E-16-X83
3

AGENT-BASED SIMULATIONS USING HUMAN PERFORMANCE MODELS FOR NATIONAL AIRSPACE SYSTEM RISK ASSESSMENT

Annual Performance Report For the Period

submitted to

NASA Ames Research Center
Moffet Field, CA 94035-1000

for the period

15 April 2004 to 31 March 2006

Amy R. Pritchett, Sci.D., Principal Investigator
Schools of Aerospace Engineering and Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0150
(Tel) 404-894-0199
(Fax) 404-894-2301
Amy.Pritchett@isye.gatech.edu

David Goldsman Ph.D. and Christos Alexopoulos Ph.D., Co-Principal Investigators
School of Industrial and Systems Engineering
Georgia Institute of Technology

Richard Fujimoto, Ph.D., Co-Principal Investigator
College of Computing
Georgia Institute of Technology

Margaret Loper, Ph.D., Co-Principal Investigator
Georgia Tech Research Institute

CA-NNA04CI30A

Technical Monitor: Dr. Irv. Statler

June 2006

ABSTRACT

While it is fortunate that the National Airspace System (NAS) has achieved levels of safety unparalleled in almost any other domain, commensurate levels of risk are so low as to be very difficult to assess. This difficulty can be represented from two different-but-corresponding viewpoints: statistical concerns with estimating very rare events, and modeling concerns with estimating very rare events. The statistical concerns stem from the sheer number of data points required to perform a truly random Monte Carlo-type simulation (order of 10^{10}), and the novel analyses that would be required to form estimates not of 'mean' values, but instead of occurrences in the 'tails' of probability distributions of *a priori* unknown form. The modeling concerns stem from the difficulty in capturing all the behaviors of every agent in the system and all types of interactions between them such that every possible 'real' behavior is captured.

This proposal instead seeks to coordinate the statistical and modeling concerns using an intermediate model fidelity: agent-based simulation of relevant aspects of the NAS. Agent-based simulation of the National Airspace System (NAS) provides a detailed prediction of system behavior. Both micro-level (agent) and macro-level (system-wide) behaviors are simulated simultaneously, highlighting system-wide issues arising from a change in NAS configuration, air traffic procedures, or air or ground technologies – as well as identifying unreasonable demands that system dynamics may place on individual agents such as controllers or pilots.

Prior developments established such detailed simulations using human performance models as the pilot and controller agents. Corresponding developments in simulation architectures also demonstrated the ability to accurately model agent interactions, to enable larger scale simulations via distributing the simulation, and to facilitate tighter linkages with the statistical issues with data collection and with establishing sufficient confidence in output parameters.

This report describes the following research tasks further developing agent-based simulation as a method for NAS risk assessment:

- Closer integration (working with San Jose State University) of human performance models with agent-based simulations, so that they can work seamlessly together.
- Implementation of a large-scale agent-based simulation of an aspect of the NAS suitable for assessing safety issues at sufficient level of detail to identify the likely impact of hazards resulting from specific conditions, and to highlight their source and potential solutions.
- A two-stage analysis process, one using very detailed agent-models for a lower number of simulation runs, and the second using less detailed agent models for a high number of data runs suitable for statistical analysis.
- Supporting research into data analysis techniques and large-scale simulation techniques.

A specific test case (analyzing aircraft arrivals into LAX using a variety of spacing techniques) was examined throughout as a test case. *It should be noted, however, that the over-arching purpose of this work was to provide the research and development base for NAS risk assessment, not to provide a single risk assessment of this one test case.*

INTRODUCTION AND PROBLEM STATEMENT

While it is fortunate that the National Airspace System (NAS) has achieved levels of safety unparalleled in almost any other domain, its commensurate low levels of risk are very difficult to assess. This can be represented from two different-but-corresponding viewpoints: statistical concerns with estimating very rare events, and modeling concerns with creating the vast number of behaviors that could create them. The statistical concerns stem from the sheer number of data points required to perform a truly random Monte Carlo-type simulation (order of 10^{10}), and the novel analyses required to form estimates not of 'mean' values, but instead of occurrences in the 'tails' of probability distributions unknown *a priori*. The modeling concerns stem from the difficulty in capturing all the behaviors of every agent in the system and all types of interactions between them such that every possible 'real' behavior is captured in a simulation.

Historically, approaches to NASA risk assessment have focused either on the statistical concerns or on the modeling concerns. When statistical methods were emphasized, very aggregate, low fidelity models were used, such as fault trees; while the statistical processes deriving the result could be verified, the accuracy of the underlying model was often based on expert opinion and approximations, and the source of and solution to a safety issue could not be robustly identified. When high fidelity models were emphasized, very few number of data points could be collected. For example, human-in-the-loop simulations of entire air traffic centers have been enacted, but they could not provide data about more than a few hours of operations.

This project instead sought to unify the statistical and modeling concerns using an intermediate level of model fidelity: agent-based simulation of relevant aspects of the NAS. Earlier developments established such detailed simulations using human performance models as the pilot and controller agents. Corresponding developments in simulation architectures accurately model;ed agent interactions, enabled larger scale simulations via distributing the simulation, and facilitated tighter linkages with the statistical issues with data collection and with establishing sufficient confidence in output parameters.

This report describes a series of research tasks conducted, in concert with San Jose State University, towards developing agent-based simulation as a method for NAS risk assessment. A specific test case (analyzing aircraft arrivals into LAX using a variety of spacing techniques) was examined throughout as a demonstration. The report concludes with the work's anticipated impact and directions for further research.

BACKGROUND

Agent-Based Simulation of the NAS

Analysis of large-scale complex systems, such as the National Airspace System (NAS), requires the use of hybrid agent-based simulation. Simulation is an important tool for evaluating the performance of systems, and provides a safe and cost-effective way of examining the impact of potential changes. Typically, large-scale complex systems include different types of entities whose interactions are as important to overall system performance as the behavior of the entities taken in isolation. For instance, the NAS can be defined as a collection of mutually dependent entities, such as aircraft, controllers, pilots, ground systems, and communication, navigation and surveillance technologies; many of these entities act as intelligent agents responding to their environment. The overall behavior of the NAS – and its safety – is thereby created as an emergent behavior of these agents in the context of their environment.

The individual behavior of these different entities and overall behavior of the NAS, therefore, can be modeled by a combination of agent models, environment models, and their interactions. This is a departure from traditional methods of simulating the NAS, which have largely been based on discrete-event models and have often been limited to specific applications, types of analyses or parts of the NAS. [1,2]

Several difficulties needed to be overcome in implementing an agent-based simulation of the NAS. The most-studied element focuses on modeling the various entities within the NAS; for example, detailed and validated models have been created of human performance in work environments and of vehicle dynamics.[3,4]

Software Architecture

Prior research at Georgia Tech and within the Aviation Safety Program has complimented the development of detailed agent models by developing a simulation software architecture with the following capabilities:

1. To accept different types of models with varying fidelity without placing unnecessary restrictions on the types of models;
2. To allow for easy reconfiguration of the simulation through the addition of new agent models to define scenarios through plain-text script files;
3. To control the timing of the agents in a computationally efficient manner when running within the architecture in a single processor; and

4. To handle interactions between entities within the system.

Capabilities 1 and 4 were created by the use of an object-oriented analysis and design approach in adapting the original Reconfigurable Flight Simulator (RFS) architecture to agent-based simulation of the NAS.[5,6] This architecture establishes base standard interfaces for agents (by which they can access the shared simulation environment, including timing controls, data-passing mechanisms, and environment models and databases), and a mechanism for extending their interface in the simulation (termed the Object Data/Methods Extension [OD/ME]). Agents meeting these interfaces and the OD/ME protocol can be entered into the simulation environment and configured by calling plain-text scripts. Capabilities 3 and 4 were resolved through development of novel within-processor timing schemes, in which agents are free to update independently of each other at their own rate (preventing unnecessary calls to agents) until times where they need to be synchronized with other agent(s) for an interaction or for a measurement of system behavior.

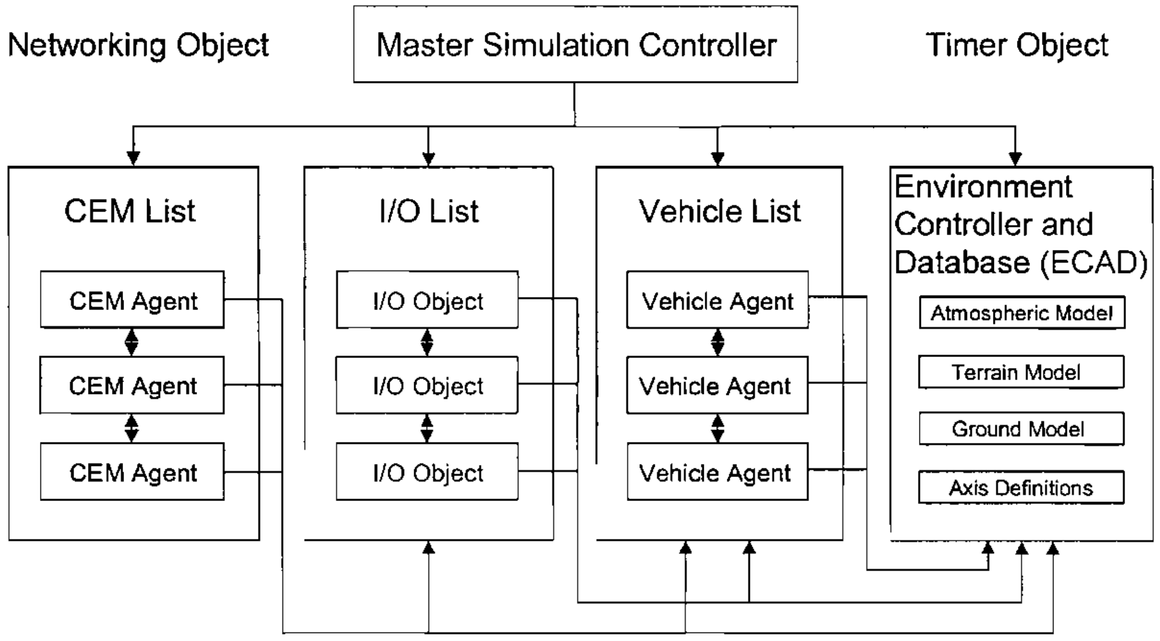

Figure 1. RFS Architecture Control Structure, With Linkages Between CEM and Vehicle Agents, I/O Objects, and the ECAD; Networking and Timing Objects Have Access to All

Simulating Interacting Agents Within and Across Networks

To further extend the utility of agent-based simulation of the NAS, preliminary efforts created networked simulations with heterogeneous agent models (i.e., including agents that are not executed with the RFS architecture, but instead control their own timing and processes, and interact with the larger simulation through data-passing across the network). Networking interfaces have been implemented using the High-Level Architecture (HLA), a Department of Defense standard for distributed simulation, which provided the Run-Time Interfaces (RTIs) for networking protocols. In one specific implementation, working with San Jose State University and ATAC Corp., the RFS was networked to the human performance model MIDAS, which provided detailed models of the behavior of an air traffic controller and pilots engaged in activities of interest; the RFS provided models of the 'procedure-following humans' (i.e., up-stream and down-stream controllers, and pilots not engaged in activities of interest), as well as relevant communication and surveillance technologies, aircraft dynamic models, and a shared simulation environment providing an atmospheric turbulence model and a navigation database.[7]

These efforts to date may be considered a success for their demonstration of linking heterogeneous simulations (including detailed human performance models) together into an agent-based simulation of a fundamental component of the NAS. However, this endeavor also highlighted the need for further reliance on networked, distributed simulations (sometimes called federated simulation systems). This simulation configuration is an increasingly popular approach to realizing scalable agent-based simulations of large, complex systems. These simulation systems are constructed by interconnecting autonomous agent-models executing on different computers that are connected via a network. Scalability is achieved in the sense that one can increase the number of agents and the number of computers to model larger and/or more complex systems, such as modeling larger segments of the NAS or including more agents for higher fidelity simulations. This technology enables one to leverage prior investments in simulation software by reusing rather than re-inventing simulators for elements of a system. It is particularly well-suited for modeling the NAS because simulators already exist that model key elements, such as the aforementioned MIDAS and RFS. Distributed simulation has been successfully employed in domains such as Defense (e.g., interconnecting flight and vehicle simulators to create virtual battlefields for training), as exemplified by recent efforts such as the HLA.[8] Some work in applying this concept to aviation simulations has also been completed.[9]

Using Large-Scale Simulations for Risk Assessment

General Issues with Risk Assessment of the National Airspace System

While it is fortunate that the National Airspace System has achieved levels of safety unparalleled in almost any other domain (with the notable exception of nuclear power), commensurate levels of risk are so low as to be very difficult to assess. This difficulty can be represented from two different-but-corresponding viewpoints: statistical concerns with estimating very rare events, and modeling concerns with estimating very rare events. The statistical concerns stem from the sheer number of data points that would be required to perform a truly random Monte Carlo-type simulation (order of 10^{10}), and the novel analyses that would be required to form estimates not of 'mean' values, but instead of occurrences in the 'tails' of probability distributions of *a priori* unknown form. The modeling concerns stem from the difficulty in capturing all the behaviors of every agent in the system and all types of interactions between them such that every possible 'real' behavior is captured in a simulation.

While sufficient number of runs of a purely-modeled, perfectly random simulation can not be achieved for classic Monte Carlo-type methods of risk assessment, a carefully coordinated and staged process can provide more insight into the sources and frequencies of the most important hazards. Specifically:

- First, a 'hazard analysis' can be performed. When using simulation, this can be based on a low number of simulation runs using comparatively high-fidelity models; other processes for risk assessment may instead use expert opinions and various model forms such as fault tree analysis, etc. The result of this first process is identification of the most relevant (i.e., dangerous and/or likely) hazards which warrant further investigation
- Second, detailed analysis of the most relevant hazards using models focused on the behaviors focused on their cause and resulting impact on safety. At this stage, greater statistical accuracy (i.e., high confidence in an estimate of their resulting frequency of 'unsafe event') can be sought through a higher number of simulation runs.

Practical Issues with Risk Assessment, and Theory-Based Solutions

To date, agent-based simulation as a method has focused on the simulation more than the analysis; however, as such simulations are enabled, we predict that the data analysis process will become the bottleneck. To date, a standard approach to data analysis has included *a priori* design of experiments, many simulation runs then conducted to replicate events of interest, and post hoc data analysis as a separate, subsequent activity to the simulation. In examining this process, two effects are noteworthy:

- First, for NAS risk assessment, efficient, appropriate methods for design of experiments and data analysis are not always known ahead of time. For example, safety analysis typically entails rare-event analysis – the use of common techniques such as Monte Carlo simulations often require a prohibitive number of data runs to create sufficient rare events for analysis, suggesting the need for more structured sampling techniques, including mechanisms for closed-loop control of the simulator configuration to steer towards data-providing runs.
- Second, when appropriate data analysis techniques are known, they often follow a standard form that can be built into the simulation, mitigating the need for post hoc data analysis. However, current techniques need to be adapted to be suitable for built-in data analysis, and must be used with an appropriate understanding of their underlying theoretical basis.

