SWARM Simulation of Multi-Agent Fault Mitigation in Large-Scale, Real-Time Embedded Systems

Derek Messie
Jae C. Oh
Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244
Email: dsmessie@syr.edu, jcoh@ecs.syr.edu

KEYWORDS

SWARM, BTeV, Emergent Behavior, Subsumption Architecture, Fault Mitigation

ABSTRACT

This paper presents a SWARM multi-agent simulation of fault mitigation within BTeV, a large-scale, real-time embedded system. The Real-Time Embedded Systems (RTES) group collaborates on designing real-time embedded intelligent software to ensure data integrity and fault tolerance within BTeV, a triggering and data acquisition system for particle-accelerator-based High Energy Physics experiments at Fermi National Laboratory. The hardware layout spans 2,500 digital signal processors and approximately 2,500 Linux computers. Adaptive and small in footprint, very lightweight agents were designed to apply both reactive and proactive rules to accomplish fault tolerance within the system. The scale and real-time requirements of this system make it ineffective to design a traditional expert system that relies on centralized processing to determine appropriate rule actions for every possible system state. Instead, a decentralized approach based on Rodney Brooks' *subsumption architecture* is presented. A SWARM simulation is used to investigate emergent behavior of the multi-layered, distributed approach to fault mitigation within the RTES/BTeV environment.

INTRODUCTION

This paper describes the design and implementation of a SWARM simulation of multi-agent fault mitigation within BTeV, a large-scale, real-time embedded system. BTeV is a particle accelerator-based High Energy Physics (HEP) experiment studying matter-antimatter asymmetries in the decays of particles containing the bottom quark. The Real-Time Embedded Systems Collaboration (RTES) was formed with the purpose of designing real-time embedded intelligent software to ensure data integrity and fault-tolerance within the BTeV data acquisition system. Multiple levels of adaptive Very Lightweight Agents (VLAs) are one of the primary components responsible for fault mitigation within this environment. The VLA design was implemented and presented in a prototype of the RTES/BTeV system at the SuperComputing 2003 (SC2003) conference.

Given the number of components and countless fault scenarios involved, it would be impossible to organize various levels of VLAs using an 'expert system' that applies mitigative actions triggered from a central processing unit acting on rules capturing every possible system state. Rather, this project uses Brooks' (Brooks 1986) multi-layer, decentralized *subsumption architecture* from mobile robot design, and adapts it to achieve specific global behavior within the BTeV fault mitigation system. The SWARM simulation is used for studying the scaling qualities of emergent behavior resulting from the subsumption approach.

This paper is divided into six sections. Section 2 provides some background about the BTeV experiment and the RTES collaboration. Section 3 presents a brief overview of Brooks' *subsumption architecture*, and describes the details of the subsumption model for multi-agent fault mitigation within RTES/BTeV.

Section 4 describes the SWARM multi-agent simulation of the RTES/BTeV environment, including details on the system components and fault scenarios modeled, VLA subsumption fault mitigation rule firing, and simulation time steps.

Next steps for the project are provided in section 5, followed by a conclusion in section 6.

RTES/BTeV

Overview

BTeV is a particle accelerator-based High Energy Physics (HEP) experiment currently under development at Fermi National Accelerator Laboratory. The goal is to study charge-particle violation, mixing, and rare decays of particles known as beauty and charm hadrons, in order to learn more about matter-antimatter asymmetries that exist in the universe to-

Proceedings 18th European Simulation Multiconference
Graham Horton (c) SCS Europe, 2004
ISBN 3-936150-35-4 (book) / ISBN 3-936150-36-2 (CD)

