ARPA Reconfigurable Simulator Initiative (ARSI)

Duke Buster
Jim King
Texas Instruments Incorporated
Plano, Texas

ABSTRACT

ARSI is a low cost, Distributed Interactive Simulator (DIS)-compliant simulation that can easily change shape into different vehicles. ARPA will use ARSI to explore the viability of such simulators for training and the research, development, and evaluation of future vehicle concepts. We contend that a single reconfigurable simulator will maintain the required fidelity and be less expensive than a collection of single configuration simulators.

ARSI has five areas of reconfigurability: mechanical enclosure, distribution of simulation functions, crew/vehicle interface, tactical interaction with other vehicles, and scenario / battlefield database. The keys to easy configuration are a flexible "core" from which hardware and software modules can be hung, and emphasizing the use of models whose behaviors are table-driven or parameter-driven. The baseline ARSI program will deliver this "reconfigurable core" and modules for five vehicle configurations: M1A1 Abrams, M1A2 Abrams, M2A1 Bradley, M2A2 Bradley, and HMMWV scout.

Keywords: simulator, reconfigurable, DIS, platoon training, concept development

BIOGRAPHIES

Duke Buster

Duke is the lead software engineer for ARSI. He is also responsible for the system integration and test. Duke has worked at Texas Instruments since graduating from Texas A&M in 1986. He has developed several fast prototype aircraft simulations for TI product development and man-in-the-loop testing. Duke's emphasis in these simulations has been the database and modelling functions. Outside of simulation work, Duke has developed a number of decision aids for mission planning and mission execution.

Jim King

Jim is the lead systems engineer for the ARSI program. He has worked at Texas Instruments since 1985. Prior to his work on ARSI, Jim worked on several projects involving simulation, mission planning, flight tests, and test equipment. Some of these projects include the AIWS Tactical Aircraft Mission Planning (TAMPS) interface, Covert Penetration System algorithm development and flight test, and the Thirsty Saber Flight Test Control System. Jim graduated in 1985 from the University of Texas at Dallas with a Bachelor's degree in Mathematical Sciences (computer science) with prior coursework in Electrical Engineering at Southern Methodist University and Texas Tech University.

ARPA Reconfigurable Simulator Initiative (ARSI)

Duke Buster
Jim King

WHAT IS A RECONFIGURABLE SIMULATOR?

Currently, a reconfigurable simulator is not a well defined object. People generally expect a reconfigurable simulator to be:

1) easily expanded, reduced, or altered in hardware and software components,
2) easily upgraded to new hardware and software,
3) modifiable by a documented procedure,
4) modifiable within typical resources - personnel, funding, schedule, etc.

With changes in the hardware and software, the simulator may be given a different appearance, function, performance, or cost.

Reconfigurability is a relative value. The more changes that can be made to a simulator given limited resources, the more reconfigurable the simulator. Any simulator can be modified with enough money, equipment, and personnel. To discuss reconfigurability, we have to define what pieces of a simulator can be modified and limits on the required resources.

WHY ARE RECONFIGURABLE SIMULATORS INTERESTING?

The basic reasons for considering reconfigurable simulators are:

1) the potential to save money for a group that wants more than one vehicle simulation,
2) the potential to save development time and cost for building new simulations.

Currently, a group that wants multiple vehicle simulations must develop or purchase multiple simulators. Much of the equipment between simulators is redundant, such as the enclosures and computer equipment. The group must also provide the storage space and maintenance for multiple simulators. One "good" reconfigurable simulator can save the cost of the redundant hardware (particularly expensive computer equipment) and support costs. The "good" phrase above hides the possible difficulties of a reconfigurable simulator. To be good, the simulator must maintain the required fidelity of all the vehicles, and require little effort to change shape between vehicles. A group that wants only one existing simulation would not need to consider reconfigurable simulators. The modules in a reconfigurable simulator are typically more expensive than single configuration simulator modules because the reconfiguration demands more flexibility.

For an organization developing a new simulation, a reconfigurable simulator is interesting because those modules that are common to the old and new simulations can be reused. This cuts down development costs, particularly if the reused models have already been validated and verified. Reusing a reconfigurable simulator may also shorten the development time for the new modules by providing pre-defined interfaces and functions that the designers do not have to create.