As agent-based simulation becomes commonplace, methods will be needed for rigorously planning the test runs and then correctly making statistical inferences from the vast amount of data that they can record. These methods need to be able to analyze for rare events with as few simulation runs as possible. Preliminary efforts to this end have been made on earlier grants, resulting in the architectures shown schematically in Figure 2.

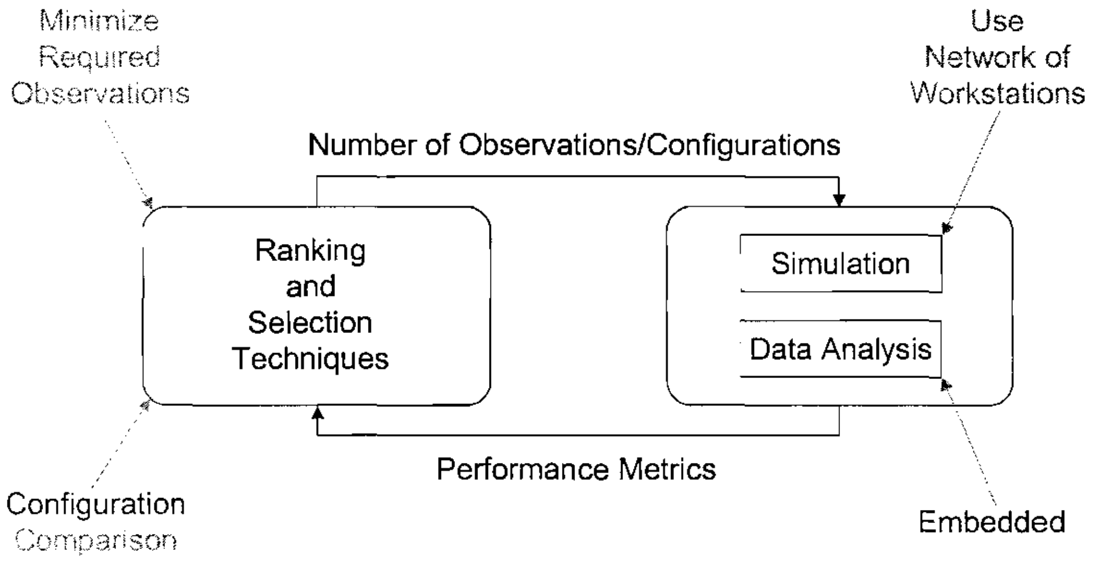
Minimize Required Observations
Use Network of Workstations
Number of Observations/Configurations
Ranking and Selection Techniques
Simulation
Data Analysis
Performance Metrics
Configuration Comparison
Embedded

Figure 2. Schematic Representation of Dynamic Linkage Between Ranking and Selection Techniques and Simulation with Embedded Data Analysis

OBJECTIVES

1. Continue to develop agent-based simulation as an effective model of NAS behavior. Specific concerns include direct use of human performance models as agents, so that they can be used seamlessly in large-scale simulations of the NAS.
2. Establish a two-stage process which (1) uses high-fidelity models to identify the most relevant (i.e., hazardous and/or likely) hazards, and then (2) streamlines the important behaviors into agent models suitable for a high number of simulation runs sufficient for statistical confidence in output parameters about a risk's frequency and impact when triggered by hazardous conditions.
3. Establish embedded data analysis and statistical control methods suitable for automatically collecting data from the simulations, and for controlling the number and type of simulation runs to achieve the desired confidence levels in the output parameters.
4. Provide the supporting research in statistical techniques and large-scale simulation development to enable not only this effort, but broader use of agent-based simulation for risk assessment.

RESEARCH TASKS

Development of agent-based simulation architecture and models suitable for large-scale simulation suitable for NAS risk assessment.

Established methods of agent-based simulation (not only within our project, but also within the larger community) have focused more on the agents than on their environment. Following an ecological viewpoint of human behavior, we believe that many aspects of human performance are driven by the environment and should be represented as such. As such, we developed specifications of the constraints and affordances implicit within the environment that can establish both a modular software mechanism for codifying environment structures relevant to many types of agents and a more-accurate conceptual representation of agent behavior, detailed in this section.

Specifically, the RFS conceptual framework builds on the principles of cognitive engineering to describe the components of the work environment, i.e., technology, processes and information, and the humans and automation, in task-relevant ways and using a structure-preserving form using the same attributes and structure as used by system designers and human operators. The framework includes both declarative models of system components and their interrelations, and computational models of those complex, dynamic behaviors that cannot be adequately described declaratively.

With this framework, the work environment can be viewed as including all components, physical or not, that define, afford or constrain the actions of the humans and automation. Each component is defined by three attributes: a set of properties whose values can be accessed $<P_c>$, a set of usage mechanisms which can be accessed to use the component to some end $<UM_c>$, and internal dynamics ID which can internally change its properties. The properties and usage mechanisms can be fully described in the declarative structures, while the internal dynamics are captured in software structures suitable for their replication computationally (e.g., a computational representation of ordinary differential equations representing aircraft flight dynamics) and the appropriate software structure is referenced in the declarative model.

Unique to this framework, air traffic operating procedures and other 'work processes' can be included as components. By defining them as structured orderings of activities, they have properties that expose the enclosed procedure and a set of usage mechanisms that specify how this component may be used; they do not have any internal dynamic that changes the values of its internal properties and thus this one attribute is set to 'NULL' for work process components.

Further, the work environment is viewed as a multidimensional space where each component is mapped onto one or more dimensions, as illustrated in Figure 3. Examining the types of human and automation functions germane to this project, (1) functions are directed towards achieving some goals, and (2) these functions are 'situated', i.e., their execution is contingent to the current state of the work environment. Thus there are two primary relationships that should be modeled when modeling work environment of the NAS: (1) the functional dimension that relates the components to the goals such that the human and automation agents can observe the environment sufficiently well to search for and prioritize environment components affording their goals (and that enables formal methods to assemble models of the tasks), and (2) the contextual dimension that defines the workspace of the human and automation in terms of which components are accessible to them by virtue of their placement in the work environment. Unlike current cognitive engineering models, there isn't any conceptual limitation on the number of dimensions that can be added into this model of the work environment, and there is no limitation on the nature of relationships in any dimension; thus, this model form is easily extensible whenever the application requires. As summarized in Table 1, components may be summarized also as containing a set of component aspects, with each component aspect representing the components' mapping of properties and usage mechanisms onto a dimension of the environment. Thus, the environment model is built 'bottom up' from the assemblage of components put into it, and their organization as defined not only by spatial properties but also by function allocation between human and automated agents.

Table 1: Summary of the RFS work environment model

Model	Description
$WE = <<C_e>, <KD_e>>$	Work environment: The work environment is modeled as a collection of all the work-relevant components in the system and all the dimensions of knowledge of relationships between these components that the workers need to accomplish their work.
$KD_e = <R_d>$	Knowledge Dimension: A knowledge dimension is a set of work-relevant relationships that associate components in the work environment. A knowledge dimension includes all instances of these relationships and thus establishes one kind of structure or network in the work environment.
$R_d = <[C_e], <P_r>>$	Work-relevant Relationship: A relationship associates components in the work environment in work-relevant ways. The knowledge-processing engine in the worker model is capable of parsing relationships to derive the model of a task.
$C_e = <ID_c, <P_c>, <UM_c>>$ $C_e = <ID_c, <CA_c>>$	Environmental Component: Physical objects, processes, technologies, procedures and regulations, information and organizational structures that affect worker activities. This model contains their internal dynamics, work-relevant properties and usage mechanisms, which may be grouped by knowledge dimension into a set of component aspects.
$CA_c = <[P_c], [UM_c]>$	Component Aspect: The component aspect is the view of the component from one knowledge dimension.

Let us consider Figure 3 to illustrate this framework's utility to analysis of function allocation. To identify the impact of particular function allocations between the human air traffic controller of sector ZLA-39 and particular automation, each allocation can be represented by changing the contextual dimension of the controller and of the automation. In Figure 3, for example, the controller's contextual dimension includes the functions of lateral and vertical separation via a specified procedure; a different function allocation may put those functions in the contextual dimension of an automated system, and replace the controller's radar screen with a decision aid. These allocations can be easily and quickly implemented in XML, either by hand or by function allocation algorithms. As will be noted later, the software architecture can respond in run-time to changes in these models to incorporate dynamic changes to the function allocation.

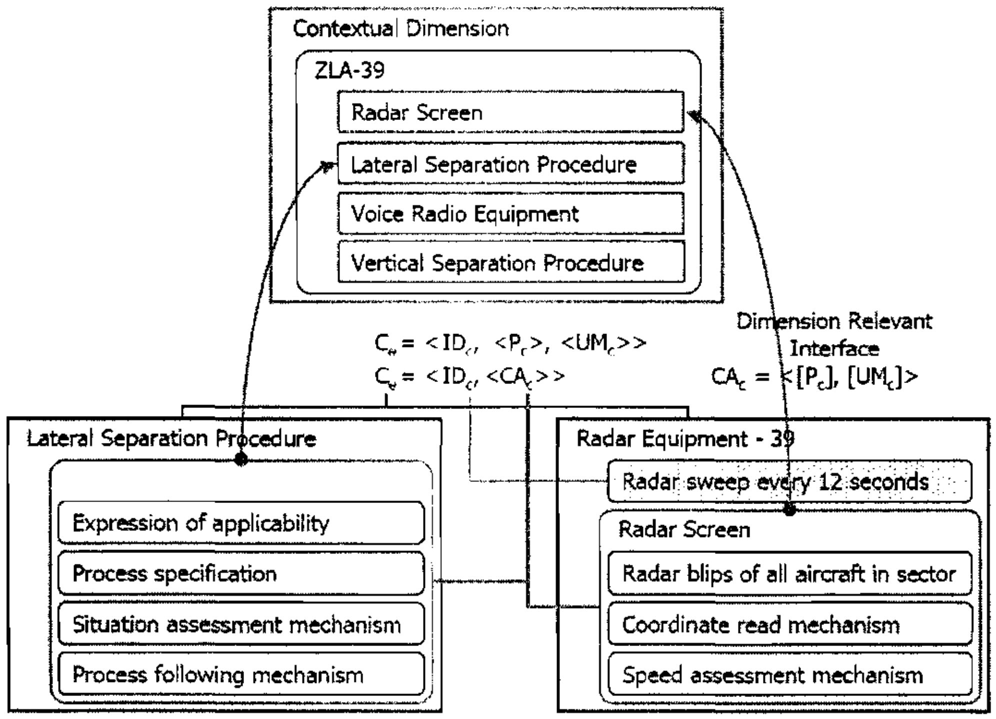
Contextual Dimension
ZLA-39
Radar Screen
Lateral Separation Procedure
Voice Radio Equipment
Vertical Separation Procedure
Dimension Relevant Interface
$C_{\theta} = <ID_c, <P_c>, <UM_c>>$
$C_{e} = <ID_c, <CA_c>>$
$CA_c = <[P_c], [UM_c]>$
Lateral Separation Procedure
Expression of applicability
Process specification
Situation assessment mechanism
Process following mechanism
Radar Equipment - 39
Radar sweep every 12 seconds
Radar Screen
Radar blips of all aircraft in sector
Coordinate read mechanism
Speed assessment mechanism

Figure 3. Describing Two Environment Components and Their Mapping onto the Contextual Dimension

The models of human and automated agents can be generated from the environment model, with the inclusion of representations of human performance where desired. The structure of the environment model, i.e., its dimensions and its components, impose certain requirements on the agent models: the specification of skills, with associated internal capabilities and processors, that the agent must have to be able to successfully interact within their given environment (which includes their function allocation). For example, to execute a procedure component, the agent must have a corresponding skill, and to use the functional dimension the agent must have the skill to parse the dimension and find the components that are affordances to their goals. These agent skills can be provided or represented by the AirMIDAS models used here in collaboration with San Jose State University, or, when simpler representations of human behavior are best suited to the analysis, streamlined models may be used.

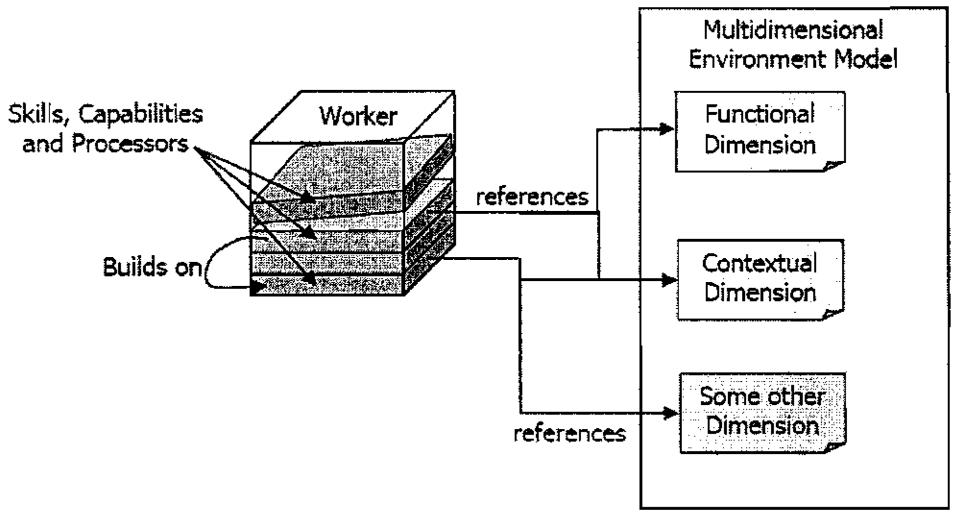

Figure 4. Constructing Human and Automation Agents (Workers) from Skills, Capabilities and Processors Referencing the Environment

This relationship between the environment and agents can be used to computationally construct each agent around the skills, capabilities and processors demanded by its work environment, and to relate the demands of the environment to the resources available in the environment, as illustrated in Figure 4. To examine how an agent model will interact with the work environment, consider again the controller of sector ZLA-39 as illustrated in Figure 3 with one dimension (contextual dimension) and one component (radar equipment) detailed. The controller agent first interacts with one or more dimensions of the work environment. From these dimensions the agent obtains the instances of the relationships that are relevant to his or her current work, and then parses these relationships to identify the components that it needs to interact with. The aspect that maps the component onto the dimension, for example, the radar screen that maps the radar equipment in sector ZLA-39 to the workspace of the sector controller, identifies the properties and the usage mechanism, i.e., the interface elements, of the component with which the agent interacts. Using his or her knowledge about parsing the relationships on a dimension and the general mechanisms to invoke the usage mechanisms on the components, the agent is able to successfully interact with the work environment. The invoking of the usage mechanisms on the components may trigger their internal dynamics.

For such interaction between agent and environment, the agent models should be equipped with some inference making and problem-solving approach that employs the knowledge in each or a combination of dimensions in the work environment model. These inference making and problem-solving approaches may be 'hard-wired', as in the case of agent models of automated systems, may include simple models of human performance (e.g., probabilistic modeling of reaction time with otherwise 'perfect' behavior), or may involve sophisticated human performance models. In our case, when detailed representations of human performance are warranted we propose using AirMIDAS, developed by NASA Ames Research Center (ARC) and San Jose State University (SJSU) primarily for aviation-related applications.[10,11] This computational human performance model contains several internal cognitive behaviors. Mechanistic models of essential psychological and physiological phenomenon such as vision, attention, working memory and motor skills capture well-understood aspects of human behavior. Domain knowledge serves as pre-established knowledge about the task, often represented as procedures and a rule-base of goals and processes for core tasks. An upgradeable world representation also acquires and maintains knowledge about the current state of the environment. Within this framework, a symbolic operator model maintains queues of tasks waiting to occur, and switches tasks between them according to knowledge and goals. The manner in which these behaviors are assembled can be determined by the selection of Cognitive Control Models in response to the demands of the environment.

