LA-13625-T
Thesis

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

LA-13625-T
Thesis

Issued: August 1999

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

Robert Y. Parker

Los Alamos
NATIONAL LABORATORY
Los Alamos, New Mexico 87545

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

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.

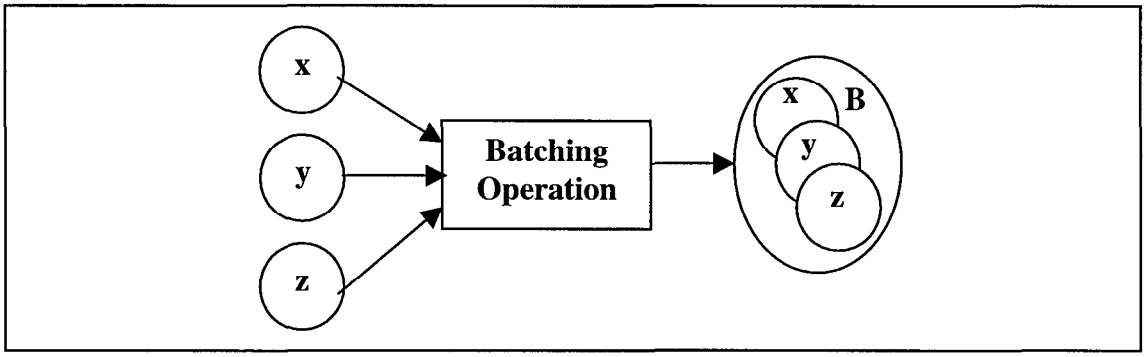
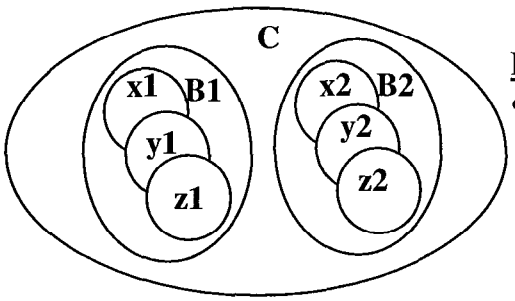


Figure 1.1 Entity Batching Operation

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.)



Entity Structure Hierarchy

- *Top Level*: Entity C
  - *First Sub-Level*: Entities B1 and B2
    - *Second Sub-Level*: Entities x1, y1, z1, x2, y2, and z2

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 Eisenhawer). 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.

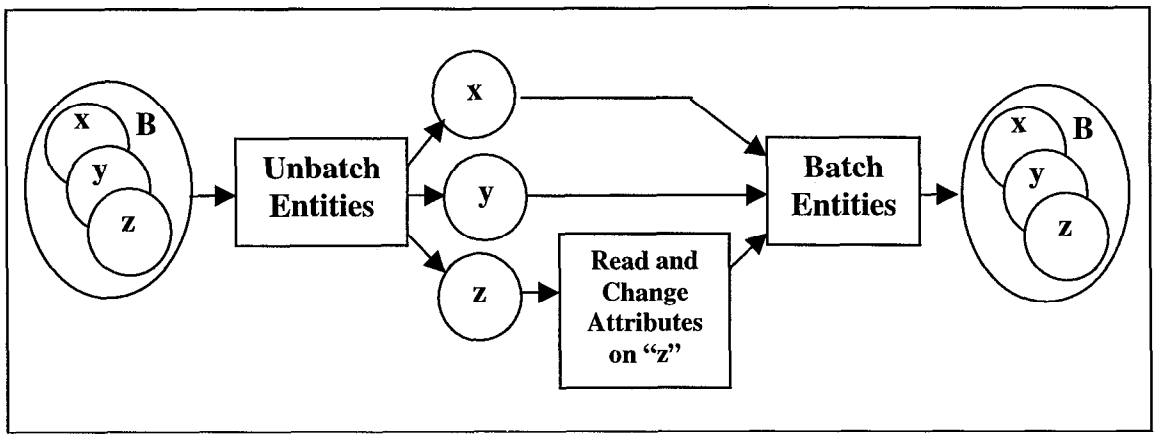


Figure 2.1 Unbatch/Batch Technique used to Access Sub-Level Entities

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.