To be effective in either role (multiple vehicles or a development platform) a reconfigurable simulator must be modular. This is the basic technical challenge to making a reconfigurable simulator: maintaining the required fidelity over a range of emulations while allowing the easy addition of new modules.

HOW RECONFIGURABLE IS ARSI?

Our customer wanted a prototype reconfigurable simulator with as broad a configuration power as we could design in 18 months and the given budget. Our customer specifically requested that ARSI initially emulate five ground vehicles, provide hooks to easily expand to other vehicles and aircraft, provide a skeleton for future concept vehicle definition, and have a low recurring cost for multiple footprints. PM-CATT and Battle Labs personnel assisted us in defining the function scope for ARSI. After defining the function scope, we identified five required areas of reconfigurability:

1) Mechanical enclosure - the enclosure includes the floor, frame, seats, monitors, etc. Reconfiguring the enclosure means we can change the cockpit layout, number of crewstations, etc.
2) Distribution of simulation functions - the software modules are distributed among

several computers across a standard network. Reconfiguring the function distribution means we can change the computers on the network and put whatever software modules we wish on each computer. This also allows us to incorporate existing systems.

3) Crew/vehicle interface - this interface includes the visuals and controls provided to the crewmembers. Reconfiguring the crew/vehicle interface means we can move or change: out-the-window visuals, actual hardware controls such as grips and pedals, or panels emulated with computer displays and touchscreens.
4) Tactical interaction with other vehicles - this interaction takes place across a network to other simulations controlling the vehicles. Reconfiguring the interaction with other vehicles means ARSI can game on Distributed Interactive Simulation (DIS) and Simulation Network (SIMNET) protocol networks.
5) Scenario / battlefield database - this database(s) includes the battlefield data for the models and visuals. Reconfiguring the battlefield database means we can create and load a new gaming area from digitized databases in formats such as the Standard Simulator Database (SSDB) Interchange Format (SIF) and the Defense Mapping Agency (DMA) formats.

We set some reconfiguration goals to perform typical changes between existing vehicle configurations. These goals are listed below. Our goals assume the people making the changes are familiar with the Operators Guide or Reconfiguration Guide (manuals for changing the simulator between defined configurations or creating a new configuration, respectively). At the time of writing this paper, we have defined some reconfigurability metrics mirroring our goals but have not collected any numbers.

Mechanical enclosure:
- Assemble the simulator from stored state to a running configuration - 2 people, 1 day
- Change from one defined vehicle configuration to another - 2 people, 1/2 day
- Add or move a monitor - 1 person, 1 day (assuming the other cockpit pieces do not have to move)

Network distribution:
- Remove a machine from the network - 1 person, 1/2 day
- Add a Unix or VMS machine to the network - 1 person, 1/2 day
- Move a software module from one Unix host to another - 1 person, 1 minute
- Move a software module to a Unix machine to VMS or vice versa - 1 person, 1 hour (assuming the code is portable, this includes copying data files and recompiling)

Crew/vehicle interface:
- Modify the out-the-window visuals fields-of-view - 1 person, 1 hour
- Add a new out-the-window visual - 1 person, 1/2 day (assuming the cockpit layout is complete and the channel limit on the image generator computer is not exceeded)
- Modify an emulated controls panel - 2 people, 2 days (assuming the modelling is complete and the change is well defined)

Scenario/database:
- Switch from one defined database to another between exercises - 1 person, 1 minute
- Build a new database from digitized data - 1 person, 2 days

Although ARSI is designed for the easy addition of new modules or cockpit layouts, we have not set any reconfiguration goals for new developments. These efforts depend too much on the design complexity and the number of the designers. We will, however, collect metrics when we develop new configurations.

HOW WILL ARSI INITIALLY BE USED?

ARPA will use ARSI to explore the viability of a reconfigurable simulator for:

1) training
2) research, development, and evaluation of future vehicle concepts.

Several government and National Guard sites will experiment with crew and platoon level combat training. We are currently identifying which government agencies will work with ARSI as a new development tool. Various Battle Labs and development Commands have been discussed. The baseline ARSI program will deliver the reconfigurable core (described below) and the modules for five vehicle configurations: M1A1 Abrams, M1A2 Abrams, M2A1 Bradley, M2A2 Bradley, and HMMWV scout.

