Consistency Maintenance using UNIFY

Anand Natrajan, Paul F. Reynolds Jr. and
Sudhir Srinivasan

Technical Report No. CS-95-28
June 16, 1995

Contact: anand@virginia.edu

Web: ftp://ftp.cs.virginia.edu/pub/techreports/CS-95-28.ps.Z

Consistency Maintenance using UNIFY

Anand Natrajan, Paul F. Reynolds Jr. and Sudhir Srinivasan

{anand, reynolds, ss7a}@virginia.edu
Dept. of Computer Science,
University of Virginia

1 Abstract

Distributed simulations comprised of aggregated entities (AEs) and disaggregated entities (DEs) pose critical consistency issues when AEs and DEs interact. Usually, meaningful interaction cannot take place without one of the two representing itself at a level of resolution compatible with the level of the other. Any approach that employs dynamic transitions between aggregated and disaggregated resolution levels suffers from not only potential consistency problems, but also chain disaggregation, network flooding, transition latency, and mapping problems between levels. Solutions meant to solve some or all of these problems leave one critical issue unresolved: proper and efficient maintenance of consistency among the levels of resolution for the same set of objects. An alternative approach, where AE and DE levels of resolution are maintained concurrently for the same objects, encounters the same consistency issues: anything that happens to a DE must be reflected accurately in the AE, and *vice versa*.

The fundamental problem is this: consistency cannot be guaranteed without a unified, coherent approach to correct, efficient consistency maintenance among levels of resolution for a set of simulated objects. In our approach, UNIFY, we contend that multiple levels of resolution should be addressed *inside* individual simulations, in order to ensure an efficient, coherent, verifiable solution. We propose the concept of MREs (Multiple Resolution Entities) *in lieu* of AEs and DEs. MREs are capable of representing simulated objects at specified levels of resolution in a consistent manner. A MRE should be able to provide, when requested, bindings for attributes at any desired level in a timely manner. A number of key issues must be addressed in order to provide this capability. The most critical issues are: identification of core data, temporal consistency, and mapping consistency.

2 Introduction

Simulationists speak of aggregated entities (AEs) and disaggregated entities (DEs). AEs typically represent abstract units such as Army battalions or Air Force squadrons, while DEs represent single objects such as a battle cruiser or a tank. Distributed simulations comprised of AEs and DEs pose critical consistency issues when AEs and DEs are allowed to interact. Usually, meaningful interaction cannot take place without one of the two representing itself at a level of resolution compatible with the level of the other. But then potential problems arise. A disaggregated AE that reaggregates itself for the purpose of interacting with another AE, and then later disaggregates (i.e., the sequence $AE_1 \rightarrow DE_1 \rightarrow AE_2 \rightarrow DE_2$ occurs), may put itself into a state (DE_2) that DE_1 could not have achieved over the same period of time. Any approach, such as this, that employs dynamic transitions between aggregated and disaggregated levels of resolution, suffers more than just potential consistency problems. Other problems include: chain disaggregation, network flooding, transition latency, and mapping problems between levels. We have found that solutions meant to solve some or all of these problems leave one critical issue unresolved: proper and efficient maintenance of consistency among the levels of resolution for the same set of objects.

An alternative approach, where AE and DE levels of resolution are maintained concurrently for the same objects (e.g. a battalion AE, with tank and wheeled vehicle DEs), encounters the same consistency issues: anything that happens to a DE must be accurately reflected in the AE, and vice versa. Many programs, for example ARPA's Synthetic Theater Of War (STOW), have advocated that AE and DE representations exist in different simulations. This approach is likely to lead to *ad hoc* solutions to the consistency problem.

The core problem is this: a fair fight cannot be guaranteed without a unified, coherent approach to correct, efficient, consistency maintenance among levels of resolution for a set of simulated objects. In UNIFY, we contend that multiple levels of resolution should be addressed *inside* individual simulations, in order to ensure an efficient, coherent, verifiable solution. Rather than think in terms of DEs or AEs, we propose the concept of MREs (Multiple Resolution Entities). MREs are capable of representing simulated objects at specified levels of resolution in a consistent manner. UNIFY is a framework that ensures consistency will be maintained across levels in a simulation.

3 Definitions

We present definitions for some terms we will use. Some of our definitions are based on those in [AMG95].

- ***Object***: A fundamental element of a conceptual representation that reflects the real world at levels of abstraction and resolution appropriate for a planned simulation.
- ***Entity***: A unit of organization at some level of abstraction, such as a tank, human, platoon, battalion, cloud or radar.
- ***Model***: A mathematical abstraction of the behavior of an object at a level appropriate for the planned simulation. Models are usually instantiated in simulation source code.
- ***Resolution***: The conceptual level at which an entity is simulated.
- ***Disaggregated Entity (DE)****:* A high-resolution entity, such as a Close Combat Tactics Trainer (`CCTT`) tank simulator.
- ***Aggregated Entity (AE)***: A low-resolution entity, such as a battalion, that simulates several aggregated objects.
- ***Multiple Resolution Entity (MRE)***: An entity that can be perceived at multiple levels of resolution concurrently.
- ***Simulation***: A dynamic representation of one or more objects, involving some combination of executing code, control/display interface hardware and interfaces to real-world equipment.
- ***Multi-level Simulation***: A simulation or exercise that involves entities at different levels of resolution.

Note: Levels of resolution and levels of aggregation are inversely related: high-resolution means low level of aggregation, and low-resolution means high level of aggregation.

