achieved from implementing *direct sub-level entity access*. However, the trend holds that significant run time improvements still result from implementing *direct sub-level entity access* in place of more common modeling methods. The actual amount of improvement is model dependent.

Table 4.3 shows the relative size differences (in terms of memory usage) between the unbatch/batch implementation of doing true defect analysis in the baseline case, and the *local* and *global* direct access implementations employed in the Mod-1A and Mod-1B

cases respectively. Since the size of the constructs remain constant between all runs, no reference to the different scenarios is necessary.

**Table 4.3** Size Contributed to Model by each True Defect Analysis Implementation

	<b>Baseline-1 Implementation</b>	<b>Mod-1A Implementation</b>	<b>Mod-1B Implementation</b>
<b>Size (Memory Usage)</b>	158 KB	148 KB	2 KB

A relatively small improvement in size of the *local* access implementation (in the Mod-1A model) occurs compared with the very large improvement in size of the *global* access implementation (in the Mod-1B model). This happens because a significant part of the true defect analysis in the Baseline-1 and Mod-1A cases involves coordinating and controlling the routing of entities to a designated location where the sub-level information is actually read and compiled. The only difference between the Mod-1A case and the Baseline-1 case (see section 3.4.1) is that the *local* implementation of direct sub-level entity access (Mod-1A) does not require the weapon entities to be unbatched and re-batched to get the appropriate information. The weapon entities still have to move through a designated point, however, to be accessed. The *global* implementation of direct sub-level entity access (Mod-1B), on the other hand, eliminates the need to move or route the weapon entities to perform true defect analysis. The overhead that goes into coordinating the movement of entities and compiling the resultant sub-level information in the Baseline-1 and Mod-1A cases is eliminated by applying *global direct sub-level entity access*. The *local* implementation still provides some improvements and benefits

over the baseline method in terms of both size and speed; however, the *global* implementation has an even greater positive impact.

With model size being an important consideration in computer simulation modeling, the Mod-1A and Mod-1B implementations of *local* and *global* sub-level entity access illustrate the potential for size improvements that can be obtained by applying *direct sub-level entity access* in place of more commonly used methods (as in the baseline case). The size is related to the number and type of programming variables used in (and, indirectly, the amount of) the underlying code required to achieve a given modeling implementation. A benefit of smaller size is the reduced hardware resources needed to run the simulation (size efficiency). Also, a much smaller implementation usually means much less code to execute, resulting in faster run times (speed efficiency). Although there is not an exclusive correlation between size and speed, both are useful measures in determining the efficiency of a particular setup.

Overall, the speed and size improvements shown in the modified True Defect Analysis models can be attributed to their more direct approach of getting entity information. Applying *direct sub-level entity access* eliminates the many extra execution steps that more common methods require to perform the same function.

#### **4.1.2 Qualitative Analysis**

The implementations of true defect analysis in the baseline model (using the unbatch/batch technique), the Mod-1A model (using *local* direct access), and the Mod-1B model (using *global* direct access) were analyzed according to the qualitative metrics identified previously. A comparison of the three cases is presented in Table 4.4.

**Table 4.4** Qualitative Metric Comparison of Defect Analysis Models

Complexity	<p><b>Baseline-1 (using batch/unbatch technique):</b> Overly complex by requiring each entity to move through a given point in the model with each entity being unbatched to gain access to sub-level attribute information and then re-batched to reestablish the proper entity structure. Complexity increases significantly to account for multiple storage sites or additional weapon and subsystem types.</p>
	<p><b>Mod-1A (using local implementation):</b> No unbatching or batching necessary, though each entity still must be moved through a given place in the model to access sub-level information. Complexity increases significantly to account for multiple storage sites (primarily due to work involved with moving the weapons to the appropriate analysis point). However, accounting for additional weapon and subsystem types is not as complicated as in the baseline case.</p>
	<p><b>Mod-1B (using global implementation):</b> No unbatching or batching necessary. No unnecessary coordination or movement of entities is required to get the sub-level information. Negligible increase in complexity to account for multiple storage sites or additional weapon and subsystem types.</p>
Scalability	<p><b>Baseline (using batch/unbatch technique):</b> Not easily scalable to account for additional stockpile storage sites or additional weapon and subsystem types. Each additional stockpile storage site requires cumbersome modeling logic to coordinate the movement of weapon entities from different sites. Additional logic is also required at the defect analysis point to account for each additional weapon or subsystem type. Significant adverse size and speed impacts result.</p>
	<p><b>Mod-1A (using local implementation):</b> Requires a smaller modeling construct at the defect analysis point to account for additional weapon and subsystem types. However, the <i>local</i> implementation of <i>direct sub-level entity access</i> still requires the additional modeling logic to coordinate movement of the weapons from each stockpile storage site. Hence, size and speed still increase, but less than in the baseline case.</p>
	<p><b>Mod-1B (using global implementation):</b> Very scalable. No modeling logic needed to coordinate movement of entities to the true defect analysis point (since no entity movement is required). Negligible speed and size impact to account for additional storage sites or weapon and subsystem types.</p>
Flexibility/Functionality	<p><b>Baseline (using batch/unbatch technique):</b> Flexibility is very limited and functionally constrained since all weapon entities in the model must be physically moved and manipulated to do true defect analysis. Care must be taken to schedule the true defect analysis when it will not interfere with or coincide with the other functions in the model requiring weapon movement</p>
	<p><b>Mod-1A (using local implementation):</b> Somewhat limited because this implementation is still constrained by how and when entities actually get to the point of analysis. However, there is more functionality and flexibility (more options) in dealing with attributes once the entities arrive at the analysis point.</p>
	<p><b>Mod-1B (using global implementation):</b> Much more flexible and functional. The analysis is not constrained by movement of entities (since no movement is necessary) and more flexibility exists for selecting weapon types and subsystem types for inclusion in the analysis.</p>

Table 4.4 indicates how well each implementation meets the objectives identified earlier for improving the 2x2 nuclear stockpile life-extension model; the objectives being to perform true defect analysis, account for multiple weapon storage sites, and include more weapon and subsystem types. All three cases were designed specifically to demonstrate alternative approaches to true defect analysis, so they all potentially fulfill the first objective. However, as indicated in the table, the complexity and scalability issues involved with accounting for multiple storage sites and additional weapon and subsystem types make the Baseline-1 and Mod-1A implementations of true defect analysis much less desirable than the *global* direct access implementation in Mod-1B. With the *global* approach, all three objectives can be appropriately addressed without the true defect analysis portion of the model becoming unduly complicated or creating an adverse impact on the rest of the model.

## **4.2 MULTIPLE STORAGE SITES MODELS**

The Multiple Storage Sites Baseline-2 concept model was compared against the corresponding modified case (Mod-2). Both quantitative and qualitative comparisons were performed according to the previously defined metrics to identify the advantages of using *direct sub-level entity access*.