These methods and models can be used synergistically for (1) formal analysis and for (2) fast-time simulations with computational human performance models capable of predicting and quantifying the performance and safety impacts of changes to the NAS. Computationally assembling the declarative representation of all components (including human and automation agents), and their dimensions enables automatic generation of a declarative model of the system which is too large for unaided analysis by a human but can be formally analyzed by computational algorithms for the metrics of interest. To facilitate formal declarative modeling and analysis, RFS has established an XML representation for the declarative models, and developed a mechanism that then automatically assembles, from the individual components' specifications and interrelations, a network model of the entire system in XML which can serve to analyze dependencies between components and requirements placed on human and automated systems. Such a construction provides a formal method supporting basic modeling and analysis, including identifying the extent to which the functions allocated to the humans and automated agents contribute to mission requirements, and identifying any issues with resource insufficiencys. For example, an environment may require a human to perform a task for which they do not have sufficient information, or may imply the human needs to have excessive training or extraordinary abilities when compared to a set of known capabilities in current operations. Likewise, this formal analysis can generate a 'network' representation of dependencies between functions and the environment components they operate upon that is potentially useful input to function allocation policies.

While this formal method can identify those issues apparent from static representations of the environment and human and automated agents, many metrics of interest – especially the detailed, complex, evolving aspects of system safety – need to be assessed over the course of time. Therefore, RFS additionally includes an object-oriented modeling framework in which complex, dynamic internal behaviors are each encapsulated as computational objects representing the internal dynamics of environment components and the processors (or other computational representations) internal to the agents. While these dynamic models need meet some interface requirements for the sake of centralized simulation clock control and data passing standards, they may internally use any of a wide range of model structures as appropriate.

The combination of the declarative and computational models also enables computational agent-based simulations to predict the dynamic system performance and metrics that will emerge when placed in a given scenario. Before and during run-time, the structure of the declarative model may be changed; the simulation will automatically update the network model of dependencies and adapt the agents' behaviors accordingly. The agent and environment models are represented with sufficient detail that we have already analyzed multi-agent air traffic concepts in which constructs such as interruptions, knowledge state, the relative timing of events and actions, and workload could be estimated. This task could thus focus on incorporating into the RFS and AirMIDAS the ability to formally assemble the models of tasks to derive and record the task demands placed on the human.

The RFS is based on the Reconfigurable Flight Simulator software architecture originally designed for HITL flight simulator development and experimentation.[12] This simulation platform was constructed using the object-oriented design approach in C++ and provides a fairly general and extensible architecture that has since been built on to incorporate novel conceptual constructs suitable for complex-agent based simulations. While HITL experimentation is not the dominant focus of the proposed work, the RFS can still be run in real-time while interacting with human participants, and thus can also provide a seamless link between fast-time computational modeling and real-time HITL experimentation-based methods. Additionally, it has built in networking features with the demonstrated capability to participate in distributed simulations using the High Level Architecture (HLA) networking protocols.

Test case to serve as a demonstration.

A test case was used throughout as a basis for developing an agent-based simulation including human performance models as agents and for using this simulation for risk assessment. Specifically, this test case examined aircraft arrivals into LAX. Several traffic management initiatives have been undertaken to increase the capacity of the system while at least maintaining, if not improving, the operational safety of the system. For example, Time-Based-Metering is one initiative that operationally changes the NAS from distance-based control of spacing in air traffic flows to time-based control.[13,14] This change is enacted through component and network level changes in the work-processes in the work environment of air traffic controllers. This operational change aims to increase the capacity of the airport in terms of the number of arrivals it can accept in any given time. The success of such a work-processes based transformation is highly dependent on the successful integration of humans, technology, work-processes and information in the system, such as the operational and structural transformations being made to the national airspace system.[15] Two different cases were examined in this test case: those where controllers are using 'Miles-In-Trail' (MIT) based spacing between aircraft, and those where controllers are 'Time-Based Metering' (TMB); however, the purpose of this analysis was not to rank one controller method relative to another, but rather to use them collectively to assess the most relevant risks involved in these types of operations. *It should be noted, that the over-arching purpose of this work was to provide the research and development base for NAS risk assessment, not to provide a single risk assessment of this one test case.*

Investigation into the agents' contributions to risk were be based on behaviors at both the system and individual level. Measures of individual behavior focused on a "human-scale" time representation (seconds, minutes, hours). System-wide measures follow large-scale response to these events over hours to days to months and years, and were defined in terms associated with airspace occupancy (arrival rates, time in sector) and dynamics (velocity traces in 4-D airspace/time dimensions). In the evaluation of human-system performance and associated safety-risk or hazard, both these scales (and other intermediate scales) of behavior must be evaluated as to the contribution of those behaviors to some reference level of safety.
The first task identified the important task descriptions and work processes to include in the simulation models, including Conflict Avoidance (CA), Miles-In-Trail (MIT), and Time-Based-Metering (TBM). Conflict avoidance procedures are used to separate any two aircraft that have been detected to possibly come too close to each other in the near future. Such an occurrence of the loss of separation is referred to as a 'violation' in this thesis. The MIT procedures are meant to space arriving aircraft by a specific in-trail spacing that is expressed in terms of nautical miles and is provided to the air traffic controllers by the air traffic management units or is agreed upon by controllers of adjoining sectors. These procedures are meant to achieve a specific arrival rate while also spacing aircraft to allow for additional aircraft to merge into the stream. The TBM procedures are also meant to achieve high arrival rates but operate on the measure of time, where desired time of arrival of an aircraft at specific fixes is used as the target to ensure the desired arrival rate. To fulfill this task, we worked with SJSU to articulate the specific behaviors of interest in the assessment of risk in air traffic control through a series of workshops and meetings. We also coordinated with ATAC Corp. on specific airspace attributes of the scenario (e.g., aircraft arrival patterns) to properly configure and initialize the scenarios.

Figure 5illustrates the air traffic control system examined here. It shows the eastern part of the airspace for Los Angeles International airport (LAX). This system consists of technological / physical components such as the aircraft (pink dots in figure) that fly through the airspace to their respective destination airports. In this case study, all aircraft arriving to LAX are termed as arrivals, while all other flights are termed as overflights. The airspace is spatially divided into multiple contiguous sectors (ZLA-39, ZLA-37, ZLA-20, ZLA-19, SCT-FDR), where the boundaries of these sectors are predefined as abstract polygons in the airspace. The air traffic, i.e., all aircraft, in each of these sectors is monitored for conflict-free and procedure-compliant operation by air traffic controllers, each working on their respective sectors. The air traffic controllers are each equipped with a radar screen that displays the traffic in their sector, a voice radio to transmit traffic control commands to the aircraft and receive requests from aircraft, and a specific set of control procedures. These elements constitute the workspace of the controller, with which the controller interacts to achieve his or her goals, i.e., maintain safer operations and ensure compliance with assigned procedures.

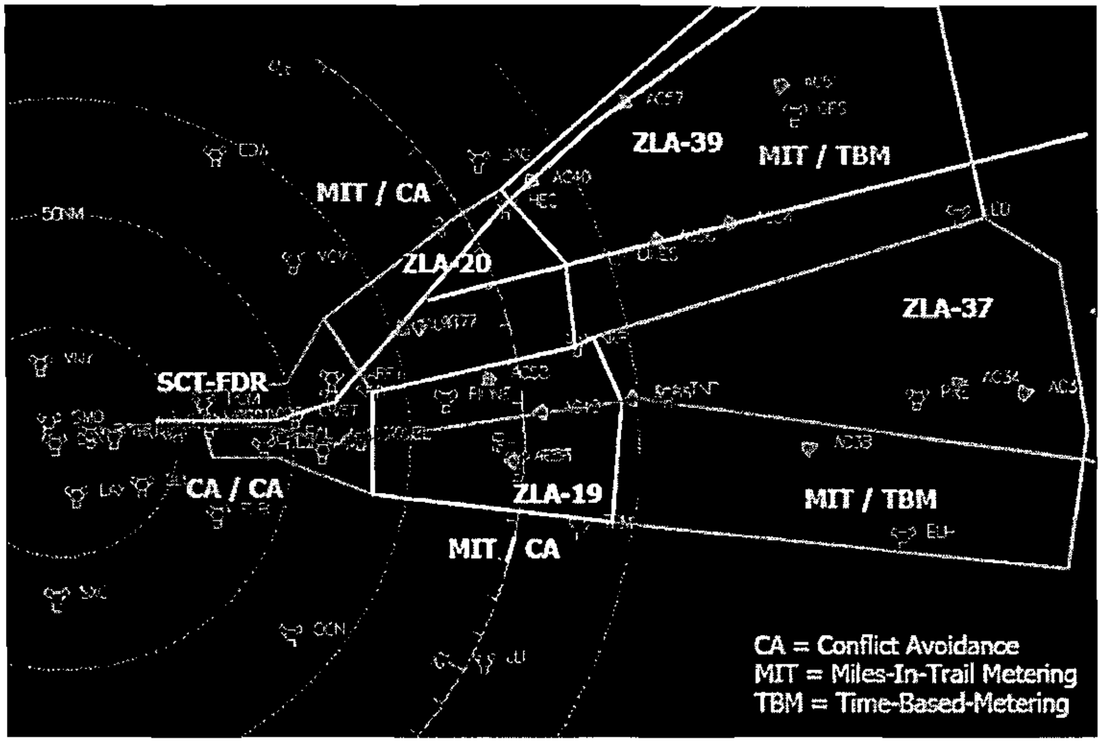

Figure 5: The eastern airspace of the Los Angeles International airport (LAX)

This case was sequentially examined in several phases. The first conducted detailed simulation runs with RFS linked in run-time to AirMIDAS, each running on distributed processors interacting via the HLA distributed simulation protocols. This effort provided initial, detailed analysis of potential safety issues, and identified those aspects of behavior most germane to further analysis with high numbers of simulation runs.

The second phase established streamlined 'mini-MIDAS' human performance models within RFS, and was used to conduct a large number of simulation runs (over 1 million flight hours). This effort involved four researchers (2 at Georgia Tech and 2 at SJSU) dedicated to developing and testing the models of both the system and its components, designing the experiments, and running the simulations. Additionally, the team was supported by experts in the field of modeling and simulation of human performance and air traffic systems. ATAC Corp. provided real-world input data and validated the models against observed reality. However, this one and a half year effort had limited success in creating validated simulations.

Examining this phase of effort, significant time was spent on developing the models, making them work with each other, and, most significantly, in analyzing the impact of changes to the models of the system's components and the system. The model of each system component and each agent was implemented in monolithic object-oriented classes where the models of the capabilities and skills of the agents and the models of the internal dynamics of environmental components were heavily intertwined. There was limited separation between models of components, work environment and the workers. As a result, one incremental change in the model of any system element could cascade through the system, requiring changes in the models of many interacting components.

The validation efforts in this phase were the least successful on two accounts:

1. The modeled system manifested two emergent behaviors that are not characteristic of the real world system that was being modeled. These behaviors were observed in a majority of simulation runs.
2. Those simulations that did not manifest those problematic emergent behaviors did not exhibit performance sufficiently close to the real world data for a third party to validate the simulations. The performance was being measured through the use of thirteen different system level metrics, such as the number of separation violations in each

controlled sector of the airspace, the average distance traveled in each sector and the average time taken by the aircraft in each sector.

There could be a number of causes of these problems.

1. The input data used to configure the simulations could have errors or biases,
2. The individual models of the system components may not adequately represent aspects of behavior they were intended to represent,
3. The intended aspects of component behavior described by the models may not have included some dynamics critical to emergent system behavior, and
4. The output data against which the system was being validated could be wrong or biased.

Though these factors were identified as possible causes during this phase, they could not be tested in practical time due to the limitations of the model forms and their implementation in a simulation platform. Instead, it was chosen to remodel and re-simulate the system to demonstrate the effectiveness of the newly-developed conceptual framework and simulation architecture in the third and final phase of research on this project.

The primary challenges for this third phase were to:

1. Eliminate unwanted and unrealistic emergent behaviors from the simulations, and
2. Create a valid simulation suitable for safety analysis.

In trying to meet these challenges, this work demonstrates the conceptual framework and simulation architecture by:

1. Developing this thesis' conceptual framework-based models of the air traffic control system,
2. Explaining emergent behavior through simulation-based evaluation of the set of probable causes at the component, network and worker levels by transforming the system's models to reflect those changes, and
3. Demonstrating how different configurations of the NAS could be compared operationally using simulation-based analysis, especially with regards to safety analysis.

The next sections describe themodel of the system and its components, document our success in explaining and eliminating the unwanted emergent behaviors, compare arrival operations using MIT versus TBM, and finally discussion validation of the system models.

Modeling the Arrivals into LAX

This section discusses a representative set of models from the case study to demonstrate how an air traffic system can be simulated in a manner suitable for risk assessment using the developments outlined earlier. In the first phases of effort, a number of model specifications were elicited from subject matter experts. This knowledge included specifications for components of the system work environment and for air traffic controller behavior. Components in the work environment included the aircraft, the surveillance and communication equipment used by controllers, the physical structure of the airspace as defined by its division into the sectors, and the air traffic control procedures (work-processes) used by the controllers (pilots were assumed to follow air traffic control clearances exactly, and thus were not modeled in detail). The specification of controller behavior included how they prioritize between tasks and their limitations believed to be intrinsic to human cognition. Apart from the components of the system, the specifications also included knowledge of the structure of the work environment, primarily with respect to the physical structure and the contextual structure. These specifications were reused for modeling the air traffic management system using this thesis' framework.

Air Traffic Controller Work-processes

This section models air traffic control procedures and regulations as work-process components of the work environment. The Federal Aviation Administration and collaborating agencies develop operational procedures to effectively and efficiently manage air traffic, and also develop regulations that the controllers should comply with. In addition, controllers can rely upon informal procedures developed and shared within their immediate community. These procedures and regulations (work-processes) are a part of the system independent of any particular air traffic controller; thus, from an air traffic controller's point of view, these work-processes are a part of their work environment. These work-processes were modeled as work-process components of the work environment.

As discussed in the previous section, a work-process has no internal dynamics, two properties (the expression of the situation in which the work-process applies and the process) and one usage mechanism. Represented as a nested tuple the conceptual model of the work-process component looks like:

Work-Process = (Null, (Situation, Process), (Procedure-Following-Mechanism))

Figure 6 shows the listing of one such procedure that describes the situation and the process' properties in natural language. When computationally represented, this component's properties are expressed in XML, including concrete schema specification for representing the situation and specific work-process instances (excerpts of the work-process shown in Figure 6 are shown in their XML specification in Figure 7 and Figure 8). The usage mechanism 'procedure-following-mechanism' is an algorithm capable of reading these properties, parsing them and using them. For example, this mechanism will make sure that the order of the activities listed in the process is adhered to. Additionally, this usage mechanism will need to access the worker's environmental context to sense the values of the contextual variables that are being used in the situation expression and the process, and to take actions in the context. Thus, the usage mechanism imposes requirements on the air traffic controller model's access to its environmental context.