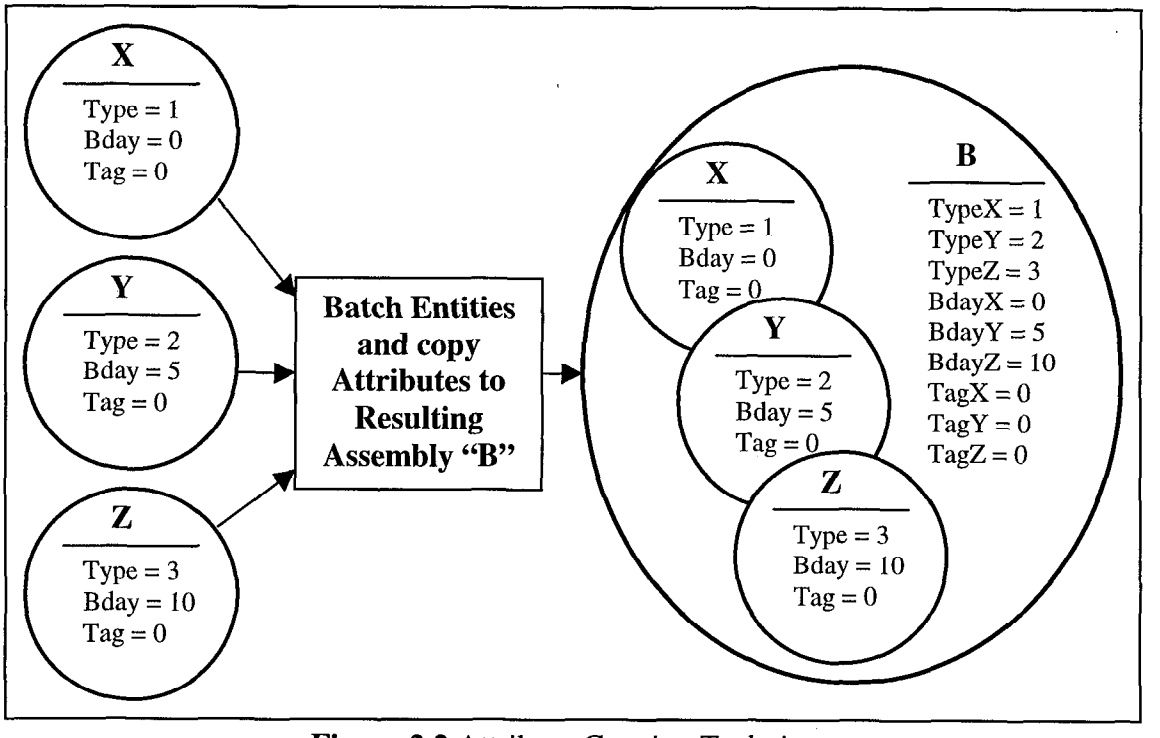


Figure 2.2 Attribute-Copying Technique

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.

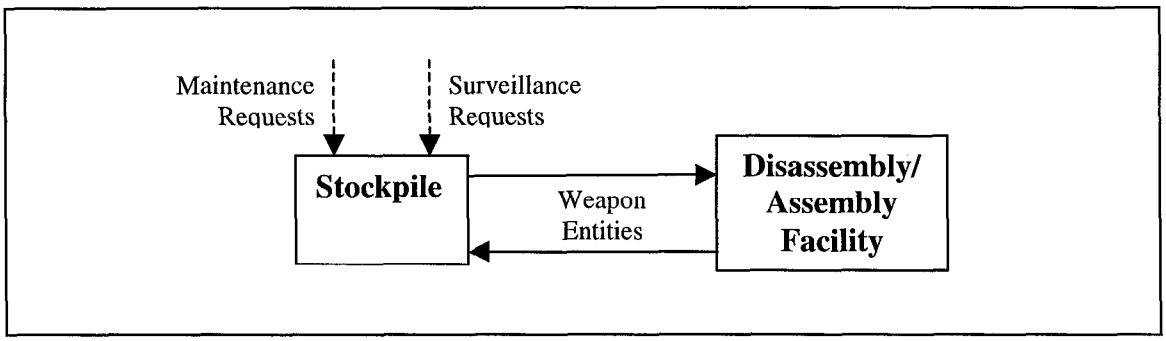


Figure 3.1 2x2 Model – Basic Entity Flow Diagram

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.