Similar to what was done previously in section 4.1, three scenarios were set up under which the two Multiple Storage Sites models were run to provide an ample base for observation and analysis under different input conditions. All input parameters were the same in each of the three scenarios with the exception of how often weapon entities had

to be selected from the multiple storage sites for maintenance. Table 4.5 lists the parameter values that varied between the scenarios.

**Table 4.5 Scenario Parameter Values for the Multiple Storage Sites Models**

	<b>Schedule Maintenance on a Set of Weapon Entities Every ...</b>
<b>Scenario 1</b>	13 weeks
<b>Scenario 2</b>	26 weeks
<b>Scenario 3</b>	52 weeks

The three scenarios represent situations where scheduled maintenance requests require the selection of weapon entities from multiple storage sites quarterly, semi-annually, and annually.

#### **4.2.1 Quantitative Analysis**

Table 4.6 lists the average run times (as well as the standard deviation) of the two Multiple Storage Sites models for each of the scenarios tested. The Baseline-2 model implements a common attribute-copying technique for making sub-level entity information available during the selection of entities from the multiple storage sites. The Mod-1 model represents a *global* implementation of *direct sub-level entity access* used to accomplish weapon entity selection. The values in the table are based on five simulation replications of each model/scenario. Again, all the models produced identical model output data within a given scenario allowing the modeling methods to be compared as opposed to the output data.

**Table 4.6** Run Time of the Multiple Storage Sites Models

	<b>Baseline-2 Model</b>	<b>Mod-2 Model</b>
<b>Scenario 1</b> (Avg. / Std Dev)	36.1 seconds <i>0.07 seconds</i>	5.3 seconds <i>0.00 seconds</i>
<b>Scenario 2</b> (Avg. / Std Dev)	72.1 seconds <i>0.11 seconds</i>	10.6 seconds <i>0.00 seconds</i>
<b>Scenario 3</b> (Avg. / Std Dev)	138.6 seconds <i>0.10 seconds</i>	20.4 seconds <i>0.05 seconds</i>

As was the case with the True Defect Analysis models, the variations in run times for the Multiple Storage Sites models are statistically insignificant for the purposes of the comparison.

The *global* implementation of *direct sub-level entity access* applied in the Mod-2 model produced a run time that was 6.8 times faster than the baseline case for all the scenarios. The run time improvement in this case was essentially constant across the scenarios because the only functions being performed in the models related directly to the selection and scheduling of weapon entities (the focus of the varied parameter) with no other operations occurring in the model. The situation was somewhat different with the True Defect Analysis models in section 4.1.1. In those models, more operations (both true defect and surveillance defect testing) were present, while only one (true defect analysis scheduling) was being varied, resulting in a range of run time improvements across the scenarios.

The run time observations made here (as well as in section 4.1.1) again illustrate the potential speed benefits of applying *direct sub-level entity access* in place of more common modeling methods while realizing that the actual amount of improvement is still model dependent.

Table 4.7 shows the relative size difference (in terms of memory usage) between the common attribute-copying implementation of selecting and sorting weapon entities from multiple storage sites (the Baseline-2 case), and the remote, *global* direct access implementation employed in the Mod-2 case. Since the size of the different constructs remain constant between all runs, no reference to the different scenarios is necessary.

**Table 4.7** Size Contributed to Model by each Sort and Select Implementation

	<b>Baseline-2 Implementation</b>	<b>Mod-2 Implementation</b>
<b>Size (Memory Usage)</b>	170 KB	31 KB

As was the case with the *global* true defect analysis implementation, the Mod-2 implementation of remote, *global* sub-level entity access illustrates the potential for significant size reductions that can be obtained by applying *direct sub-level entity access* in place of more commonly used methods.

The speed and size improvements shown in the modified Multiple Storage Sites model can largely be attributed to its direct and efficient approach of accessing and using sub-level entity information. Applying *direct sub-level entity access* in this case eliminates – as in other direct sub-level entity access implementations – the many extra execution steps that more common methods require to perform the same function.

#### **4.2.2 Qualitative Analysis**

The implementations of sorting and selecting entities from multiple storage sites in the Baseline-2 model (using attribute-copying to access sub-level information) and the Mod-2 model (using remote, *global* direct access) were analyzed qualitatively. A

comparison of the two cases – in terms of the qualitative metrics complexity, scalability, and flexibility/functionality – is presented in Table 4.8.

**Table 4.8 Qualitative Metric Comparison of Multiple Storage Sites Models**

<b>Complexity</b>	<b>Baseline-2 (using attribute-copying):</b> Overly complex by requiring all entities of a requested type to move from all storage sites to a given location in the model to sort and select the appropriate ones to send on. Complexity increases significantly to account for additional weapon and subsystem types.
	<b>Mod-2 (using global implementation):</b> No movement of entities is required to sort and select the weapons. Sorting and selection are done globally and remotely, independent of where the weapon entities are in the model and independent of the number of storage sites involved. The complexity remains essentially constant even when more weapon and subsystem types are added.
<b>Scalability</b>	<b>Baseline-2 (using attribute-copying):</b> Not easily scalable to account for additional stockpile storage sites or additional weapon and subsystem types. Each additional stockpile storage site and weapon and subsystem type requires additional cumbersome modeling logic to coordinate the movement of the appropriate weapon entities from the different sites. Each storage site must be augmented with additional holding queues and routing logic to keep additional weapon types segregated.
	<b>Mod-2 (using global implementation):</b> Very scalable. No modeling logic is needed to coordinate movement of entities since no movement of entities is necessary to sort and select. No segregation of weapon entity types is required. All weapon entities can be kept in a single holding queue at a given storage site (independent of the number of weapon types).
<b>Flexibility/Functionality</b>	<b>Baseline-2 (using attribute-copying):</b> Flexibility is very limited and functionally constrained in part because all weapon entities in the model must be physically moved and manipulated to sort and select the appropriate weapon entities. Furthermore, the common modeling constructs used are limited in the number of criteria considered and the ability to sort based only on top-level entity attributes.
	<b>Mod-2 (using global implementation):</b> Much more flexible and functional since sorting and selection are unconstrained by the movement of entities (no movement is necessary). Many more sorting and selection options are available, including the ability to select entities based on both top-level and sub-level attribute information simultaneously.