Heading Merge:
Situation:

1. A conflict is predicted in the future
2. The AC have a common waypoint. i.e. their routes merge ahead.
3. The difference in their heading is more than 15 degrees but less than 135 degrees.

Process:

If conflict time is more than 4 minutes ahead
→

- Spawn Speed control procedure

If conflict time is more than 2 minutes but less than 4 minutes AND
→

- Vector trailing AC by 15 degrees away from the merge
- Speed up the leading AC by upto 50 Kts IAS if it is not already on a speed increase command ELSE slow down the trailing AC by upto 50 knots IAS if it is not already on speed reduction command
- Resume course of vectored AC when appropriate
- Resume speed of the AC whose speed was commanded when appropriate

If conflict time is less than 2 minutes AND
Current vertical separation between them is less than a 1000 ft
→

- Climb higher AC by 2000 ft with VS of 2000 FPM
- Vector trailing AC by 30 degrees away from merge
- Speed up the leading AC by upto 50 Kts IAS if it is not already on a speed increase command ELSE slow down the trailing AC by upto 50 knots IAS if it is not already on speed reduction command
- When vertical separation achieved, command altitude altered AC to approach target altitude with VS of lower AC
- Resume course of vectored AC when appropriate
- Resume flight plan following for AC whose vertical speed was altered
- Resume speed of speed altered AC

If conflict time is less than 2 minutes AND
Current vertical separation between them is greater than a 1000 ft
→

- Spawn Altitude control procedure

Figure 6: Natural language listing of the heading merge procedure for conflict avoidance

```
<KB:Object xsi:type="Proc:typeProcedure" Name="SolveHeadingMerge">

 <Proc:Arguments>
  <Proc:Argument xsi:type="BWT:typeNamedOperand" Type="STRING"
    Name="[.]FirstACInConflict" ArgumentID="FIRSTAC"/>
  <Proc:Argument xsi:type="BWT:typeNamedOperand" Type="STRING"
    Name="[.]SecondACInConflict" ArgumentID="SECONDAC"/>
 </Proc:Arguments>

 <Proc:WorkCondition CheckConditionAt="START" EvaluateToValue="true"
   EvaluateToType="BOOLEAN">
  <BWT:Expression xsi:type="BWT:typeOperator" Name="&amp;"
   Type="BOOLEAN" DLL="" Value="">
   <!-- The given AC must be in conflict at time ahead -->
   <BWT:Argument xsi:type="BWT:typeOperator" Name="areACInConflict"
     Type="BOOLEAN" DLL="" Value="">
    <BWT:Argument xsi:type="BWT:typeNamedOperand" Name="[@PA]FIRSTAC"
     Type="STRING"/>
    <BWT:Argument xsi:type="BWT:typeNamedOperand" Name="[@PA]SECONDAC"
     Type="STRING"/>
   </BWT:Argument>

   <!-- Current heading difference must be greater than 15 degrees -->
   <BWT:Argument xsi:type="BWT:typeOperator" Name="&gt;"
     Type="BOOLEAN" DLL="" Value="">
    <BWT:Argument xsi:type="BWT:typeOperator" Name="fabs"
     Type="DOUBLE" DLL="" Value="">
     <BWT:Argument xsi:type="BWT:typeOperator" Name="-" Type="DOUBLE"
       DLL="" Value="">
      <BWT:Argument xsi:type="BWT:typeNamedOperand"
        Name="[//]RadarData{[@PA]FIRSTAC|STRING}::PtrHeading_deg"
        Type="DOUBLE"/>
      <BWT:Argument xsi:type="BWT:typeNamedOperand"
        Name="[//]RadarData{[@PA]SECONDAC|STRING}::PtrHeading_deg"
        Type="DOUBLE"/>
     </BWT:Argument>
    </BWT:Argument>
    <BWT:Argument xsi:type="BWT:typeValueOperand" Type="DOUBLE"
     Value="15.0"/>
   </BWT:Argument>

   <!-- Current heading difference must be less than 135 degrees -->
   <BWT:Argument xsi:type="BWT:typeOperator" Name="&lt;"
     Type="BOOLEAN" DLL="" Value="">
    ... ...
   </BWT:Argument>
   ... ...
  </BWT:Expression>
 </Proc:WorkCondition>
 ... ...
```

Figure 7: Partial XML specification of the heading merge procedure shown in Figure 6, listing the partial specification of the applicable situation.

```
<KB:Object xsi:type="Proc:typeProcedure" Name="SolveHeadingMerge">
</Proc:WorkCondition>
 ... ...
</Proc:WorkCondition>

<Proc:Process>
 <!-- Define variables -->
  ... ...

 <!-- Start the process with forking amongst the four conditional
      resolutions -->
 <Proc:Fork NodeID="ResolveHeadingMerge">

   <!-- CONFLICTTIMEAHEAD > 4 mins, spawn speed control  -->
   <Proc:Choice>
    <!-- specify condition expression  -->
    <Proc:Expression EvaluateToValue="true" EvaluateToType="BOOLEAN">
     <BWT:Expression xsi:type="BWT:typeOperator" Name="&gt;"
       Type="BOOLEAN" DLL="" Value="">
      <BWT:Argument xsi:type="BWT:typeNamedOperand"
        Name="[@VA]CONFLICTTIMEAHEAD" Type="DOUBLE"/>
      <BWT:Argument xsi:type="BWT:typeValueOperand" Type="DOUBLE"
        Value="240"/>
     </BWT:Expression>
    </Proc:Expression>

    <!—Spawn speed control procedure  -->
    <Proc:Spawn ProcedureName="SpeedControl" NodeID="spawnSpdCntrl"
      Parallel="true">
     <Proc:Arguments>
      <Proc:Argument xsi:type="BWT:typeNamedOperand"
        Name="[@PA]FIRSTAC" Type="STRING" ArgumentID="FIRSTAC"/>
      <Proc:Argument xsi:type="BWT:typeNamedOperand"
        Name="[@PA]SECONDAC" Type="STRING" ArgumentID="SECONDAC"/>
     </Proc:Arguments>
    </Proc:Spawn>
   </Proc:Choice>
   ... ...
 </Proc:Fork>
 ... ...
</Proc:Process>
</KB:Object>
```

Figure 8: Partial XML specification of the heading merge procedure shown in Figure 6, listing the partial specification of the process.

This demonstration included three work-processes used by the air traffic controller:

1. Miles-in-trail (MIT): These work-processes help the air traffic controllers space aircraft on the same route by a given distance. In real-life the desired distance is provided to the controllers by outside traffic management units.
2. Time-Based-Metering (TBM): In time based metering the Traffic Management Advisor (TMA), a technological aid, provides the controllers with 'delay times' that specify when each aircraft should arrive over a specific fix. The time-based-metering work-processes are meant to help the controller slow and space the aircraft to 'absorb' their delay times.
3. Conflict Avoidance: These work-processes are meant to help the air traffic controller avoid conflicts between aircraft. A conflict is a situation when two aircraft come closer than five nautical miles horizontally and one thousand feet vertically above an altitude of 18,000 feet, and three nautical miles horizontally and one thousand feet vertically below 18,000 feet.

The assignment of work-processes to each controller was one way of specifying the system's network level structure. For example, when exercising MIT control, the controllers of the higher sectors, i.e., ZLA-37, ZLA-39, ZLA-20 and ZLA-19, were given the MIT work-processes, while the SCT-FDR controller was given the Conflict Avoidance work-processes. On the other hand, when exercising TBM control, controllers of sectors ZLA-37 and ZLA-39 were given TBM work-processes while the other three were given conflict avoidance work-processes.

Multidimensional Model of the Work environment

The system modeled in this demonstration consisted of five air traffic controllers, each controlling the airspace in one of the five contiguous sectors on the eastern approach to the Los Angeles International Airport (LAX) (Figure 9). There are a number of flights that fly through these sectors: some of them are arrivals into LAX (as shown by the red and green lines that extend from the right hand side of Figure 9 to the left hand side); others are considered 'over-flights'. Each sector controller has a display of flights in his or her sector, and a voice radio by which he or she issues commands to pilots onboard the aircraft. The controllers have little knowledge of the aircraft in other sectors and do not frequently communicate with other controllers; therefore, the other controllers and aircraft outside their sector are not considered to be in their context.

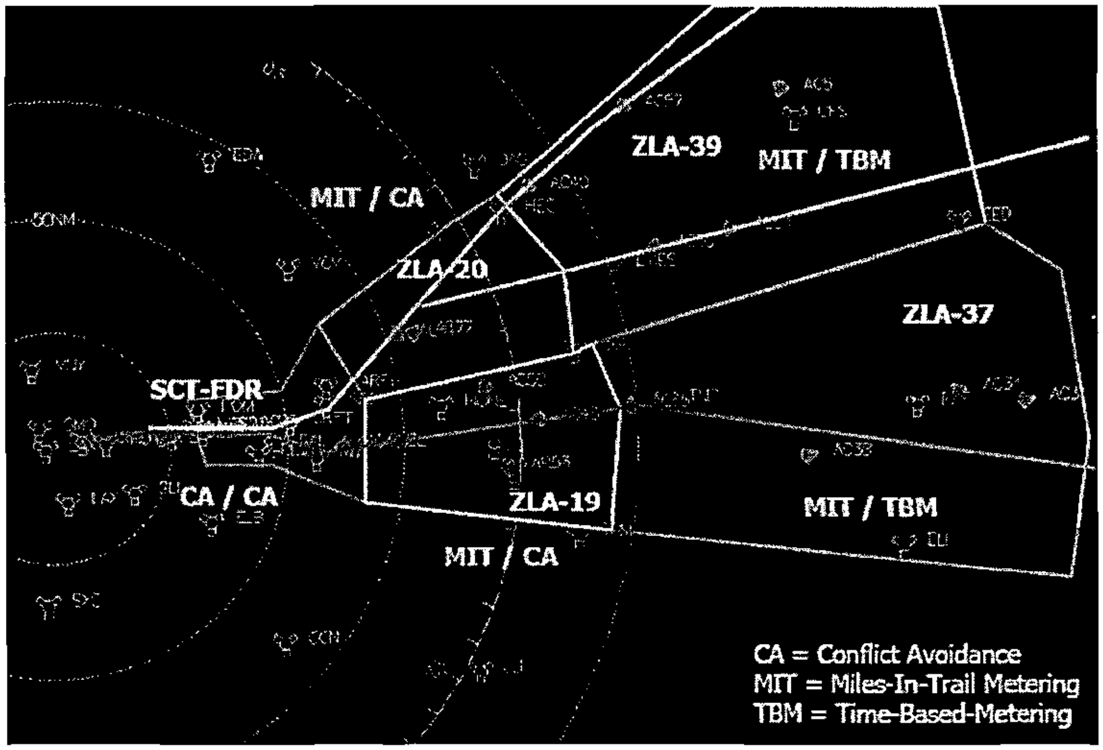

Figure 9: Sectors and arrivals at LAX

Based on this specification we know that the components in the work environment of the air traffic controllers include the aircraft, the surveillance and voice radio equipment and the work-processes'. The reader should note that not all these components have the same lifespan as the controller: aircraft appear in the work environment according to their schedule and then exit the airspace. Thus, this is a dynamic work environment.

Viewed from top-down, the contextual dimension is constructed by assembling contextual nodes for each component of the work-environment and then associating them through contextual-compositions. In this case, the contextual dimension includes one contextual node for each control sector, which further subsumes the contextual nodes for the surveillance equipment, the control equipment, the wind measurement equipment, the work-process information system and the flight strips management system (Figure 10). These further subsume contextual nodes containing those parts of each component available to the contextual node.

Viewed from the bottom-up, the model of each component of the work environment includes a set of properties and usage mechanisms. These attributes are grouped into aspects that map them onto the contextual dimension of the work environment.

For example, let us consider the aircraft component. For the purpose of air traffic control, the work-relevant properties of the aircraft may be summarized as:

1. Latitude,
2. Longitude,
3. Altitude,
4. Ground speed,
5. Vertical speed,
6. Heading,
7. Flight Path Angle, and
8. Flight Plan

Each aircraft model has a set of contextual aspects that map these attributes onto the contextual nodes comprising the contextual dimension. The surveillance equipment includes a set of contextual aspects of each aircraft within its monitored airspace through a contextual-composition relationship. The internal dynamics of the surveillance equipment represents a daemon that makes sure that at any given point of time in the simulation the contextual nodes of all aircraft in the sector are included in the contextual node of the surveillance object. Similarly, contextual nodes are created for flight strips available in the context of a controller (Figure 10).

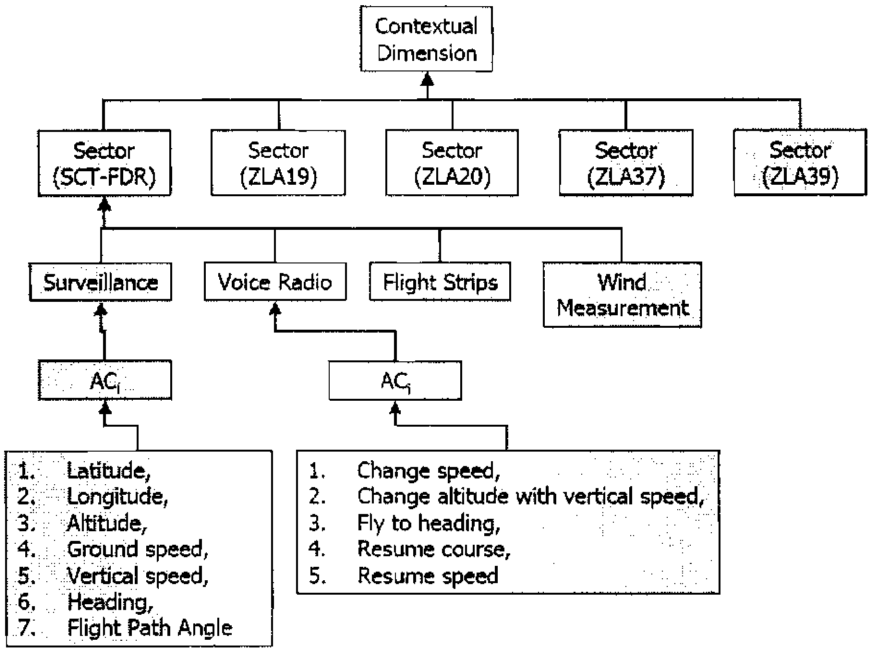

Figure 10: Part of the contextual dimension in the case study

The usage mechanisms of an aircraft include:

1. Change speed,
2. Change altitude with vertical speed,
3. Fly to heading,
4. Resume course, and
5. Resume speed.

In the real world, these usage mechanisms are available to the pilot through controls in the cockpit or through flight management systems. For example, pilots tune their communication equipment to the frequency assigned to the control sector that they are flying through and thus come into the contextual dimension of the controller, where they can be given commands by the controller. In this case study, these pilot-controller communication mechanisms were assumed not to be important for the analysis and these usage mechanisms are directly made available in the context of a controller by mapping them over to their contextual dimension through their voice radio equipment.

