ITR: COLLABORATIVE RESEARCH - (ECS+ASE) - (dmc/int)
Live Wires: Always On Adaptive and Editable Simulations With Live Feed

Maria Hybinette (PI) and Eileen T. Kraemer (Co-PI)
Department of Computer Science
The University of Georgia
Athens, GA 30602-7404

maria@cs.uga.edu and eileen@cs.uga.edu

Tucker Balch (Collaborative Co-PI)
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332-0280

tucker@cc.gatech.edu

Project Summary

Intellectual Merit

We will develop new algorithms that enable live, collaborative, dynamic and scalable simulations. We envision large-scale, complex simulation systems connected to real-world live feed data that can be used by lay persons via the Internet, and viewed and edited simultaneously by collaborating scientists. Real-time live-feed data streams will be provided by world-wide-web sources and other Internet sources. In addition we will apply machine learning techniques to improve predictive performance of the simulations Thus we propose innovative approaches to the integration of data-driven applications for use in prediction, risk-assessment and decision making (the dmc technical focus area). Our work is focused on both the economic prosperity and vibrant civil society (ECS) and the advances in science and engineering national priority areas (ASE).

We will demonstrate the generality of our research by applying it to two quite different applications that we will implement: simulation of economic markets and simulation of automobile traffic in the Atlanta, Georgia area. In both applications the world wide web provides real time data on the progress of the corresponding real systems. In particular, for automobile traffic, we will use real-time data from Georgia's Department of Transportation (GDOT) traffic website, as well as data from IQStat, Inc as input to our simulator. For the purpose of providing real-time marketing research on car radio use, IQStat has equipped hundreds of cars in the Atlanta area with GPS receivers, radio monitors, and data transmitters to relay their position back to IQStat's headquarters. The IQStat and GDOT's live-feeds will serve as input to our simulator, enabling it to be more accurate.

The traffic simulation will be accessible to the general public via the world wide web. Individual commuters, for instance, will be able to enter proposed routes and our system will simulate their travel to provide an estimate of the time it will take. Commuters could then compare multiple routes to determine which path they should take to work.

From a practical point of view, we expect our simulation systems to provide helpful information to investors, commuters and other lay persons. From a scientific point of view, several challenging research questions arise and will be addressed in this work:

- How can editing of "live" complex simulations be accomplished?
- How can complex, running simulations be effectively viewed by multiple users simultaneously?
- How can live-feed data be easily connected and synchronized to running simulations?
- How can machine learning technologies be integrated into a simulation system to make it more accurate as a forecasting tool?

By answering these questions we address our second technical focus area which concerns the integration of human-computer interfaces, information management and computing to support complex distributed systems (i.e. int technical focus area).

Broader Impact

This research will have broader impact both within our institutions (the University of Georgia and Georgia Tech) and externally. In the short term, our simulations will be made available through the web to enable commuters to travel more quickly and consequently increasing our economic prosperity by reducing fuel consumption (ECS).

In the long term our technologies will become an important new collaborative medium for scientific investigation in a number of applications and as such address the advances in science and engineering (ASE). Example applications include air traffic control systems, communication networks, and simulation of world-wide manufacturing and sales distribution systems.

Locally this work will include participants from under-represented groups (two of the PIs are female, and students currently involved include African-American, Asian and female students). Additionally, we will integrate this research into several of our graduate and undergraduate courses in the form of research projects for the students.

Contents

Project Description

1 Introduction

The Web abounds with useful time-sensitive data: stock prices, weather radar, web-cams, traffic reports and so on. Millions use this information every day to guide their activities. Furthermore, with the advent of new wireless technologies, access to online information is even more pervasive. People can access the web from their cars, the mall, gas stations, and even from airliners. Presently, however, we can only consider the current state of the world to inform our plans. **What if we could make decisions based on what the world will be like an hour in the future?**

As one example, consider the plight of a commuter on his way to work. At his present location traffic is flowing well, but the radio says there is a traffic jam near his destination. Does it matter? Maybe the jam will be gone by the time he gets there. On the other hand, maybe it won't and an alternative route is called for.

To help this motorist we would need a fusion of online technologies like those found at `mapquest.com` and `georgia-navigator.com` combined with a faster-than-real-time simulation kernel like GTW (Das et al. 1994). It is easy to imagine that such technologies could be fused to provide more powerful services; it is another matter to make it so. In fact, there are critical gaps in current technologies and deep research questions that must be addressed for this to happen.

Our research will enable interactive, persistent (always on) web-based simulations that leverage live feed data to inform users about the predicted state of the world. Furthermore, these simulations will be adaptive on the basis of live feed data, and modifiable by users *while they are running*. These capabilities will support applications like:

- Online automobile route planners that account for current *and future* traffic conditions.
- Adaptable market prediction systems that allow individual investors to test hypotheses regarding the effect of various real-time market indicators on future stock prices.
- Support for commanders in disaster relief, terror response, or battlefield scenarios via predictions regarding the impact of action choices on a rapidly evolving situation.

This research spans several disciplines including discrete event simulation, human computer interaction and machine learning. We have assembled a multi-disciplinary team at the University of Georgia and Georgia Tech with the appropriate skills and interests for success. The project will be led by Professor Maria Hybinette at UGA. Professor Hybinette is a co-creator of Georgia Tech TimeWarp (GTW), an influential discrete event simulation system; currently her research is focused on efficient interactive distributed computing. Professor Eileen Kraemer studies the monitoring, visualization, and interactive steering of long-running computations, looking both at issues in the human interactions with displays and steering and at the concurrency control and performance issues inherent in the dynamic modification of distributed computations. Issues in machine learning and adaptive simulation will be investigated by Professor Tucker Balch at Georgia Tech. Professor Balch's core research centers on learning models of social system behavior by observation.

In this work we will address research questions in three interrelated areas:

- **How can an "always on" simulation respond effectively and efficiently to multiple asynchronous live feed data streams?** We envision simulations that utilize multiple online data sources which provide periodic snapshots of the true state of the world. If the new information differs from what the simulation had predicted, the simulation must be repaired. Furthermore, the data sources may be at multiple resolutions. Numerous subsidiary questions arise, especially when the live feed information can arrive in different forms from distributed sources at differing rates.
- **How can an "always on" simulation be usable and editable while it is running?** In order for an online simulation to be most effective for an individual user, he should be able to revise the structure of the simulation so that the information is tailored to his needs. For instance, a financial analyst may have a strong belief that the price of oil affects the price of GM stock. He should be able to revise the simulation online to reflect such hypotheses. Issues in the presentation of simulation state and the representation of and comparison with "real-world" state will be considered. These issues include both human factors issues in terms of displays and allowable interactions and performance issues in terms of the granularity at which modifications may be applied, the frequency at which real and simulated

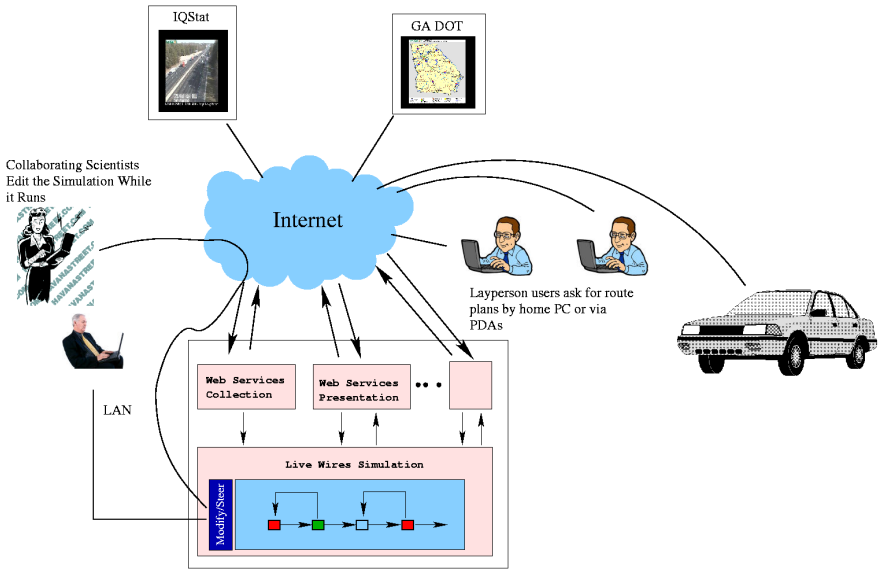


Figure 1: Live Wire System Architecture

states are compared, and the criteria to be used in determining that real and simulated states have diverged.

- **How can we make the most effective use of live feed data to produce a more accurate simulation?** Live feed data provides an unparalleled opportunity for simulation model refinement. In this work we will assume that the structure of the simulation is given by a designer – e.g. Peachtree Street intersects North Avenue at a certain latitude and longitude. Furthermore we assume the various parameters concerning relationships between nodes in the simulation can be learned by observation of live feed data. We will consider several established machine learning techniques, including reinforcement learning, hidden Markov models, and artificial neural nets.

2 Experimental Application Domains

We will apply our research in two application domains: interactive traffic simulation and economic markets. Each of these domains offers unique challenges that will also enable us to demonstrate the broad applicability of our ideas. Both domains offer potential broader benefits society as well.

