ISSN:2231-0711
www.ijcset.net

Rivet on Disseminated Problem Solving Techniques in Airtraffic Flow Management

M.Hema Madhuri[#1], Divya Vadlamudi[#1], Movva N.V Kiran Babu[#2], Anusha Kurra[#3]

[#1]*Department of CSE, K L University, Andhrapradesh*

[#2]*Department of CSE, Mother Theresa Institute of Science &Technology,Sathupally,AP.*

[#3]*Department of CSE, QIS college of Engineering & Technology,Ongole*

[#1]*hemamadhuri@live.com*, [#2]*divya.movva@kluniversity.in* [#3]*kiranbabuonline@yahoo.co.in*, [#4]*anusha1264@gmail.com* ,

Abstract:-Air traffic control (ATC) is a service provided by ground-based controllers who direct aircraft on the ground and in the air. The primary purpose of ATC systems worldwide is to separate aircraft to prevent collisions, to organize and expedite the flow of traffic, and to provide information and other support for pilots when able. The geographical and functional distribution and the highly dynamic nature of air traffic control (ATC) make it an ideal candidate with many potential applications that can be modeled in distributed environment, but they increased the complexity of the systems. Hence it leads to some difficulties in problem solving ATC.In this paper we adress those problem solving difficulties.Finally we discuss some problem solving techniques that can overcome these difficulties based in Agent technology.

Key words: Air traffic control, ATC management, Agents,

I. INTRODUCTION

Agent can be defined to be independent, problem solving computational entities capable of effective operation in dynamic and open environment[1] .agents are often deployed in environment in which they interact, and work together with other agent including people and software they have possibly contradictory aims such environments are known as the multi agent systems .agents can operate without the direct involvement of humans and others.[2] Agent can be used as a design symbol for designers and developers in the way of structuring an application around autonomous, communicative elements, and elements, and lead to the constructing of software tools and infrastructure to support the design symbol. In this sense, they offer a new and often more appropriate route to the development of complex systems, especially in open and changing environments. Agent technologies span a range of specific techniques and algorithm for dealing with interactions with others in changing and open environment. These include issues such as balancing reaction. Agent-based systems are one of the most effective and important areas in research and development to have emerged in information technology in the 1990s.[3]an agent is a computer system that is capable of flexible autonomous action in dynamic, unpredictable, typically multiage domains the characteristics of dynamic and open environments in which, for example, heterogeneous systems must interact, span organizational boundaries, and operate effectively within rapidly changing circumstances and with dramatically increasing quantities of available information, suggest that improvements on traditional computing models and paradigms are required .thus the need for some degree of autonomy , to allow components to respond dynamically to changing circumstances while trying to achieve over-arching objectives, is seen by many as fundamental. Many observers therefore believe that an agent represents the most important new paradigm for software development since object orientation. The concept of an agent has found wide range of sub-disciplines of information technology, including computer networks, software engineering, artificial intelligence, human-computer interaction, distributed and concurrent systems, mobile systems, telematics, computer-supported cooperative work, control systems, decision support, information retrieval and management, and electronic commerce. In practical developments, web services, for example, now offer fundamentally new ways of doing business through a set of standardized tools, and support a service-oriented view of different and independent software components interacting to provide valuable functionality. In the context of such developments, agent technologies have gradually come to the foreground. Because of its horizontal nature, it is likely that the successful adoption of agent technology will have a profound, long-term impact both on the competiveness and capability of IT industries, and on the way in which future computer systems will be conceptualized and implemented. Many researchers and programmers see agents as programs roaming a network to collect business-related data in order help users to buy goods, or implement platform independent code-on-demand, for example this need for mobile agents is acknowledged, and builds on European strengths, but mobility brings added security problems. The research effort concentrates on how to guarantee termination, security or exactly-once protocols. To protect against malicious hosts, agents should contain time limit validity, and electronic money with an expiration date. A key issue that needs to be addressed here is administrability of mobile agent systems, e.g., authorization policies; this has been a major reason why mobile agents have not yet been taken up by the mainstream. Note also that hosts need to be protected as well as agents End users already encounter the situation that, while ample bandwidth is available on the backbones of network service providers, their experience is limited by the constraints of the infamous last mile.