4 Problems with Current Approaches

A common approach for handling interactions between AEs and DEs in a multi-level simulation has been to designate some areas of the battlefield as “virtual playboxes” in which all interactions are performed at the DE level [Karr94]. When an AE enters the playbox, it undergoes a disaggregation process whereby the AE is separated into its constituent DEs. Upon leaving the playbox, these DEs may reaggregate. The virtual playbox approach has several shortcomings: (1) the playboxes must be chosen *a priori*, (2) their boundaries are static in many cases, which means that AEs that “stray” into a playbox, but do not interact with other entities inside it, will disaggregate unnecessarily, and (3) by definition, no aggregate-level simulations may occur inside. However, this approach is simple, since aggregation and disaggregation decisions are reduced to determining when the boundary of the playbox is crossed. A more generic scheme, where aggregation decisions are made dynamically, is clearly preferable.

The playbox approach hides some critical issues such as temporal inconsistency and chain disaggregation. We elaborate on these issues using the following example scenario. Consider an airborne reconnaissance (T) over a battalion (A_1). The aircraft is interested only in the positions of the constituent DEs and does not affect them in any way. Most schemes to date would require a disaggregation sequence as the aircraft flies over the AE’s location, because the positions of constituent DEs are not maintained at the aggregated level. When the aircraft is out of range, the DEs would reaggregate. Further, the aircraft could return shortly thereafter, causing the aggregation/ disaggregation cycle to repeat. Obviously, this case can become pathological.

4.1 Temporal Inconsistency*

When simulations run on different time-steps, i.e., simulations at different resolution levels proceed at time steps that differ by orders of magnitude, inconsistency may arise. In particular, inconsistency may occur during attrition

* The definition of Temporal Inconsistency has changed since our last publication [Nat95].

computation, while perceiving the state of another entity, during line-of-sight computations or during dead-reckoning. In computing attrition, the problem occurs due to the time spent in solving Lanchester equations. These equations are easily computed in their simplest form, but in order to model the capabilities of the aggregated entities better, a number of factors are added to the equations, making them more time-consuming to compute [Karr83]. It is then possible to reach a state where entities have an inconsistent view of each other, as follows. In the airborne reconnaissance example, suppose battalion A_1 is engaged in a battle with another battalion A_2. A_1 and A_2 interact at the battalion level. Aircraft T, coordinating an attack on A_1, observes A_1 at the tank level and relays information about A_1 to other entities which intend to attack A_1. Now consider the following sequence of events: A_1 communicates its current strength to A_2, which computes attrition on A_1 using Lanchester equations. In the meantime, T requests tank-level information from A_1. T's requests may be satisfied in a much shorter time than that taken to compute the effect of A_2's interactions with A_1. By the time A_2 completes its computations, the results will be quite meaningless, because the computations were performed with data that are now stale. A_2 and T would now have inconsistent views of A_1. We believe the temporal inconsistency problem will become very significant as large multi-level simulations such as STOW are planned and executed.

4.2 Chain Disaggregation

In a two-level simulation, interactions between AEs should naturally occur at the aggregate level and those between DEs at the disaggregate level. However, several options arise when AEs interact with DEs. A naïve approach is to disaggregate an AE whenever it comes into sensor proximity of a DE. However, this could cause chain disaggregation, wherein many AEs are forced to disaggregate in a short period of time [Smith94]. Consider a simple case where four AEs are interacting in the following linear fashion ($A \leftrightarrow B$ indicates that entities A and B interact with each other), i.e. $AE_1 \leftrightarrow AE_2 \leftrightarrow AE_3 \leftrightarrow AE_4$. AE_1 comes into contact with a DE, causing AE_1 to disaggregate. This forces AE_2 to disaggregate, followed by AE_3 and AE_4. The problem can be translated easily to the airborne reconnaissance example. When the aircraft T begins interactions with battalion A_1, the latter disaggregates. This causes other AEs interacting with A_1 to also disaggregate. The naïve approach causes unnecessary disaggregation and puts a burden on computing and network resources.

Figure 1, based on [Smith94], shows different approaches to solving the chain disaggregation problem. The colored entities are simulated at the highest level of resolution, i.e., they are fully disaggregated.

The Null solution demonstrates the chain disaggregation problem. In this solution, an AE must disaggregate if it interacts with a DE. As can be seen from the figure, *all* of the AEs disaggregate unnecessarily by transitivity. The Direct Contact solution suggests that only those AEs that directly interact with the DE should be disaggregated. In effect, this limits the propagation along any chain to one step. However, the question of how the disaggregated AEs (colored) interact with the "un-disaggregated" AEs (un-colored) is left open. One option is for the "un-disaggregated" AEs to also disaggregate. This degenerates to the Null solution and hence is unacceptable. Another option is for all AEs to be able to handle interactions between disaggregated AEs and "un-disaggregated" AEs. This approach implies every AE should be able to handle interactions with all types of AEs. Given that there may be many different types of AEs in a simulation, this is clearly not a scalable solution. The Horizons approach stipulates that all AEs within some range of the DE should disaggregate. This approach has the drawback of the Direct Contact solution — it is not clear how disaggregated AEs and "un-disaggregated" AEs interact. In the Local Virtual approach, the DE receives aggregate information from the AE. The DE then locally disaggregates the AE information in order to obtain the level of information it needs. This is also known as "pseudo-disaggregation" and is employed by the JPSD program [Calder95]. It solves the chain disaggregation problem, but exhibits other problems. Temporal inconsistency could occur if two DEs locally disaggregate the same AE using different algorithms. Furthermore, this solution is not scalable, because each DE may have to know how to disaggregate every AE in the simulation. The last solution presented here — Partial Directed — also solves the chain disaggregation problem, but ignores other problems. In this solution, the AEs interacting with the DE are partially disaggregated, i.e., some part of the AE remains an AE, while the rest disaggregates into DEs. However, the algorithms for partial disaggregation are quite complex. Also, consecutive disaggregation and aggregation might not produce the same state. Temporal inconsistency is introduced once again, but this time within (the partly-colored) AEs themselves.