Our applications will serve two audiences:

1. Researchers and simulation designers who use simulation to build accurate models of real world systems.
2. Laypersons who seek easy online access to the results of these simulations; commuters, for example, who want to know what the traffic will be like.

We envision an architecture like that illustrated in Figure 1. In this architecture, a faster than real time simulation runs on one of our servers (or on several in the case of a parallel simulation). Through a web server, Internet users are able to query the simulation. In some cases, as in the traffic simulation, users will be able to request specific simulation runs for their personal use.

In our paradigm, the underlying simulations are composed of Logical Processes (LPs) that correspond to objects or activities in the real world. In the case of an economic simulation, an LP might correspond to a particular commodity. Of course the value of one commodity might depend on another, so our simulation must enable a researcher to indicate such relationships. Creation of such editing interfaces is one of our proposed research thrusts.

Researchers will be provided a web-based interface for adding LPs and revising the links between them, similar to the example in Figure 2. At this level the editor allows a user to add or remove LPs, and to add or remove data links between them. In the example in the figure, a researcher has indicated that the value of gold depends on the Dow Jones Industrial average as well as on crude oil prices. You will also notice that live

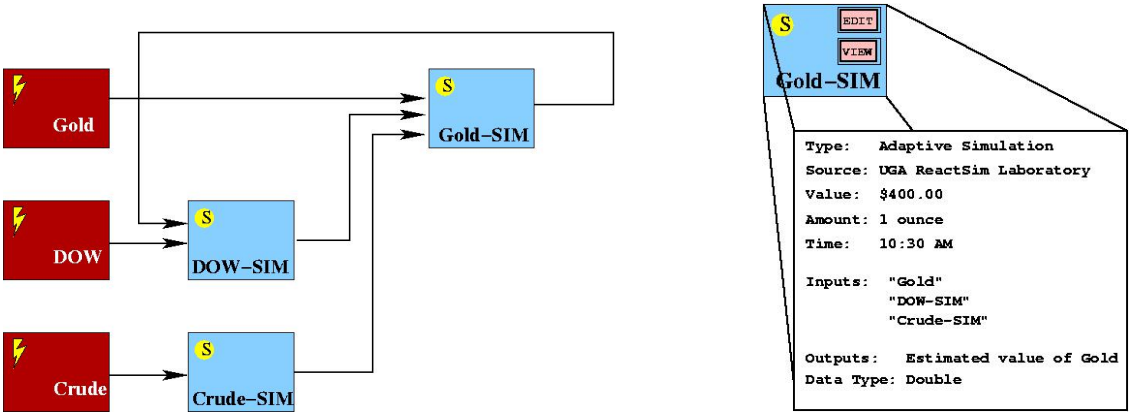


Figure 2: Our vision for the Web based interface for editing simulations. On left live-feed LPs are indicated by a flash, while "simulated" LPs are indicated by an S.

feed information is entering the simulation for gold, DOW and NASDAQ. These live feeds are "buffered" by LPs that simulate their value further into the future. We envision that the buffering LPs will use machine learning techniques to improve their accuracy over time.

The system must provide synchronization points in order to provide coordinated access between the scientists as well as with the simulation allowing updates to the simulation only at "safe" points. We will use different interactive steering approaches to enforce this "safety" in different ways.

We now describe each simulation application, and our research plans regarding them.

2.1 Interactive Traffic Simulation

U.S. transportation systems, including rail, air and roadways are frequently congested. Congestion can lead to severe delays, frustration, and expense for all involved — from the traveler to the transportation authority. Congestion related delays cost highway commuters billions of dollars each year; and businesses whose workers commute are impacted financially as well. In many cases such delays and their related costs could be reduced or avoided with informed planning.

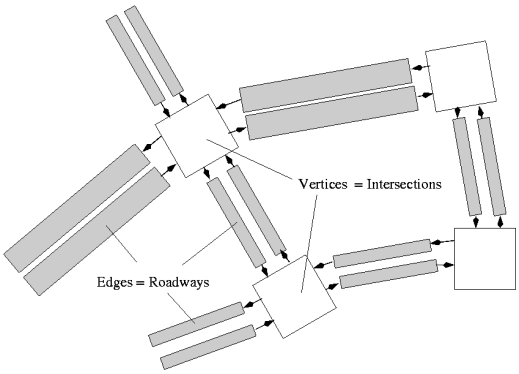
Previously, we have collaborated with researchers at MITRE on a related problem: air traffic simulation. MITRE developed and validated an air traffic simulator called the Detailed Policy Assessment Tool (DPAT). DPAT executes on our parallel simulation executive GTW, developed by one of the PIs (MH) at the Georgia Institute Technology (Das et al. 1994). This work demonstrated the ability to simulate several hours of air traffic in the continental U.S. in less than a minute using a shared memory multiprocessor. DPAT simulates air traffic patterns and computes congestion related delays and through-puts. The software can simulate both international airspace, as well as the continental US (CONUS). For more detail on DPAT please see (Wieland 1998).

We will leverage our experience with transportation simulation in the creation of an automobile traffic simulator. Several such simulations have been created before (Ben-Akiva and Bierlaire 1999; Bottom et al. 1999; Wahle and Schreckenberg 2001). Ours is distinct primarily because it includes consideration for multiple live feeds form different sources, and adaptation or learning (more on those topics later). Please note that it is not our objective to create the "best" traffic simulator – instead we seek to show how a typical traffic simulator can benefits from live feed.

The transportation network is represented as a graph composed of intersections at the vertices, and roads as edges between them. Vehicles are represented as messages that are passed from edges to intersections, where they are distributed back out to edges. Figure 3 illustrates how such a traffic network can be composed.

LPs that represent intersections are parameterized with variables that describe in what ratios incoming vehicles are distributed to outgoing edges. Such parameters can be estimated by a simulation designer *a priori*; they could be based on traffic studies at each intersection; or they could be learned using adaptive techniques coupled with live feed.

Vehicles can be simulated individually, or at a larger scale as groups of 10s or 100s of vehicles. This approach is known as multi-resolution simulation (Reynolds, Jr. et al. 1997). When vehicles are aggregated



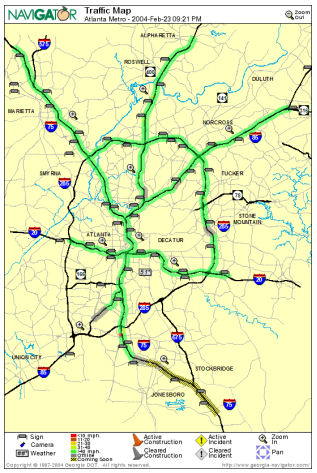


Figure 3: On Left: The traffic simulation will be composed of intersections (squares) and roadways between them. Each roadway link supports unidirectional travel for vehicles. On Right: An example snapshot of the georgia-navigator.com website. The speed of traffic in each direction for each highway is indicated by the color of that stretch of road.

in groups, they can be dis-aggregated at intersections, re-grouped as appropriate for the intersection, then sent out to the roads.

How the System Will Work Our idea is to provide a service similar to that provided by `mapquest.com`: namely an online route planning service. The primary difference is that our system will provide recommendations on the basis of current and *future* traffic conditions.

For demonstration purposes, we will reduce the complexity of route planning by restricting the map to include only primary roadways. Users will click on the map to indicate the major intersection near their start and end points. Travelers will also be asked to enter the start time of their trip. Using this information, as well as current and predicted traffic conditions, a simple route planning algorithm (e.g. A-star) will generate a proposed path for the traveler. We will also enable users to enter and save their own routes and to compare expected travel times between candidate routes on a simple text page. This simplified interface might be used by those with wireless PDAs.

We will allow different interfaces for different types of users by providing three URLs: one URL for those with full-screen capabilities – desktop or laptop users, another URL for those with PDAs and a third for those accessing by cell phone internet service. The interfaces differ in terms of display complexity and interaction style.

Sources of Live Feed Data A key motivation for experiments in this domain is our access to multiple sources of relevant live feed data. In particular we will employ:

- **georgia-navigator.com:** This website provides real time information about traffic speeds on all of Atlanta's major commuter access highways. See Figure 3.
- **IQStat, Inc.:** IQStat has instrumented 100 cars in the Atlanta area with GPS receivers and digital communication devices. By January 2005, IQStat expects to have 500 vehicles in Atlanta equipped with their technology. They are able to report the locations of all vehicles every two minutes. Please see the attached letter of support from IQStat.
- **Internet users:** Our hope is to attract a significant number of Internet users to the service. In addition to our system providing them with information, they will also provide us with information about their planned activities. This information can be used in the simulation just like the other data sources.

These disparate sources of data provide information at multiple resolutions and they are fully asynchronous.

One of the key research issues we will address is how to support multi-resolution live feed simulation.

2.2 Online Markets

In addition to traffic, we will simulate economic markets. Economic market simulations are data-flow driven, (as opposed to traffic simulation, which is spatially oriented). Our goal in addressing this type of application is to provide a more obvious opportunity for editing simulations. We would like to provide an environment in which users can experiment with different hypotheses regarding the relationships between economic entities.

Again, the approach is to represent key objects in the world by Logical Processes (LPs) in a simulation. In the economic simulation, LPs correspond to commodities, equities, and other objects that generate numerical values (e.g. the price gold or the current Dow Jones average). We will enable a user to revise simulations by adding or deleting directed links between the LPs.