Mobile agents may improve the end user experience by offloading application-specific filtering, media adaptation, and other pre-processing to a node with high bandwidth connectivity. This is particularly interesting for mobile phones and portable devices. One of the commercial application areas in which the added value of mobile agents is very high, is large-scale distributed or decentralized system integration with highly adaptive and dynamic business logic. Existing solutions are generally centralized, pulling everything onto one platform, limiting the complexity and changes that can be handled. A decentralized agent approach divides and conquers complexity by pushing a large part of the business logic out onto source systems so that much monitoring and aggregation can be done on each. This distributes workload and increases robustness because the local processing can be performed independently of other systems, resulting in fewer and more relevant interactions with these systems, at a higher level of abstraction. In turn, mobility, mainly single hop, is the answer to the increasing need for flexibility and adaptability in business logic. Agents can easily be deployed to source systems, carrying new database drivers, code to interact with new application or file types, or new data processing rules. Software is updated at the component-level, at runtime, proving a level of dynamism and flexibility that goes far beyond current release policies. Agent communication and behavior capabilities complete the picture, being very well suited to high-level service-based Interactions, the decentralized implementation of business logic, and for adapting and handling change in their environment. A nice property of the dynamic, component-level approach is that it naturally fits step-by-step system integration, with each step resulting in added value for the business. This is a particularly significant advantage in the current economic climate, in which many companies have seen mega-projects fail. For example, Global IDs Inc in the US offers a next-generation product suite for data integration based on the Tryllian mobile agent platform. Their data integration products are capable of simultaneously monitoring many hundreds of enterprise systems for relevant changes in data or metadata, by deploying mobile agents onto those systems. The agents tap into local databases or applications, keep track of changes, can pre-process data and only forward relevant events or structured derived data to centralized collectors – in real time if required. The mobility of the agents allows highly customized functionality, which can be dynamically updated. Thus, the business user can change the business rules that are being executed at any point in time, while only relevant drivers and adapters are transferred to a source system. Agents can assess the impact of changes in the business rules and handle that impact throughout the integration process.

DISTRIBUTED PROBLEM SOLVING DIFFICULTIES

To understand the difficulties facing groups solving problems it helps to note some important characteristics of distributed problems. The following traits are true of a wide variety of group situations [4]:

Most situations consist of a collection of agents, each with various *skills*, including sensing, communication (often over limited- bandwidth channels), planning, and acting.

- The group as a whole has a set of assigned *tasks.* As in single agent problem-solving situations, these tasks may need to be decomposed into subtasks, not all of which may be logically independent. The group must somehow assign subtasks to appropriate agents.
- Typically each agent has only *limited knowledge.* An agent may be subject to several kinds of limitations: limited Knowledge of the environment (e.g., because of restricted sensing horizons), limited knowledge of the tasks of the Group or limited knowledge of the intentions of other agents.
- There are often shared *limited resources* with which each agent can attack tasks. For example, if the agents are in a blocks world environment, the shared resources are the blocks out of which their constructions must be made.
- Agents typically have differing *appropriateness* for a given task. The appropriateness of an agent for a task is a function of how well the agent's skills match the expertise required to do the task, the extent to which its limited Knowledge is adequate for the task, and current processing resources of the agent.

Several kinds of distributed problem-solving difficulties follow from this characterization.