Table 4.8 indicates how well each implementation meets the objectives identified earlier for improving the 2x2 nuclear stockpile, life-extension model; the objectives being to perform true defect analysis, account for multiple weapon storage sites, and include

more weapon and subsystem types. Both of the models were designed specifically to demonstrate how selection of specific weapon entities (in response to a surveillance or maintenance request) can take place in a multiple storage site environment, so both potentially fulfill the second objective. However, as alluded to in the table, the complexity, scalability, and functionality issues involved with accounting for multiple storage sites and additional weapon and subsystem types make the Baseline-2 implementation of sorting/selection much less desirable than the *global* direct access implementation of the Mod-2 model. Using the *global* approach, the two objectives relating to multiple storage sites and increased weapon and subsystem types can be appropriately addressed without contributing adversely to or being an undue burden on any other part of the model (including true defect analysis).

#### **4.3 ORIGINAL 2X2 MODEL VS. IMPROVED 7X7 MODEL**

One of the goals of the concept models was to determine the best ways to improve the original 2x2 nuclear stockpile model relative to the key deficiencies originally identified. The analyses of the various implementations in the concept models led to applying *direct sub-level entity access* to upgrade the 2x2 model – resulting in the improved 7x7 model.

Applying the same quantitative metrics to a comparison of the 2x2 and 7x7 models (as was done with the concept models) is not valid because of the differences between the 2x2 and 7x7. However, the improvements made possible by *direct sub-level entity access* as implemented in the 7x7 model can be qualitatively contrasted with the 2x2 model. Table 4.9 compares the two models on the bases of the qualitative metrics

complexity, scalability, and flexibility/functionality – highlighting the deficiencies previously identified in the 2x2 model and the ability of the 7x7 model to address those deficiencies.

**Table 4.9** Qualitative Metric Comparison of the 2x2 versus the 7x7 Model

<b>Complexity</b>	<p><b>2x2 Model:</b> The simplifying assumptions made in the model are intended to reduce the complexity for the case of two weapon types and two subsystem types. However, the complexity increases significantly if the common modeling constructs and methods used in the 2x2 are merely extended to account for more weapons, subsystems, and storage sites.</p>
	<p><b>7x7 Model:</b> Fundamentally no more complex than the 2x2 model – and in many regards, less complex. “Hard wired” components of scheduling and routing in the 2x2 model are handled more gracefully and less intrusively in the 7x7 model.</p>
<b>Scalability</b>	<p><b>2x2 Model:</b> Not easily scalable within the existing modeling constructs to account for additional stockpile storage sites or additional weapon and subsystem types. An increase in weapon types, subsystem types, and storage sites alone significantly increases the burden on the model even before being able to address all the functional deficiencies (like true defect analysis).</p>
	<p><b>7x7 Model:</b> Very scalable with the application of direct sub-level entity access techniques. The model can be further scaled with additional storage sites and more weapon and subsystem types without adversely impacting the key objectives identified earlier (true defect analysis, multiple storage sites, more weapon and subsystem types). The <i>direct sub-level entity access</i> implementations are more easily expandable (with less loss of efficiency and function) than the common modeling techniques used in the 2x2 and baseline concept models.</p>
<b>Flexibility/Functionality</b>	<p><b>2x2 Model:</b> Flexibility is very limited. With most functions “hard-wired” using common modeling methods, changes and improvements are more difficult and tedious to make if done within the existing architecture using the same common modeling methods. The functionality of the model is limited to that supported by the common modeling constructs (limited sorting capabilities, only top-level entity manipulation, routing constraints, etc.). In its present state, the model is obviously deficient in defect analysis capabilities and in its ability to handle more storage sites and more weapon and subsystem types.</p>
	<p><b>7x7 Model:</b> Addresses all the key deficiencies from the 2x2 model with the flexibility to expand and change the model more gracefully and robustly. Functionally more capable, enabling more flexible and powerful selection criteria for maintenance and surveillance requests, and offering more control over defect analysis information.</p>

The 2x2 model and the baseline concept models all employ similar common modeling techniques to perform their functions. Likewise, the 7x7 model and the modified concept models all employ direct sub-level entity access techniques to perform certain key functions. So analogous qualitative observations can be made between the 2x2 and 7x7 models as were made between the baseline and modified concept models. Consequently, Table 4.9 does not attempt to list all the similar observations that can be made based on the observations in Tables 4.4 and 4.8. Also, the observations made in Table 4.9 are primarily focused on those functions relating to the key deficiencies/objectives identified for the 2x2 model.

The observations made between the 2x2 model and the 7x7 model help demonstrate the improvements and impact that *direct sub-level entity access* can have when seeking more flexible, functional, useful stockpile life-extension models. All the advantages observed from applying *direct sub-level entity access* over more common modeling techniques in the concept models are combined in the 7x7 model and result in a more informative, efficient, functional, and robust model than could have been achieved using the more common modeling methods. The 7x7 model performs true defect analysis, enabling more accurate and insightful surveillance program analysis (by being able to compare surveillance data to expected actual defect rates). The 7x7 model can more easily handle the complexities of routing, selecting, sorting, etc. that accompany a stockpile consisting of many weapon and subsystem types. The hierarchical weapon entity structure represented and supported in the 7x7 model more closely depicts the actual situation and lets the model more effectively communicate information to model users and developers. The implementations of *direct sub-level entity access* in the 7x7

model enhance all of these functions – resulting in a model that is more easily expandable and flexible to change, making the model even more useful for the future.

#### **4.4 DISADVANTAGES OF DIRECT SUB-LEVEL ENTITY ACCESS**

Several advantages of using *direct sub-level entity access* have been addressed in the previous sections. Applying the concept resulted in smaller, faster, more efficient, and more flexible model implementations. However, a few potential disadvantages of *direct sub-level entity access* should be mentioned.

The primary disadvantage of *direct sub-level entity access* (as experienced in this thesis) is the initial effort required to enable the capability in a given simulation software product. This is because *direct sub-level entity access* is not currently a very common modeling technique. Most discrete-event simulation software products do not already have the built-in capability for direct sub-level entity access. That is, the functionality is not already a pre-defined function readily available to the user. Depending on the software being used, enabling direct sub-level entity access requires a significant amount of initial effort – including a substantial amount of programming to get things set up properly. For example, the special functional blocks (see Appendices B, C, and D) that were used in the modified concept models required substantial custom programming to interface with the existing sub-level entity model data properly. On the other hand, a few simulation software products do support some degree of sub-level entity access more readily than others. In such cases, the initial effort that is required to enable *direct sub-level entity access* in the desired way may be significantly reduced.

If *direct sub-level entity access* is not a built-in function of a given simulation product, software maintenance and support for the functionality (as built-in by the user) is left to the user. Unless it is a built-in function of a simulation software product, the user must undertake many of the technical support and maintenance issues relating to the direct sub-level entity access capability.