A directed link from one LP to another indicates that the output of one LP (e.g. crude oil prices) affects the state of another LP (e.g. the price of gold). We assume implicitly that each LP generates a single output value (e.g. the price of the corresponding commodity). In this simulation there are several special types of LPs:

- Live feed LPs that inject live feed data into the simulation,
- "Regular" LPs that periodically compute a value for their associated commodity according to code provided by the simulation designer,
- Learning LPs that can learn an association between the other commodities they monitor and their own output. Learning will be covered in a later section.

Using these various types of LPs in the simulation editor, a researcher can assemble a running simulation of an economic market (see Figure 2).

3 Simulation

The two major classes of simulations, continuous and discrete, differ in their treatment of simulation time. Continuous simulations treat state changes as occurring continuously over time. The behavior of the system is often characterized as a set of differential equations. Applications typically simulated by continuous approaches include modeling of weather, surface transformations and viscous fluids. In contrast to continuous simulations, discrete simulations treat state changes as occurring at discrete points in time. "Events" define system state changes and represent important activities in the system, such as an arrival or a departure event in an airport simulation. Each event has an associated time stamp indicating the simulated time when the event occurs. This research is concerned with discrete event simulation.

Discrete event simulations typically maintain data structures of state variables, an event list of forthcoming time-stamped events and a global clock that indicates the progress of the simulation. The simulation advances by repeatedly processing the event containing the smallest time stamp from the event list. Processing an event may cause one or more state variables to be modified, and/or new events to be scheduled. For example, a discrete event simulation may model an air traffic system where state variables indicate the number of airplanes at each airport. Departure and arrival events modify these variables as new aircraft arrive or depart from the airport. Typical discrete event simulation applications include modeling ground and air traffic, battlefield management, and communication networks.

3.1 Parallel Discrete Event Simulation

Distributed parallel simulation provides two advantages. First, multiple processors can be used to reduce the execution time of the simulation. Second they may be required to support distributed personnel or resources (e.g., a combat simulator with multiple human participants at different locations). Distributed simulation also facilitates linking existing simulators developed for different platforms to model large systems.

In this work, a parallel simulation is composed of distinct components called logical processes or LPs. Each LP models some portion of the system under investigation. For example in an air traffic simulation each airport might be represented by an LP. The logical processes may be mapped to different processors. As in a sequential simulation, a change in system state is defined by an event. The "scheduling" of an event is accomplished by sending a message from one LP that may request the destination LP to change its state or schedule additional events. For example processing a departure event may result in scheduling a new arrival event at another airport. Since events are scheduled by sending messages, "message" and "event" are

used interchangeably in our discussion. To summarize: distinct components in the simulation are modeled by logical processes and the simulation progresses as LPs exchange time-stamped event messages that cause changes in the system state at discrete points in time.

A synchronization mechanism is used to ensure each LP processes events in time-stamp order. The two leading classes of synchronization protocols are *conservative* and *optimistic* approaches. A conservative protocol enforces consistency by avoiding the possibility of an LP ever receiving an event from its past (as measured in simulated time). LPs *wait* to process events until reception of an out-of-order event is impossible. The optimistic protocol, in contrast, uses a detect-and-recover scheme. When an event is received in an LP's past, an LP recovers by rolling back previously processed events with later time-stamps than the one that was just received.

A benefit of optimistic simulation is that they can be used to enable the inclusion of computationally intensive operations in real-time interactive environments such as synthetic environments (Hybinette and Fujimoto 1999; Hybinette and Fujimoto 2002). In most other cases these environments preclude calculations that run slower than real-time. While optimistic protocols are more complex and require more memory than conservative protocols, they offer greater concurrency by being less reliant on lookahead.

3.2 Cloning Parallel Simulation Programs

Simulation cloning is a technique that replicates a running sequential or parallel simulation program during its execution in order to explore different possible futures of the simulation. It can be used, for instance, to concurrently explore alternative courses of action to deal with emerging events. For example, in an air traffic control setting, one might wish to use simulation to compare alternative approaches to managing the flow of aircraft when inclement weather is forecast for one portion of the air space. This may require controllers to restrict the number of aircraft entering this portion of the airspace. Operationally, this is handled by increasing the spacing between aircraft (called the *miles in trail* or *MIT*) restriction. One might wish to evaluate the overall impact of such restrictions on the flow of traffic throughout the entire traffic network, since delaying certain flights will have a "ripple" effect that propagates throughout the system.

In this example, the simulation can be initialized with the current state of the traffic space, and executed forward until reaching the point where new restrictions are to be imposed. The simulation can then be cloned (replicated), with each clone simulating the traffic space with a different MIT restriction. The point where the simulation is cloned is referred to as a *decision point*. The cloned simulations can execute concurrently, and will produce identical results as a traditional replicated simulation experiment where the entire simulation is executed, from the beginning, using different MIT restrictions that are imposed at the time of the decision point.

Simulation cloning can improve the performance of the simulation in two different ways. First, it is clear that the computation prior to the decision point is performed only once, and its results are shared among the different clones. This is in contrast to a replication experiment where this computation will be repeated for each replication. Second, it is often the case that there is much computation that is common among the clones, even *after* the decision point has been reached. For example, traffic congestion in the eastern part of the U.S. will not affect traffic on the west coast for some time. Therefore, the simulation of air traffic in the west coast will be identical immediately after the clones are created. One would like to also perform these computations only once, and share their results, rather than repeat them within the different clones.

A technique called *incremental cloning* has been developed to allow computations after the decision point to be shared among the different clones (Hybinette and Fujimoto 2001). The basic idea in incremental cloning, is to provide mechanisms to detect when portions of the cloned simulations diverge from each other, and replicate portions of the simulation only as needed.

The benefit of cloning has been demonstrated by evaluating a commercial air traffic control simulation in (Hybinette and Fujimoto 2001). The performance results show that cloning can significantly reduce the time required to compute multiple alternate futures. These results also show that dynamic cloning becomes more advantageous as the number of alternatives increases or the later the decision point is inserted. It is also demonstrated that virtual logical processes are especially efficient when the influence of a message introduced by a logical process does not spread to other processes in the simulation at a high rate. This is shown by using a benchmark that varies the rate of cloning other LPs. It is also shown both analytically and empirically that the advantages of cloning are preserved regardless of the number of clones (or execution

paths).

3.3 Proposed Research in Simulation

The objective of this portion of the research is to address the systems-level challenges associated with enabling simulations to serve as collaborative tools, and for them to accept and respond to live feed data. The tool will *react* to the current state of the world by initializing or correcting a faster-than-real-time simulation based on live-feed data from online sources. The tool must be *interactive* to enable the large-scale simulation to be viewed and edited simultaneously by multiple collaborating scientists. The running simulation will *adapt* by evaluating and responding to differences between state predictions and real-time live-feed data. Achieving these objectives is not trivial, because in order for such a system to provide useful information to a decision maker, it must provide it *quickly*. Accordingly, these advances must be made in the context of a high performance implementation.

The research concerning the simulation executive will focus on four main thrusts:

- *Multi-resolution live feed simulations*
- *Compensatory simulation* to correct and re-use the simulation results with respect to live-feed data.
- *Simulation merging* to enable the efficient, concurrent investigation of multiple execution paths.

The key research challenges for the simulation software arise from the simultaneous requirements that we must simulate *relevant* situations for multiple distributed users at *faster than real-time* rates. "Relevant" means that the simulation must predict future outcomes arising from the current real-world state of the system that is simulated.

The problem of integrating online time sensitive, real-world data (live feed) with faster-than-real-time simulation introduces new, challenging research issues. As an example, simulation forecasting tools must provide information on the basis of accurate world state information.

This work will enable *collaborative simulation* where numerous clients interact with each other through a shared running simulation. Issues we want to explore in this context include: How can efficiency be improved by allowing clients to "share" part of a global running simulation, or can sharing be prevented without impacting the performance of the simulation?

3.3.1 Multi-Resolution Live Feed Simulations

Previous work in multi-resolution simulation (Reynolds, Jr. et al. 1997) is primarily concerned with applications that require aggregation and disaggregation of groups of agents. For example an army platoon can be simulated at one level as a single unit, but at another level as individual soldiers. The platoon is simulated as an aggregated group for movement purposes, but then disaggregated when a combat activity begins.

Observe that live feed data may be available at different resolutions. For example, in our automobile simulation application the `georgia-navigator.com` website will provide information about the average speed of hundreds of cars at five minute intervals. On the other hand data from IQStat will provide the locations of individual vehicles every two minutes. These data are distinct in their (1) spatial coverage, (2) the number of vehicles they concern and (3) the rate at which the information arrives. Our simulation systems must address all of these differences. Accordingly we will investigate how our simulation can allow high and low resolution entities to be present in the simulation at once. Related work include (Chen and Szymanski 2002; Liljenstam et al. 2002), where the first adopts a component based view allows components to be contructed using a C++ class library.

However, a difference in our research is that the challenges here concerns the use of live feed data streams to correct an ongoing simulation. In previous work (Hybinette and Fujimoto 2002) we have shown how an interactive human-in-the-loop simulation can schedule I/O processing in anticipation of the simulation's need for it. The simulation can also respond to user requests that differ from those that the simulation had anticipated. We will leverage this earlier work, but carry it further in two important ways: first, the incoming live feed will be distributed among many LPs, and it will arrive at different resolutions and rates. These issues have not been addressed before. Second, the LPs themselves will *learn* from the live feed data so that they will provide a more accurate simulation as time goes on (see section on adaptive simulation).