It is important to note that in attempting to solve the chain disaggregation problem, most of these approaches introduce the temporal inconsistency problem, thus strengthening our claim that the temporal consistency problem is critical.

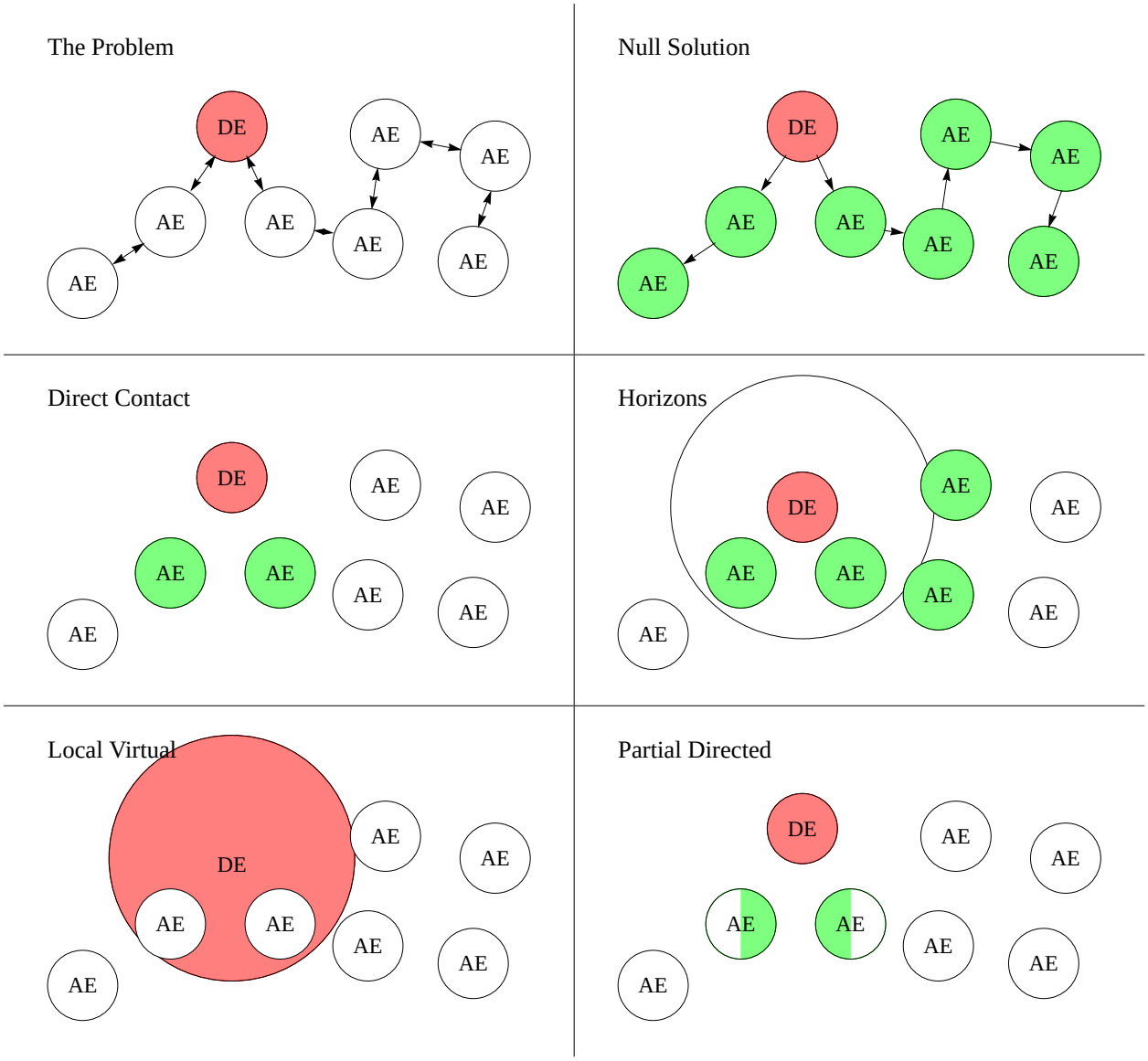

FIGURE 1. Some solutions to chain disaggregation

4.3 Network Flooding

The network is projected to be a bottleneck in distributed simulations. Network resources may be strained by the acts of aggregation and disaggregation, depending on the scheme used. Disaggregation creates new entities, each of which could be a sender and/or receiver of messages. Clearly, even if only the entity state messages (for example, ESPDUs in `DIS`) generated by all the entities are taken into account, this is an increase in network traffic. Also, aggregation and disaggregation protocols typically require a number of messages to be sent, such as “Request to disaggregate”, “Refuse to disaggregate” and “Request to aggregate”. In our example scenario, if the aircraft T returns every few minutes, the battalion A_1 would undergo repeated aggregation/disaggregation cycles, flooding the network with protocol messages. Thus, these messages represent an unacceptable overhead. Finally, by disaggregating an AE, we lose the opportunity of message bundling, resulting in many short messages. This can reduce the effective throughput of the network.