First, there are difficulties with *optimal task assignment.* Many mappings of decomposed subtasks to agents are possible, but, because agents have differing appropriateness with respect to a given task, only a few agents will be acceptable for any task. In many distributed problems it is crucial for agents to adopt the right *role.* In addition to insuring that any given task is assigned to an appropriate agent, the group has to achieve *task coverage.* All subtasks should be assigned to some agent (complete role assignment) and multiple unnecessary agents should not be assigned a task (consistent role assignment). The limited knowledge of agents compounds the difficulty of optimal task assignment and task coverage. Incomplete knowledge prevents consistent and complete role assignment because no one agent may have a global knowledge of all the roles or subtasks that need to be assigned. Optimal task assignment is also threatened since agents may not know about tasks for which they are the most appropriate.

Second, *task coordination problems* arise because tasks assigned to agents may not be independent. For example, if two Blocks world agents are each to build towers (as subtasks of a larger task), the plan that one agent produces might negatively interact with the plan of another if both intended to use the same block. While single-agent problem solvers have difficulties in handling Non-independent tasks or sub goals, these difficulties multiply for distributed problem solvers. Again, limited knowledge is the reason. If two agents have only local knowledge—if they know only the local environment, know only their tasks and intentions then they will not know of, or be able to prevent, negative interactions between their roles and those of other agents. In summary, a main challenge to distributed problem solving is that the solutions which a distributed agent produces must Not only be locally acceptable, achieving the assigned tasks, but also they must be interfaced correctly with the actions of other Agents solving dependent tasks. The solutions must not only be reasonable with respect to the local task, they must be *globally Coherent* and this global coherence must be achieved by *local computation alone*

DISTRIBUTED PROBLEM SOLVING IN AIR-TRAFFIC CONTROL

Problem solving in air-traffic control may be distributed in several ways elsewhere we discuss a variety of architectures of distribution. Currently we have implemented only *object-centered* systems, where one agent is associated with each aircraft. In our air-traffic control task, aircraft enter rectangular (14 x 23 mile) airspace at any time, either at one of 10 infixes on the borders of the airspace, or from one of two airports. The main goal of the agent associated with each aircraft is to traverse the airspace to an assigned destination—either a boundary out fix, or an airport. Each aircraft has only a limited sensory horizon, hence its knowledge of the world is never complete and it must continually gather information as it moves through the airspace. Information may be accumulated either by sensing or communication. Agents are allowed to communicate over a limited bandwidth channel to other aircraft for purposes of exchanging information and instructions

Distributed ATC is a group problem not only because agents may help one another gather information but also because the goals Of one agent may interact with those of another Coal interactions come in the form of shared conflicts A conflict between two or More agents arises when, according to their current plans, the two will violate minimum separation requirements at some point in the future. When shared conflicts arise, agents must negotiate to solve them. In a crowded airspace, such goal conflicts can get particularly complex, and involve several aircraft, thus necessitating a high degree of group cooperation

AIR TRAFFICFLOW MANAGEMENT

Air traffic control (ATC) is a service that provides safe, orderly, and efficient flow of aircraft operating within airspace. Generally, an Air Traffic Service Provider (ATSP) is the authority responsible for providing air traffic control; in the U.S.A., the FAA is the ASTP for the National Airspace System (NAS). The FAA has four types of facilities that participate in ATC. ATC towers manage the aircraft arriving, departing, and taxiing on the ground. Terminal Radar Approach Control Facilities (TRACONs) control airspace approximately within thirty miles of an airport. Air Route Traffic Control Centers (ARTCCs) are responsible for the remainder of controlled airspace in the NAS. Our interest in ATFM is primarily at this level, which consists mostly of “en route” traffic flying on instrument flight rules (meaning they rely on instrumentation and ATSP guidance). En route traffic usually follows predefined air routes (essentially “sky highways”) in order to increase the predictability of the traffic flow. There are twenty such ARTCCs in the continental United States, and each ARTCC is further subdivided into sectors. At the national level, the Air Traffic Control System Command Center (ATCSCC) develops strategic plans for traffic flow Management throughout the NAS. It has final approval of all national traffic management initiatives (TMIs) and is responsible for resolving inter-facility issues.