Intially we will use time as the basis for integrating multi-resolution data, developing a dynamic scheduler that deals both with regularly scheduled updates and with interrupt-driven type updates, similar to

approaches in scheduling animations. We will also consider real-time aspects of live-feed. For example we may store older, as-yet-unprocessed live feed for our learning algorithms, in order to better refine the learning process. Real-time, closest-to-reality constraints might dictate what we need to garbage collect and keep only the most recent.

3.3.2 Compensatory Simulation and Merging

We propose two new techniques, called merging and compensatory simulations, to improve the performance of simulation by reducing redundant computations. Merging allows replicated computations to *merge* portions of the simulation state so that future computation can share state while compensatory simulations allow new simulations to *re-use* portions of a simulation that have already been constructed or modeled. In these approaches we address how simulations may reduce the number of computations by sharing or re-using common computations.

We are proposing to use a mechanism called *compensatory simulations* to accommodate live-feed data. Here, we will exploit a distributed caching scheme where we will leverage on *previous* simulated computations in order to gain in performance, just as simulation cloning leverage computations by other clones.

A possible consequence of incorporating live-feed data into a simulation is that the system may "thrash" as it repeatedly corrects events that conflict with progress in the real-world. This is likely to happen when simulation application code does not correspond closely enough with the actual process it is simulating. One way to address situations like this is to enable the simulation to *learn* from live-feed data. For instance, if, in a particular situation, a real-world process causes a particular event repeatedly, the simulation should respond in the same way. There are a number of machine learning techniques that can learn from example, and would therefore apply in this domain.

This work will also enable collaborative simulation where numerous clients interact with each other through a shared running simulation. Issues we want to explore in this context include: How can efficiency be improved by allowing clients to "share" part of a global running simulation, or can sharing be prevented without impacting the performance of the simulation?

We are proposing a number of techniques and approaches that vary both in the amount of effort and expertise required. Consequently, we can accommodate both undergraduate, master students and doctoral students.

Compensatory Simulations Simulation forecasting tools must provide information on the basis of accurate world state information. But, as the simulation progresses, the state of the world may change in an unexpected direction, thus invalidating portions of the simulated outcome. To address this, we are developing a new approach, called *compensatory simulation*, to synchronize a simulation's assumptions with quickly changing information about the state of the world.

For example, an air traffic controller may wish to use faster-than-real time simulation to evaluate how a restriction affects the flow of traffic throughout the entire traffic network. Here, the simulation is initialized with the current state of the traffic space, and executed forward until reaching the point where new restrictions are to be imposed. The controller can then evaluate the effect of the restriction via the simulation.

A problem with this approach is that the results of the simulation may be inaccurate because the simulation may not have accounted for recent changes in the state of the world. As an example, suppose a simulation takes 15 minutes to run and that the controller starts the simulation, say, at 10:00 AM and wants to evaluate the impact of restrictions he might impose at 10:30 AM. If at 10:10 AM the state of the air traffic control network changes in a significant manner, the results provided at 10:15 AM will be wrong. However, if the simulation could respond efficiently to live-feed data, it would correct erroneous portions of the simulation, while retaining the results of computations that are unaffected by the change.

To accommodate live feed data we propose *compensatory simulations.* A compensatory simulation share pre-simulated results while accounting for live-feed data. One approach is to rollback the portion of the simulation that is affected by the live feed data and then re-execute forward. A second approach is to start from the beginning of the simulation and examine the results from the original simulation. Both of these approaches may gain performance by leveraging a derivation of the cloning mechanism. Both of these approaches may suffer from excessive re-computation. However, it is likely that there are some conditions

where one approach or the other is effective. We may also study techniques for managing information that may be stored for later re-use. We propose a new scheme called: distributed LP caching for computational re-use and to study efficient storage schemes to address data storage management. These are described below:

- **Distributed LP caching:** Each LP manage a database (cache) of data that informs it of newly scheduled events. Here, the ideas of an LP cache is for the LP to avoid computations that have already been computed.
- **Time Sensitive LP caching:** The performance and usefulness of an LP managing its own database may be dwarfed with the "simplicity" of the LP's computation. We propose to design an LP that monitors how long it takes to calculate itself versus the time of looking up in the database.
- **Efficient Information Storage:** Study techniques for managing information that may be stored for later re-use. The information that needs to be stored may be large and consequently we need new techniques on how to incrementally store (possibly to secondary storage), and how to compress the data and how to load and re-use this information.

Related work in caching include (Walsh and Sirer 2003; Riley et al. 2000; Ferenci et al. 2002) implemented in network simulators. The first is a language based scheme that improves the performance of a sequential network simulator. This approach is similar to function caching as described in (Pugh and Teitelbaum 1989). The second approach caches "routes" in order to avoid maintaining a complete routing table. The third caches the results of a previous simulation runs and re-uses results according to protocol specific knowledge. We propose a more general approach that is a distributed approach and is sensitive to time constraints of computations. We also propose to implement a "learning" cache that learns when to check the cache and when not to. We may use optimization

Simulation Merging Merging of clones would allow cloned computations to *revert* to an un-cloned state. This will increase efficiency because the new un-cloned LPs will complete computations that would otherwise be duplicated. In this part of the research we propose to merge clones when they are sufficiently similar. This idea is the reverse of cloning, as we merge logical processes that have been previously cloned.

One necessary contribution of this work will be a mechanism for evaluating the similarity of clones. We expect that reasonably complex LPs will never again be identical after cloning. However the quantitative outcome of a simulation may be statistically identical if "similar" clones can be merged.

A problem that we will look at here is to determine when a cloned object is sufficiently similar to be un-cloned. We will leverage techniques discovered in the exploration of Just-In-Time cloning (Hybinette 2004). For example we will look at mechanisms that incrementally re-assemble a cloned object (the object may reconstructed from portions of the object that converges). We will also look at techniques that may further improve the efficiency of simulations such as lookback that loosens the constraints of rolling back computatitions (Chen and Szymanski 2003).

4 Usable and Editable Simulations

Interactive simulation allows users to not only view the output of an ongoing simulation, but also permits multiple users to view and edit the internals of he simulation itself, in either a collaborative or independent mode. Thus, two classes of users must be supported:

- **consumers**: those who wish to query (view the output of) the simulation: the drivers in the traffic application, the traders in the economic market application
- **simulation scientists**: those who wish to view/alter the parameters or components of the simulation itself

4.1 Supporting the consumer

It has been said that the purpose of computing is intuition rather than numbers. Similarly, the purpose of each of the live-feed applications that we propose is to provide humans with insight into developing situations and to serve as a decision aid. Each of the experimental application domains described above(Atlanta Road

Network, Online Markets) presents information to and solicits information from users. As with any presentation of data or processes, the interfaces to these simulations must be designed with usability considerations in mind, and the quality of the interface through which the users interact with these applications is critical to their success. High-level design goals for these interfaces are that the software be intuitive(easy to learn, easy to remember, easy to apply to new problems) and powerful(efficient and effective).

These goals must be achieved in an application-specific way. In the Atlanta traffic application, safety is a primary concern. We note that it is best not to distract people who are driving a 4,000-lb vehicle at 70+ miles per hour on a highway with six lanes in each direction and intersections with nicknames such as "Spaghetti Junction". That is, the traffic displays will be used under dangerous, high-stress conditions. The driver must be able to obtain needed information with no more than a quick glance at a screen. Interactions with the display are necessarily limited at drive time, but may be more detailed during a planning phase (before the driver pulls out). Furthermore, the size of the display is constrained, likely the size of a PDA screen, perhaps somewhat larger.

In developing the consumer's interface to the traffic application, we will apply the results of in-depth studies and guidelines of these types of displays, such as (FHW ; Smith and Mosier 1986; MIL 1989), as well as standard human factors design and evaluation techniques including task analyses, heuristic evaluation, and user studies. Although the displays we will develop will share many of the characteristics of other PDA and small-screen displays studied in HCI research, the complexity of the driving task and the potential for risk forces the application of principles from avionic human factors (Marshak et al. 1987). However, the target users of this device will differ substantially from the airline population. Thus, lab research and field studies will be necessary to understand both the task of driving with advanced displays and the cognitively complex task of way finding (Wetherell 1979; Zwahlen and DeBald 1986; Streeter et al. 1985). Further, many differences will exist among individual drivers and many different driving situations will occur, making modeling and generalization difficult (Dingus et al. 1990; Avolio et al. 1985; Pauzie et al. 1989). A major human factors concern for navigation systems (Rockwell 1972; Dingus et al. 1986) is that the addition of visually displayed information, especially moving maps, may divert visual attention from primary visual tasks such as lane tracking and obstacle detection. Although drivers may often have spare attentional resources(Dingus et al. 1988), there may be situations in which processing and response demands exceed capacity. This may result in increased mental workload and errors.

In both the traffic and economic market applications we will develop displays and interactions based on an initial task analysis, followed by cycles of prototyping and user evaluation, in order to develop user interfaces that work well for the tasks at hand.