4.4 Transition Latency

We refer to the time taken to effect an aggregation or disaggregation as the *transition period*. Transition periods can be significantly long depending on the complexity of the protocol. For example, a proposal in [Robkin92] requires on the order of 10 seconds to complete the aggregation process. This is because each DE could request to be reaggregated, and each could also refuse to be reaggregated, thus stopping the entire process. Long transition periods are incompatible with real-time constraints in human-in-the-loop simulations because they may cause visual or conceptual inconsistencies. An entity that does not change position during a transition period, and then suddenly undergoes a large displacement at the end of the transition period can cause a visual inconsistency or "jump". A conceptual inconsistency may arise in the airborne reconnaissance example as follows. In the 10 seconds that it might take battalion A_1 to disaggregate, aircraft T might have flown away from it and disappeared from the scenario. T would not be able to relay any information about A_1 because A_1 did not disaggregate in time!

4.5 Mapping Inconsistency

The mapping inconsistency problem exists because the attributes at one level of resolution are not consistently mapped to the attributes at other levels. The problem is observed when an entity performs actions in an interval of time in a simulation that it could not have performed in reality. This may happen, for example, during an aggregation-disaggregation sequence. The information stored at an aggregated level may not be sufficient to provide consistency at the disaggregated level. In other words, in the first transition, i.e., disaggregated to aggregated, some information pertaining to the DEs may be lost. Consequently, the second transition may result in a disaggregated state that is inconsistent with the first disaggregated state.

In the case of the airborne reconnaissance, after the aircraft goes away, the DEs in the battalion may reaggregate. While reaggregating, for example, the actual positions of the DEs may be lost. If the aircraft returns within a short time after reaggregation, a disaggregation must be effected. On disaggregating again, a standard algorithm or doctrine [France93] [Clark94] would be applied to position the entities. This might cause unrealistic discontinuities or "jumps" in position.

5 Proposed Approach

It is clear from the above discussions that temporal inconsistency must be solved in order to make multi-level simulations feasible. The temporal inconsistency problem is caused because there is not enough information at any one level that can be translated to information at other levels. Traditional approaches towards aggregation/ disaggregation maintain, at any given time, the attributes at only *one* level of resolution — the level at which the entity is being simulated. This is unsatisfactory for two reasons:

- When the entity is simulated at a certain level of resolution, the attributes at the other levels are unused or lost.
- There is an implicit assumption that the resolution level at which the entity is being simulated (simulation level) is also the resolution level at which it is being perceived (perception level). Thus, entities explicitly aggregate or disaggregate in order to keep their simulation level the same as their perception level.

We believe the perception level of an entity should be uncoupled from its simulation level. This implies that attributes from the simulation level should be consistently transformed into the attributes at the perception level(s). It then follows that each entity should possess attributes at multiple levels of perception. We call such an entity a Multiple Resolution Entity (MRE). By definition, a MRE can be perceived at multiple resolutions.

We propose `UNIFY` as an approach to solving the aforementioned problems. `UNIFY` is based on the use of MREs rather than AEs and DEs. Each MRE either maintains state information at all desired levels (as determined, perhaps, as part of the High Level Architecture subscription process) of resolution or furnishes information at a requested level in a timely manner. Simulation of the MRE entails handling incoming interactions at all desired levels. Each MRE is responsible for enforcing logical consistency across resolution levels: the effect of any incoming interaction should be reflected consistently in the attributes of all levels of the MRE. For example, a platoon unit — a typical MRE — composed of four tanks would contain information regarding the platoon as well as the individual tanks (Figure 2). Similarly, a battalion unit would have information at the battalion level regarding each of its platoons. In turn, each of the platoons would contain information regarding the individual tanks.

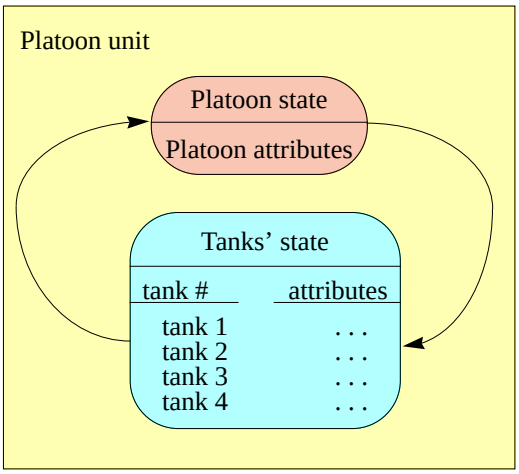

FIGURE 2. Design of a MRE

Let A_1 and A_2 be platoons of tanks and T be a solitary tank. The interactions between A_1 and A_2 occur at an aggregated level. For instance, platoon state information such as velocity and strength may be exchanged and acted upon. When T comes into contact with A_1 it requests tank-level information. A_1 proceeds to send information regarding the tanks of interest to T (Figure 3). Typically, this information would be culled from data the MRE maintains on each tank. A_2 receives information sent from A_1's "global" fields — the fields that are either common to all entities or can be deduced from the individual attributes of the tanks (Figure 3). For example, if T is absent, the velocity of the individual tanks in A_1 would not be important, and a global velocity vector in A_1 could be sent to A_2. However, if T is present, then A_1's velocity vector for A_2 could be computed as a weighted average of the individual velocities of its constituent tanks.