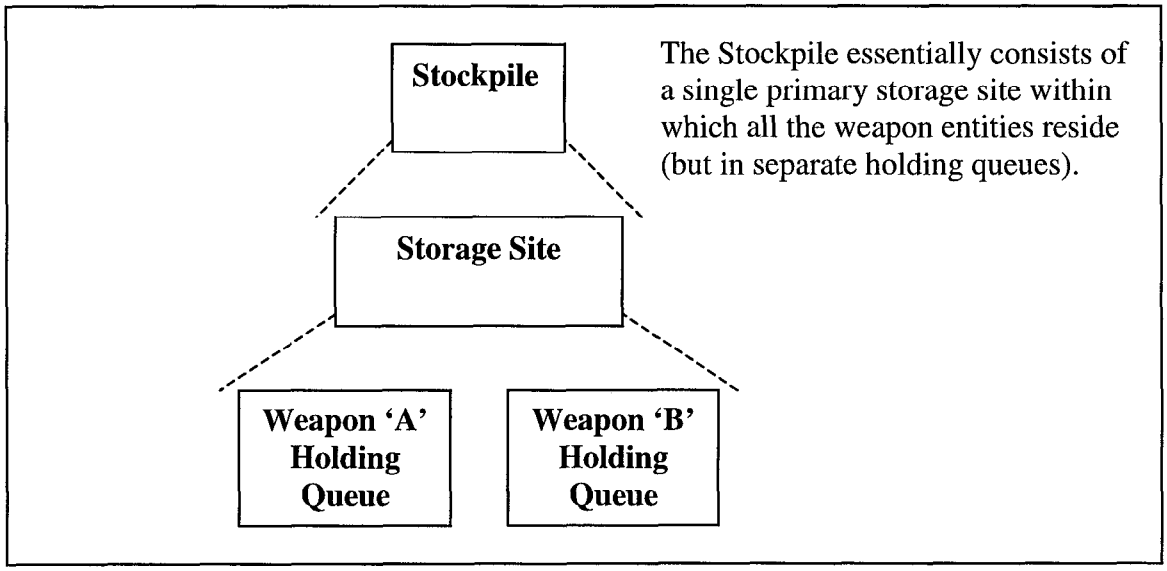


Figure 3.2 Simplified Stockpile Representation in 2x2 Model – Entity Segregation

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.

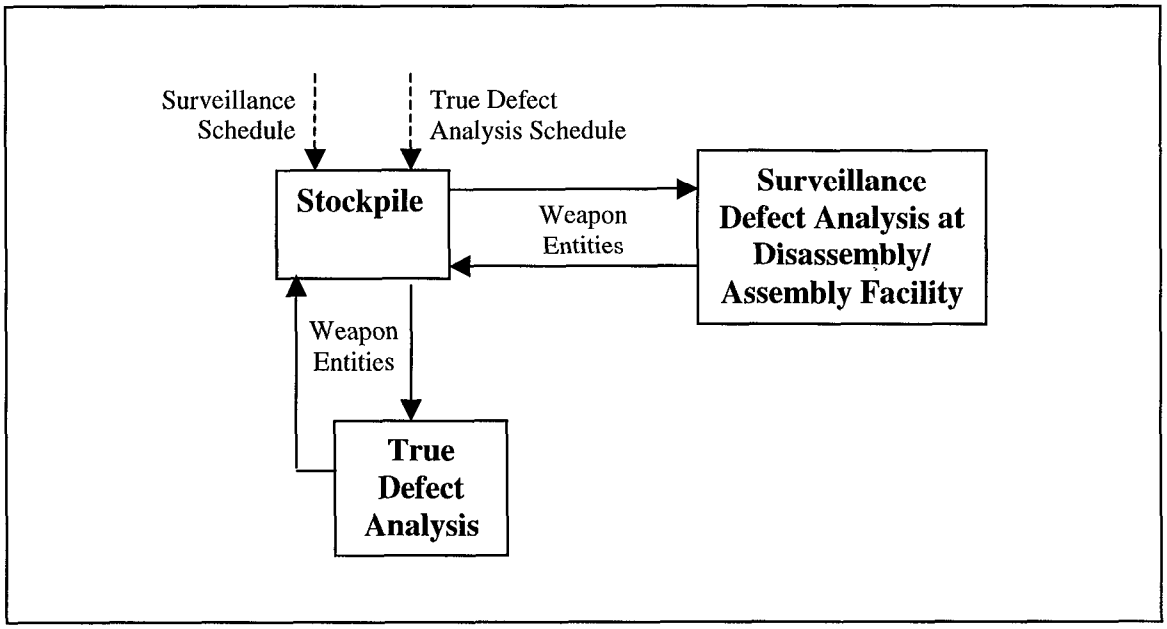


Figure 3.3 True Defect Analysis "Baseline-1" Model – Basic Flow Diagram

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.