4.2 Supporting the simulation scientist

Interactive simulation bears many similarities to the practice of interactive steering of computations(Kraemer et al. 1998; Kohn et al. 2000; Vuppula et al. 2000), which permits users to monitor a program's execution and to adjust both application parameters and the allocation of resources in an online fashion. In the case of interactive steering of computations, the user's ability to monitor the program in execution, observe intermediate results, and to "tweak" application parameters, select or install new control algorithms, and direct the allocation of resources allows users to attain increased quality or reduced execution time. Further, the use of well-designed views and controls can provide users with the opportunity to explore the application's behavior and to develop intuition about aspects of it that require remediation or are ripe for optimization. In the case of interactive simulation, the simulation scientist may seek to adjust performance, but is more likely to be making adjustments in order to obtain a higher-fidelity simulation. The interface through which the simulation scientist interacts must provide a suite of display views in support of the tasks of configuring, monitoring, and controlling the editable simulation. Essential views include:

- Statistical presentations, including parameters of the simulated entities (traffic, stock market), deviation of selected simulation parameters from their corresponding live-feed values over time, and performance statistics. The scientist may choose to see these values aggregated since the beginning of the simulation or over some sliding window of time selected by the user.
- An animated display of the simulation in action, at varying levels of detail, represented as LP nodes with messages of various types passing across channels between LPs, again over a user-selected time window. Such a view can help the scientist to develop intuition about the performance of the simulation

- An animated display of the simulation in action, at varying levels of detail, represented in terms of the simulated entities (traffic intersections, moving market averages, for example). Such a view can help the scientist to develop intuition about the fidelity of the simulation (adherence to live feed data). Side-by-side comparisons of the simulated versus actual environments over time might also prove illuminating.
- Detailed views of individual LPs or canonical LPs. The names and values of selected variables at the LPs will be displayed. Associated with these variables are attributes live feed input, writable (anytime), writable (at a synchronization point), evaluation variable, etc.. Such views are useful in verifying the behavior of the simulation, as well as identifying variables that may be used or affected in editing the simulation.
- A display of the simulation environment parameters, such as variables involved in the management of cloning, merging, or rollback.
- A display of the learning modules in use, as well as those available for use.
- A display of available live-feed inputs.
- A display of the evaluation modules and parameters in use, and their parameters.

These views should provide sufficient information for the scientist to monitor the state of the simulation, to make decisions about dynamic changes to the simulation, and to enact these changes. Through this interface, the simulation scientist will be able to enact:

- **changes to LPs:** These include changes to parameter values at an LP, adding, removing, or replacing an LP, directing that clones be created or merged, or altering the policy or mechanism by which LPs are cloned or merged.
- **changes to live-feed status:** These include the addition, removal, or update of live-feeds either to simulation components or to evaluation modules.
- **changes to learning modules:** These include updating parameters of the current learning module(s), as well as addition or removal of learning modules.
- **changes to evaluation modules:** These include updating parameters of the current evaluation module(s), as well as addition or removal of evaluation modules.

Technical challenges exist in realizing such an editable simulation, among these are **safety** in the application of changes to the various components in the simulation system, **performance effects** associated with these dynamic changes, and **usability concerns** in the implementation of the interface. Our prior work in the interactive steering of distributed computations prepares us well to address this similar problem in interactive simulation(Kraemer et al. 1998; Hart et al. 1999; Miller et al. 2001). *Safety* in this context means that dynamic changes to the simulation are applied in a way that maintains the correctness of the computation. Note that although some dynamic changes may be applied at any of the participating LPs at any point in the simulation(altering a local parameter at one LP) , others may be correctly applied only at certain points in the simulation(between transactions in a transaction-based simulation), and still others may require some coordination among LP updates (enforcing simultaneous update of a simulation policy, or transferring responsibility for some task from one LP to another). It should be noted that the approach to safety, including both the degree of control, if any, that is attempted or guaranteed, and the method by which safety is implemented, is dependent upon other choices, related to the structure of the simulation components, instrumentation and data collection methods, and the programming model assumed in the simulation.

In related work in interactive steering of computations, researchers have addressed consistency concerns at a variety of levels. Some systems leave such concerns entirely up to the user or programmer(Winfield 1998)(Beazley and Lomdahl 1997), or rely on the application to provide the necessary consistency checks(Eisenhauer and Schwan 1998). Others, typified by a "scripting approach" , limit the points in the program at which steering may take place, and still others limit the steering system to a particular programming model, i.e., simple iterative computations with a single main loop(Geist et al. 1996). Application-specific steering systems typically solve the consistency problem in an application-specific manner.

At the other end of the spectrum, more complex schemes for controlling interaction points(Vetter and Schwan 1997),(Oberhuber et al.) and detecting changes that affect consistency(Parker and Johnson 1995) have been developed. However, few of these systems address the coordination of steering actions across multiple processes in a distributed system.

In prior work we developed Pathfinder, an architecture for interactive steering of distributed systems that attempts to meet the diverse needs of different users, program models, and applications. In the design of this architecture we emphasized configurability (permitting the user to select an appropriate balance among safety, lag, and perturbation in both monitoring and steering) and support for the coordinated, consistent steering of distributed programs. The programming model assumed in this work was an event-based model, with the option to specify the computation as consisting of a set of (possibly nested) logical actions. These logical action boundaries form good locations at which to both collect data and apply steering changes. For example, in a phased computation, each phase would perform a single logical action across multiple processes. A steering change across multiple processes then could be applied such that it takes place between the same phases at each process. Multiple algorithms for the construction of these logical actions based on event streams were implemented and evaluated(Vuppula et al. 2000) and codes were developed to perform consistent updates to these distributed computations(Miller et al. 2001). Each of these studies involved an evaluation of the performance effects of the implementation choices employed.

Usability of the interface is important issue if the tools we develop are to gain acceptance. Here, our prior work with the development of tools for visualization and interaction with complex systems will guide our approach. We will follow the procedure employed in our prior tool development for Pathfinder and for problems in computational biology(Liu and Kraemer 2001),(Wu and Kraemer 2001),(Zhang et al. 2001), (Farahi et al. 2003),(Wang and Kraemer 2003). That is, we will meet with members of our target user group (simulation scientists),walk through with them the tasks they seek to perform and the interactions and displays they desire, develop prototypes (first on paper, later in software), get feedback, and then redesign. This process is performed iteratively, with functionality added or moved as the design progresses.

5 Adaptive Simulation

Simulations with live feed provide a unique challenge because the simulation can be held back at any time when the live feed data at a particular time contradicts what had been predicted by the simulator for that time. Whenever this happens the simulation (or at least part of it) must be reset (rolled back) and run forward again from the corrected point. Clearly, if we can reduce the frequency of such occurrences, the simulation could run much further into the future, and do so more efficiently.

If the simulation could "learn" not to make mistakes, or, put another way, if it could predict the future more accurately, it could simulate further into the future before a rollback event. Machine learning techniques can address this.

Our approach is to apply machine learning at the Logical Process (LP) level. In particular, consider that each LP is provided a set of input variables (as incoming messages), and from those it should compute an action (or output) that is exported to other LPs. In the case of a market simulation the inputs correspond to incoming information on commodity values that affect the price of the commodity under consideration.

At least two types of machine learning techniques are appropriate to this task: *supervised learning* (Sutton and Barto 1998) and *reinforcement learning* (Mitchell 1997).

In a *supervised learning* system the learner is provided a set input-output pairs for training, where the provided training output is the desired outcome. Object recognition or classification is a common task for such systems (e.g. given an image of a face, whose face is it?). Artificial Neural Nets (ANNs) (Mitchell 1997) are often used for these types of problems. ANNs are *stateless* in that each classification task is independent from those that came before.

In a *reinforcement learning* system the learner is provided sensor inputs and a *reward* for its last action. On the basis of this information it must choose an output. The learner is not provided the "correct answer" as is the case with supervised learning. Instead it must search for actions in particular states that provide positive rewards.

5.1 Proposed Research in Adaptive Simulation

Machine learning is not new to simulation. For instance there is a great deal of work concerning the use of Artificial Neural Nets (ANNs) as a simulation modeling infrastructure (e.g. (Alon et al. 1991; Hajjar et al. 1998)). Others have investigated how ANNs can be simulated efficiently (e.g. (Garca-Orellana et al. 2001;

Barbosa and Lima 1990)). In our own work, we have trained agents that use Reinforcement Learning (RL) in simulation before they were used to control robots (Balch 1998).

Our work is distinct from this other research however, as we focus on the application of ANNs and RL to the task of online adaptive modeling in a running simulation with live feed. We will use live feed information to "train" LPs so that they make more accurate predictions. Presumably, the longer such a simulation runs, the more accurate it will become. Our objective is to investigate how machine learning can be used to create simulations that are faster (because of fewer roll backs) and more accurate.

In the context of discrete event simulation, we can view an LP as an input-output process for which the mapping from input (incoming message) to output (outgoing messages) is carried out by a section of code that follows a possibly stochastic, but nonetheless predictable pattern. The mapping from input to output is usually coded by an application programmer who has substantial experience regarding the behavior of the real-world process that the LP corresponds to.