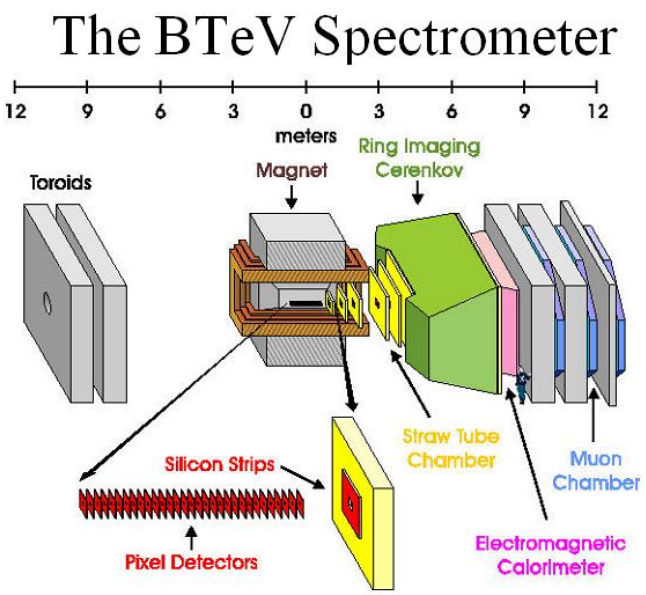

Figure 1: BTeV pixel detector layout.

day (Kwan 2002). The project is sponsored by the National Science Foundation.

The BTeV experiment will exist inside a particle accelerator where the collision of protons with antiprotons can be recorded and examined for detached secondary vertices from charm and beauty hadron decays. The layout for the BTeV detector is shown in figure 1.

The experiment uses 30 planar silicon pixel detectors to record interactions between colliding protons and antiprotons in the presence of a large magnetic field. A schematic view of the pixel detector is shown in figure 2. These detectors, along with readout sensors are embedded in the accelerator, which are connected to specialized field-programmable gate arrays (FPGAs). The FPGAs are connected to approximately 2,500 digital signal processors (DSPs).

The interactions resulting from the collision of protons and antiprotons are carried via custom circuitry hardware to localized processors that reconstruct the 3-dimensional crossing data from the 30 silicon pixel detectors in order to examine the trajectories for detached secondary vertices (Nordstrom 2003). These detached vertices are indicators of the likely presence of beauty or charm decays.

BTeV will operate at a luminosity of $2\text{x}10^{32}\text{cm}^{-2}\text{s}^{-1}$ corresponding to about 2 interactions per 7.6 MHz beam crossing rate (Kwan 2002). Average event sizes will be around 200 Kilobytes after zero-suppression of data is performed on-the-fly by front-end detector electronics. Every beam crossing will be processed, which translates into the extremely high data rate of approximately 1.5 Terabytes of data every second, from a total of $20\text{x}10^6$ data channels.

A three tier hierarchical trigger architecture will be used to handle this high rate. Data from the pixel detector and muon detector will be sent to the level 1 trigger processor, where an accept or reject decision will be made. The level 1 vertex trigger processor will perform pattern recognition, track, and vertex reconstruction on the pixel data for every interaction (Kwan 2002). It has been estimated that 99% of all minimum-bias events will be rejected by the level 1 vertex trigger, while 60-70% of the events containing beauty or charm decay will still be accepted for further evaluation.

Level 2 and 3 will be implemented on a cluster of CPU nodes, and data that makes it past the level 1 filter will be assigned to one of these level 2/3 processors for further analysis. Data that survives level 2 will be passed to level 3 algorithms to determine whether or not it should be recorded on archival media (Butler 2002). It is estimated that level 2 will decrease the data rate by a factor of 10, and level 3 will further reduce the incoming rate by a factor of 2. Once data is filtered through all three levels, and additional data compression is performed, it is expected that the resulting data rate will be approximately 200 Megabytes per second.

The events that are actually accepted within this system occur very infrequently, and the cost of operating this environment is high. The extremely large streams of data resulting from the BTeV environment must be processed real-time with highly resilient adaptive fault tolerant systems (Butler 2002). For these reasons, a Real-Time Embedded Systems Collaboration (RTES) was formed with the purpose of designing real-time embedded intelligent software to ensure data integrity and fault-tolerance within this data acquisition system.