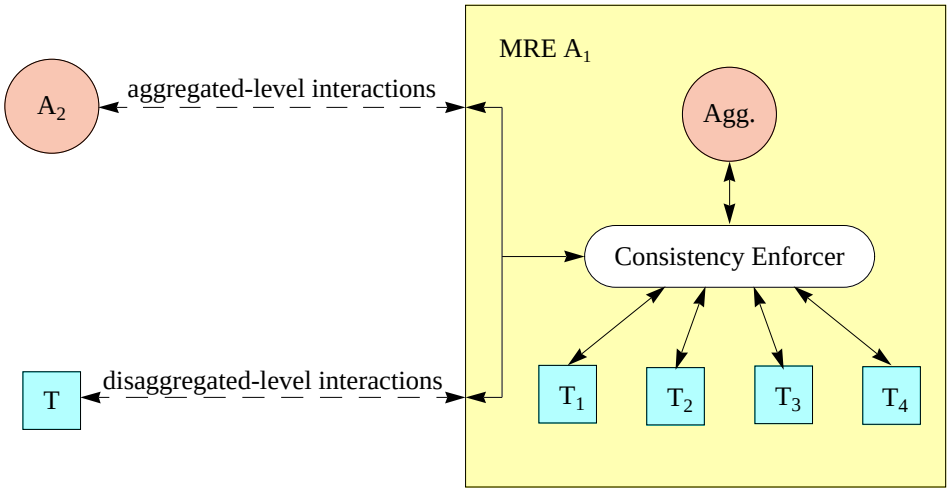

FIGURE 3. Multiple levels of resolution

Consistency maintenance is the key aspect of the `UNIFY` approach. Consistency must be maintained among levels of aggregation. There are two aspects to consistency — temporal and mapping. We address the issue of temporal consistency by imposing an atomicity constraint on interactions between MREs. A_1's interactions with T and A_2 must be serialized and applied to A_1 atomically. When A_1 receives a message regarding an interaction from any other MRE, it must process that message completely before beginning to process any other message that might

arrive. The atomicity constraint* is required because interactions at any level may affect the other levels. If an interaction is from T, then its effect on A_1 should be reflected in subsequent interactions between A_1 and A_2. In this instance, information flows from the high-resolution level to the low-resolution level in A_1. Likewise, if an interaction comes from A_2, then the state of each high-resolution entity in A_1 may have to be updated. In effect, the entity deals with every interaction "atomically" — it completes evaluating the effects of the interaction at *all* levels before it can begin the next one. Atomicity of interactions, while strict, ensures that all levels are consistent with each other. It is worthwhile to note that we have reduced the problem of maintaining consistency between aggregated and disaggregated entities to the task of serializing and atomically handling each request arriving at the unit.

Mapping consistency pertains to designing a pair of functions f and f^{-1} such that f maps a set of attributes at the disaggregated level to a set at the aggregated level while f^{-1} maps attributes from the aggregated level to the disaggregated level.

We propose a model for consistency in Figure 4. A_1, A_2 and A_3 are MREs with multiple levels of resolution, while T is a MRE with one level of resolution. If two entities perceive† a particular MRE at different levels of resolution at overlapping simulation times, then the entities' respective perceptions can be translated from one to the other with the same result as if the entities had perceived the MRE at the other level of resolution. Also, if two entities view the MRE at the same level of resolution at overlapping simulation times, the entities would perceive the MRE in exactly the same way. Note that in the figure, an arrow does *not* imply an interaction between the corresponding entities. A dotted box surrounding an MRE indicates that any view of the MRE from outside this box is temporally consistent with any other view overlapping in time. The left-to-right arrows indicate "perception", while the vertical arrows denote mapping consistency. Also, note that the dashed bidirectional arrow represents "exactly the same", whereas the dashed curved arrows represent "can be translated".

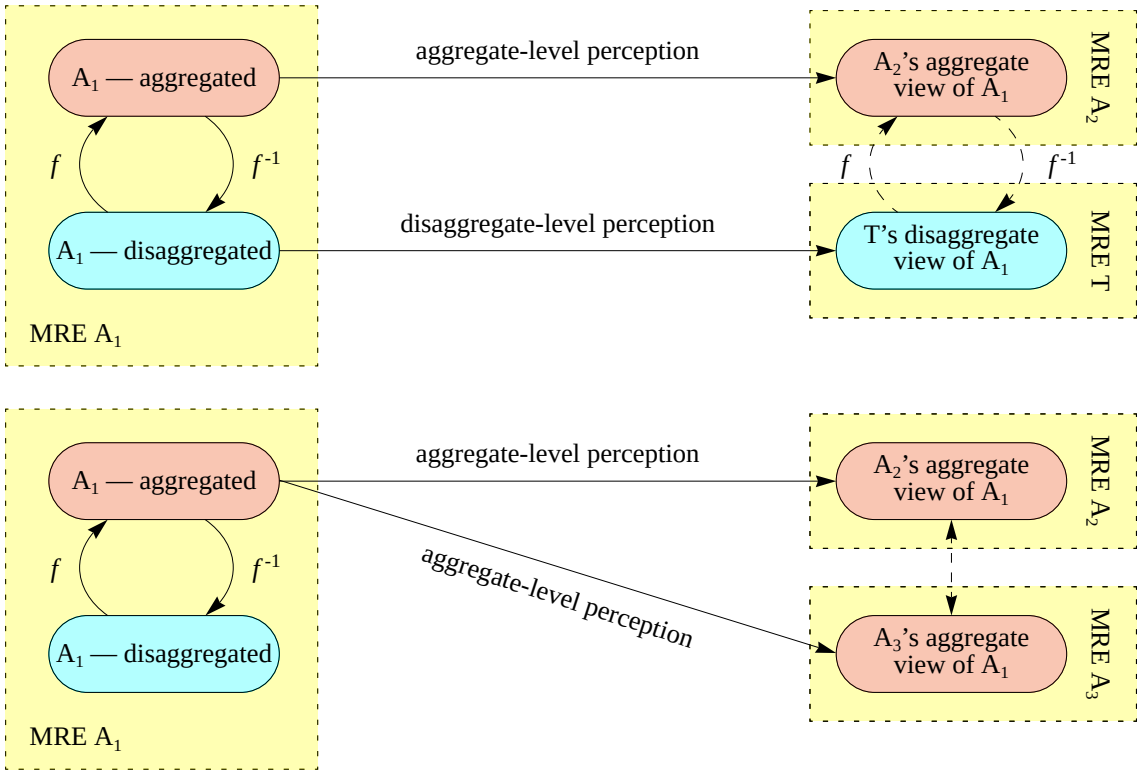

FIGURE 4. Models for consistency

* The concept of atomicity is borrowed from other areas in Computer Science. For example, in the context of memory systems, an "atomic read/write" means that no other read (or write) can be effected until the current read (or write) completes.