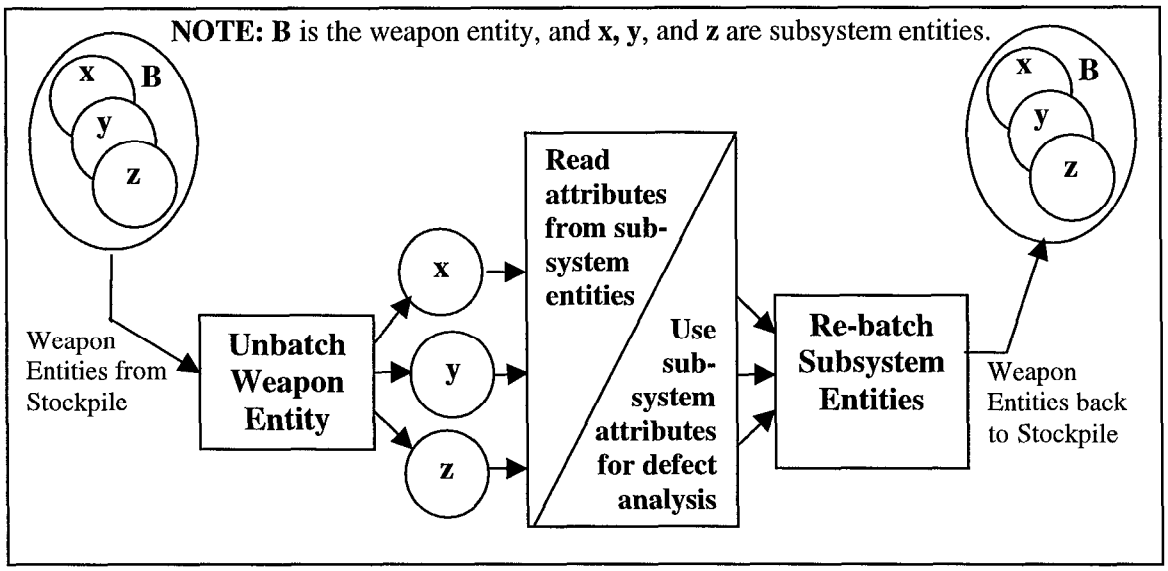


Figure 3.4 Accessing Sub-Level Attributes for Defect Analysis ("Baseline-1" Model)

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.