Air Traffic Flow Management (ATFM) is a system level function within ATC to manage the flow based on capacity and demand. ATFM is the responsibility of a Traffic Management Unit (TMU) within each ARTCC and at the ATCSCC. The ATCSCC develops strategic plans to ensure balanced flow throughout the NAS over a planning horizon of two to eight hours. The center TMUs develops tactical plans to manage air traffic within their local airspace over a planning horizon of up to two hours that are consistent with any relevant ATCSCC initiatives. The TMUs constantly monitor for potential conditions that could reduce airspace capacity such as adverse weather and for excessive traffic demand that could overload a sector controller’s ability to safely handle traffic. For example, a TMU may identify a Flow Constrained Area where a capacity-demand imbalance may occur due to severe convective weather. The TMU would then analyze which type of traffic management initiative should be invoked to alleviate the traffic imbalance. Since TMIs may affect adjacent centers, either directly or through ripple effects, ATCSCC approval is needed before invoking a TMI. TFM issues are discussed in a bi-hourly planning teleconference, involving representatives from the ATCSCC, each ARTCC, and airlines A variety of TMIs are available to the FAA, depending on the nature of the traffic flow problem; we describe some commonly used TMIs. A re-route procedure directs an aircraft onto a new route to avoid a problem area, such as a severe thunderstorm or congested airspace. This is the only TMI we have implemented in our current simulation. Re-routing can impact both ATC and NAS users: workload of the sector controllers receiving the diverted traffic increases, expected aircraft arrival times may change, and more fuel may be needed for the aircraft to follow the new route. A Ground Delay Program (GDP) delays aircraft at the departure airport in order to manage the demand at the arrival airport. Flights are assigned new (delayed) departure times, thus changing their expected arrival time at the impacted airport. GDPs are implemented when capacity at an arrival airport has been reduced for a sustained period, due to weather or

Excessive demand. Miles-in-Trail (MIT) restrictions enforce a certain separation between aircraft transiting through some point (e.g., an airport, sector boundary, or route). MITs are used to apportion traffic into a manageable flow, as well as provide space for addition traffic (merging or departing) to enter the flow of traffic.

A MIT procedure can cause the traffic flow to back up, potentially resulting in a larger MIT restriction in the upstream center (known as a pass back) or delayed departures. Airlines manage their fleet of aircraft in an Airline Operations Center (AOC). The AOC typically has a position called the ATC coordinator that monitors the TMIs issued by the TMUs and ATCSCC and participates in the planning teleconference to make their concerns visible to the FAA. A major thrust of the CATFM concept is to increase the role of the AOC in ATFM

Agent-based ATFM Simulations

The Airspace Concept Evaluation System (ACES)[6]is a distributed agent-based simulation of the NAS that uses a "activity centric paradigm". ACES support the Department of Defense's High Level Architecture (HLA), which has enabled the integration of several simulations into the overall system. As ACES is focused on the entire NAS, the simulation includes ATFM [7], but is not its only focus. In ACES, individual reasoning entities are represented as agents, as well as the different simulation layers connected through HLA.

IMPACT (Intelligent agent-based Model for Policy Analysis and of Collaborative Traffic flow management) is a swarm-based agent model of FAA agents and airline agents, simulating several possible responses to capacity reductions: no TMIs, GDPs without information sharing, and GDPs with shared airline schedules [8]. In each scenario, the FAA agents decide to impose GDPs or not based on predefined policies, and the airline agents choose actions that minimize the estimated cost to their operations. As expected, their simulation measured the best performance when schedule information was shared, but surprisingly found that GDPs without shared information (as occurs in today's operations) resulted in a *greater* average cost per flight than when no TMIs were instituted.