Each of the 2500 DSPs are assigned a unique Very Lightweight Agent (VLA) responsible for a specific set of proactive and reactive fault mitigation rules. Figure 3 shows one such VLA, along with the two other components (Local Manager , Physics Application(PA)) found at every DSP.

The group of components found at each DSP are referred to as individual *Workers*. Multiple Workers are grouped into a single *Farmlet* of nodes. Similarly, proceeding up the chain, regional managers

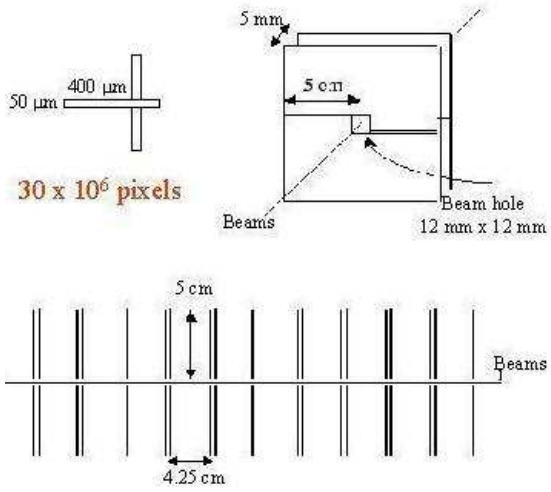

Figure 2: Schematic drawing of the pixel detector.

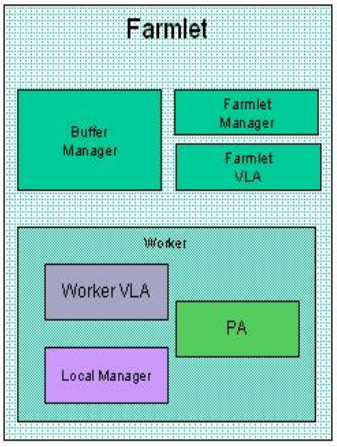

Figure 3: BTeV Level 1 Worker

are assigned groups of Farmlets. Individual Farmlet VLAs are capable of communicating with each of the associated Worker VLAs under them. Likewise, regional VLAs communicate directly with all Farmlet VLAs within their region.

Very Lightweight Agents (VLAs)

Multiple levels of very lightweight agents (VLAs) as described in (Tamhankar et al. 2003) are one of the primary components responsible for fault mitigation across the BTeV data acquisition system.

The primary objective of the VLA is to provide the BTeV environment with a lightweight, adaptive layer of fault mitigation. One of the latest phases of work at Syracuse University has involved implementing individual proactive and reactive rules for specific system failure scenarios.

A scaled prototype of the level 1 RTES/BTeV environment was presented at the SuperComputing 2003 (SC2003) conference. The prototype hardware consisted of a 7-slot VME crate with 4 fully populated motherboards and 16 DSPs. The DSPs were Texas Instruments C6711 with 64MB of RAM each, running at 166 MHz. Graphical Modeling Environment (GME) software was used to model the RTES/BTeV data aquisition system (Bapty et al. 2003). GME was developed by the Institute for Software Integrated Systems (ISIS) at Vanderbilt University as part of research investigating core technology for model-integrated computing, and applications for software integrated systems.

Reactive and proactive VLA rules were integrated within this level 1 prototype and served a primary role in demonstrating the embedded fault tolerant capabilities of the system.

Challenges

While the SC2003 prototype was effective for demonstrating the real-time fault mitigation capabilities of VLAs on limited hardware utilizing 16 DSPs, one of the major challenges is to find out how the behavior of the various levels of VLAs will scale when implemented across the 2500 DSPs projected for BTeV. In particular, how will agent rules within each VLA interact as they are activated in parallel at multiple layers of the system, and how will this affect other components and the overall behavior of a large-scale, real-time, embedded system such as BTeV.

Given the number of components and countless fault scenarios involved, it would be impossible to design an 'expert system' that applies mitigative actions triggered from a central processing unit acting on rules capturing every possible system state. Rather, one alternative is to use Rodney Brooks' multi-layer, decentralized subsumption approach for mobile robot design, and adapt it to achieve specific layers of global behavior within large-scale fault mitigation systems.

