Aerogility - An intelligent decision support system for managing service-based aftermarkets

Viet Dung Dang
Lost Wax
Avalon House, 72 Lower Mortlake Road
Richmond, Surrey TW9 2JY
United Kingdom
v.d.dang@lostwax.com

Steve Osborn
Lost Wax
Avalon House, 72 Lower Mortlake Road
Richmond, Surrey TW9 2JY
United Kingdom
steveo@lostwax.com

Gary Vickers
Lost Wax
Avalon House, 72 Lower Mortlake Road
Richmond, Surrey TW9 2JY
United Kingdom
garyv@lostwax.com

Malcolm Bridgeford
Lost Wax
Avalon House, 72 Lower Mortlake Road
Richmond, Surrey TW9 2JY
United Kingdom
malcolmb@lostwax.com

ABSTRACT

This paper describes Aerogility - an intelligent decision support system for managing service-based aftermarkets.
Aerogility uses multi-agent software technology to implement a dynamic model of an aerospace aftermarket. The model is realistic and includes all the assets, people and processes in a dynamic and interactive representation of the business. An initial model can be implemented quickly then enhanced and extended in stages. Individual stakeholders can contribute their understanding of the business to their part of the model. The operational parameters and business rules that drive the behaviour of the model can be easily changed. A management team can work through a variety of what-if scenarios and comparative simulations to evaluate options and policy innovations. Each scenario can be benchmarked with various analytical information that facilitates detailed analysis and powerful insight.

Categories and Subject Descriptors

INDUSTRIAL SOFTWARE [**Industrial and military applications**]

Keywords

Aerospace, Aftermarket, Decision support system, Multi-agent system.

1. INTRODUCTION

Maximising fleet availability and Maintenance Repair Overhaul (MRO) utilisation, while minimising costs, is a major pressure in today's global aerospace and defense industry. Fleet operators are driving this trend and demanding Total Care, Availability-contracts and Performance Based Logistics, transferring the operational and financial risks back to the product supplier.

In this model, servicing and maintenance becomes a key driver of long term profitability. The large number of present and future variables that influence service levels and aftermarket economics make planning and operating an aftermarket an imprecise art. Managers frequently have to make difficult and expensive decisions in the face of these complexities. These decisions make or break profitability.

Short, medium and long term questions and issues arise:

- Short term: these concern operational decisions, scheduling and policies. For example, how should engineering teams be deployed? how should the jobs be prioritised?
- Medium term: these relate to sourcing and partnering options. For example, what is the impact of different inventory levels? where should components and parts be located? should components and parts be refurbished?
- Long term: these regard investment evaluation and planning. For example, how many MRO facilities are needed for projected product sales? Where should the MRO facilities be located? What capabilities should an MRO facility support?

Aerogility provides managers with the decision support they need to answer these questions by enabling them to analyse and evaluate the complex balance of aftermarket resources to improve customer service and increase profits. Built on multi-agent technology, it enables business users to simulate the aftermarket in a dynamic and realistic way, try different what-if scenarios, gain powerful insights from analytical output and optimise the business strategies. The adaptability of multi-agent software technology also means that the model can be changed and improved easily - as new

Cite as: Aerogility – An intelligent Decision Support System for Managing Service-Based Aftermarkets, Viet Dung Dang, Steve Osborn, Gary Vickers, Malcolm Bridgeford, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra andCastelfranchi(eds.),May,10–15,2009,Budapest,Hungary,pp.1411–1412

factors arise and assumptions change. The system can be evaluated by using historical data to project forward and compare the analytical result from the simulation with the actual data.

The next section will look at Aerogility's technical architecture.

2. AEROGILITY ARCHITECTURE

Aerogility contains the following main components: Aerogility Tools, Model Executor, Customer Model and Lost Wax Agent Framework.

2.1 Aerogility Tools

Aerogility User Tools is implemented as a Web application running on a standard Java EE Web application server. It enables the user to configure agents and objects in the system, manage scenarios, analyse the results and optimise the scenarios.

2.1.1 Agent Configuration

This tool allows the user to create and configure agents and objects in the system. The configuration data for the agents and objects are used at run-time to determine the behaviour of the model, for example, inventory levels, component costs or process timing assumptions. As Aerogility incorporates a rule engine, agent behaviour can be configured either through setting of various parameters and tables or by specifying business logic through a rule-based interface. The latter allows the user to express business rules in an English-like domain-specific language.

2.1.2 Scenario Management

The user can use the scenario management tool to organise the agents into scenarios as well as manage and run them in any timeframe - this can range from hours to decades, singly or in batch mode. He or she thus can explore different business policies, operational configurations and process innovations.

2.1.3 Analytics

The Analytics system takes output from scenario runs and displays it in a rich set of formats, such as Key Performance Indicators (KPIs), tables, reports, charts etc. The output from two scenarios can be displayed simultaneously to enable one to be compared against another.

2.1.4 Optimisation

This provides the user with the ability to optimise the scenarios, i.e. determine the best set of values to maximise, minimise or reach a set of objectives when the model is run. Different optimisation techniques can be incorporated within the architecture. The current release uses an exhaustive search approach; however, hill climbing and genetic algorithms approaches have also been developed to be deployed in a future release.

2.2 Model Executor

The Model Executor executes scenario runs submitted by the user via the Scenario Management tool. The executor allows various configuration options, such as run mode and whether to show the run-time User Interface (UI).

There are two run modes: regular demo time and fast execution:

- In the regular demo time mode, the model runs in a user-configured simulation time. This is useful if the user wants to look at the run-time UI and see the events as they happen or for debugging purposes.
- In the fast execution mode, the model runs as fast as it can, skipping the time when no event occurs. This is the more frequently used mode, as the user usually submits multiple runs then look at the Analytics output to analyse the results and revise the strategies.

The run-time UI usually contains three main parts:

- World View: this shows the model world with all the agents on their respective location and some visual indication of their status.
- Agent List: this lists all agents and objects in the system alphabetically for ease of access to agent details.
- Various Agent Detail windows that show the internal details of the agents.

2.3 Customer Model

A customer's model is developed in a series of iterations to represent accurately the aftermarket processes and customer requirements. A key benefit of the underlying multi-agent architecture is flexibility. Any element of the aftermarket represented in the model can be easily added, modified or deleted using the above configuration tools. Thus the business user is able to gain maximum benefit from the model, without requiring the support of specialist development resource. When a sufficiently realistic model has been implemented, the business user is able to change any of the operating parameters used by the model and create different scenarios.

2.4 Lost Wax Agent Framework

Aerogility is implemented on top of the Lost Wax Agent Framework. The Agent Framework is implemented in Java and provides the core framework required to implement multi-agent systems, as well as an agent library to support their development:

- The core framework provides containers in which agents run and the management facilities for agent execution, including naming, discovery, life cycle and messaging. It is lightweight and can support thousands of concurrent agents.
- The agent library provides a set of commonly used agents such as logistics, production centre, maintenance centre, warehouse agents etc. Thus a customer model can be quickly set up by reusing or extending the agents in the library.

3. THE DEMONSTRATION

The demonstration presents a generic model of the Aerospace aftermarket. It models aircrafts in details with various modules and components and a sophisticated network of airports, operators, manufacturing facilities, MRO facilities, warehouses and supply chains. Similar models have been implemented for leading companies in the aerospace sector.