STEAM [9] has been used to model a collaborative system for real-time traffic synchronization [10] Real-time traffic synchronization is the work of the individual sector controllers as they manage flights that run through multiple sectors, and is complementary to our focus on the traffic flow level. Unlike our model, where the collaboration also includes the airspace users, only the sector controllers and a few higher-level coordinating entities coordinate to find solutions to the ATFM problems. The Man-Machine Integrated Design and Analysis System (MIDAS) is an agent-based model of human performance when coupled with machine interfaces. MIDAS has been applied to ATFM[11][, but at a different level of granularity than in our work, as it emphasizes the capabilities and limitations of human cognitive ability instead of complex decision making at a more abstract level.

CONCLUSION:

In this paper we addressed the problem solving issues in ATC and problem solving twchniques based on agent technology. Finally we conclude that, Although some Existing techniques offers a number of advantages, such as decentralization and collaboration in ATC, it also increases the complexity of the system in distributed environment.Research has shown that there is a need to solve this complexity and problem solving issues in ATC with proper ATC management and Agent technology is the best option to solve these problems in distributed environment.

ACKNOWLEDGEMENTS

We like to express our gratitude to all those who gave us the possibility to carry out the paper. We would like to thank Mr.K.Satyanarayana, chancellor of K.L.University, Dr.K.Raja Sekhara Rao, Dean, and K.L.University for stimulating suggestions and encouragement. We have further more to thank Prof.S.Venkateswarlu, Dr.K.Subrahmanyam, who encouraged us to go ahead with this paper.

REFERENCE:

[1] Agent Technology: Enabling Next Generation Computing: A Roadmap for Agent-Based Computing by Michael Luck, Peter McBurney and Chris Priest January 2003, Version 1.0, ISBN 0854 327886

[2] Bo Chen, Member, IEEE, and Harry H. Cheng, Senior Member, IEEE," A Review of the Applications of Agent Technology in Traffic and Transportation Systems" IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 11, NO. 2, JUNE 2010.

[3] Agent Technology: Computing as Interaction: A Roadmap for Agent Based Computing by Michael Luck, Peter McBurney, Onn Shehory, Steve Willmott and the AgentLink Community, © September 2005, AgentLink III ISBN 085432 845 9

[4]S. Cammarata, D. McArthur, and R. Steeb, "Strategies of cooperation indistributed problem solving," in Proc. 8th Int. Joint Conf. Artif. Intell.,1983, pp. 767–770.

[5]K. Tumer and A. Agogino, "Distributed agent-based air traffic flow management," in Proc. Int. Conf. Auton. Agents, 2007, pp. 342–349

[6]Sweet, D. N., Manikonda, V., Aronson, J. S., Roth, K., & Blake, M. (2002, August). Fast-Time Simulation System for Analysis of Advanced Air Transportation Concepts. Paper presented at the AIAA Modeling and Simulation Technologies Conference and Exhibit, Monterey, California

[7]Couluris, G. J., Hunter, C. G., Blake, M., Roth, K., Sweet, D. N., Stassart, P. A., et al. (2003, August). NationalAirspace System Simulation Capturing the Interactions of Air Traffic Management and Flight Trajectories. Paperpresented at the AIAA Guidance, Navigation, and Control (GNC) Conference, Austin, Texas

[8]Keith C, C., Cooper, W. W., Greenbaum, D. P., & Wojcik, L. A. (2000). Modeling Distributed Human Decision- Making in Traffic Flow Management Operations. Paper presented at the 3rd USA/Europe Air Traffic Management R&D Seminar

[9]Tambe, M. (1997). Agent architectures for flexible, practical teamwork. Paper presented at the American Association for Artificial Intelligence Conference (AAAI-2007).

[10]Nguyen-Duc, M., Briot, J.-P., Drogoul, A., & Duong, V. (2003, October 13-17). An application of Multi-Agent Coordination Techniques in Air Traffic Management. Paper presented at the 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), Halifax, Canada.

[11]Corker, K. M. (1999). Human Performance Simulation in the Analysis of Advanced Air Traffic Management. Paper presented at the I999 Winter Simulation Conference