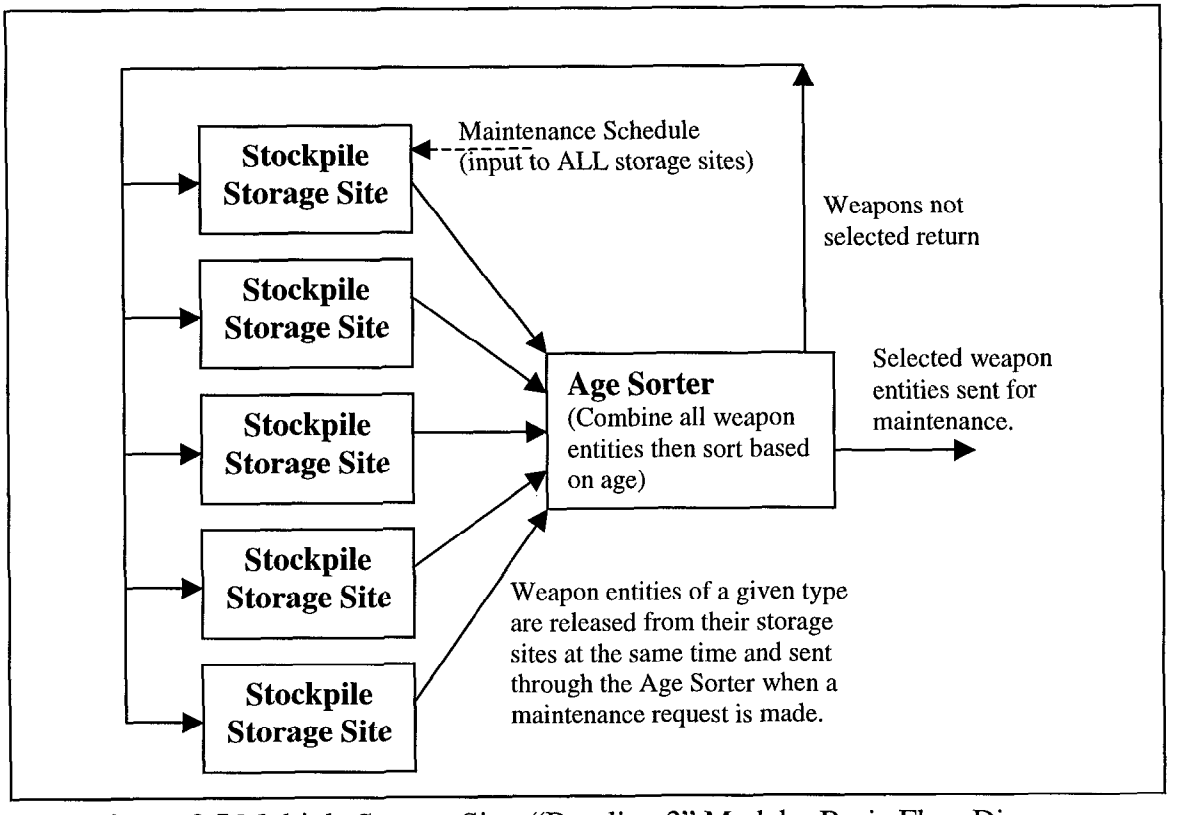


Figure 3.5 Multiple Storage Sites "Baseline-2" Model – Basic Flow Diagram

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.

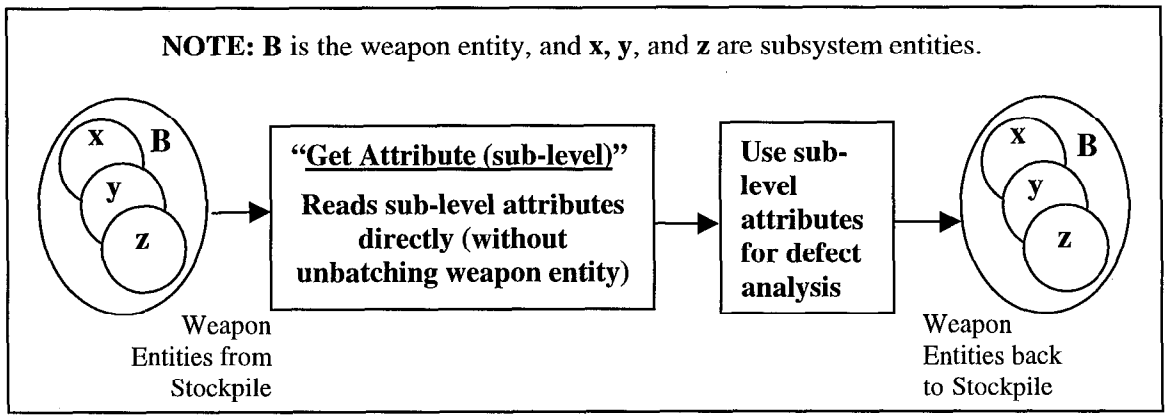


Figure 3.6 Accessing Sub-Level Attributes for Defect Analysis ("Mod-1A" Model)

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.