In our case, with live-feed, we have access to information about the real-world process itself. The general approach is to embed a learning algorithm into LPs to enable them to learn the input-output mapping, possibly without the assistance of an application programmer. As live feed information becomes available it is used online to improve the accuracy of the corresponding LP. For this part of the project we will address the following research questions:

1. How can machine learning algorithms be integrated into the LPs within a discrete event simulation to use live feed to improve its performance?

2. Which types of machine learning algorithms are best for particular applications?

3. To what extent do machine learning augmented LPs improve the run time performance of simulations with live feed?

4. To what extent do machine learning augmented LPs improve the accuracy of simulations with live feed?

As described above, we have a plan already for how to integrate ANNs and RL with LPs in a discrete event simulation (Question 1). With regard to Question 2 we suspect that ANNs will be most appropriate for LPs that perform simple input-output processing and that they can be trained by directly applying live feed as the correct "output;" while RL will be appropriate when LPs act as agents, and must learn to move through a set of states to accomplish a goal. We will also evaluate the impact of this approach by measuring how he resulting simulations improve, both in runtime efficiency and in accuracy.

6 Research Plan

Research will be carried out in three laboratories: the Re-Active simulation laboratory, the BORG laboratory and the Viz Eval laboratory of which the proposers are directors. We have proposed research to span 3 years. This work will be conducted by the proposers, one senior personal, four PhD students and several undergraduate students. Our research plan per year is as follows:

Year 1: UGA: UGA will focus on developing the simulation infrastructure, and on prototyping the various user interface components of Live-Wire. This includes both agent based simulations and event oriented simulators and multiple live feed data streams. We will simulate live feed data by using log files provided by IQStat and the Georgia DOT.

Georgia Tech: Georgia Tech will focus on initial implementations of learning algorithms within discrete event simulations, using log files as simulated live feed.

Year 2: UGA: This year we will integrate real live feed data and evaluate how well our systems work. year we will turn our attention on how to merge similar simulations. We will leverage lessons learned from the compensatory scheme. We will extend the compensatory scheme to include adaptive simulation technologies. We believe that augmenting live-feed with machine learning technologies will have a large impact in our community, and is an ambitious part of our proposal. We will also work on a hybrid cloning and compensatory scheme.

Georgia Tech: GT will continue working on machine learning technologies, now with real live feed.

We will also start working on collaborative applications. We believe that collaborative applications may also benefit from machine learning and cloning technologies in order to improve performance.

Year 3: In this final year we will address optimizations and further refinements of techniques developed in the previous years. We will concentrate on collaborative applications and how to integrate and leverage machine learning technologies.

The PIs teaches several senior/graduate level courses each year related to their research: Operating Systems Concepts, Modeling and Simulation, Advanced Concepts in Parallel and Distributed Simulation, Performance Evaluation of Parallel and Distributed systems, Human Computer Interaction and several courses in robot and multirobot systems.

In all of these courses, students are required to develop principled evaluations of their hypotheses through experimentation. In simulation, human interaction and robotic courses students will gain background knowledge and skills for the on-going research of the PI's simulation laboratories. The best students will be recruited to participate in the PIs laboratories.

7 Impact and Relevance

- **Impact on science and industry:** The proposed research can be applied to any domain where decision makers must make *informed decisions* in time constrained situations. Example applications include air traffic control, battle-management, simulation-based training and communication network management.

  An important element of our previous work, including cloning and latency hiding (work that we are continuing here), is that it has been applied to *real-world problems*. We collaborated with industry and the government using their *real world data*. The driving applications for this work included an air traffic simulator called the Detailed Policy Assessment Tool (DPAT), a military large-scale battle-management system called the THAAD Integrated System Effectiveness Simulation (TISES), and an image sensing application called the Synthetic Scene Generation Model (SSGM).

  DPAT is an air traffic control application that computes congestion related delays and through-puts in the air traffic control system, TISES is a high-fidelity simulation that models all activities performed by a collection of THAAD missile batteries during an engagement scenario, and SSGM is used by Ballistic Missile Defense Organization researchers to produce phenomenologically correct images for testing missile and satellite sensor systems. Details of our work in these domains are described in (Carothers et al. 1998), (Hybinette and Fujimoto 2002) and (Hybinette and Fujimoto 2001).

  It is critical that realistic benchmarks be used in the evaluation of simulation performance. For example in previous work we used PCS, a realistic benchmark that simulates personal communication services networks. A driving goal of our research is to assist users of the the ground transportation network, economic markets and sensor nets, we have already forged collaboration with IQStat which will provide traffic live feeds.

- **Impact on education and under-represented groups:** Three full-time graduate students will be funded at the University of Georgia. We will involve undergraduates as well. The University of Georgia attracts minority and women students to the Department of Computer science by issuing a program called "Institute Fellowships". The PI worked with a graduate female Africa-America student from this program on a project involving cloned simulations, which involved publication of conference paper. The proposed activity will engage both undergraduate and graduate students in research both in computer science and perceptual psychology and include participants from under-represented groups (two of the PIs are female, students currently involved include African-American, Asian and female students). Currently, a bright undergraduate student is working on merging.

  An important component of this course is learning through experimentation with multi-threaded programs on multi-processor machines. The PI taught an undergraduate/graduate course in *Parallel and Distributed Simulation Systems* for the first time in the Spring of 2003. In this course students gain

background knowledge and skills for the on-going research in the PIs simulation laboratory. Due to the popularity of the first course taught over 40 students signed up for the PIs next course on simulation.

The research will also mesh with the educational mission at both the University of Georgia and Georgia Institute of Technology.

- **Outreach on infrastructure:** We will extend the impact of our work in three ways: publications in the popular press and scientific journals, educational exhibits on the web and through open distribution of the software developed under this grant.

  Our simulator GTW, that was developed while the PI was Ph.D. student at Georgia Tech has been distributed to nearly 50 institutions. We will exploit these previous contacts in order to further disseminate our work.

8 Prior Research Accomplishments

- Query-Based Visualization of Executing Distributed Computations NSF Award 9619831, 7-1-97 to 6-30-01, $270,053 (Co-PI with G.-C. Roman).
- CAREER: An Infrastructure in Support of Configurable, Consistent, Interactive Computational Steering, NSF Award 9996082, 9/1/98 to 4/30/2003, $201,617 (Single-PI).
- Instrumentation Grant for Research in Parallel and Distributed Computing NSF Award 9986032, 3/15/2000 to 2/28/2003, $76,303 (Co-PI with D. Lowenthal)
- Collaborative Research: Program Visualization: Using Perceptual and Cognitive Concepts to Quantify Quality, Support Instruction, and Improve Interactions NSF Award 0308063, 6/15/2003 to 4/30/2006, $303,606 Collaborator: Beth Davis, Georgia Tech.

The first of these NSF-funded projects addresses Exploratory Visualization, a novel exploratory approach to program understanding of large, long-running distributed computations. Products of the research include a tool for the monitoring and visualization of distributed computations, the development of a suite of algorithms for the collection of consistent snapshots in distributed computations, and the design and implementation of visualization and user interface techniques to support the user in achieving understanding.

The second project, funded by the NSF CAREER award, addresses the development of algorithms and tools for interactive steering. Interactive steering permits users to monitor a program's execution and adjust application parameters and resource allocation in an online fashion. Steering tools must address issues related to the consistency, lag, scalability, and induced perturbation of monitoring, display, and interaction components. This research focuses on the design and development of infrastructure for a steering environment that permits users to specify the configuration to provide the desired balance among consistency, lag, and perturbation. This project includes the development of modular algorithms for the collection of snapshots with varying degrees of consistency, and algorithms that permit the consistent application of changes to the executing program, while minimizing resulting lag and perturbation. Instrumentation provided through the third grant has permitted the evaluation of performance and usability in a wider variety of configurations.

The fourth project, which is early in its implementation, seeks to evaluate perceptual, attentional, and cognitive aspects of program visualization and to study the effects of these factors on the extent to which program visualizations achieve their goals of communicating information about a program's state and behavior.

Most of Professor Balch's previous research has focused on behavior-based multi-robot systems. He has investigated communication in multi-robot systems (Balch and Arkin 1995), formation control (Balch and Hybinette 2000), and behavioral diversity in multi-robot systems (Balch 2000). Of relevance to this proposal, Balch developed and deployed a multi-robot simulation system called TeamBots (see `www.teambots.org` for more information).

Recently Balch has begun a research program focused on the problem of visual tracking of multiple interacting targets (e.g. ants) using behavior models that is funded by NSF ITR award 0219850. This work has resulted in 5 publications at significant AI and computer vision conferences, and two submissions for journal publication.

9 Roles and Coordination and Management Plan

9.1 Roles

Proposed funding will support two faculty (two months for the PI and one month for Co-PI per year) and three Ph.D. students at the University of Georgia, and one faculty (one month per year) and one Ph.D. student at Georgia Tech.

Professor Maria Hybinette (PI) at the University of Georgia will lead the project. She is also primarily responsible for research issues concerning simulation. Hybinette is the creator of the idea of simulation cloning and one of the creators of the influential discrete event simulation kernel GTW. We plan to leverage simulation cloning in this work.

Professor Eileen Kraemer (Co-PI) at the University of Georgia is primarily responsible for human factors research in the proposed work. Professor Kraemer's research focuses on interactive visualization of large, scientific programs. Professor Kraemer also investigates how such programs can be "steered" while they are running.