The functional dimension relates the environmental components to the goals through means-ends-constraints relationships; in other words, it identifies them as affordances and constraints towards one or more goals. The construction of the functional dimensions starts with identification of the goals (Figure 11).

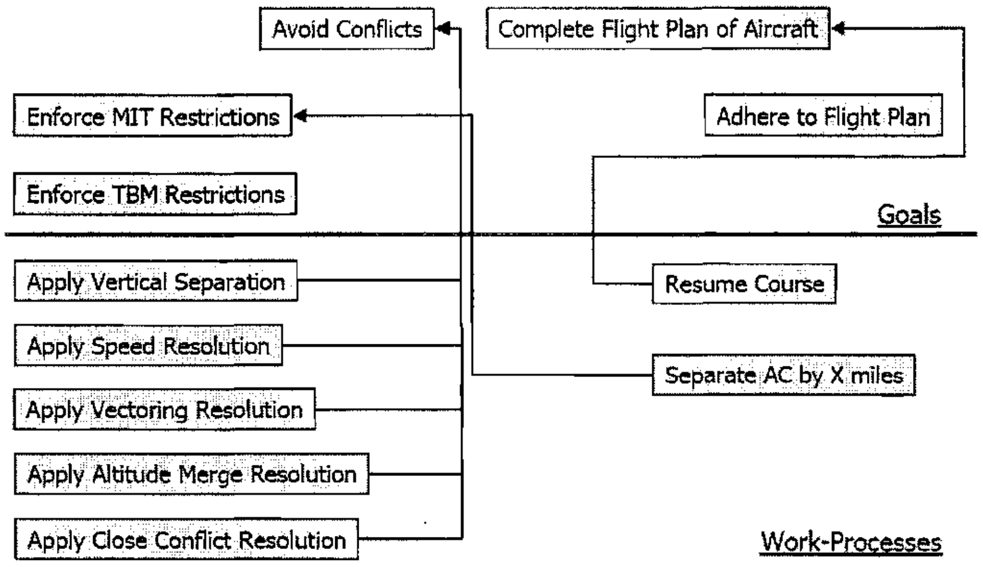

Figure 11: Relating goals and work-processes via the functional dimension

Once the goals have been identified, the environmental components are tagged as means and constraints. In this case we have two kinds of environmental components: technological artifacts and work-processes. For the sake of brevity of this discussion, Figure 11 shows a partial model including only the goals and the work-processes; the artifacts are included in the full model in an identical manner as the work-processes. In the figure, the associations with arrowheads identify that the component at the tail of the arrow is a means to the work-objective at the head of the arrow. If the arrowhead is a dot, it identifies the component at the tail of the arrow as a constraint.

Environment-Centered Construction of Agents

In this section, models of the air traffic controllers are automatically generated from the work environment model and the assignment of goals and context to workers. Air traffic controllers are the only workers modeled here. There are five of these workers, one for each of the control sectors. They are each allotted to a specific sector in the contextual dimension (Figure 10), thus making the appropriate contextual node (and any it subsumes) available to each controller. They are each allotted a subset of goals from the set shown in Figure 11. For instance, when the system design represents an MIT configuration, the air traffic controller for sector ZLA19 (one of the four higher sectors that enforce MIT restrictions between arrivals) is allotted the following objectives: Enforce MIT Restrictions, and Complete Flight Plan of

Aircraft. For an environment-centered design, making these allotments is the only activity required when designing the system.

However, an agent also needs to be constructed as a configuration of skills, capabilities and processors. This thesis' simulation platform is capable of building the agent based on the system-level design. Specifically, using the model composition architecture discussed earlier, the model construction engine first identifies all environmental components associated with the allotted goals on the functional dimension and with all components in the contextual dimension that are associated with the allotted context of the worker. The usage mechanisms of these components are identified and the skills and capabilities associated with them are picked out from the computational model registry and added onto either an empty agent model or to a template agent model which may be hand-picked by the designer. When a template model is picked by the designer, the model construction engine identifies those skill and capability implementations from the computational model registry that correspond to the template. These skills and capabilities are then aggregated into the agent. At this point the agent model is ready to be used in the simulation for operational analysis; in addition, the assembled list of skills and capabilities can be examined during network analysis to examine the feasibility of the set for the intended worker and to identify training and information requirements for the worker.

Table 2 shows the full set of skills and capabilities in the controller for sector ZLA-39 when given MIT procedures. Each of these skills and capabilities was coded using C++ into facets that can be aggregated into the agent model.

Table 2: The full set of skills and capabilities of the controller of sector ZLA-39 when given MIT procedures

	Skill Name	Skill Description
1	AreACInConflict	Check if two aircraft are in conflict
2	GetIfCommonWaypoint	Check if two aircraft have a common waypoint
3	GetDistanceBtPts	Get horizontal distance between two points
4	GetHdgBtPts	Get heading between two waypoints, as measured from the North
5	GetIfFailedResolution	Makes sure if a particular resolution failed for given aircraft in a given conflict
6	IsPointInSector	Check if a given point is in sector
7	GetDstncToMergePoint	Calculate the distance between an aircraft's current position and the merge point of that aircraft and another aircraft.
8	GetHdgFromMergePoint	Get the orientation of an aircraft from the merge point of that aircraft and another aircraft
9	ChangeSpeed	Command a given aircraft to change speed
10	ChangeAltWithVS	Command a given aircraft to change altitude with a given vertical speed
11	ChangeHeading	Command a given aircraft to change heading to a given heading
12	ResumeCourse	Command an aircraft to resume course
13	ResumeSpeed	Command an aircraft to resume waypoint speed
14	ResumeAltitude	Command an aircraft to meet waypoint altitude restriction
15	Wait	Wait for a given amount of time
16	GetDoubleProperty	Read a variable from the context
17	GetFlightPlan	Read flight plan of a given aircraft from the context
18	MonitorConformance	Monitor boundary conformance for all aircraft sent off-course by the controller
19	MonitorTraffic	Monitor traffic in sector for possible conflicts
20	IsChangeConflictFree	Judge if a particular maneuver for a particular aircraft would be conflict free in given time frame

21	CalculateLatLongAltAtTimeAhead	Calculate the position of an aircraft in given future time
22	IsACReadyToResumeAlt	Make sure that the aircraft can resume altitude without creating future conflicts
23	IsACReadyToResumeSpeed	Make sure that the aircraft can resume speed without creating future conflicts
24	IsACReadyToResumeHeading	Make sure that the aircraft can resume course in two dimensions (not vertically) without creating future conflicts
25	IsACReadyToResumeCourse	Make sure that the aircraft can resume three-dimensional course without creating future conflicts
26	IsACOnAltChange	Check if the aircraft has been commanded altitude changes
27	IsACOnHeadingChange	Check if the aircraft has been sent off-course
28	IsACOnSpeedChange	Check if the aircraft has been commanded to change speed
29	MonitorMITTraffic	Monitor arrivals for in-trail spacing violations
30	IsACInMITViolation	Check if a particular aircraft is closer to any other arriving aircraft than the required in-trail spacing
31	GetViolationDistance	Calculate the distance that a particular aircraft will have to absorb to avoid in-trail separation violation
32	GetDistanceToDestination	Calculate aircrafts along track distance from destination
33	GetHeadingFromDestination	Calculate orientation of a given aircraft from the destination
34	IsPastILSMerge	Calculate if an arrival aircraft is past the ILS merge point for the destination
35	GetHeadingFromILSMerge	Calculate the orientation of the aircraft from the ILS merge
36	GetDistanceToILSMerge	Calculate the along track distance of an aircraft from the ILS merge point
37	GetConflictTimeAhead	Calculate time to conflict
38	FollowProcedures	Follow the procedures in the context

The agent models were constructed from the template for the rudimentary human performance model described earlier. This template was populated with the skills and capabilities that are specifically needed of an air traffic controller based on the specifications just discussed. Except when a worker-level transformation was examined, the activity parameters were as shown in Table 3.

Table 3: Specification of activity parameters for the air traffic controller model

Agent Type	Limited resources (Max Resources = 7)					
Source of Variability	Duration		Accuracy	# Resources		
Distribution	Normal		Uniform	Normal		
Activity	Mean	Stdev	Probability	Mean	Stdev	Resource Acquisition Priority Rating
Monitor Traffic for Conflicts	6	4	0.8	3	1	5
Monitor Traffic for MIT Spacing	3	2	0.8	3	1	4
Monitor Traffic for TBM Compliance	3	2	N/A	2	1	4
Change Speed	6	2	N/A	4	1	8
Change Heading	6	2	N/A	4	1	8
Change Altitude	6	2	N/A	4	1	8
Resume Course	3	1	N/A	4	1	8
Resume Speed	3	1	N/A	4	1	8
Monitor Sector Boundary Conformance	2	6	N/A	2	1	10
Wait	Var	5	N/A	1	1	1
Follow Procedures	Unlimited		N/A	2	2	1

Explaining Emergent Behaviors

This section demonstrates how the two unrealistic emergent behaviors observed in the previous phases were explained and, through use of simulation of the models, used to correct the models:

1. Occurrences of unplanned steep descents in flight paths, and
2. Occurrences of unplanned horizontal "loops" in flight paths.

These emergent behaviors could arise from either the aircraft or the controller behaviors, mismatched or untimely interaction between the two, the work-process specifications, or a combination of these factors. Since the previous modeling architecture was not well suited to quick modifications of its models of environment components and agent behaviors, exploring the full set of possible changes to the models would have required significant software modification and a prohibitive duration of development time.

Once the system was modeled using this framework and implemented over the software and simulation architecture, the level of effort to test several design variables was significantly reduced and a sufficient range of conditions was examined to not only explain the causes of those emergent behaviors but also eliminate them in the simulation, as described in the following sub-sections.

Explaining Steep Descents

The altitude profiles of arrivals using the previous simulation are shown in Figure 12. It was observed in the previous simulation that to avoid conflicts some aircraft would be commanded to either hold altitude or climb to higher altitudes, and then later be commanded to resume course and meet waypoint altitude restrictions. In such situations, the aircraft would sometimes exhibit steep descents. Approximately 98% of the 11,200 simulation runs, each with approximately two hour of simulated time and an average of approximately 28 arrivals per simulation, manifested this behavior for one or more aircraft irrespective of which procedures (MIT or TBM) were in place.

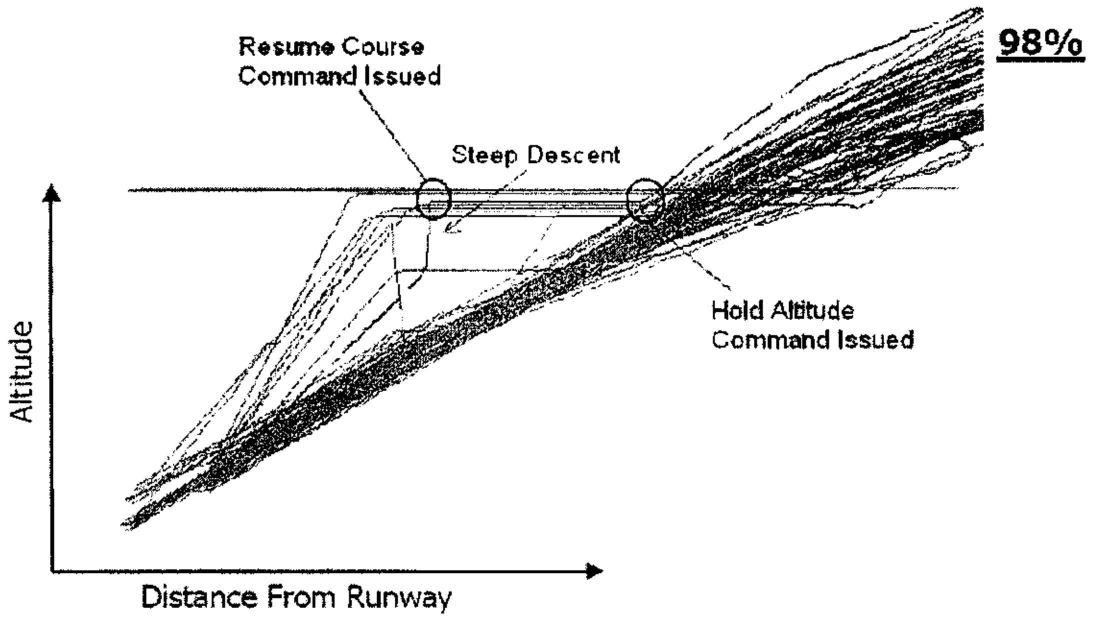

Figure 12: Vertical profile of arrivals using the previous models

As a first step in finding the source of and eliminating these unwanted emergent behaviors, the internal dynamics of aircraft were modified to have internal limits on their descent rates. Changing the system model for such a component level transformation in the system was as simple as replacing the facet that models the basic flight dynamics on the aircraft model with a new facet that enforces limits on behavior. Although this change corrected the problem of steep descents, it resulted in the aircraft not being able to meet their altitude restrictions (Figure 13), another unrealistic emergent behavior. In addition, the traffic flow within the airspace changed in a direction that led to too many conflicts and too many altitude hold commands. This component level transformation was therefore discarded as a possible candidate for eliminating the unwanted emergent behavior.

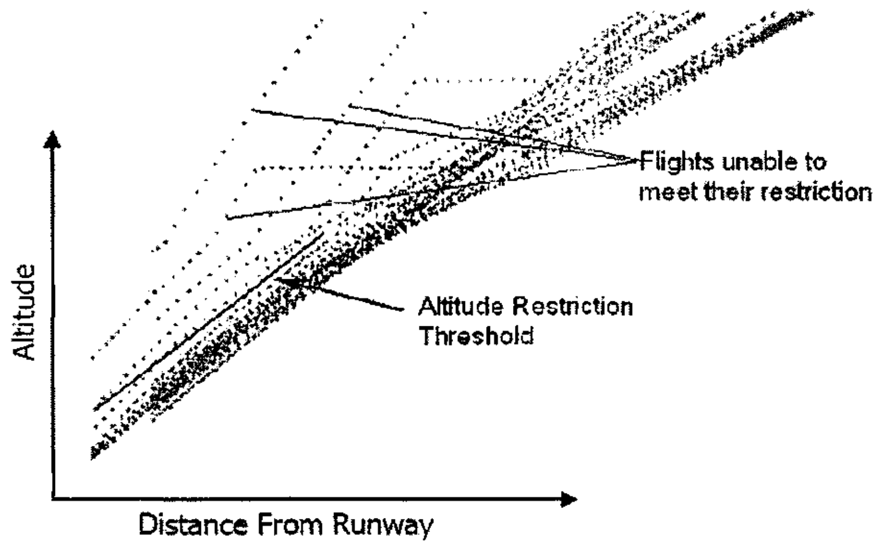

Figure 13: Vertical profile for arrivals when limits on vertical speeds were encoded in aircraft internal dynamics

Since transforming aircraft behavior was not the correct solution to the problem, the next step examined transformations in work-processes. As shown in Figure 12, the aircraft were commanded to hold altitude in order to avoid conflicts. As a result, the aircraft drew close to their next horizontal waypoint while staying at a much higher altitude. Thus, when commanded to resume course and meet the altitude restriction of their next waypoint, they tended to exhibit very steep descents. This observation was used to transform the work-processes to not hold aircraft at the same altitude for very long, but instead descend aircraft at fixed vertical speeds as soon as they are vertically separated (Figure 14). The overall system performance in terms of separation violations and number of occurrences of steep descents improved with such a change: although the aircraft continued to sometimes exhibit descents at a high rate, these descents were not as drastic and as frequent as before. Examining the recorded behaviors found the remaining steep descents occurred when the aircraft were commanded to descend at the given vertical speed at lower altitudes where the true air speeds are lower. Modifying the work-processes further to schedule the commanded vertical speed by altitude resulted in reduction of steep descents from occurring in 98% of simulations to 0.3% (Figure 14), while also improving system performance in terms of reducing the number of violations recorded per run and the number of total diversions from planned flight paths of aircraft. Figure 15shows the output of one simulation where one flight was held at a given altitude but, when later brought back on course, its rate of descent was within tolerance.