## CHAPTER 5

### CONCLUSIONS AND RECOMMENDATIONS

#### 5.1 CONCLUSIONS

The purpose of this thesis was to demonstrate that *direct sub-level entity access* is indeed a useful concept that is often preferable in modeling nuclear stockpile life-extension issues. *Direct sub-level entity access* – a seldom-used concept – had not previously been applied to nuclear stockpile models. However, the application of the concept to such models in this thesis showed that key functions of nuclear stockpile, life-extension models are well suited to, and can benefit greatly from, the advantages *direct sub-level entity access* offers over more common modeling techniques.

Concept models that were designed to address key deficiencies identified in current stockpile life-extension modeling efforts embodied in the 2x2 model were used to help analyze the effectiveness of common modeling techniques compared with direct sub-level entity access techniques. The baseline concept models employed common modeling techniques, including unbatch/batch and attribute-copying, to access and use sub-level entity information from weapon subsystems. The modified concept models used direct sub-level entity access methods to access sub-level entity information. In all cases, the direct sub-level access approach outperformed the more common approaches. *Direct sub-level entity access*, in these cases, resulted in models that were significantly faster, smaller, more efficient, more flexible to change, more scalable, and more capable

than their baseline counterparts. The analysis of the concept model implementations indicated that the best way to meet the objectives of improving the deficient 2x2 nuclear stockpile, life-extension model was to apply global, *direct sub-level entity access*.

Applying *direct sub-level entity access* to the 7x7 model completely addressed the key deficiencies identified from the 2x2 model; namely true defect analysis, multiple storage sites, and additional weapon and subsystem types. Had common modeling techniques been used to expand the 2x2 model (as was demonstrated in part in the baseline concept models), the resulting limitations would have made improving the 2x2 model very difficult and even impractical. Such limitations include cumbersome routing constraints, restrictive entity sorting capabilities, inefficient entity information access, and excessive model size growth; all leading to a slow, inflexible, and overly complicated model. Instead, the 7x7 model's use of *direct sub-level entity access* resulted in a model that is more capable, flexible, scalable, and informative than the original 2x2 model. The 7x7 model is also faster, more efficient, and better poised for future growth than it would have otherwise been had common modeling techniques been used.

The 7x7 model embodies some of the real complexities involved with the nuclear stockpile. Many different weapons and subsystems spread amongst many different storage sites can make modeling the system quite difficult. However, *direct sub-level entity access* (particularly *global access*) was shown to be well suited to handle such complex representations and proved critical to achieving the objectives of the improved stockpile life-extension model.

Some additional observations should be reiterated at this point relating to *direct sub-level entity access*. The concept, as presented in this thesis, is intended to make the

most use of model information as it already exists in the simulation software. The intent is not for users to create or set up their own custom databases (or something similar) to hold model information – essentially making a redundant set of data that the simulation software already manages. Instead, *direct sub-level entity access* strives to take advantage of the fact that the software already keeps track of sub-level entity information, and that this information can and should be more accessible to the modeler – without requiring that everything be done at the top entity level. Consequently, the concept may require a greater initial investment of effort to interface appropriately with the existing sub-level entity model data. The amount of effort required depends largely on the simulation software being used and its existing capabilities relating to sub-level entity access. However, considering the advantages that *direct sub-level entity access* provides, the time and effort spent to make it work in a model is worth it.

## 5.2 RECOMMENDATIONS

Some recommendations for further research that relate to the thesis include deeper sub-level implementation of direct entity access, database integration, and additional applications for *direct sub-level entity access*. These ideas will be briefly described below.

Using *direct sub-level entity access* to improve nuclear stockpile, life-extension models was the primary focus of this thesis. However, the concept is not restricted to this area of application and would be well suited to a myriad of other interesting simulation modeling areas outside of the nuclear weapon domain. Many things can be and are modeled as hierarchical entity structures (for example, automobiles). Any model with

such entity structures could potentially benefit from *direct sub-level entity access* (depending on the objectives of the model). Investigating different areas of application would be worthy of further consideration.

The specific implementations of *direct sub-level entity access* in this thesis only dealt with two-level entity structures (as in Figure 1.1). The special functional blocks that were created to apply *direct sub-level entity access* in the modified concept models were only designed to support access to information from model entities existing at the first sub-level of a given entity hierarchy. This is because the entity structures in the models analyzed herein were not composed of more than two levels. Supporting *direct sub-level entity access* of deeper sub-levels (as in Figure 1.2) would likely require significantly more effort to apply. Investigating the impact of deeper *direct sub-level entity access* and implementing support for it would be a good candidate for further research. Such an effort could be applied to nuclear stockpile models as was done in this thesis, or to any other appropriate area of application.

Setting up a custom database and interface that tracks sub-level entity model data was mentioned previously as another seldom-used alternative for accessing and using entity information. While this option was judged to require more effort to implement than doing single-depth *direct sub-level entity access*, it would still be interesting to investigate various database options that could be more fully integrated into simulation models from any application domain (weapon or otherwise).

## APPENDIX A

### VENDOR QUESTIONNAIRE

#### Batched (Sub-Level) Entity Access Questionnaire

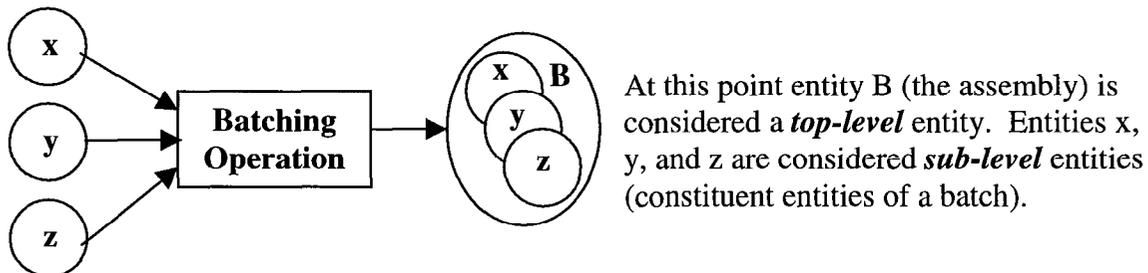
Thank you for taking the time to complete this questionnaire. Feel free to qualify any responses or to attach/include additional explanations as you deem necessary.

#### **Background Information/Example:**

Consider a simulation model representing three entities (x, y, and z) each with unique characteristics (attributes). For example, each entity carries the *attributes* “type” and “age” and “changeFlag”, but with different attribute values:

	“type” value	“age” value	“changeFlag” value
<b>Entity x</b>	0	12	0
<b>Entity y</b>	1	8	0
<b>Entity z</b>	2	17	0

The three entities are batched together in an assembly-type operation. The resulting assembly is a new entity ‘B’. Because the assembly may at some future time be separated into its original constituencies, entities x, y, and z must maintain their unique characteristics while batched together. Entity B (the batch) may even have unique attributes assigned to it (such as “AssemblyDate” and “type”, etc.). The process being modeled is illustrated below.