SUBSUMPTION MODEL FOR MULTI-AGENT FAULT MITIGATION

Layers of Subsumption

The phrase *subsumption architecture* was first used by Brooks (Brooks 1986) to describe a bottom-up approach for mobile robot design that relies on multiple layers of distributed sensors for determining actions. Until that time, designs relied heavily on a centralized location where most, if not all, of the decision making process took place. In fact, only initial sensor perception and motor control were left to distributed components. As a result, the success and adaptability of these systems was almost entirely dependent on the accuracy of the model and actions represented within the central processing unit (Brooks 1991).

In contrast, Brooks proposed that there should be essentially no central control. Rather, there should be independent layers each made up of a large number of sensors, with each layer responsible for distinct behavior. Communication and representation is developed in the form of action and inaction at each of the individual layers, with certain layers subsuming other layers when necessary. In this way, layer after layer is added to achieve what Brooks refers to as *increasing levels of competence*. This in turn breaks the problem down into what Brooks describes as'desired external manifestations', as opposed to slicing the problem on the basis of 'internal workings' of the solution as was typically done in the past.

Figure 4 shows the *levels of competence* that Brooks defined for his mobile robots. In order to implement the subsumption design across a multi-agent fault mitigation system, the first step is to clearly define the distinct layers of subsumption that may be required to achieve the effective fault mitigation behavior desired. Just as Brooks' provided, a description that distinguishes each layer needs to be outlined, along with detail on the specific roles and responsibilities of each.

Figure 5 defines these layers for the system fault mitigation implementation presented in this paper.

0. *Avoid contact with objects (moving or stationary).*
1. *Wander aimlessly around without hitting things.*
2. *"Explore" the world by seeing places in the distance which look reachable and heading for them.*
3. *Build a map of the environment and plan routes from one place to another.*
4. *Notice changes in the 'static' environment.*
5. *Reason about the world in terms of identifiable objects and perform tasks related to certain objects.*
6. *Formulate and execute plans which involve changing the state of the world in some desirable way.*
7. *Reason about the behavior of objects in the world and modify plans accordingly.*

Figure 4: Layers of competence for mobile robots as defined by Brooks.

0. *Act on local reactive rules.*
1. *Monitor local components and activate local proactive rules.*
2. *Analyze and act on statistics gathered from rule firing.*
3. *Map the local state of the environment.*
4. *Reason about the system in terms of accumulated statistics and the state of connected components.*
5. *Formulate and execute actions which involve changing the state of the system in some desirable way.*
6. *Reason about the behavior of components in the system and modify plans accordingly.*

Figure 5: Layers of competence for fault mitigation within BTeV.

The first layer (layer 0) is responsible for simple reactive rule activation within each component of the system. Just as layer 0 for mobile robots is responsible for the rudimentary task of avoiding objects, layer 0 here takes the most basic actions as defined in the local reactive rule base of each corresponding component. At this layer, incoming messages from connected components are typically reacted to by either sending simple remediary actions to problem components, or by forwarding error messages to higher level components within the system. The roles and responsibilities for each of the higher layers is also provided.

Implementation

In (Brooks 1986), layers 0 and 1 were implemented, and work had just begun on designing rules that would begin to address attributes of the levels of competence associated with layer 2. Similarly, the fault mitigation simulation described in this paper has implemented layers 0 and 1, along with a limited number of rules addressing layer 2.

The first step in this approach was to define individual VLA rules within the system that will contribute to meeting the responsibilities assigned to the lower layers of the subsumption architecture. Basic fault mitigation rules that address levels 0 and 1 define both reactive (level 0) and proactive (level 1) rules for the system. Ten of the error scenarios that are simulated in this implementation are listed in figure 6, and figure 7 lists the specific fault mitigation rules activated in each case.