Professor Tucker Balch (Co-PI) at Georgia Tech is responsible for applying machine learning techniques to the problem of adaptive simulations. Professor Balch's Ph.D. thesis work included the creation of a large scale multi-robot simulation system. He also has a great deal of experience in multi-agent learning.

Research Scientist TBD at Georgia Tech will support the project as a programmer in the first two years.

9.2 Management and Coordination

The primary driving force in out management plan is the shared goal of creating two "always on" simulation applications. Each researcher has a vested research interest in creating the simulations. Overall, the lead, Professor Hybinette, will direct the creation of the simulation. Integration with user interfaces and learning techniques will be enabled through specification of APIs between the various components.

The University of Georgia and Georgia Tech are approximately 70 miles apart. Our plan is to hold monthly meetings at which all PIs and students attend. We will alternate these meetings between UGA and Georgia Tech. It is our hope that these campus visits will also facilitate additional collaborations between Georgia Tech and UGA. Cost of the monthly meetings will be covered within the travel budgets for each university.

10 Biographies

Prof. Maria Hybinette, PI

Maria Hybinette is an assistant professor in the Computer Science Department at the University of Georgia. Her research area is in the general area of system, that centers on large-scale, high performance, discrete event simulation on parallel shared-memory multiprocessors. A novel contribution of her work is the idea of **cloning ongoing simulations**. This enable running simulations to be split into multiple, parallel simulations at a decision point. The work has been implemented and applied to a **real** large-scale simulation of the air traffic network.

Professional preparation

Emory University	Mathematics and Computer Science	B.S.	1991
Georgia Institute of Technology	Computer Science	M.S.	1994
Georgia Institute of Technology	Computer Science	Ph.D.	2000

Appointments

Assistant Professor	The University of Georgia	2002-
Research Scientist II	Georgia Institute of Technology	1998-2001
Staff Simulation and Modeling Engineer	The MITRE Corporation	1997
Research Scientist I	Georgia Tech Research Institute (GTRI)	1995-1996

Publications most related to proposed research

- **Cloning Parallel Simulations**, Maria Hybinette and Richard M. Fujimoto, ACM Transactions on Modeling and Computer Simulation (TOMACS), October 2001 (Volume 11, Number 4)
- **Latency Hiding with Optimistic Computations**, Maria Hybinette and Richard M. Fujimoto, Journal of Parallel and Distributed Computing, March 2002 (Volume 62, Number 3)
- **Computing Global Virtual Time in Shared-Memory Multiprocessors**, Richard M. Fujimoto and Maria Hybinette, ACM Transactions on Modeling and Computer Simulation, October 1997
- **Just-In-Time Cloning** Maria Hybinette, 18th Workshop on Parallel and Distributed Simulation (PADS-2004) (to appear) Kufstein, Austria, May 2004.
- **Scalability of Parallel Simulation Cloning** Maria Hybinette and Richard M. Fujimoto, In Proceedings of the IEEE 35th Annual Simulation Symposium, San Diego, CA., April 2002
- **GTW: A Time Warp System for Shared Memory Multiprocessors**, Samir Das, Richard M. Fujimoto, Kiran Panesar, Don Allison and Maria Hybinette, In Proceedings of the 1994 Winter Simulation Conference, Orlando, Florida, December 1994

Synergistic activities

Professor Hybinette has participated on the organization committee for RoboCup Junior 2001. She has served as a reviewer for journals such as JPDC, TOMACS, and the International Journal of Formal Methods, and several conferences.

Collaborators and Other Affiliations

Collaborators

Tucker Balch (Georgia Tech), Frank Dellaert (Georgia Tech), Richard Fujimoto (Georgia Tech), John Miller (UGA), Eileen Kraemer (UGA), George Riley (Georgia Tech, ECE), Karsten Schwan (Georgia Tech), Fred Wieland (MITRE Corporation).

Graduate Advisors

Richard Fujimoto (Georgia Tech), Karsten Schwan (Georgia Tech).

Advising Students

MS in progress: Abhishek Agarwal, Abhishek Chugh **MS completed**: 2003 - Vinay Sachdev. **PhD in progress**: Yin Xiong. I also advice 5 undergraduate students regarding their course work and career goals.

Graduate Committee Member

David Miller (MS 2001), Arumugaraja Selvaraj (MS 2002), Nan Li and Chetna Warade (MS 2003)

11 Facilities and Equipment

Prof. Hybinette's laboratory, the Re-Active Simulation lab, houses a 5 node Myrinet-2000 cluster. Four nodes in the clusters are 4 Dell Power Edge 6450 Pentium III Xeon, the fifth node is a Dell PowerEdge 2550, which serves as a front end machine. The cluster is also connected via Fast Ethernet. The laboratory also contains a Dell Precision 220 MiniTower Pentium III, and two Dell Precision 620 MiniTowers which are dedicated for student research. The laboratory also include two printers, a HP laserjet and a HP color laserjet printer.

Prof. Kraemer's laboratory includes a 1 Gdb Network with the following computing equipment:

- 1 Cluster of Eight quad Pentium Xeon 500 MHz processors
- 1 Dell Server
- 4 VA Research linux workstations
- 10 Dell Intel Pentium IV computers
- 1 Silicon Graphics Octane
- 2 Hewlett Packard 4000N Laserjet Printer
- 1 Canon color laser printer
- 1 Virtual Research V6 VR Helmet with Polhemus IsoTrac II
- 1 Liberty 6-degree-of-freedom tracker with 8 receivers also an eye tracker, data gloves, and controllers

The Computer Science Department operates three TCP/IP networks of UNIX machines and two IPX Novell networks in a configuration which is undergoing continual expansion. Utilizing standard software components, computers from several vendors can be linked into a powerful computing resource that includes advanced networking services and high quality programming environments.

The backbone of the Department's UNIX network consists of several SUN Microsystems servers, including several multiprocessors. Connected via Ethernet/Fast Ethernet to these servers are over 80 UNIX systems (primarily SUN Ultra's). For output, the networks connect several laser printers, color printers, line printers and pen plotters. For specialized input, the networks connect a Hewlett-Packard scanner, a Sony video camera and Data Translation image capture board, and 8 SUN video camera/image board systems. The Novell networks are served by high-end Personal Computers (PCs) and connect over 80 Pentium PCs.

The Computer Science Department currently houses three parallel machines:

- Intel i860 Hypercube with 8 processors
- MasPar 2202 SIMD system with 2048 processors
- Big Red Machine (a high-speed cluster of PCs)

This equipment enables the faculty and graduate students to conduct research in advanced areas of Computer Science.

Outside the Department, University Computing and Network Services (UCNS) supports mainframe computers, research machines and computing clusters for general campus-wide use by students and faculty. This include a SGI rack-mount Origin 2000 with 24 X 300 MHz MIPS R12000 processors with 4MB cache memory and 8 GB of system memory, two IBM SP2 systems, both with nodes containing dual 375 MHz processors. The first system (Frame 1) is made up of 16 nodes connected by high-speed switch. Each node has 512 MB memory. The other system (Frame 2), has eight nodes, each with 1 GB memory.

Prof. Hybinette laboratory and the Computer Science Department facilities is currently run by the computer support staff presently at the Computer Science Department. The staff presently consist of three full time Computer Services Specialists. Their duties include support for instructional computing as well as for research.

References

Development of human factors guidelines for advanced traveler information systems and commercial vehicle operations: Comparable systems analysis. Technical Report FHWA-RD-95-197. http://www.fhwa.dot.gov/tfhrc/safety/pubs/95197/index.html.

1989. Human engineering design criteria for military systems, equipment, and facilities. Technical Report MIL-STD 1472D, U.S. Government Printing Office, Washington, D.C.

Alon, N., Dewdney, A. K., and Ott, T. J. 1991. Efficient simulation of finite automata by neural nets. *JACM 38*, 2, 495–514.

Avolio, B., Kroech, K., , and Panek, P. 1985. Individual differences in information processing ability as a predictor of motor vehicle accidents. *Information Processing Letters 27*, 557–585.

Balch, T. 1998. Behavioral diversity in learning robot teams. PhD Thesis, College of Computing, Georgia Institute of Technology, Atlanta, Georgia.

Balch, T. 2000. Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous Robots 8*, 3 (July).

Balch, T. and Arkin, R. 1995. Communication in reactive multiagent robotic systems. *Autonomous Robots 1*, 1.

Balch, T. and Hybinette, M. 2000. Social potentials for scalable multi-robot formations. In *Proceedings of the 2000 IEEE International Conference on Roboti cs and Automation* (San Francisco, CA, May 2000).

Barbosa, V. C. and Lima, P. M. V. 1990. On the distributed parallel simulation of hopfield's neural networks. *Software - Practice and Experience 20*, 10, 967–983.

Beazley, D. and Lomdahl, P. 1997. Extensible message passing application development and debugging with python. In *Proceedings, International Parallel Processing Symposium (IPPS'97)* (Geneva, Switzerland, 1997).

Ben-Akiva, M. and Bierlaire, M. 1999. Discrete choice methods and their applications to short-term travel decisions.

Bottom, J., Ben-Akiva, M., Bierlaire, M., Chabini, I., Koutsopoulos, H., and Yang, Q. 1999. Investigation of route guidance generation issues by simulation with dynamit.