The effort involved in making these component level transformations in the work-processes was as little as changing its XML representation of which components to include and modifying isolated models of activities, without any cascading interactions between the various component and agent models.

Process to vertically separate two arrivals

(a) Hold altitude of higher aircraft

(b) Recheck every 30 seconds if resuming course is conflict free

(c) Resume course when appropriate

Transformed process to vertically separate two arrivals

(a) Hold altitude of higher aircraft until vertical separation is achieved

(b) Once separated vertically, descend higher aircraft to next waypoint altitude at vertical speeds less than that of lower aircraft. When assigning vertical speed, compensate for the following

   (1) If true airspeed of aircraft is greater than 250 knots, assign 2000 feet per minute

   (2) If true airspeed of aircraft is less than 250 knots, assign 1500 feet per minute

(c) Recheck every 30 seconds if resuming course is conflict free

(d) Resume course when appropriate

Figure 14: Vertical separation procedures before and after transformation

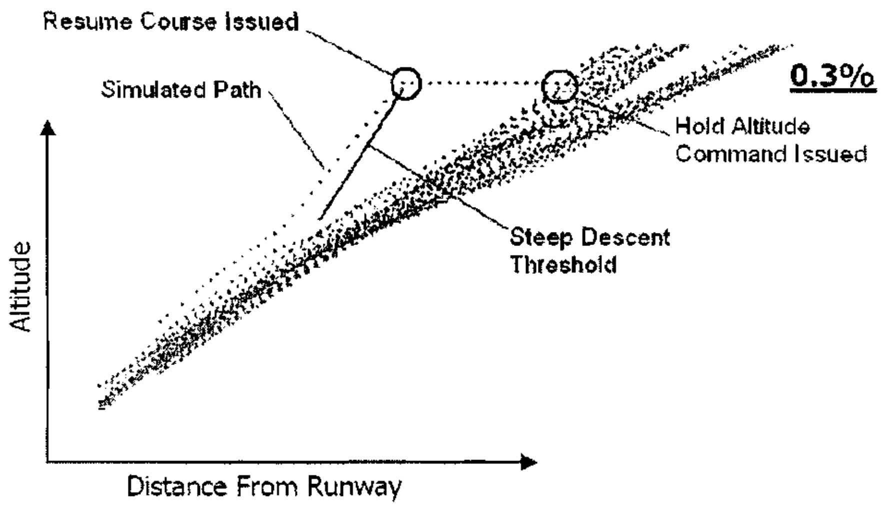

Figure 15: Vertical profile of arrivals when air traffic controller work-processes were changed to not command high vertical speeds

Explaining Loops

It was observed in the previous modeling effort that, to avoid conflicts, some aircraft were vectored off their routes and then be commanded to resume course. In such situations the aircraft would sometimes exhibit a "horizontal looping" behavior (Figure 16) where the aircraft turns back to the waypoint it was originally enroute to. Approximately 51% of the 11,200 simulation runs manifested this behavior.

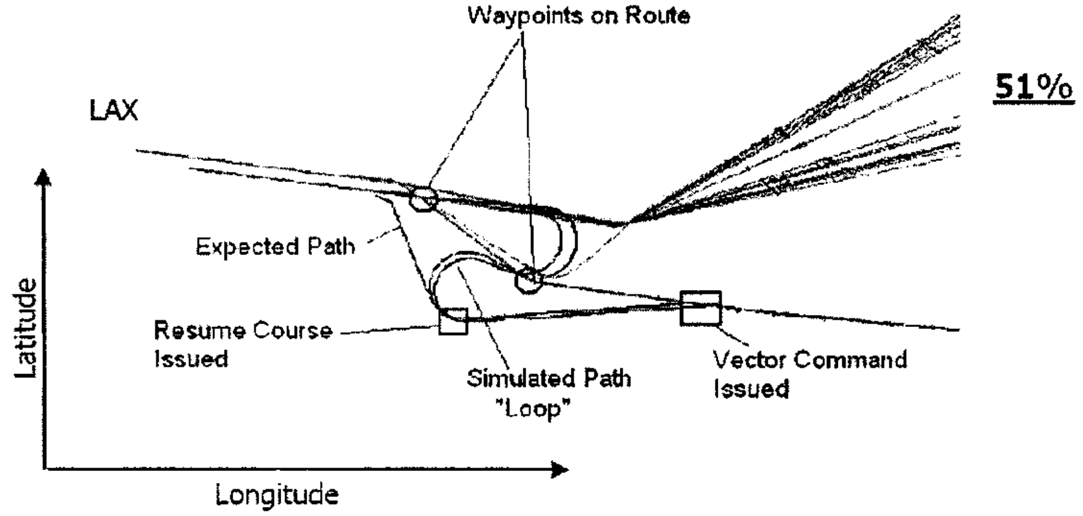

Figure 16: Horizontal profile of arrivals in previous simulation exhibiting the horizontal looping behavior

Similar to the case of steep descents, any of the aircraft and controller behavior and the work-processes models could account for such a behavior. Each one of these possible causes were examined. Examination of the event logs showed that the aircraft, when commanded to resume course after the earlier deviation from it, resumed their path to the waypoint that they were enroute to before they were vectored off. If that waypoint was now behind them they had to horizontally loop back to reach it. This was obviously a mismatch between the internal dynamics in the flight management system (FMS) of the aircraft and the expectations implicit in the controller's command to resume course. Figure 17 illustrates the internal mechanism of the aircraft's model of its FMS. The pink tag on the usage mechanism of the aircraft that is used to command the aircraft to vector off route at a given heading invokes the heading override behavior in the internal dynamics of the aircraft. This event-based (responsive) behavior further changes the aircraft's autonomous waypoint following behavior to steer it off the route. When a resume course command is issued through the use of the corresponding usage mechanism (identified by the green tag), it invokes a course-resumption behavior in the internal dynamics that again affects the autonomous waypoint following behavior of the aircraft. It was noted that the previous model's course-resumption behavior always reverted to the waypoint before the off-route deflection. On the other hand, the controller assumed that the aircraft would resume course towards the next waypoint in its flight plan that has a bearing between +90 degrees and –90 degrees relative to its current heading, without the controller needing to explicitly direct the aircraft to this new waypoint.

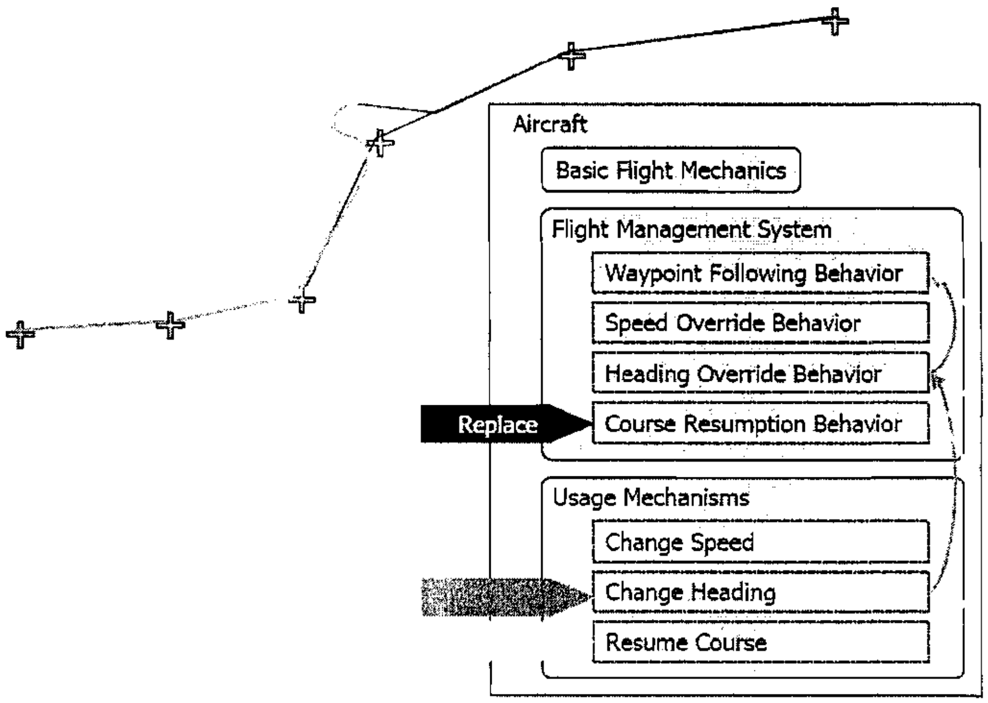

Figure 17: Illustration of the internal mechanism of the aircraft model that leads to horizontal looping behavior

In order to test this hypothesis, a component level transformation was introduced in the system through changing the course resumption behavior in the internal dynamics of the aircraft model to match the assumptions of the controller. Using the architecture of this thesis, this transformation was easily introduced by simply replacing the facet that implemented the course resumption behavior, within the internal dynamics of the aircraft, with a new one that exhibited the desired FMS characteristics. Due to the modular nature of the model elements, this change did not require any concomitant changes in any other elements of the aircraft or the system models. Furthermore, this component level change completely eliminated the "horizontal loop" (Figure 18), thus also explaining the cause of this unwanted emergent behavior.

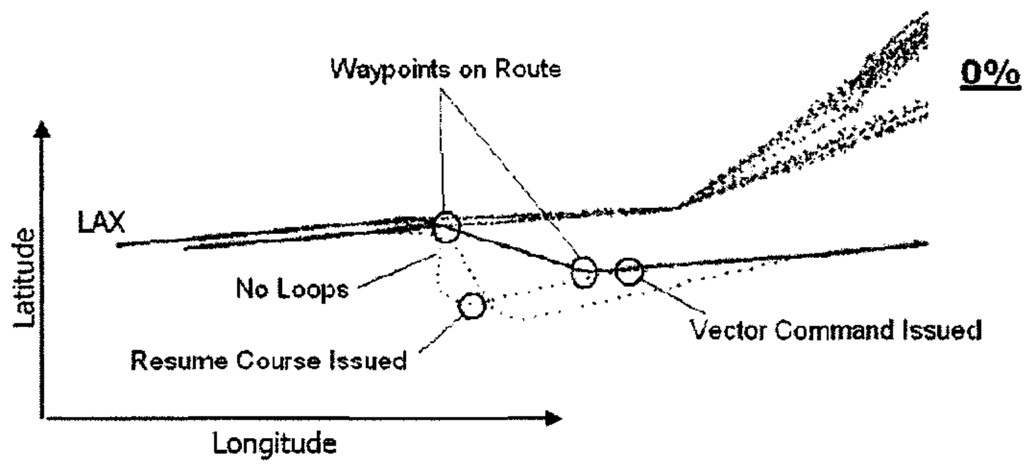

Figure 18: Horizontal profile of arrivals with the changed internal dynamics of the flight management system of aircraft

The time and effort spent in this phase was far less compared to the prohibitive development time required in the previous phases. Using this project's conceptual framework and simulation platform, once the basic modeling constructs for the domain had been created, it took one researcher less than one week to test the intuitions, i.e., create new models for component internal dynamics or agent behavior (through the use of facets specific to those behaviors), run a number of simulations to analyze behavior, and feed back the changes in the models. This exercise demonstrates the ease, flexibility and efficiency of using this project's conceptual and practical constructs in explaining emergent behavior and testing different design alternatives through the use of component level transformations.

Figure 19 and Figure 20compare the previous model's performance with that of this thesis' model after correcting the unrealistic emergent behaviors. The graphs show the number of times aircraft got too close to each other ("violations") in each of the five sectors for each of the four scenarios (i.e., different arrival streams in different wind conditions) in both the MIT and the TBM work-processes. The graphs clearly show an improvement in performance as measured by a reduction in the number of violations.

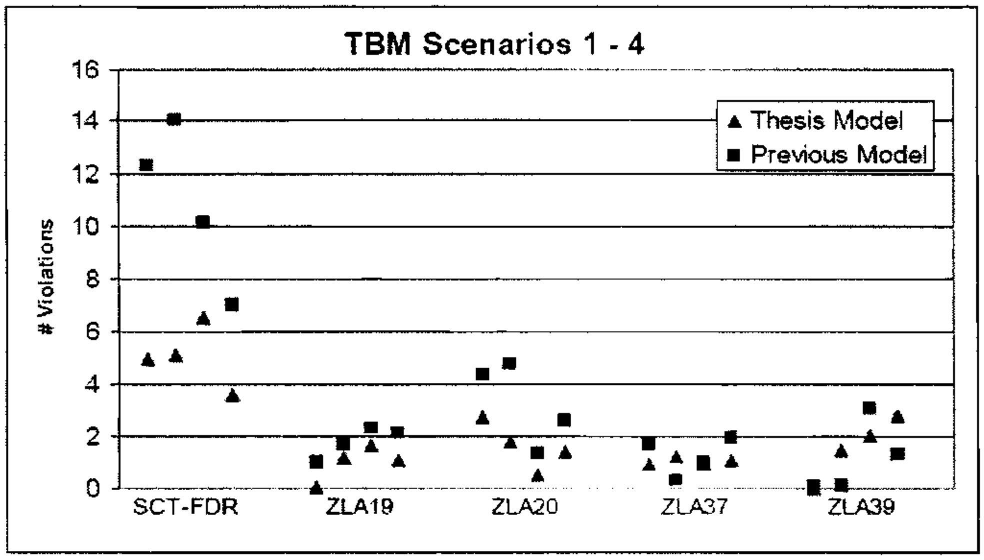

Figure 19: Comparison of average number of separation violations per sector per scenario for previous and corrected TBM models

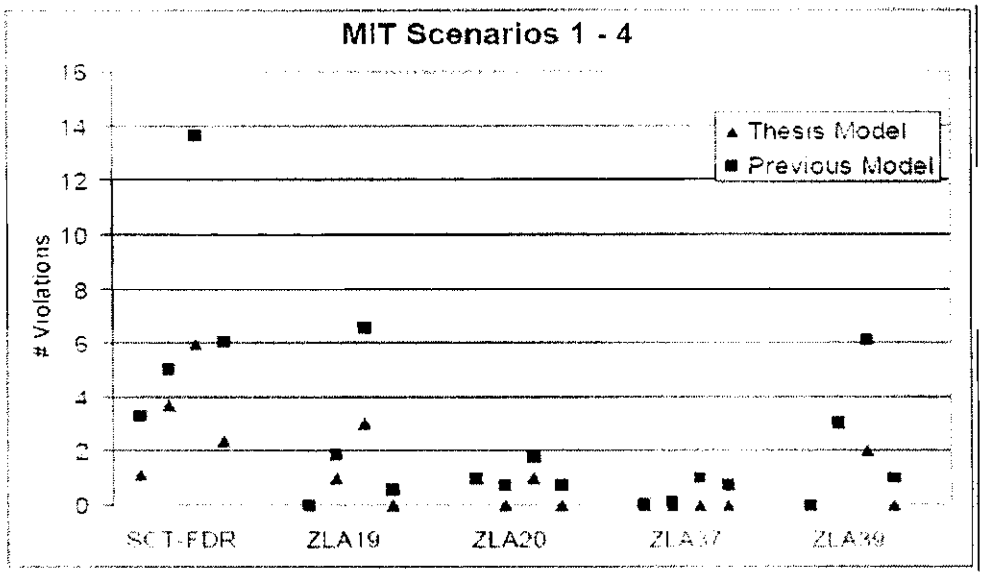

Figure 20: Comparison of average number of separation violations per sector per scenario for previous and corrected MIT models

Analyzing Configurations of and Changes to Air Traffic Systems

This section describes two additional analyses of transformations to this air traffic control system. A network level transformation in the work environment is exercised through changing the set of work-processes available to each air traffic controller, including MIT versus TBM as well as examining the implications of extra conflict avoidance procedures. The impact of estimated human performance parameters were also examined.