Actions triggered from rules deemed to contribute to higher levels of competence are given priority at each individual component, and in this way subsume lower level rules when conflicts arise. At each layer, rules are experimented with and adjusted to compliment emerging behavior that contributes to meeting the fault mitigative responsibilities of each layer.

The general procedure for adding higher levels of competence is described by Brooks (Brooks 1986). Layers of the control system that correspond to specific levels of competence are built, and new layers are added to the existing set in order to move to the next higher level of overall competence. A complete control system that accomplishes level 0 is first implemented. It is debugged thoroughly, and is not altered from that point on. Another separate layer of control is then built on top of the zeroth level, and is called control level 1. It is capable of accessing and examining data from the level 0 interface, essentially suppressing the normal data flow at level 0. This layer, along with the behavior of level 0, achieves level 1 competence. However, level 0 continues to run unaware of the layer above that may or may not have interfered with its data paths. This process is repeated to achieve higher and higher levels of competence.

SWARM

Overview

The primary focus of this simulation is on effective ways to implement the multi-agent subsumption approach within a complex system such as BTeV so that qualities of emergent fault mitigation behavior can be evaluated. A high volume of rules at various levels will be used in order to evaluate the interaction and behavior of the VLAs and other components across 2500 DSPs. This requires a simulation environment that will allow abstract representation of some of the complex integration within BTeV. However, for the results to be of any use at all, this abstraction must be done in a way that still accurately models the actual behavior of the components involved. The level of success of the results are directly dependent on this accuracy.

SWARM (http://www.swarm.org), distributed under the GNU General Public License, is software available as a Java or Objective-C development kit that allows for the multi-agent simulation of complex systems. It consists of a set of libraries that facilitate implementation of agent-based models.

The basic architecture of SWARM provides for the

ID	Description	Possible Causes
e1	DSP over time budget on crossing processing.	Crossing was too complex to complete and developer was not careful to give up in time.
e2	PA is stuck in a loop (within software timer control)	Improper error handling caused the program to get stuck in an infinite loop.
e3	DSP application framework is stuck in a loop (outside of software timer control)	Logic error in code that manipulates the board's communications facilities.
e4	DSP application branches to an illegal instruction.	Logic error any place in the code that causes corruption of memory.
e5	Processing times per crossing are too long.	SAF reported crossing processing times are consistently falling out of range.
e6	Too many track segments. Not necessarily a fault at the source.	The front-end hardware is malfunctioning, more particles collided than can be managed, bug in the upstream algorithms.
e7	Corrupt data in a crossing (truncated, misaligned, or bad header)	Bad checksum or incorrect header data in a crossing due to transmission failure or upstream logic error.
e8	Corrupt data - no such channels in the detector.	Logic error in the front-end electronics or firmware. (byte swapping)
e9	Crossing data lost.	DSP was reset or reboot while an event was being processed, FPGA input queue overflow, FPGA output queue overflow.
e10	Failed to transfer results down the DSP L1 buffer link. (buffer ready flag not set in time)	The level-1 buffers were not ready to receive data, the farmlet output queues overflowed.

Figure 6: 10 BTeV Error Scenarios

ID	How and where detected	Action
e1	Timer interrupt on DSP caught by VLA.	VLA resets PA. VLA sends m08 to FM VLA. VLA sends m29 if rate of these is too high.
e2	Many m08 messages over a time period containing PC within a region of code. Detected by Farmlet Manager VLA.	Send m26 to RM VLA. Notify OP.
e3	Many m06, m07 messages over a time period containing PC within a region of code. (FM,RM)	FM sends m26, RM notifies OP.
e4	m06, m07 message contains PC that is not in program space. (FM)	FM sends m26, RM notifies OP.
e5	Ensemble of m15 messages does not match known distribution.	DSP or FM send m14, RM notify OP.
e6	PA discovers this. (DSP)	PA sends m18, DSP/FM records and sends m18 if rate too high. RM notifies OP.
e7	FPGA cannot verify checksums or lengths, SAF cannot unpack header or sees bad status from FPGA (DSP)	DSP sends m16, FM records and sends m16 if rate too high, RM notifies OP.
e8	PA discovers this while unpacking the crossing data. (DSP)	DSP sends m17, FM records and sends m17 if rate too high, RM notifies OP.
e9	Discovered during the reset/reboot sequence using NVRAM history, crossing with only a header appears. (DSP)	DSP sends m11, FM records and sends m11 if rate too high, RM notifies OP.
e10	Timeout during transfer of algorithm results in SAF. (DSP)	DSP sends m10 and m11, FM records and sends m10 if rate too high, RM notifies OP.