† It is assumed that no information degradation occurs in the perception of another entity.

6 Expected Benefits of UNIFY

UNIFY solves or alleviates many of the problems described in Section 4. The following sections discuss the benefits we expect.

6.1 Temporal Consistency

UNIFY directly addresses the temporal inconsistency problem. For the most part, this problem has been ignored in the distributed simulation community. The few attempts at solving this problem have followed *ad hoc* approaches, designing customized translators that maintain consistency. These approaches are conceptually unscalable. As pointed out in Section 4, the temporal inconsistency problem lies at the core of the challenges faced in designing multi-level simulations: *it must be solved in an efficient and scalable manner to make large-scale multi-level simulations feasible.* UNIFY is the first step in this direction.

Temporal consistency guarantees that all entities view a particular MRE in a consistent manner. Also, the perception level is uncoupled from the simulation level. The MRE may be thought of as being simulated at all levels of resolution, irrespective of the levels at which it is being perceived. Temporal consistency has a salutary effect on attrition computation, line-of-sight computations, dead-reckoning and other key aspects of a simulation. These aspects strongly influence the decision-making process in a simulation and hence play a role in establishing the validity of the simulation. Decisions made on the basis of incorrect data affect the usefulness of the simulation.

6.2 Elimination of Chain Disaggregation

Recall the chain disaggregation problem (Section 4.2) shown in Figure 5. UNIFY effectively eliminates this problem since it has no concept of aggregation or disaggregation. Each MRE determines the level of resolution at which it perceives another MRE, and the perceived MRE is able to present consistent views of itself to its perceivers. As seen in Figure 5, each UNIFY entity is an MRE to begin with, so no new entities are created when MREs interact. The interactions between MREs are defined in the design of the simulation. Eliminating chain disaggregation reduces the number of entities participating in the simulation. This reduces the demands made on network and processor resources (discussed below).

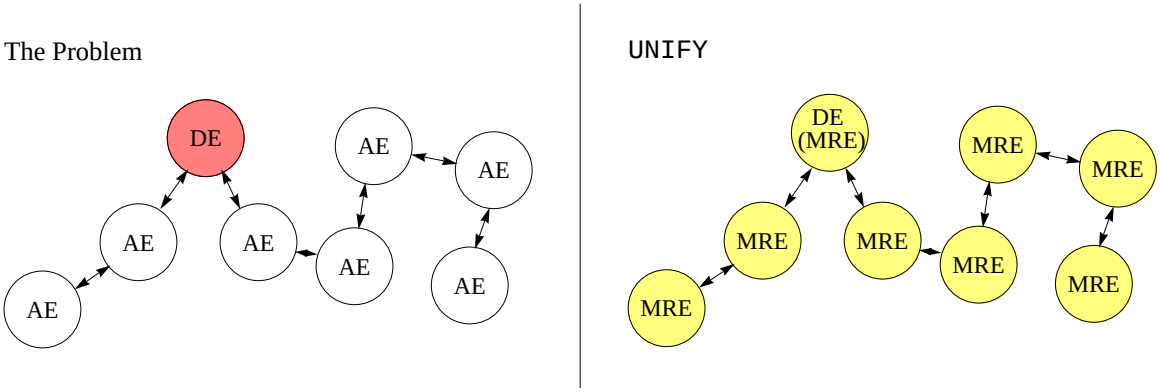

FIGURE 5. UNIFY's approach to solving chain disaggregation

6.3 Reduced Network Traffic

The network flooding problem is alleviated by reducing the message traffic in the simulation in a number of ways:

- UNIFY does not have aggregation/disaggregation cycles. Thus, the number of protocol messages is reduced. This is a significant reduction of overhead.

- UNIFY prevents unnecessary disaggregation, thereby reducing the number of entities in the simulation. This means that there are fewer receivers and senders of messages. This in turn reduces the number of messages sent over the network.
- Message bundling is a technique in which a number of short messages can be packed into a single long message. The packed messages should have some common characteristics — the same set of receivers, for instance. This can increase the effective throughput of the network by causing fewer long messages to be sent rather than many short messages. Since a MRE is responsible for interactions with all of its constituent objects at a particular level, the MRE can bundle messages about those objects. For example, a MRE simulating a platoon of tanks can bundle the state update messages (at the disaggregate level) of all its tanks.

Reducing network traffic affects scalability as well. Network bandwidth places a limit on the number of entities that can be simulated. Reducing network traffic increases this limit, thus improving scalability.

Typically, schemes using dynamic aggregation and disaggregation cause "bursty" traffic on the network, because the aggregation/disaggregation cycles add to the network traffic for short periods. By eliminating aggregation/disaggregation cycles, UNIFY imposes a more "uniform" load on the network, ensuring less variance in latency. Fixed latencies are compatible with real-time constraints in a simulation.