Carothers, C., Hybinette, M., and Fujimoto, R. M. 1998. Towards parallelization of large-scale ada simulations using time warp. In *Proceedings of the 1998 International Summer Computer Simulation Conference* (July 1998).

Chen, G. and Szymanski, B. K. 2002. Cost: A component-oriented discrete event simulator. In *Proceedings of the 2002 Winter Simulation Conference* (2002), 776–782.

Chen, G. and Szymanski, B. K. 2003. Four types of lookback. In *Proceedings of the* 17^{th} *Workshop on Parallel and Distributed Simulation (PADS-2003)* (June 2003), 4–10.

Das, S., Fujimoto, R., Panesar, K., Allison, D., and Hybinette, M. 1994. GTW: A Time Warp system for shared memory multiprocessors. In *Proceedings of the 1994 Winter Simulation Conference Proceedings* (December 1994), 1332–1339.

Dingus, T., Antin, J., Hulse, M., and Wierwille, W. 1986. Human factors test and evaluation of an automobile moving-map navigation system. part 1: Attentional demand requirements. Technical Report CR-86/12/05, General Motors Research Laboratories, Warren, MI.

Dingus, T., Antin, J., Hulse, M., and Wierwille, W. 1988. Human factors associated with in-car navigation system usage. In *Proceedings of the Human Factors Society 32nd Annual Meeting* (Santa Monica, CA, 1988), 1448–1452. Human Factors Society.

Dingus, T., Antin, J., Hulse, M., and Wierwille, W. 1990. Attention and performance while driving with auxiliary in-vehicle displays. Technical Report TP-10727, Transport Canada, Ottawa, CA.

Eisenhauer, G. and Schwan, K. 1998. An object-based infrastructure for program monitoring and steering. In *2nd SIGMETRICS Symposium on Parallel and Distributed Toos (SPDT'98)* (Welches, Oregon, USA, August 1998).

Farahi, K., Whitman, W. B., and Kraemer, E. T. 2003. Red-t: Utilizing the ratios of evolutionary distances for determination of alternative phylogenetic events. *Bioinformatics 19*, 16 (Nov), 2152–2154.

Ferenci, S. L., Fujimoto, R. M., Ammar, M. H., and Perumalla, K. 2002. Updateable simulation of communication networks. In *Proceedings of the* 16^{th} *Workshop on Parallel and Distributed Simulation (PADS-2002)* (May 2002), 107–114.

Garca-Orellana, C. J., Aligu, F. J. L., Velasco, H. M. G., Macas, M. M., and Sotoca, M. I. A. 2001. Large neural network: Object modeling and parallel simulation. *International Journal on Artificial Intelligence Tools 10*, 3, 373–385.

Geist, G. A., Kohl, J. A., and Papadopoulos, P. M. 1996. CUMULVS: Providing fault-tolerance, visualization, and steering of parallel applications. *SIAM*.

Hajjar, D., AbouRizk, S. M., and Mather, K. 1998. Integrating neural networks with special purpose simulation. In *Winter Simulation Conference* (1998), 1325–1332.

Hart, D., Kraemer, E., and Roman, G.-C. 1999. Consistency considerations in the interactive steering of computations. *Internation Journal of Parallel and Distributed Systems and Networks 2*, 3, 171–179.

Hybinette, M. 2004. Just-in-time cloning. In *Proceedings of the* 18^{th} *Workshop on Parallel and Distributed Simulation (PADS-2004)* (May 2004). to appear.

Hybinette, M. and Fujimoto, R. M. 1999. Optimistic computations in virtual environments. In *Proceedings of the 1999 International Conference on Virtual Worlds and Simulation* (January 1999), 39–44.

Hybinette, M. and Fujimoto, R. M. 2001. Cloning parallel simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 11*, 4, 378–407.

Hybinette, M. and Fujimoto, R. M. 2002. Latency hiding with optimistic computations. *Journal of Parallel and Distributed Computing 62*, 3, 427–445.

Kohn, B., Kraemer, E., Hart, D., and Miller, D. 2000. An agent-based approach to dynamic monitoring and steering of distributed computations. In *IASTED Parallel and Distributed Computing and Systems 2000* (Las Vegas, Nevada, Nov 2000).

Kraemer, E., Hart, D., and Roman, G.-C. 1998. Balancing consistency and lag in transaction-based computational steering. In *Proceedings of the 31st Hawaii International Conference on System Sciences* (Kailua-Kona, HI, Jan. 1998), 137–147.

Liljenstam, M., Yuan, Y., Premore, B., and Nicol, D. 2002. A mixed abstraction level simulation model of large-scale internet worm infestations. In *Proceedings of the Tenth IEEE/ACM Symposium on Modeling, Analysis, and Simulation of Computer Telecommunication Systems* (October 2002).

Liu, R. and Kraemer, E. 2001. Strategies for improving multiple alignment of retrotransposon sequences. Poster Presentation.

Marshak, W., Kuperman, G., Ramsey, E., and Wilson, D. 1987. Situational awareness in map displays. In *Proceedings of the Human Factors Society 31st Annual Meeting* (New York, NY, 1987), 533–535. Human Factors Society.

Miller, D. W., Guo, J., Kraemer, E., and Xiong, Y. 2001. On-the-fly calculation and verification of consistent steering transactions. In *Supercomputing 2001* (Denver, Colorado, Nov 2001).

Mitchell, T. M. 1997. *Machine Learning*. McGraw-Hill, New York.

Oberhuber, M., Rathmayer, S., and Bode, A. Tuning parallel programs with computational steering and controlled execution. In *Proceedings of the Thirty-First Hawaii International Conference on System Sciences* (Maui, HI), 157–166.

Parker, S. and Johnson, C. 1995. SCIRun: A scientific programming environment for computational steering. In *Supercomputing '95* (1995).

Pauzie, A., Marin-Lamellett, C., and Trauchessec, R. 1989. Analysis of aging drivers' behaviors navigating with in-vehicle visual display systems. In *Vehicle Navigation and Information Systems (VNIS '91)* (New York, NY, 1989), 61–67. Institute of Electrical and Electronic Engineers.

Pugh, W. and Teitelbaum, T. 1989. Incremental computation via function caching. In *Conference Record of the 16th Annual ACM Symposium on Principles of Programming Languages,(POPL-1989)* (May 1989), 315–328.

Reynolds, Jr., P. F., Natrajan, A., and Srinivasan, S. 1997. Consistency maintenance in multiresolution simulation. *ACM Trans. Model. Comput. Simul. 7*, 3, 368–392.

Riley, G. F., Ammar, M. H., and Fujimoto, R. 2000. Stateless routing in network simulations. In *Proceedings of the Eigth International Symposium on Modeling, Analysis, and Simulation of Computer Telecommunication Systems(MASCOTS)* (August 2000), 524–531.

Rockwell, T. 1972. *Skills, judgment, and information acquisition in driving*, 133–164. Wiley Interscience, New York, NY.

Smith, S. and Mosier, J. 1986. Guidelines for designing user interface software. Technical Report ESD-TR-86-278, The MITRE Corp., Bedford, MA.

Streeter, L., Vitello, D., and Wonsiewicz, S. 1985. How to tell people where to go: Comparing navigation aids. *International Journal of Man-Machine Studies 22*, 549–562.

Sutton, R. S. and Barto, A. G. 1998. *Introduction to Reinforcement Learning*. MIT Press.

Vetter, J. and Schwan, K. 1997. High performance computational steering of physical simulations. In *Proceedings of the 11th International Parallel Processing Symposium* (Geneva, Switzerland, April 1997).

Vuppula, H., Kraemer, E., and Hart, D. 2000. Algorithms for the collection of global snapshots in a distributed system: An empirical evaluation (Las Vegas, NV, Aug 2000). 197–204.

Wahle, J. and Schreckenberg, M. 2001. A multi-agent system for on-line simulations based on real-world traffic data. In *HICSS* (2001).

Walsh, K. and Sirer, E. G. 2003. Staged simulation for improving the scale and performance of wireless network simulations. In *Proceedings of the 2003 Winter Simulation Conference* (December 2003).

Wang, J. and Kraemer, E. 2003. Gfpe: Gene-finding program evaluation. *Bioinformatics 19*, 13 (Sep), 1712–1713.

Wetherell, A. 1979. Short-term memory for verbal and graphic route information. In *Proceedings of the Human Factors Society 23rd Annual Meeting* (Santa Monica, CA, 1979), 464–469. Human Factors Society.

Wieland, F. 1998. Parallel simulation for aviation applications. In *Proceedings of the IEEE Winter Simulation Conference* (December 1998), 1191–1198.

Winfield, A. 1998. A virtual laboratory notebook for simulation models. In *Proceedings, Pacific Symposium on Biocomputing '98* (Maui, HI, 1998), 177–188.

Wu, T. and Kraemer, E. 2001. Expression profiler: Software to analyze and visualize gene expression profiles. Poster Presentation.

Zhang, Y., Tian, H., Arnold, J., and Kraemer, E. 2001. A visualization system for protein interaction mapping. Poster Presentation.

Zwahlen, H. and DeBald, D. 1986. Safety aspects of sophisticated in-vehicle information displays and controls. In *Proceedings of the Human Factors Society 30th Annual Meeting* (Santa Monica, CA, 1986), 256–260. Human Factors Society.