In the MIT configuration, the controllers of the four higher-altitude sectors (ZLA-39, ZLA-37, ZLA-20 and ZLA-19) use procedures for distance-based separation between aircraft (MIT procedures) which help them issue air traffic clearances to ensure that no two aircraft come closer than the desired in-trail spacing. The air traffic controllers' primary goal was to enforce this MIT restriction and make sure that each aircraft was resumed on its course before it left their sector, if they had earlier deviated it off course. The prior phases had assumed this was the only work-process stipulated to the controllers of these sectors.

In the transformed work environment, each controller was provided with an additional set of work-processes, the conflict avoidance procedures, which are meant to additionally prevent separation violations. To transform the system in this way, the only changes in the system model were (1) to change the goal assignment of the controllers (workers) of the ZLA sectors to also include violation prevention and (2) to add these procedures in the contextual dimension to the context of the ZLA sector controllers. The functional dimension did not need any changes because the procedures already existed in the system for the SCT sector and thus were already associated by means-ends-constraints relationships with the goals. Once these changes were made in the declarative model, i.e., the XML representation of the contextual dimension, the system and agent construction architecture automatically constructed the computational models needed for the simulation. The agent models did not have to change because they were already capable of processing whatever set of work-processes is assigned to them.

Figure 21 compares the performance of the new design alternative with the existing system in terms of the number of separation violations found in each sector over multiple runs in several scenarios. Though performance is improved for the new design alternative, i.e., the average number of violations is reduced, these results could not be compared with reality due to lack of data.

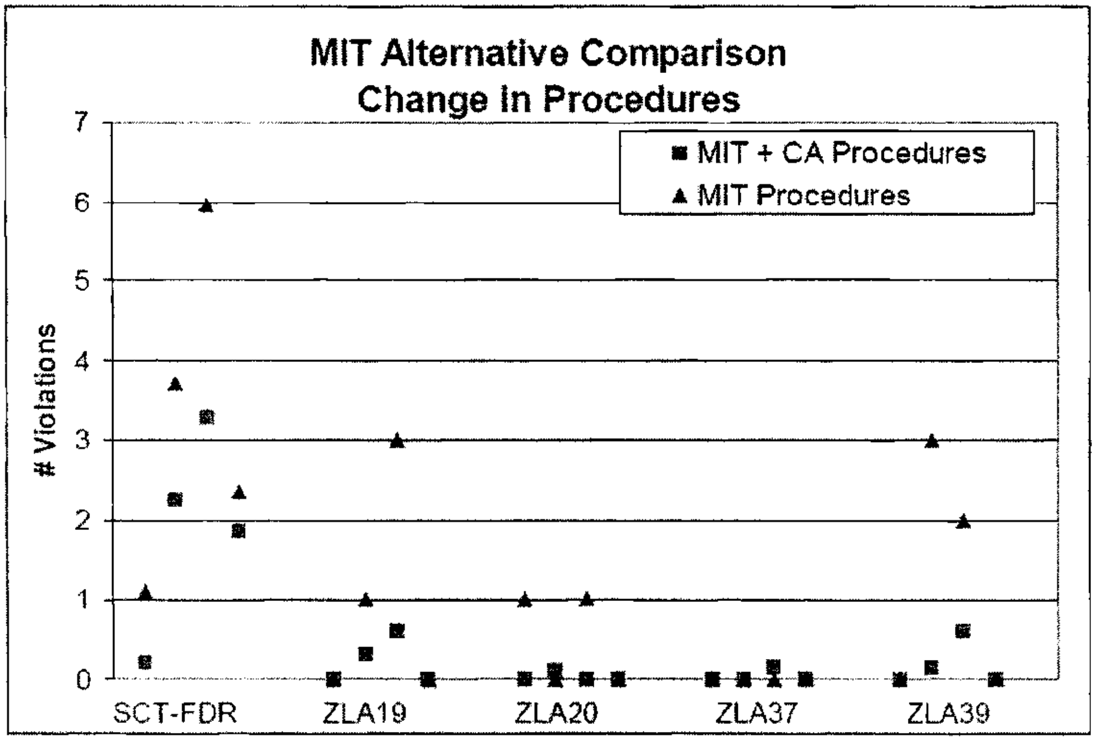

Figure 21: Comparison of average number of separation violations per sector per scenario for changes in the changes in air traffic procedures

To assess the effectiveness of the procedures by themselves in controlling the airspace relative to important considerations in human performance, the human performance model was replaced with a worker model with resource limits and without any stochastically induced inaccuracies in its activities in terms of the probability of detecting a conflict or a spacing problem in every scan of the radar display (Table 4). This transformation was enacted using the same declarative model of the worker by simply (1) replacing the reference in declarative model to the facet for the original resource provider with a modified version that did not limit resources and (2) replaced the reference in the declarative model to the facets for the activities involving inaccuracies to slightly modified facets which did not have inaccuracies. Rerunning the model constructor on the new declarative model of the system created the computational model of the transformed worker and the system.

Table 4: Summary of worker-level transformation in air traffic controller model

Agent Type	**Unlimited resources**	**Limited resources (Max Resources = 7)**		
Source of Variability	Accuracy	Accuracy	# Resources	
Distribution	Uniform	Uniform	Normal	
Activity	Probability	Probability	Mean	Stdev
Monitor Traffic for Conflicts	1	0.8	3	1
Monitor Traffic for MIT Spacing	1	0.8	3	1
Monitor Traffic for TBM Compliance	N/A	N/A	2	1
Change Speed	N/A	N/A	4	1
Change Heading	N/A	N/A	4	1
Change Altitude	N/A	N/A	4	1
Resume Course	N/A	N/A	4	1
Resume Speed	N/A	N/A	4	1
Monitor Sector Boundary Conformance	N/A	N/A	2	1
Wait	N/A	N/A	1	1
Follow Procedures	N/A	N/A	2	1

Simulations for the same scenarios as before yielded the results shown in Figure 22. The improvement in performance (i.e., reduction in violations) was expected due to increased accuracy and improved response time of the worker. As before, this one cannot be validated since the real air traffic system is always operated by resource constrained controllers, but this exercise does demonstrate the ease with which the impact of various human performance parameters can be examined during risk assessment.

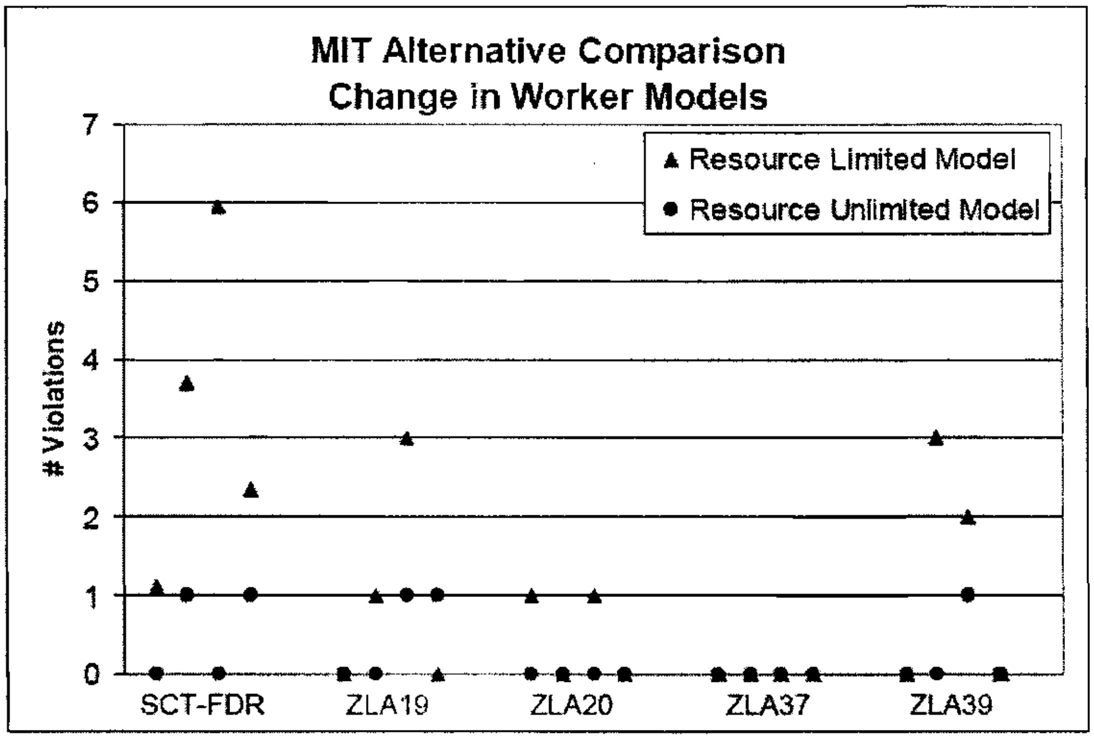

Figure 22: Comparison of average number of separation violations per sector per scenario for changes in worker models

In terms of efficiency it took one researcher about two days to enact the worker level transformation, configure and run 40 simulations for each scenario on each of eight different machines, post process the data, and analyze it. It took about three days to do the same for worker level transformations. This level of effort is sufficiently low as to motivate such analyses as a regular, integral part of risk assessment of air traffic systems.

Validating System Models

This section discusses this project's efforts to validate the simulation. One of the challenges posed before was to create a valid simulation, or to identify causes of discrepancies between simulation behavior and output data that go beyond the scope of the modeling and simulation capability. From the simulation results shown in Figure 23 and Figure 24we can see that the number of violations in the simulated system is greater than the recorded radar data. In fact, this number of violations would in reality be considered unsafe. To statistically compare the simulation output with observed data from the real system, t-tests were performed between the metrics from the radar data and the simulated system. For more than half of the thirteen metrics collected for each sector the t-tests found significant differences between the outputs from the simulation and the radar data. Thus validation of the modeled system failed, both for the previous simulation that had unrealistic emergent behaviors in flight profiles and for that developed in this thesis.

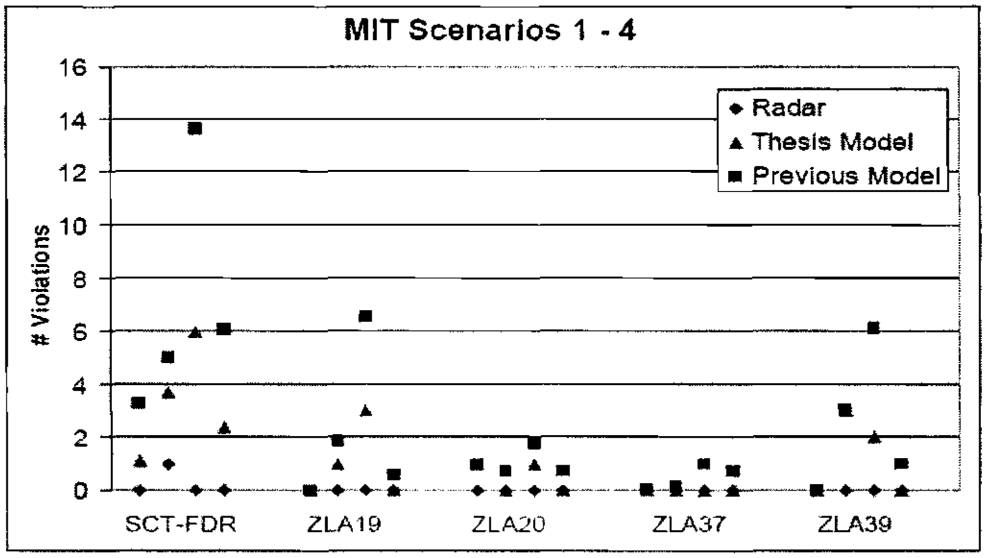

Figure 23: Comparison of average number of separation violations for simulations and the observed radar data for MIT scenarios

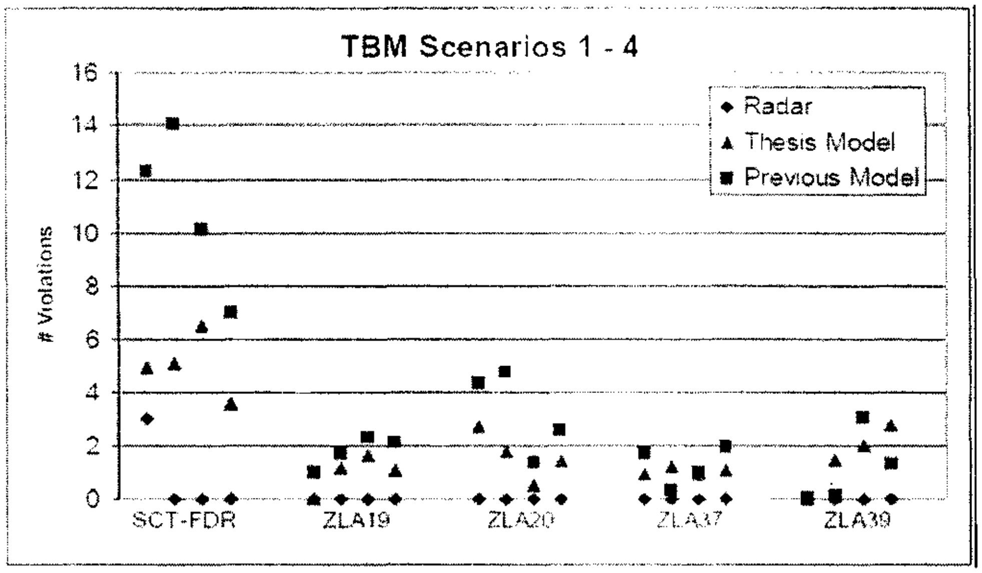

Figure 24: Comparison of average number of separation violations for simulations and the recorded radar data for TBM scenarios

As discussed early, there could be a number of causes of these failures:

1. *The input data used to configure the simulations could have errors or biases*: Most elements of the input data were crosschecked with a third party and found to be correct. However, with regards to the inputs provided for delay times associated with each aircraft in TBM operations, two problems were found. First, some of the times were sufficiently small that the aircraft could not meet them without an unrealistic increase in speed during a phase of flight where their speed should be reduced in preparation for landing. Second, some of the delay times required aircraft to get sufficiently close that they created separation violations. These delay times were provided by a third party as input data for these simulations. As such, it was impossible for this thesis' research to identify whether the problems with this input data reflect poor representation by the third party of the output of TMA system that generates them, or whether the TMA system itself is prone to these issues. However, a crosscheck by this third party with radar observations of the real system showed changes in the flight profiles of the aircraft found to have difficulty with delay times in the simulation, suggesting that controllers need to perform more than just the prescribed TBM work-processes for the delay times to be met.