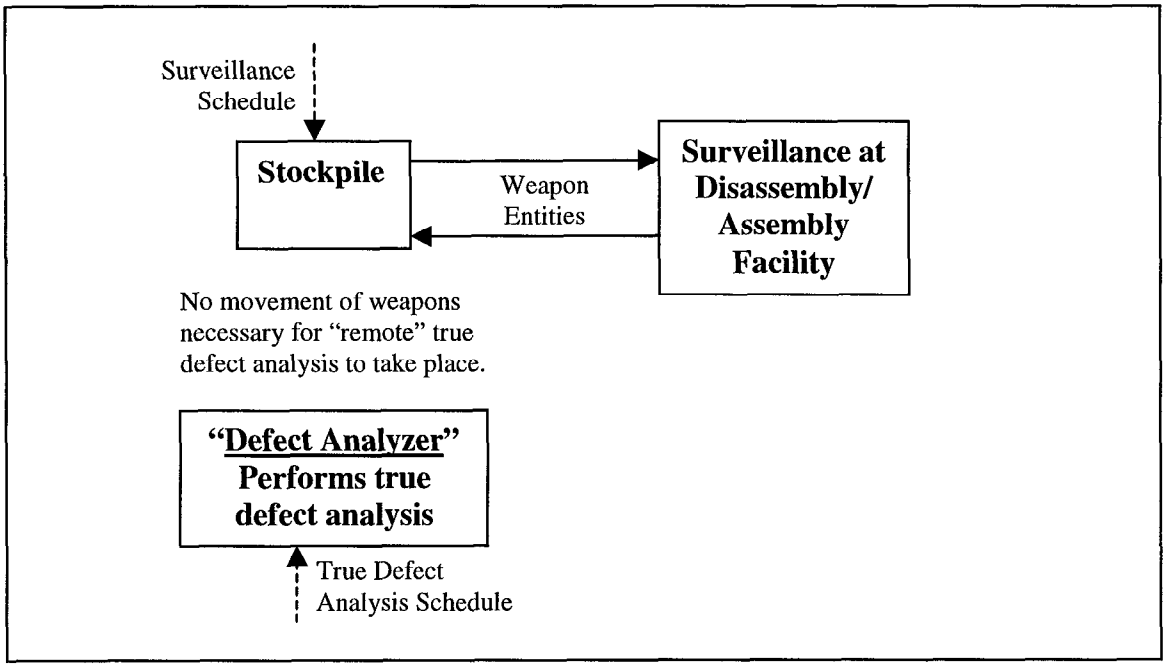


Figure 3.7 True Defect Analysis "Mod-1B" Model – Basic Flow Diagram

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.

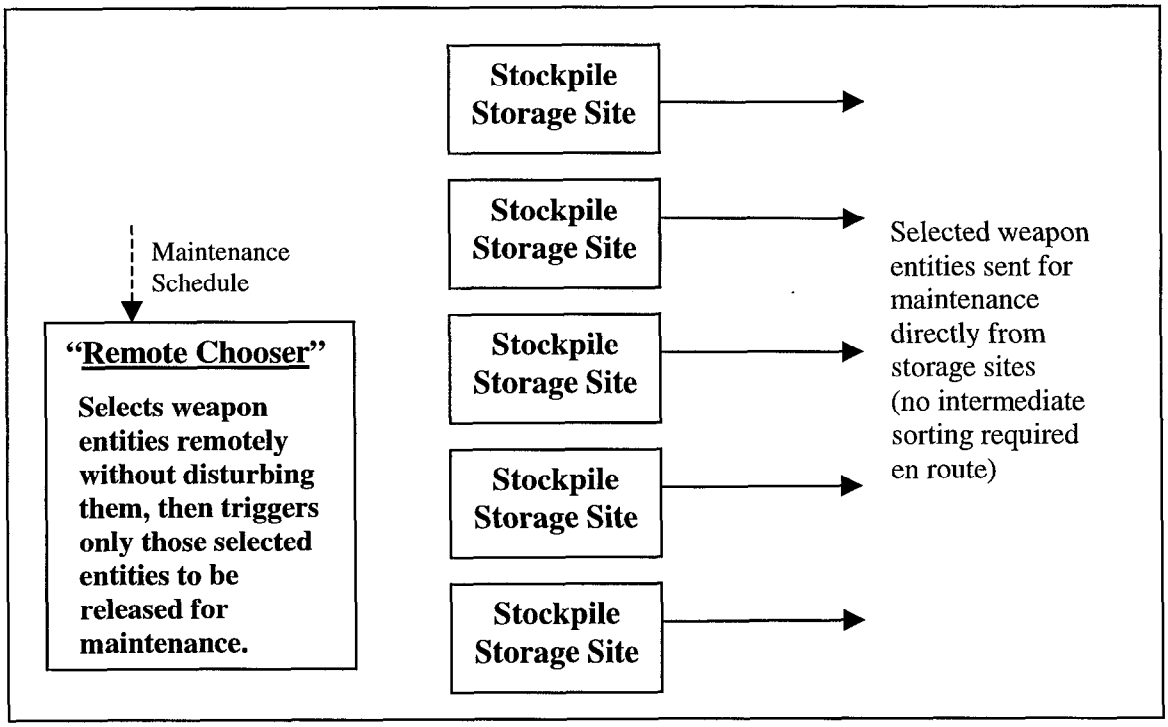


Figure 3.8 Multiple Storage Sites "Mod-2" Model – Basic Flow Diagram

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

Metric	Type	Description
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.