As you proceed with the questionnaire, please keep in mind the above example and how it would be implemented using your simulation software. The questions refer primarily to access and control of sub-level entity information (i.e. the attributes on x, y, and z) while in the *batched* state. This means accessing/controlling the sub-level entity

attributes without unbatching the assembly to gain access to them, or without copying the sub-level information onto the top-level entity where it might be more easily accessible. (**Access to** the attributes implies the ability to read/audit the attribute information. **Control of** the attributes implies the ability to change and add/remove attribute information.)

**Questions:**

1. **Please name the discrete-event simulation product, including version number, for which information is being provided in this questionnaire:**
  
2. **Is the primary interface to modeling with your product (for the typical user):**
  - 2a) **Programming (writing code)?**  
 YES  
 NO
  
  - 2b) **Or is it primarily graphically based (icon/menu/dialog based, point/click, etc.)?**  
 YES  
 NO
  
3. **Is the typical user of your product usually required to program (write code) in order to build substantial models?**  
 YES  
 NO
  
4. **How would you characterize your product: (mark all that apply)**  
 Simulator  
 Simulation Language  
 Other (please specify)
  
5. **Assume a model builder with no programming experience. Would s/he be a typical user of your product?**  
 YES  
 NO
  
6. **In a model built with your product, do individual entities carry their own attributes?**  
 YES  
 NO (If NO, please explain how entity specific characteristics are tracked)



- 10. If sub-level entity attribute access/control IS indeed possible:**
- 10a) Could this access be invoked from a single place in the model to act on entities at other places in the model (i.e. global control)?**
- YES  
 NO
- 10b) Or would the batch entities have to be passed/moved through specific locations/activities in the model in order to have their sub-entities accessed?**
- YES  
 NO
- 11. Do you think users would benefit, or see as useful, built-in capabilities to allow sub-level entity access/control (as it has been discussed in this questionnaire)?**
- YES  
 NO
- 12. A couple of ways to have some degree of access to sub-level attribute information without directly accessing the sub-level entities while in the batched state were alluded to in the background example. That is, one could unbatch the batched entity (essentially bringing the sub-level entities to the top level again), manipulate the attributes as desired, and then batch the entities back together again. Another way would be to copy the sub-level entity attributes onto the top-level batch during the batching operation such that the information would be available at the top-level. Aside from these workarounds, is there another straightforward way to achieve the same goal using your product that has not already been addressed in this questionnaire?**
- YES (If YES, please explain)  
 NO
- 13. Please list any other information that you think would be relevant, or any other related work (published papers, white papers, etc.) that you are aware of pertaining to the topic of sub-level entity access/control.**

## APPENDIX B

### DIALOG OF GET ATTRIBUTE (SUB-LEVEL) BLOCK

This appendix shows the dialog (the primary user interface) of the *Get Attribute (sub-level)* custom Extend block. Only relevant dialog tabs are shown.

**Attribute**   **Animate**   **Comments**

Finds attributes on specific sub-level items (which are part of the top-level batch passing through the block).

Unique Sub-Item Identifier	Sub-Level Attribute	Display value
0	DefectDate	Read Only
1	DefectDate	Read Only
2	DefectDate	Read Only
3	DefectDate	Read Only
	None	Read Only

If the named sub-level attribute is not found, use:

a NoValue (blank) as the value.

the number 0 as the value.

Do not retain attribute values between items

Help

## APPENDIX C

### DIALOG OF DEFECT ANALYZER BLOCK

This appendix shows the dialog (the primary user interface) of the *Defect Analyzer* custom Extend block. Only relevant dialog tabs are shown.

**Defect Analyzer [SL0]**

**Analyzer** | **Comments**

Periodically audits a group of weapons and reports the percentage of subsystems (by type) that are defective.

Do first audit at time =  time units  
then repeat every  time units

Search Group Criteria

1) **Weapon Type (top level) identified by Attrib name**   
Audit weapons with weapon type value of

2)  **Limit audit group to weapons with the Attribute**   
equal to

3) **Defect Date of sub-system identified by Attribute**

4) **Sub-system Type (sub level) identified by Attribute**

=====  
**Output defect percentages for the following sub-system type ID's:**

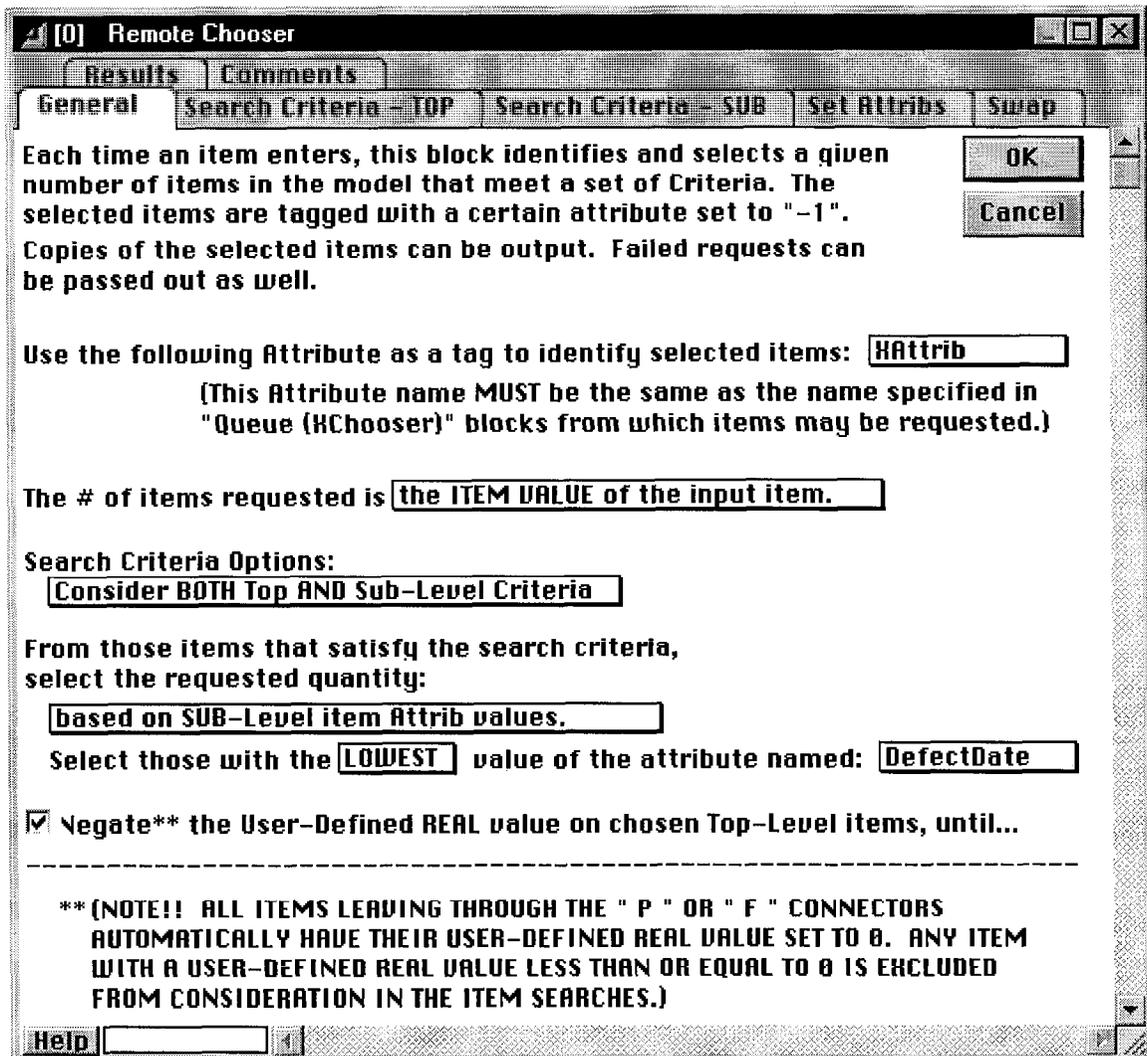
Sub-Type ID #	# Audited	% Defective
1		
2		
3		