We anticipate that ARSI will become a tool for software development methodologies and environments, such as those forwarded by SEI for reuse (domain modelling), Joint Modeling and

Simulation System (J-MASS), and Software Technology for Adaptable Reliable Systems (STARS). Texas Instruments will apply ARSI within its own Integrated Product Development Process (IPDP).

HOW DOES ARSI WORK?

ARSI has two basic states - exercise and configuration. In its exercise state, ARSI works as a simple "turn it on and start training" simulation. In its configuration state, ARSI works as a toolset with a set of software and hardware building blocks. The user of the configuration state will either be defining a new database or a new vehicle/subsystem.

Throughout the design phase, we have made choices of performance versus reconfigurability. We emphasize reconfigurability as long as the simulation fidelity "satisfies training requirements". At the time of writing this paper, we have scheduled tankers to assess the configurations for training. We use existing model algorithms and data tables for the parts of ARSI that must be validated and verified: weapon models, motion models, and damage assessment models.

Exercise state

We will describe how ARSI works in the exercise state by describing the modules involved. We divide ARSI modules into these categories:

- Software: reconfigurable core, generic modules, vehicle specific modules.
- Hardware: reconfigurable core, generic modules, vehicle specific modules.

The reconfigurable core modules, both software and hardware, are the simulator skeleton. The core modules are in every configuration: simulation executive, comm process, spatial database query body, enclosure, frame, and computer network. All the other simulation modules hang on this core, so the core modules must have simple, flexible interfaces.

The generic modules provide functions common to multiple vehicle configurations. These modules are generic because they are either vehicle independent or they are table/parameter driven. The generic modules include: image generator software, scene driver, DIS/SIMNET communications process, hardware controls handler, damage assessment model, ballistic gun model, sound generator process, test/calibration process, gunsight eyepiece, radio/intercom, sound equipment, cockpit displays, and seats.

The vehicle specific modules, software and hardware, are unique to the emulated vehicle. These modules are not required to be easily ported or modified. The vehicle specific modules include hardware controls (grips), computer-emulated control panels, missile models, vehicle system models, etc.

Figure 1 shows how the software modules are connected. The software modules communicate by broadcast messages. This is the most reconfigurable design, but we recognize that it is not the most efficient. The communications process on each computer handles the message passing. Each module receives and sends messages through its standard input and output channels (clean and simple interface). The DIS (or SIMNET) communications module is the gateway between the local ARSI network and the DIS/SIMNET network. It filters and translates messages going both ways. Other special communication channels can be added to ARSI, such as shared memory pools between software modules, bus-to-bus links, and reflective memory. However, special channels restrict reconfigurability and are usually more complicated than the existing communications architecture.

