Modelling the Organisation in CIS Applications

P.J. Wright

CSS1 Division
DRA Malvern,
St Andrews Road,
Malvern, Worcs. WR14 3PS.
U.K.

Tel: 0684 895489.

1. Introduction

The approach adopted by the future air battle management system studies at DRA Malvern is built upon the philosophy of "understanding the organisation in order to be able to define its computer requirements". The approach adopted to organisational analysis is Shlaer Mellor object-oriented analysis (OOA) [1,2].

OOA is used to model a generic CIS organisation in terms of the activities undertaken by the organisation, the tasks performed and the real-world entities required by the organisation. An operator domain has also been developed that models the human behaviour within the system, in terms of workload, operator efficiency and rôles and responsibilities.

An advanced CASE tool, SES Objectbench, is used to aid the OOA process. This also allows the models produced to be compiled and animated automatically. Scenarios are then defined to run against the C2 models and the outcome is analysed against a set of measures of effectiveness (MOEs). This modelling is undertaken in conjunction with an existing in-house modelling framework (the Total System Model) which provides an executable "environment" of the operational context. The progress of the model simulation is displayed using a graphics tool (Altia design). This provides a means of verifying the model's behaviour with military personnel present. The graphical interface also facilitates the interaction of operators with the model.

This paper presents organisational modelling using OOA and Objectbench, emphasising the generic aspects of the modelling concerning task analysis and operator simulation.

2. Object-oriented Analysis

A key step in building any object-oriented system is the analysis of the customer's requirements [3]. Analysis concentrates on the *what* of the system, rather than the *how*. The how or solution to the problem is addressed during the design phase.

In analysis the three commonly accepted orthogonal views of any systems are data, behaviour and function. These are listed in the relative order of importance placed upon them by the object-oriented approach to analysis. In general, traditional methods have ranked these in reverse order of importance (i.e. primarily functionally based (e.g. [4])).

The modelling techniques employed for these views, in Shlaer Mellor OOA, are not new. Extended entity-relationship modelling is employed to represent data as an *information model* (objects and their relationships). Each object is associated with a *behavioural model*, represented as a deterministic finite state machine [5]. Finally, function is depicted by familiar *action data flow diagrams* (ADFDs); each state in the state machine, for an object, is associated with a number of functions (data stores represent persistent object data in the ADFD). Whilst the notation employed is not generally new (this may be

viewed as a positive advantage), the consistent and complete integration of all three views is novel, providing a coherent and complete specification of the system requirements.

3. The Advantages of an OO Approach.

An important aspect of the object-oriented development life-cycle is the ability to analyse, design and implement various separate areas of the system in (partial) isolation, allowing a separation of concerns. This is performed as the first activity of Object-oriented Analysis. The problem area is split into *domains*. A domain is a separate real, or hypothetical, "world", populated by a distinct set of objects which behave according to the rules and policies characteristic of the domain. A domain represents an area of the overall problem space that can be considered in (partial) isolation and so should be representative of a universe of discourse. As an example we would split an air defence domain (in which the subject matter concerns air vehicles, targets, threats etc.) from its visualisation domain (in which we talk about plots, tracks etc.).

A domain chart is built for the overall system, with the identified domains shown in bubbles. Specialised domains are shown at the top of the chart, whilst highly reusable implementation domains appear at the bottom. Bridges are identified between domains. Bridges identify capabilities provided by one domain to another. Domains form client-server relationships with one another, implemented by the bridges, which are shown as extended arcs on the domain diagram.

Shlaer Mellor OOA provides a continuum in the development space, allowing the analysis to reflect the full detail of the final implementation, in terms of a number of OOA domains. This requires one complete model of the final system, that is kept up to date as system maintenance takes place. This facilitates future reuse of domains, objects and functions as well as allowing any re-engineering, extensions and improvements to be incorporated into the system by future development teams, with the minimum of effort.

OOA is a rigorous and complete engineering approach to modelling system requirements, producing analysis models that are executable. This eliminates many of the problems that tend to be associated with the implementation stage. In addition the models can be dynamically tested for data inconsistencies, dead and live-lock etc.