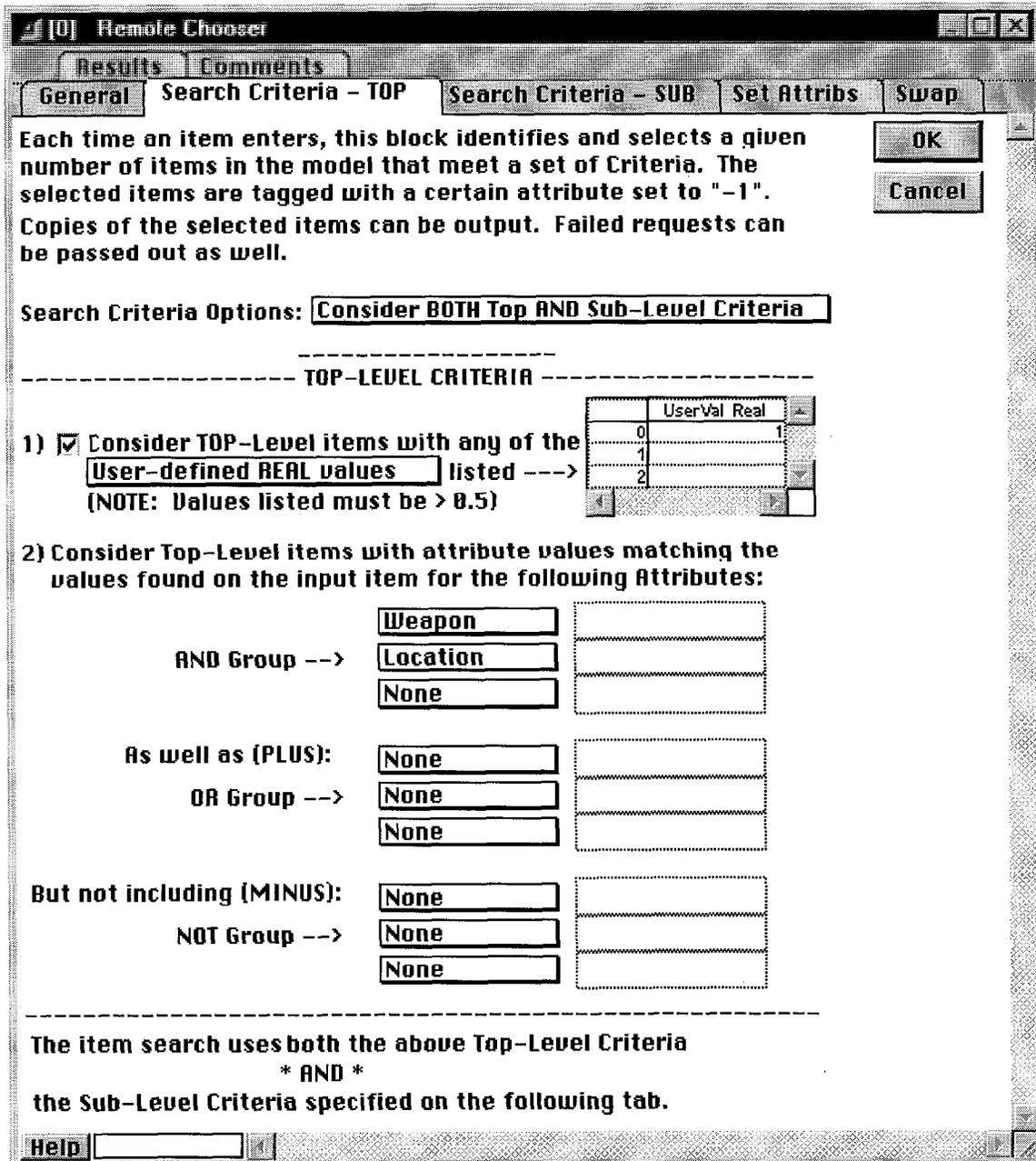
Help

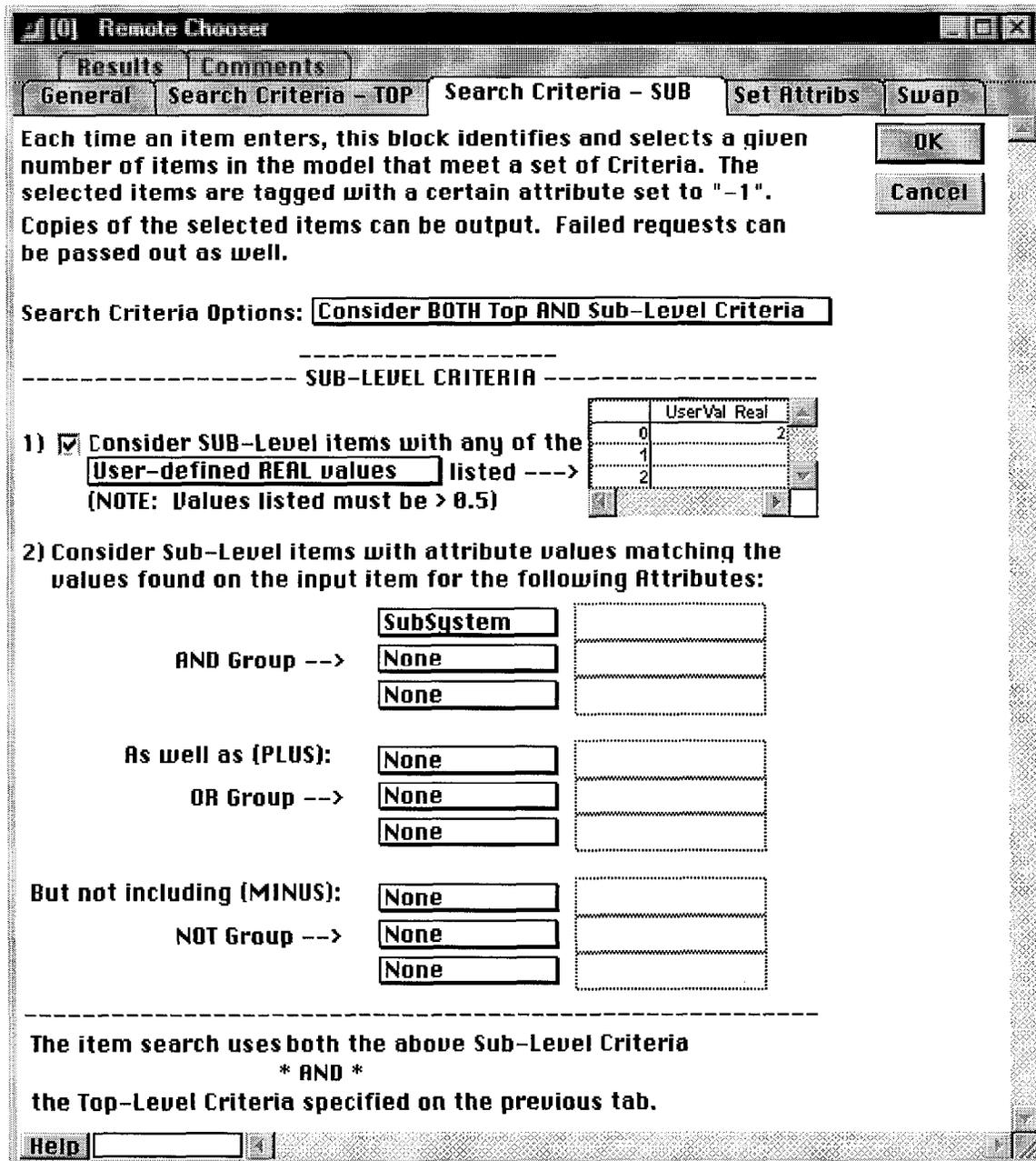
## APPENDIX D

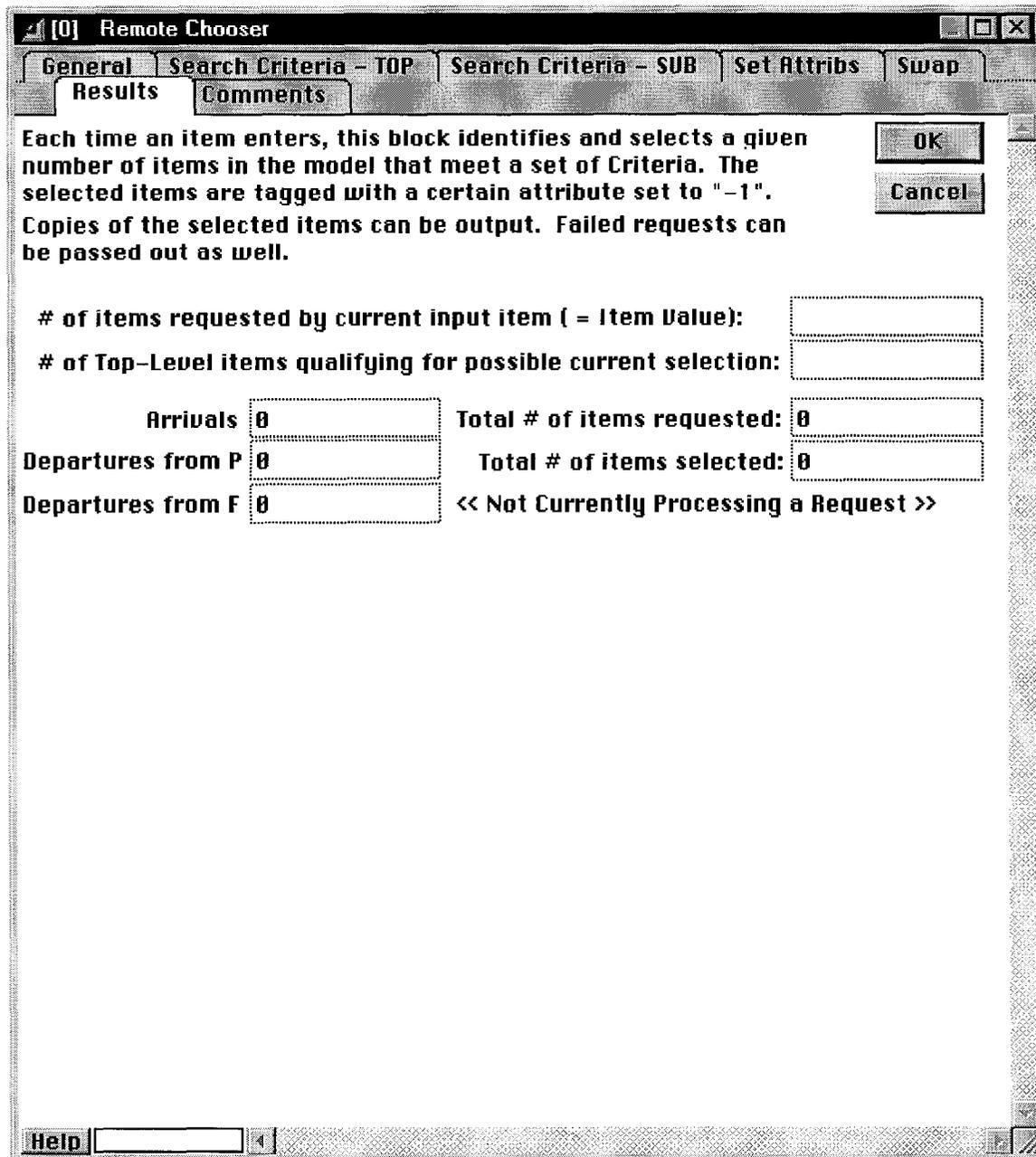
### DIALOG OF REMOTE CHOOSER BLOCK

This appendix shows the dialog (the primary user interface) of the *Remote Chooser* custom Extend block. Only relevant dialog tabs are shown.