Figure 7: Fault Mitigation Rules Activated for 10 Error Scenarios

simulation of collections of concurrently interacting agents (Daniels 2000). It provides an environment that can model various components of the BTeV system, assigning dynamic states to each agent, which can then be altered in time steps following various user-specified rules (Burkhart 1997). Both proactive and reactive rules are triggered after the current state of a given agent(or component) is evaluated against the state of other connected agents(or components).

SWARM Simulation of RTES/BTeV

Since the VLA is the primary focus for evaluating and acting on a number of sample fault mitigation scenarios, VLAs across the system are simulated. Another main component of the simulation is the Physics Application (PA). Since it is central to many of the defined fault scenarios, it is vital that the state and behavior of the PA is simulated accurately. The PA is located at the Worker level, where the local manager and Worker VLA (WVLA) also reside. All three of these components are modeled. In addition, Farmlet managers and Farmlet VLAs (FVLAs), as well as corresponding regional managers and regional VLAs (RVLAs) are also simulated.

A sample screen shot of the RTES/BTeV SWARM Simulation is shown in Figure 8. This particular configuration has defined 6 Workers per Farmlet, 10 Farmlets per region, and groups consisting of 7 regions. This provides a simulation of 2,520 DSPs, just over the actual target number projected for RTES/BTeV. Each Farmlet column is made of 6 boxes, with each box representing 1 Worker within the Farmlet. The Farmlet is represented by the yellow box located slightly above the column of 6 Workers, and simulates the actions taken by the Farmlet manager and Farmlet VLA. The centered green box at the top of the region simulates the regional manager and regional VLA.

Random fault scenarios activate specific system error messages at individual Worker, Farmlet, and regional nodes. The frequency with which each particular fault occurs is user-defined. This enables the system to simulate multiple fault scenarios in parallel, allowing for the demonstration of how simultaneous errors occurring at various levels within the system are processed. Errors introduced across the system are detected both proactively and reactively by individual components within the model, and are indicated within the simulation when the color of a given box turns red. A log of the sequence of errors occuring at each component is tracked, and can be viewed by clicking on the box representing the particular node of interest. In addition to a real-time status screen available for each Worker, Farmlet, region, and group node, a full error log tracks all error messages occurring at each time step across the system.

Individual proactive and reactive VLA rules are responsible for fault mitigation within the simulation at