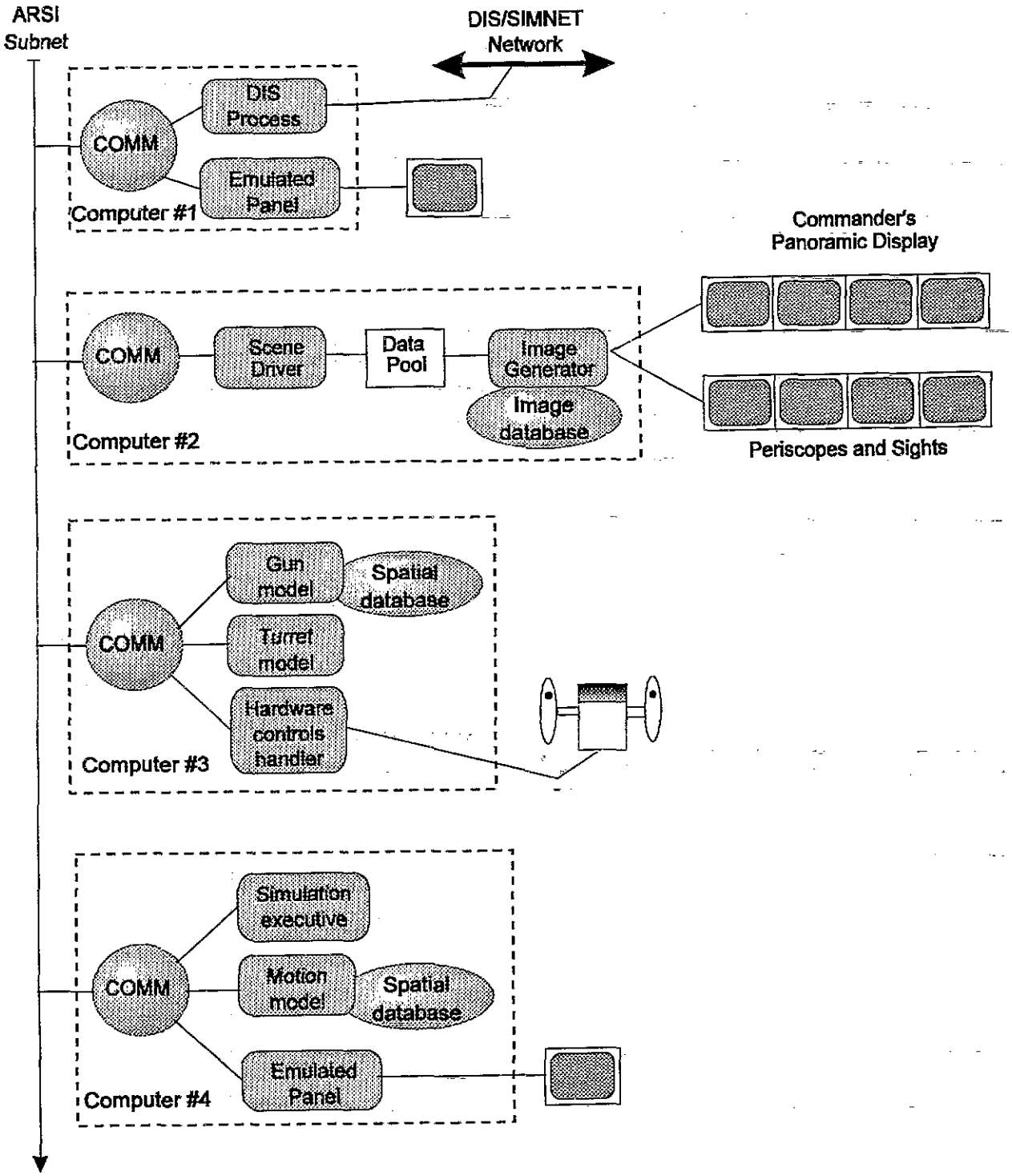

Figure 1. This shows an example of the ARSI software module architecture. The software processes communicate by broadcast message passing through the comm processes. Special communications channels can be used in the modules, such as the shared data pool on the image generator computer. The software modules can be moved between machines (given that any related hardware connections can be moved).

The simulation software modules are not synchronized. Each module is given an update rate parameter, and is responsible for running at the given rate. An unsynchronized system is much more reconfigurable and efficient because it does not use a master timing process to track all the other processes. The configuration must be carefully designed and tested before use to make sure no processes are failing their update rates. Synchronization can be added to specific modules or even the reconfigurable core, but reconfigurability is restricted.

One of the biggest restrictions to reconfigurability in existing simulators is forcing all models to use the database on the image generator. Consequently, we have separate databases for the models and the image generator. Each model has a copy of a query body which runs as a subroutine and answers queries about battlefield data. Separate databases give us the following benefits:

1) The models are free to run on computers other than the typically very expensive image generator.
2) The image generator processes have more CPU time to draw scenes instead of handling model queries.
3) The image database does not have to maintain data that is not necessary for display, eg., trafficability, etc.
4) The spatial database query body has a clean, simple interface that different applications can use. New queries and models can be developed without affecting the existing functions.
5) The spatial database query body answers spatial queries quickly with ANSI C code.

We guarantee correlation between the databases by using the same polygon and model set for producing both the image database and spatial database. This is discussed below. Using separate databases has the disadvantage of using extra computer RAM for storing multiple copies of the battlefield data. This disadvantage is not a problem in ARSI where we have multiple computers to distribute the load. This becomes a problem if a number of models are placed on one host. The query body would have to be elevated to a process that the other models could access.