There are emerging CASE tools that support Shlaer Mellor OOA and allow automatic simulation of the analysis models. (Two such tools are Intelligent OOA from Kennedy Carter and SES/Objectbench). The ability to automatically "compile" the model and execute it derives directly from the rigorous semantics of Shlaer Mellor OOA.

With such automated CASE tool support, this not only allows extensive testing of the analysis models, but performance engineering requirements can also be incorporated into the analysis models at this stage. This allows performance issues (often a critical non-functional requirement) to be addressed at a much earlier stage than a traditional approach of fixing the final implementation.

Automated code generation has represented the Holy Grail of CASE tool research for the past decade. It has also proven to be as elusive a quest as seeking the Grail. However, with the advent of the object-oriented paradigm the situation has changed.

Because of the rigour incorporated into the analysis, fully automated code generation is now a reality, though for constrained implementation architectures. Fully tailorable implementations, involving hybrid architectures, are now actively being researched and should be available in commercial CASE technology in the very near future.

Modelling military systems often involves reasoning about sophisticated real-time systems, set in a particularly unconstrained environment. A high degree of concurrency is often desirable as a basis for modelling such systems. The object-oriented paradigm offers a natural model for such highly concurrent systems. Each object instance "runs" in parallel with all other instances. This enables the system designer to choose the grain of parallelism in the final system, ranging from object instance level, through object (multiple instance) to single threaded.

It is widely recognised that in developing large software systems, such as C2 models, it is very likely that the requirements driving the development life-cycle will prove to be unstable. It is also widely accepted that such requirement changes usually involve the functionality of the system, rather than the data upon which the functionality is based (i.e. data is stable, functionality is unstable to requirements change). The object-oriented paradigm is data centred, unlike the traditional functional decomposition methodologies and is therefore more stable to requirements change.

4. Objectbench

As has been outlined above, one of the most important advantages of using Shlaer Mellor OOA is the ability to analyse a concurrent systems, such as a C2 organisation, and build an analysis model that has all the necessary dynamic constructs and semantic rigour to compile and run. SES/Objectbench exploits this facet of OOA and is capable of automatically compiling and executing the analysis model. The C2 organisational study uses Objectbench in conjunction with a graphical front end tool (Altia) to produce working C2 models. The graphical tool allows an interface to be built that is intuitive to the non-technical user of the model, facilitating the verification and use of the model.

Altia also allows "man-in-the-loop" experiments to be performed, with human operators replacing one or more rôles that are simulated by the computer model.

5. Organisation modelling

The modelling effort at DRA Malvern has concentrated on producing a fully flexible generic C2 model. However, in order to proceed with a well grounded proof of concept, a current C2 organisation, the UK Air Defence Ground Environment (UKADGE), has been modelled.

The model itself concentrates on the control team that responsible for the near time planning and control of the air battle. This aspect of the command hierarchy provides a suitable basis for modelling generic C2 organisations.

The main structure of the OOA comprises three separate domains, the main "application" domain, where the various aspects of the C2 organisation are addressed and two service domains.

The service domains, that provide support to the application domain, are generic in nature. The first is the *operator domain*, this models human operators, fulfilling rôles and performing tasks. The domain has been modelled to capture the various facets of operator tasking, including work load aggregation, operator efficiency and efficacy, task priority and pre-emption. This modelling has been carried out largely in

isolation to the main application domain to provide maximum flexibility and extensibility. At the moment only simple workload modelling has been implemented, but it is envisaged that the domain may be easily extended to include sophisticated work load concepts, if this is required. The operator domain also captures the essence of the command hierarchy and so determines if a particular operator, fulfilling a certain rôle, has the permission to take on a new task that is offered. These permissions may be changed very easily, allowing for the re-definition of rôles within the organisation.

The second main service domain, the graphics domain, provides for the Altia graphical front end to the model. At present this displays the significant events within the simulation, a mission summary tote, graphical displays depicting operator work load and a display showing the operator command hierarchy and the currently active lines of communication. The domain is fully generic, allowing other graphical GUI building tools to be used and alternative displays of information from the model to be implemented.