6.4 Reduced Transition Latency

An aggregation/disaggregation cycle can consume a significant amount of time (which we call the transition period), depending on the complexity of the protocol. Large transition periods are incompatible with real-time constraints in simulations. Elimination of aggregation/disaggregation cycles also eliminates the transition period. By design, an MRE has all necessary information to satisfy requests at any level. Thus the major source of latency would be in extracting (or computing) this information, which can be done efficiently.

7 Technical Challenges

In order to demonstrate the feasibility of UNIFY, some issues must be addressed. Issues relating to consistency maintenance are of primary importance.

7.1 Core Attribute Identification

An important issue in the design of MREs is to identify attributes required for maintaining multi-level consistency. While the MRE concept may be beneficial in maintaining consistency, we expect additional benefits to accrue from exploitation of redundancies and high-speed generation of attribute values not explicitly simulated all of the time. In UNIFY, MREs maintain a set of "core" attributes, as shown in Figure 6. These attributes form a minimal set from which other attributes at all levels can be directly found, or generated on request. Core attributes are application-specific. In our research, we expect to gain insights into general guidelines for identifying core attributes for classes of MREs. A rich set of core attributes represents a higher demand on memory resources. Alternatively, a sparse set implies that more attributes will have to be computed requiring more computing time.

7.2 Consistency Maintenance

In order that many entities perceive a particular MRE consistently, the attributes at all levels of resolution of the MRE must be consistent. We have proposed serialization and atomicity as a mechanism for maintaining consistency between levels of resolution (Section 5). We believe consistency maintenance is the key to successfully implementing multi-level simulations. The following sections detail some of the issues regarding consistency maintenance.

7.2.1 Time-step Differential

In multi-level simulations, different levels of resolution may proceed at different time-steps (Section 4.1). The time-steps usually determine the mean time between events of interest at that level of resolution. Paradoxically, the complexity of the effects of the events at a particular level of resolution determines the time taken to compute those effects. In turn, this time plays a role in determining the time-step at which that level of resolution should proceed. A

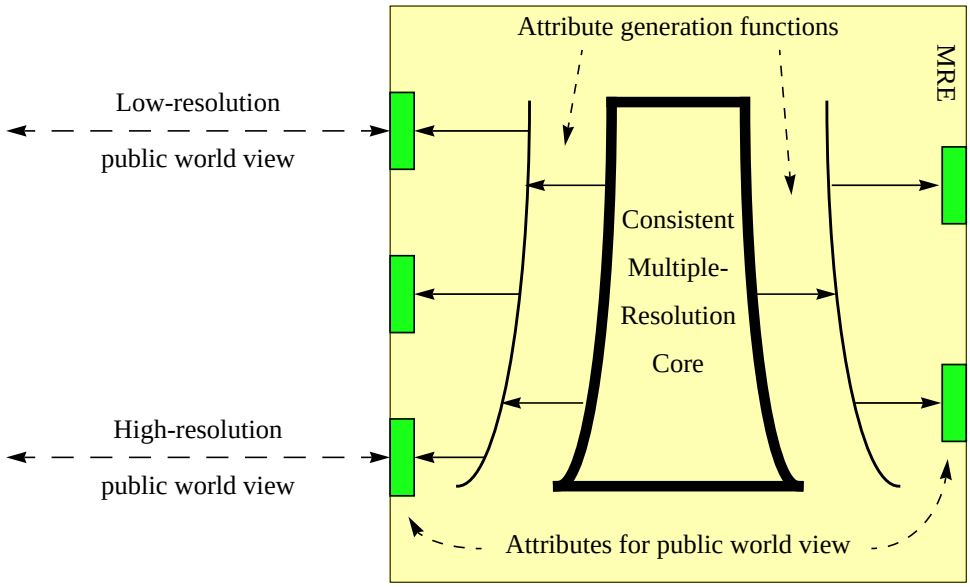

FIGURE 6. Core attributes

substantial difference in the time-steps of different levels of resolution, combined with the serialization requirement poses real-time challenges. There are two approaches to solving this problem: (i) reducing the time-step differential, and (ii) carefully relaxing the serialization and atomicity constraints in order to allow multiple, but consistent, interactions affecting a MRE at overlapping times.

7.2.2 Mapping Functions

The mapping functions f and f^{-1} are required for any approach to multi-level simulations, including an aggregation/disaggregation approach. For a MRE with two levels of resolution, if A and D are the set of attributes at the aggregated and disaggregated levels respectively, then $\langle f \mid 2^{A \cup D} \rightarrow 2^{A} \rangle$ and $\langle f^{-1} \mid 2^{A \cup D} \rightarrow 2^{D} \rangle$, where 2^{S} is the power set of any set S. These functions provide a translation mechanism from one level of attributes to another. Hence they should be considered as an integral part of the design of multi-level simulations. The "rules-of-thumb" suggested in [Allen92] are useful guidelines in the design of f and f^{-1}.

7.3 Load Considerations

In a typical disaggregation approach, computing resources must be available to simulate the newly generated DEs when an AE is disaggregated. Correspondingly, when an MRE begins interactions with other MREs at a high level of resolution, sufficient computing resources must be available to simulate these interactions. One approach to handling this additional load would be to simulate the high resolution interactions using Computer Generated Forces. Multiprocessor implementations of MREs using a small number of processors are also worth investigating. In this respect, `UNIFY` has an advantage over other approaches: since `UNIFY` creates fewer entities, some computations such as dead-reckoning and line-of-sight may be simplified.