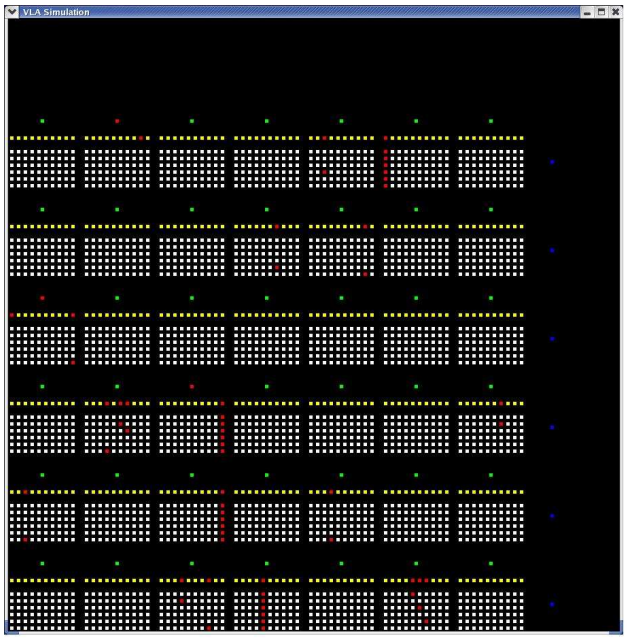

Figure 8: RTES/BTeV SWARM Simulation

each node. Every Worker VLA has a unique set of rules that it follows at each time step. Each rule is assigned a layer within the subsumption model with which it is associated. Rules assigned to higher layers that are responsible for higher *levels of competence* as defined earlier, subsume rules associated with lower layers. Likewise, every Farmlet, region, and group VLA has a unique set of rules that it follows, with higher layer rules subsuming lower ones.

As mentioned earlier, since scalability issues are a primary concern, the results must detect the degree to which any exhibited behavior is tied to specific system configurations. The SWARM model includes dynamic variables that can be modified to reflect various hardware layout configurations, such as the number of Workers per Farmlet, the number of Farmlets per region, and the number of regions per group.

Determining how to appropriately simulate time steps within any multi-agent simulation typically proves challenging. It is a particularly difficult issue in this case since the behavior of a large-scale, real-time, embedded system must be modeled. Since the primary goal is to simulate the interaction of component rules at various levels within the system, it suffices to demonstrate this interaction without explicitly capturing many of the exact processing rates and timing issues involved. Again, it is the general interaction of the rules at the various layers of subsumption that the implementation is primarily interested in simulating.

NEXT STEPS

After implementing all of the basic reactive and proactive rules of layers 0 and 1, the next step is to continue to evaluate and act on the rule firing statistics gathered at various levels of the system as defined for layer 2. From there, higher layers of the subsumption model will be implemented in order to continue to classify characteristics of observed emergent behavior. It is expected that each layer that is implemented will provide further insight into effective strategies and techniques that can be used for adjusting individual agents towards desired behavior.

CONCLUSION

A SWARM multi-agent simulation is used to model BTeV, a large-scale, real-time, embedded system. The subsumption model detailed provides a framework for implementing specific levels of competence for multi-agent systems in incremental steps. The SWARM simulation of the RTES/BTeV environment provides an effective way for experimenting with local agent rules throughout the system to evaluate the impact each have on these individual layers. In this way, more may be understood about the specific qualities and structure of local rules and actions at these various layers that lead to global fault mitigative emergent behavior.

REFERENCES

Brooks, R.A., 'Intelligence Without Representation', *Artificial Intelligence*, 47, 1991, 139-160.

Brooks, R.A., 'A Robust Layered Control System for a Mobile Robot', *IEEE Journal of Robotics and Automation*, RA-2, (April 1986).

Bapty, T. et. al, 'Modeling and Generation Tools for Large-Scale, Real-Time Embedded Systems'. IEEE Conference on the Engineering of Computer Based Systems (ECBS), Huntsville, Alabama, (April, 2003).

Burkhart, R., 'Schedules of Activity in the SWARM Simulation System'. Position Paper for OOPSLA Workshop on OO Behavioral Semantics. (1997).

Butler, J.N., et. al, 'Fault Tolerant Issues in the BTeV Trigger'. FERMILAB-Conf-01/427, (December 2002).

Daniels, M., 'An Open Framework for Agent-based Modeling'. Applications of Multi-Agent Systems in Defense Analysis, a workshop held at Los Alamos Labs. (April 2000).

Kwan, S., 'The BTeV Pixel Detector and Trigger System'. FERMILAB-Conf-02/313, (December 2002).

Nordstrom, S., *A Runtime Environment to Support Fault Mitigative Large-Scale Real-Time Embedded Systems Research*, Master's thesis, Graduate School of Vanderbilt University, May 2003.

Tamhankar, S., Oh, J., Mosse, D., 'Design of Very Lightweight Agents for Reactive Embedded Systems'. IEEE Conference on the Engineering of Computer Based Systems (ECBS), Huntsville, Alabama, (April 2003).