The graphical front end also facilitates "man-in-the-loop" interaction. Certain tasks may be designated (in the external scenario data file) as either machine or human simulated. This serves two purposes: first it allows tasks to be modelled where human intuition, conjecture or experience is heavily relied upon. These tasks are inherently extremely difficult to model using state based automata. The model allows the operator to interact with the model at times when such tasks are performed. When the automatic option is selected for such tasks then a much simplified decision making process, based upon a simple automatic state based reasoning system, is executed by the simulation. This facility allows an operator, if necessary, to concentrate on a limited sub-set of tasks; secondly, operators can specify certain tasks to concentrate on in order to investigate the man-machine interface requirements for such operations.

The use of automatic or man-in-the-loop decision making facilitates sensitivity analysis of the model (and so identifies potential areas for further, higher fidelity modelling).

The main application domain captures the essence of the C2 system. It comprises a large number of objects (currently over one hundred) but in essence these consist of three types:

first, the real world, or *environmental* objects. These are the main tangible entities within a C2 system, such as air vehicle, threat, airfield, control unit etc.;

secondly, there is a class of objects that describe the main *tasks* of the system. Tasks are defined in such a way as to be a cohesive unit of work, that has a well defined outcome or objective. These include such tasks as scramble fighters, move a CAP position, order an engagement etc.;

thirdly, *activity* objects bind sets of tasks together. These form relationships with the environmental objects and so provide a framework for sets of related task objects to communicate. In general an activity meets some form of higher objective within the C2 system. Included in the model as activities are such objects as threat prioritisation, threat allocation, mission management etc.

The three canonical forms of object described above form the basis for an extensible and flexible framework to model any C2 system or organisation. The framework has been carefully designed to allow additional tasks, activities and environmental objects to be added with the minimum of impact on the rest of the model.

As part of the application domain a generalised model of areas of responsibility (A.O.R.s) has been developed. This allows A.O.R.s to be specified as arbitrary polygons for any particular command unit. This mechanism is employed to assign assets from a particular control unit to a specified threat. The concept

may be extended to assign any resource or asset to various controlling agencies, depending upon geographical location.

The resource elements of the scenarios that drive the model have been established in data files, separate from the main model. This allows the assets and other resources employed for any particular run to be readily changed.

In addition to this modelling a further domain has been developed to interact with an external model, the Total System Model (T.S.M.). This handles inter-process communication with the T.S.M. This domain has been especially tailored to achieve an efficient communication mechanism (i.e. performance considerations were important in developing this domain). The T.S.M. provides an external environment in which the C2 model can operate. It provides services to the main application domain, such as the ability to fly air vehicles, produce radar pictures etc. The T.S.M. also provides the capability to produce a real-time air picture. This further enhances the operator interface with the C2 model.

6. Summary

The precise execution semantics of object-oriented analysis (OOA), coupled with the power of automatic model compilation and animation offered by advanced CASE tools, provides a flexible modelling environment for the development of C2 models. The method has been proven for this use at DRA Malvern and a number of benefits over traditional code-centred modelling approaches have been observed:

first, the analysis models are always in step with the design and code, since the OOA is compiled by the CASE tool to produce the final executable;

secondly, the ability to free the analyst from detailed coding considerations allows greater effort to be placed upon the analysis stage of the problem. This reaps befits in capturing potential problems at an earlier stage of the life cycle;

thirdly, the ability to easily integrate a graphical front end onto the executable model facilitates both the inclusion of human operators within the system and allows the models to be validated by non-technical personnel.

References

[1] S. Shlaer and S.J. Mellor. "Object-Oriented Systems, Modelling the World in Data". Prentice-Hall U.S.A. 1988.

[2] S. Shlaer and S.J. Mellor. "Object Lifecycles, Modelling the World in States". Prentice-Hall U.S.A. 1992.

[3] A.J. Alston, P.J. Wright & D.W. Bustard. "COBRA - A CIS Object-oriented Business Requirements Analysis methodology". Proceedings of the first BCS Conference on the Integrative Aspects of Methodologies. Herriot-Watt University, Edinburgh, U.K. September 1993.

[4] T. DeMarco. "Structured Analysis and System Specification", Yourdon Press, 1979.

[5] E.F. Moore. "Gedanken-experiments on Sequential Machineso". Automata Studies. Princeton University Press. 1956.