The ARSI enclosure can hold the whole hardware assembly, including the computers. It is designed to operate indoors. The enclosure has its own air-conditioning. An erector set of floor, frame, and joint pieces brace the assembly. These pieces support the crewstation equipment and support equipment: seats, hardware grips, gunsight eyepieces, cockpit displays, air conditioners, sound equipment, etc. The whole enclosure is covered by canvas panels. Figure 2 shows a picture of the hardware assembly for the M2 configuration.

Figure 2. This is a photograph of the ARSI hardware assembly for the M2 configuration. The whole assembly, including computer equipment and air conditioning, fits in the enclosure. We have rolled up a few of the enclosure fabric panels so the inside is visible.

Configuration state

When ARSI is in the configuration state, it provides a path for defining a new ARSI database based on the S1000 system. We had various reasons for choosing the S1000 toolset:

1) ARSI will be able to game against SIMNET nodes in their existing databases. This was a customer requirement.
2) The spatial database query body and the image generator software use the same set of polygons and models from the S1000 database. This guarantees database consistency between ARSI modules.
3) We do not want to develop yet another database generation toolset and database format. We want to use existing applicable standards.

There are three ARSI routines that leverage the S1000 toolset. Figure 3 shows the tools and the generation path.

For defining a new vehicle configuration, ARSI provides a number of well-documented hardware and software hooks. Here are a few:

1) the erector set of floor, frame, and joint pieces. These pieces can be rearranged into a very wide variety of enclosures The simulator can be broken into separate sections, such as putting the computers in a computer room, or separating the crewstations. Modifying the enclosure typically requires mechanical design time.

2) a configuration file that defines what software modules run on the computers and the parameters for each module. Its format is easily read and edited. Any modules from the library may be included in a simulation.
3) standard communications functions that third parties can use to build new software modules.
4) a configuration file defining the out-the-window visuals. This file sets the number of channels, the fields-of-view and fields-of-regard, sensor type, and a minimum scene update rate. The baseline ARSI prototype is limited to the eight channels on a Silicon Graphics Reality Engine II with two graphics pipelines and multi-channel options. This file is also easily read and edited.
5) the VAPS commercial product from Virtual Prototypes, Inc., for modifying or creating soft controls emulations. This includes the C Code Generator option so that the panels can be executed during an exercise with more efficient code and without the VAPS runtime environment.
6) some table/parameter driven modules. One example is a ballistics model which uses firing table data to model a variety of guns and ammunition types. Another is the hardware controls handler process which has a list of parameters defining how to interpret the analog and digital signals from a crewmember's grip.

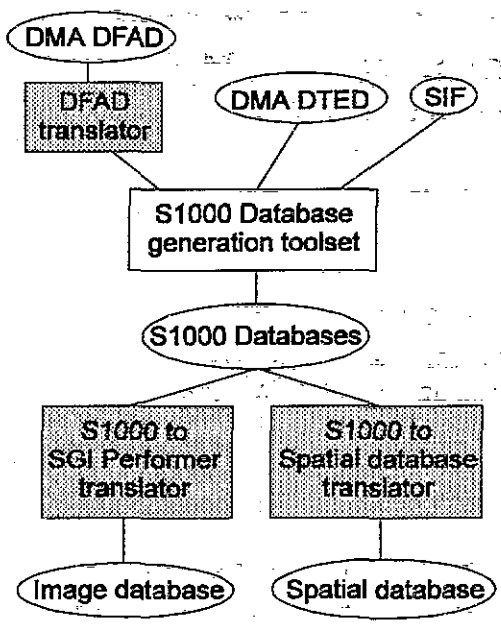

Figure 3. This shows the generation path for the ARSI database. ARSI adds a few tools (shaded) to the S1000 system. Note that the spatial database, which the models use, and the image database use the same polygon set. This guarantees database correlation between the models and visuals.

WHAT IS ARSI'S STATUS?

At the time of writing this paper, we have begun integration and test on the M2 configuration. All five required vehicle configurations will be completed December 1994. The baseline ARSI program is finished in February 1995. Several contract options have been exercised to build eight ARSI footprints (two platoons), design ARSI for a motion platform, and add a setup/logger module.

A number of other changes and additions to ARSI have been discussed. More ground vehicle configurations and aircraft configurations may be added. ARSI is an ARPA-sponsored program directed by MICOM, contract no. DAAH01-93-C-R168.