Another aspect of load is memory consumption. For comparison, we assume a memory-efficient traditional scheme exists. This scheme allocates memory only for the entities at the level of resolution that is being simulated. `UNIFY` requires memory for the attributes at *every* level of resolution. If n_i is the number of i-level entities per $i+1$-level entity, the memory requirements for a memory-efficient traditional scheme and `UNIFY` would be

$$O\left(\prod_{i=1}^{l} n_i\right) \text{ and } O\left(\sum_{j=1}^{l}\prod_{i=j}^{l} n_i\right)$$

respectively, where l is the number of levels of aggregation. The constants for UNIFY are expected to be large since more data is stored per MRE in order to maintain consistency. It appears that MREs would require more memory than AEs or DEs, but we expect this will be offset by the decreasing cost of memory.

8 Summary

The advent of large-scale distributed simulation programs (such as STOW) requires that entities at different levels of resolution co-exist and interact in a single distributed simulation. This requirement raises several problems that *must* be addressed in a systematic, unified manner for such large-scale simulations to be feasible. At the heart of these problems is the issue of maintaining consistency among levels of resolution. Other problems include chain disaggregation, network flooding, transition latency and mapping consistency.

We have proposed UNIFY, an approach to address these problems. UNIFY defines a new kind of entity, a Multiple Resolution Entity (MRE), which replaces traditional AEs and DEs. A MRE maintains a set of core attributes from which other attributes at all desired levels of resolution can be generated in a timely manner. There are several immediate benefits of the UNIFY approach: temporal consistency, elimination of chain disaggregation and reduction in network traffic and latency. The technical challenges in demonstrating the feasibility of UNIFY include: identifying the set of core attributes, resolving the time-step differential and managing load. We propose research into these challenges with the goal of establishing the feasibility and utility of UNIFY.

9 References

[Allen92] Allen, P. D., *Combining Deterministic and Stochastic Elements in Variable Resolution Models*, Proceedings of Conference on Variable-Resolution Modeling, Washington, DC, May 1992.

[AMG95] Architecture Management Group, *Preliminary Definition*, High Level Architecture Briefings, March 31, 1995.

[Calder95] Calder, R. B., Peacock, J. C., Wise, B. P. Jr., Stanzione, T., Chamberlain, F., and Panagos, J., *Implementation of a Dynamic Aggregation/Deaggregation Process in the JPSD CLCGF*, Proceedings of the 5TH Conference on Computer Generated Forces & Behavioral Representation, Orlando, Florida, May 1995.

[Clark94] Clark, K. J. and Brewer, D., *Bridging the Gap Between Aggregate Level and Object Level Exercises*, Proceedings of the 4TH Conference on Computer Generated Forces & Behavioral Representation, Orlando, Florida, 1994

[Davis92] Davis, P. K., *An Introduction to Variable-Resolution Modeling and Cross-Resolution Model Connection*, Proceedings of Conference on Variable-Resolution Modeling, Washington, DC, May 1992.

[Davis93] Davis, P. K. and Hillestad, R. J., *Families of Models that Cross Levels of Resolution: Issues for Design, Calibration and Management*, Proceedings of the 1993 Winter Simulation Conference, 1993

[DIS93] DIS Steering Committee, *The DIS Vision, A Map to the Future of Distributed Simulation*, Comment Draft, October 1993.

[DoD94] Under Secretary of Defense (Acquisition and Technology), *Modeling and Simulation (M&S) Master Plan*, Dept. of Defense, September 30, 1994.

[France93] Franceschini, R. W., *Intelligent Placement of Disaggregated Entities*, Institute for Simulation and Training, 12424 Research Parkway, Suite 300, Orlando FL 32826.

[Hill92] Hillestad, R. J. and Juncosa, M. J., *Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models*, Proceedings of Conference on Variable-Resolution Modeling, Washington, DC, May 1992.

[Horr92] Horrigan, T. J., *The "Configuration Problem" and Challenges for Aggregation*, Proceedings of Conference on Variable-Resolution Modeling, Washington, DC, May 1992.

[Karr83] Karr, A. F., *Lanchester Attrition Processes and Theater-Level Combat Models*, Mathematics of Conflict, Elsevier Science Publishers B.V. (North-Holland), 1983, ISBN: 0 444 86678 7.

[Karr94] Karr, C. R. and Root, E., *Integrating Aggregate and Vehicle Level Simulations*, Proceedings of the 4TH Conference on Computer Generated Forces & Behavioral Representation, Orlando, Florida, 1994

[Nat95] Natrajan, A., Nguyen-Tuong, A., *To disaggregate or not to disaggregate, that is not the question*, ELECSIM 95, Internet, April-June, 1995.

[Petty95] Petty, M. D., and Franceschini, R. W., *Disaggregation Overload and Spreading Disaggregation in Constructive+Virtual Linkages*, Proceedings of the 5TH Conference on Computer Generated Forces & Behavioral Representation, Orlando, Florida, May 1995.

[Robkin92] Robkin, M., *A proposal to Modify the Distributed Interactive Simulation Aggregate PDU*, Hughes Training, Inc., February 28, 1992.

[Sher92] Sherman, R. and Butler, B., *Segmenting the Battlefield*, Loral WDL, June 9, 1992.

[Smith94] Smith, R., Mystech Associates, *Invited speaker to the Department of Computer Science, University of Virginia*, December 1, 1994.

[Stein94] Steinman, J. S. (Jet Propulsion Laboratory, California Institute of Technology) and Wieland, F. (Naval Research Laboratory), *Parallel Proximity Detection and the Distribution List Algorithm*.

[Weat93] Weatherly, R. M., Wilson, A. L. and Griffin, S. P., *ALSP — Theory, Experience and Future Directions*, Proceedings of the 1993 Winter Simulation Conference, 1993.