2. *The individual models of the system components may not adequately represent aspects of behavior they were intended to represent*: Each individual component was therefore checked for adequate representation of intended aspects of behavior and was found to be valid for the given specifications. Most of these individual models had been validated during previous simulation developments. For example, the winds calculated by the simulated wind model were compared against radar data through the use of paired t-tests and were found to have similar distributions. The aircraft models were validated in terms of their ability to meet their waypoint restrictions both in terms of their spatial constraints and in terms of speeds, and their speed profiles had been verified. Furthermore, these models had previously been used in other successful research, thus lending further credibility to their correctness. Thus, there was reasonable confidence in the validity of individual models of system components.
3. *The intended aspects of component behavior described by the models may not have included some dynamics critical to emergent system behavior*: A few conversations with subject matter experts revealed that there may have been some un-modeled dynamics. Specifically, there is a significant level of communication and coordination between the controllers in the use of both MIT and TBM procedures, but this communication had not been specified when preparing the model specifications and therefore had not been modeled. In case of MIT they have letters of agreement between contiguous sectors that specify the exact miles-in-trail restriction which may not be fully represented in the model specifications. In case of TBM there is also evidence of significant coordination and communication amongst controllers to exercise the TBM restrictions. These explorations suggest the likelihood of un-modeled dynamics beyond those the models were intended to cover. Such behavior could be attributed to the creativity of and learning by air traffic controllers, or to commonplace but undocumented practices.
4. *The output data against which the system was being validated could be wrong or biased*: In addition, simulation runs identified some apparent biases in the data about real system behavior provided by a third party for simulation validation. Figure 25and Figure 26compare the average distance-in-sector validation measures with those recorded by the simulations. As the graphs show, for MIT there is negative bias in sectors ZLA19 and ZLA20 and positive bias in sectors ZLA37 and ZLA39. The measured values are

different by about 12 miles: radar data records a value of approximately 20 miles, but simulated data records a value of nearly 32 miles, an increase of 60%. Examining the profiles of the planned routes (profiles obtained by drawing straight lines through the sectors) it became evident that in reality the aircraft could not have taken such short routes. Likewise, examining the simulated aircraft profiles, most of the aircraft were not diverted by the controller from their flight plan; the few that were diverted did not divert so far as to increase the average distance of flight of all aircraft by 60%. As a further test, since the number of vectoring commands given in the simulation is significantly small, the average distance-in-sector for flight paths when there is no control by air traffic controllers (i.e., simulating whence the aircraft coast down their flight path) should dominate this measure. As shown in Figure 25 and Figure 26 the simulated data for both the previous and this thesis' simulation is dominated by the no-air-traffic-control scenario as expected. A similar issue manifested in the TBM validation data. Finally, the average total-flight-distance (i.e., the total distance flown by each aircraft through all sectors it traverses during arrival) is almost the same between simulations and validation models, suggesting the radar data intended for validation may have been systematically biased high in sectors ZLA37 and ZLA39, and low in sectors ZLA19 and ZLA20.

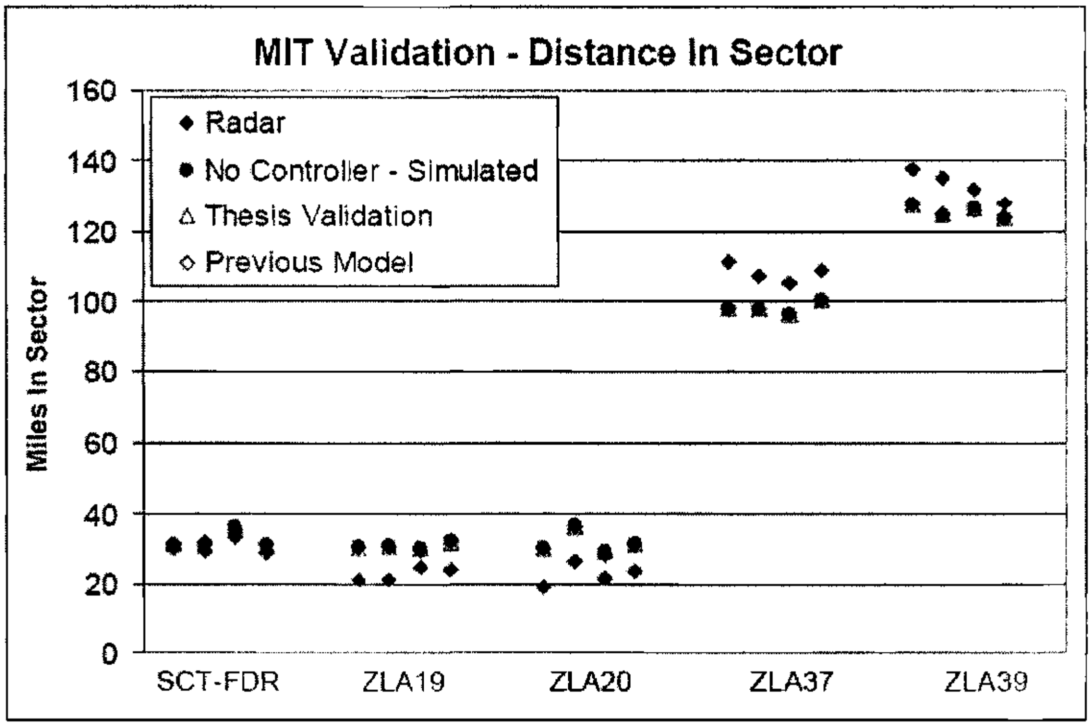

Figure 25: Bias in MIT validation data

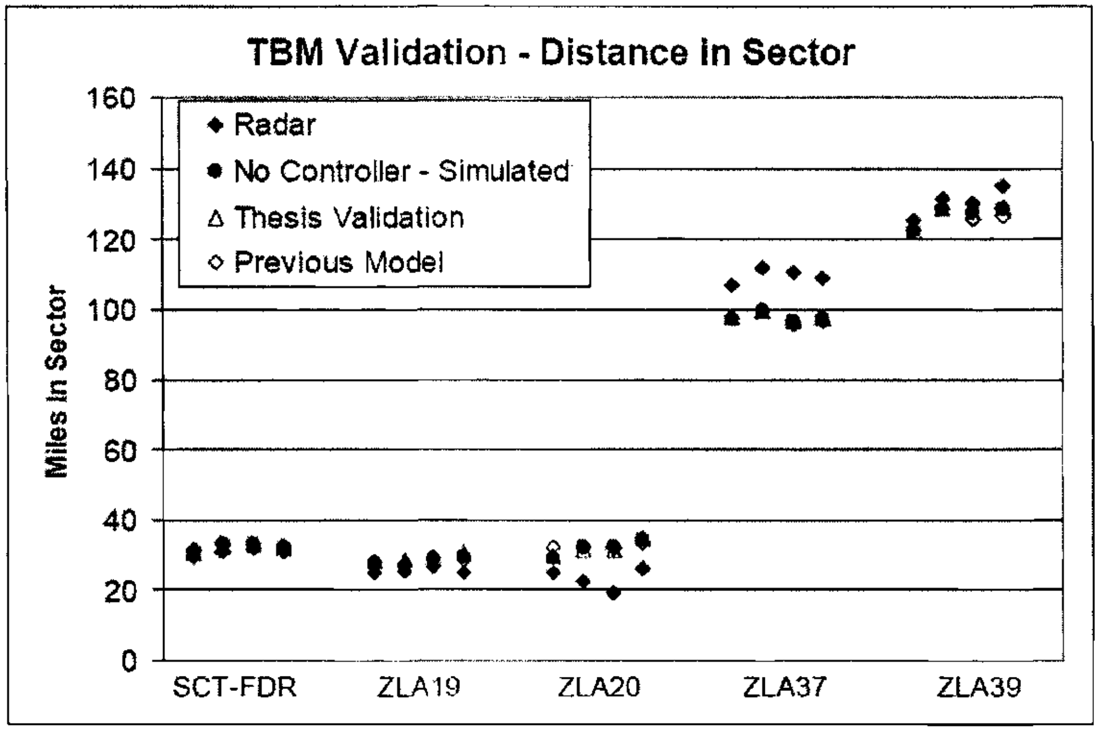

Figure 26: Bias in TBM validation data

Issues in Efficiency of Model Development and Risk Assessment

This section compares the modeling and simulation efforts in modeling the air traffic control system using this thesis' conceptual framework and simulation platform and the previous simulation. The previous modeling, simulation and analysis effort was spread out over a span of about one and half year, which is significantly large compared to this thesis' three and a half month effort. Table 5 compares the total efforts in modeling and in simulation and analysis. The modeling effort includes: conceptual modeling, development of computational models, troubleshooting, debugging and model verification. The simulation and analysis effort include: modification of models for testing transformations, configuring simulations, conducting simulations, and analyzing their results.

Table 5: Comparison of estimated modeling and analysis efforts (man-months)

	Earlier Phases	Final Phase
System Modeling	17.5	1.75
Simulation and Analysis	16	1

This final phase of the project definitely required significantly less time and effort, and was able to explore more model settings and system configurations than earlier. We mostly attribute this gain in efficiency to the ability to model the system in a structure-preserving manner, thus being able to make and test transformations much quicker. However, some exogenous factors should be noted. First, there was learning from the earlier phases. Second, some of the models and analysis capabilities developed in the earlier phases were available for this final phase's effort; attempts to exclude their development time from the time estimates for the previous simulation were conducted in good faith but required subjective assessments.

Summary

This demonstration highlighted the ability to model socio-technical systems in a structure-preserving manner, explain and predict their emergent behavior, and compare performance of different system design alternatives transformed through component, worker and network level changes. These developments provide detailed representations of system behavior and human performance suitable for risk assessment. This demonstration employed a case study in modeling and simulating the air traffic control system at the Los Angeles International Airport (LAX). Using this conceptual framework and simulation platform it is possible to transform a system at each of the component, network and worker levels. Likewise, there are significant gains in ease and efficiency in modeling and simulating a socio-technical system using the conceptual framework and simulation platform developed in this project.

CONTRIBUTIONS

The primary, intended contributions of this research focused on expanding the application and feasibility of agent-based simulation of the NAS for safety analysis. In addition, as shown below in Table 1, these developments have several likely side effects for theory and for practice.

Likewise, these research efforts are intended to support safety analysis of the NAS using agent-based simulation. However, these developments may also have broader applicability. For example, other data sets describing NAS behavior may be available which can be analyzed or data-mined using the same data analysis methods as described here. The use of agent-based simulation to describe and analyze other aspects of NAS performance may be demonstrated. Likewise, these mechanisms and statistical methods may also be applicable to other large-scale, distributed systems.

Table 1: Primary Contributions and Likely Side Effects of the Proposed Effort

Primary Contributions	Likely Side Effects
Use of agent-based simulation, integrating human performance models into a larger simulated environmental context, with output suitable for guiding subsequent hazard analysis and risk assessment processes.	Improved human performance modeling due to the ability to place human behavior in a larger environment context with interactions with other agents. Improved methods of representing the environmental context of and interactions between agents in large-scale simulations. Improved methods of timing agent interactions within and across processors, for better computational efficiency and tighter integration with the temporal dynamics within complex agent models.
Use of agent-based simulation in a multi-stage process which progressively streamlines and focuses the simulations on situations and conditions requiring risk assessment, resulting in estimates of the impact of identified hazards.	Procedures for rare-event analysis, suitable for data-mining other similar 'messy' data sets Compact models of high-level NAS behavior suitable for high number simulation runs for fast-time analyses.

REFERENCES

[1] Odoni, A. R., et al. 1997. Existing and required modeling capabilities for evaluating ATM systems and concepts. Technical Report, MIT International Center for Air Transportation.

[2] Jim, H. K. and Z. Y. Chang. 1998. An airport passenger terminal simulator: A planning and design tool. *Simulation Practice and Theory* 6: 387–396.

[3] Corker, K.M. and G. Pisanich. 1998. Cognitive Performance For Multiple Operators In Complex Dynamic Airspace Systems: Computational Representation And Empirical Analyses. *Proceedings of the 1998 42nd Annual Meeting Human Factors and Ergonomics Society*, Santa Monica, CA.

[4] Stevens, B.L, and F. Lewis. 1992. *Aircraft Control and Simulation*, John Wiley, New York.

[5] Ippolito, C.A. and Pritchett, A.R. 2000. Software architecture for a Reconfigurable Flight Simulator. *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, Denver, CO.

[6] Pritchett, A. R., S. M. Lee, and D. Goldsman. 2001. Hybrid-system simulation for safety analysis of the National Airspace System. *Journal of Aircraft*, 38(5): 835-841.

[7] Pritchett, A.R., S.M. Lee, K.C. Corker, M.A. Abkin, T.R. Reynolds, G. Gosling and A.Z. Gilgur, October 2002. Examining Air Transportation Safety Issues Through Agent-Based Simulation Incorporating Human Performance Models, *Proceedings of the IEEE/AIAA 21st Digital Avionics Systems Conference*, Irvine, CA.

[8] Kuhl, F., R. Weatherly, et al. 1999. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture for Simulation*, Prentice Hall.

[9] Bodoh, D. J. and F. Wieland, 2001. Self-Federating an Aviation Simulation Using HLA: Is it Feasible? *Proceedings of the Fifth Workshop on Distributed Simulation and Real-Time Applications*: 84-91.

[10] Corker, K.M., Man-machine integration design and analysis system (MIDAS) applied to a computer-based procedure-aiding system, in *38th Annual Meeting of the Human Factors and Ergonomics Society*, Santa Monica, CA: Human Factors and Ergonomics Society, 1994.

[11] Laughery, K.R. and Corker, K.M., Computer modeling and simulation of human/system performance, in *Handbook of Human Factors*, G. Salvendy, Editor, New York, John Wiley, 1997

[12] C.A. Ippolito and A.R. Pritchett, Software Architecture for a Reconfigurable Flight Simulator, *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, Denver, CO, August 2000.

[13] Farley, T., J. D. Foster, et al. (2001). A Time-Based Approach to Metering Arrival Traffic to Philadelphia. First AIAA Aircraft Technology, Integration, and Operations Forum, Los Angeles, CA.

[14] Mann, J., C. Stevenson, et al. (2002). Introducing Time-Based Metering at Los Angeles Air Route Traffic Control Center. Available: http://www.faa.gov/programs/oep/v7/library/Technologies/1/TMA-ATCA%2008-23-02%20km.pdf. Accessed: May 2005.

[15] FAA (2005). National Airspace System Operational Evolution Plan 2005-2015: Executive Summary. Available: http://www.faa.gov/programs/oep/v7/Executive%20Summary/Executive%20Summary%20v7.pdf. Accessed: April 2005.