## REFERENCES

- “1997 Simulation Software Survey.” OR/MS Today Oct. 1997. 26 Jan. 1999  
<<http://lionhrtpub.com/orms/surveys/Simulation/Simulation.html>>.
- “1998 Simulation Software Survey.” APICS - The Performance Advantage July 1998:  
56-59.
- “APICS - The Performance Advantage 1997 Simulation Software Survey.” APICS - The Performance Advantage July 1997. 5 Feb. 1998  
<<http://lionhrtpub.com/apics/surveys/Simulation/APICS-Simulation.html>>.
- Balci, Osman. Questionnaire response for Visual Simulation Environment. 23 June 1998.
- Balci, Osman, et al. “Dynamic Object Decomposition in the Visual Simulation Environment.” Proceedings of the 11<sup>th</sup> European Simulation Multiconference. Ed. Ali. R. Kaylan and Axel Lehman. San Diego: Society for Computer Simulation, 1997. 69-73.
- Balci, Osman, et al. “Visual Simulation Environment” Proceedings of the 1998 Winter Simulation Conference. Ed. D. J. Medeiros, Edward F. Watson, John S. Carson, and Mani S. Manivannan. Vol. 1. San Diego: Society for Computer Simulation, 1998. 279-287.
- Banks, Jerry, John S. Carson, II, and Barry L. Nelson. Discrete-Event System Simulation. Upper Saddle River: Prentice Hall, 1996.
- Barnes, Catherine. Questionnaire response for Micro Saint. 16 June 1998.
- Barnes, Martin. Questionnaire response for QUEST. 26 June 1998.
- Boerigter, Stephen. Personal Interview. 22 Jan. 1999.
- Centeno, Martha A., and M. Florencia Reyes. “So You Have Your Model: What To Do Next – A Tutorial on Simulation Output Analysis.” Proceedings of the 1998 Winter Simulation Conference. Ed. D. J. Medeiros, Edward F. Watson, John S. Carson, and Mani S. Manivannan. Vol. 1. San Diego: Society for Computer Simulation, 1998. 23-29.

- Demuth, Nelson, Stephen Nielson, and Stephen Boerigter. "Hierarchical Recursive Modeling Approach to Facility and Mission Requirements for the Nuclear Weapons Complex." Presentation at Los Alamos National Laboratory, Los Alamos. 21 July 1995.
- Diamond, Bob. "Concepts of Modeling and Simulation." PC AI Sept.-Oct. 1996: 22-26.
- Floss, Peter. Questionnaire response for ReThink. 26 June 1998.
- Goble, John, and Brian Wood. "MODSIM III: A Tutorial with Advances in Database Access and HLA Support." Proceedings of the 1998 Winter Simulation Conference. Ed. D. J. Medeiros, Edward F. Watson, John S. Carson, and Mani S. Manivannan. Vol. 1. San Diego: Society for Computer Simulation, 1998. 199-204.
- Hammer, Scott. Questionnaire response for ALPHA/Sim. 23 June 1998.
- Harrell, Charles. Questionnaire response for ProModel. 5 June 1998.
- Harrell, Charles R., and Kerim Tumay. Simulation Made Easy. Norcross: Institute of Industrial Engineers, 1995.
- Helm, Terry, Stephen Boerigter, and Stephen Eisenhower. Stockpile Life-Extension Modeling Presentation at Los Alamos National Laboratory, Los Alamos. 12 Dec. 1997.
- Hench, Karen W., J. David Olivas, and Paul R. Finch. Computer Modeling for Optimal Placement of Gloveboxes. LA-UR-97-2187. Los Alamos: Los Alamos National Laboratory, 1997.
- Imagine That, Inc. Extend User's Manual. San Jose: Imagine That, Inc. 1997.
- Imagine That, Inc. Extend+Manufacturing User's Manual. San Jose: Imagine That, Inc. 1997.
- Kelton, W. David, Randall P. Sadowski, and Deborah A. Sadowski. Simulation with Arena. Boston: McGraw-Hill, 1998.
- Kjeldgaard, Edwin A., et. al. Planning and Scheduling for Agile Manufacturers: The Pantex Process Model. SAND98-0030. Albuquerque: Sandia National Laboratories, 1998.
- Law, Averill M., and W. David Kelton. Simulation Modeling and Analysis. 2nd ed. New York: McGraw-Hill, 1991.

Lawrence Livermore National Laboratory. "Keeping the Nuclear Stockpile Safe, Secure, and Reliable." Science & Technology Review Aug. 1996:7-16.

Lawrence Livermore National Laboratory. Through the science-based Stockpile Stewardship and Management Program, we will ensure the continued safety, security, and reliability of the U.S. nuclear stockpile. UCRL-MI-121423-DNT-1 Rev 1. Livermore: Feb. 1996.

Lenz, John. Questionnaire response for MAST. 25 June 1998.

Meyers, Mike. Questionnaire response for SIMPROCESS. 9 July 1998.

Nordgren, Bill. Questionnaire response for Taylor ED. 1 July 1998.

Orca Computer. Web Site. 26 Jan. 1999 <<http://www.OrcaComputer.com/>>.

O'Reilly, Jean. Questionnaire response for AweSim and FACTOR/AIM. 17 June 1998.

Pantex Plant. Web Site. 22 Jan. 1999 <<http://www.Pantex.gov/>>.

Parker, Robert Y. Extend Customization – Experiences and Issues. LA-UR-97-5032. Los Alamos: Los Alamos National Laboratory, 1997.

Parker, Robert Y. Questionnaire response for Extend. 16 June 1998.

Rohrer, Matthew. Questionnaire response for AutoMod and AutoSched. 17 June 1998.

Sadowski, Deborah. Questionnaire response for Arena. 28 July 1998.

Siprelle, Andrew J., Richard A. Phelps, and M. Michelle Barnes. "SDI Industry: An Extend-Based Tool for Continuous and High-Speed Manufacturing." Proceedings of the 1998 Winter Simulation Conference. Ed. D. J. Medeiros, Edward F. Watson, John S. Carson, and Mani S. Manivannan. Vol. 1. San Diego: Society for Computer Simulation, 1998. 349-356.

Swain, James J. "Simulation Goes Mainstream." OR/MS Today Oct. 1997. 26 Jan. 1999 <<http://lionhrtpub.com/orms/orms-10-97/Simulation-story.html>>.

United States. Dept. of Energy. Plutonium: The First 50 Years. DOE/DP-0137. Washington: GPO, 1996.

Waller, Tony. Questionnaire response for WITNESS. 6 Oct. 1998.

*(NOTE: Trademarks or registered trademarks used herein are property of their respective owners.)*

This report has been reproduced directly from the best available copy. It is available electronically on the Web (<http://www.doe.gov/bridge>).

Copies are available for sale to U.S. Department of Energy employees and contractors from—

Office of Scientific and Technical Information  
P.O. Box 62  
Oak Ridge, TN 37831  
(423) 576-8401

Copies are available for sale to the public from—

National Technical Information Service  
US Department of Commerce  
5285 Port Royal Road  
Springfield, VA 22616  
(800) 553-6847

