Experimental Results from Integrating Planning Systems and Simulation Models

David Dyke, Mark Salt, Roberto Desimone
DERA Malvern,
St Andrews Road
Malvern WR14 3PS, UK
Tel: +44 1684 895447
dsdyke@dera.gov.uk

Peter Jarvis[†]
AI Applications Institute
80 South Bridge,
Edinburgh EH1 1HN, UK
Tel: +44 131 650 2732
Peter.Jarvis@ed.ac.uk

Abstract

This paper presents results from the implementation of a prototype based on the architecture for integrating planning and simulators proposed last year. The paper describes exactly what has been implemented and uses a non-combatant evacuation scenario (NEO) to show the benefits of such integration. Also discussed are the lessons learned from the results obtained and, finally, a description of a revised and improved architecture, which is believed will address some of the outstanding research issues.

1. Introduction

Large planning problems demand the co-operation and collaboration of many disparate and geographically dispersed experts. As part of the planning process, it is desirable to 'play out' fragments of the evolving plan, obtain and evaluate feedback, and compare alternative courses of action (COA). One way of achieving this is to use simulators to exercise the plan and provide feedback on the planning process by incorporating measures of effectiveness into plans. However, existing planning systems do not support collaborative planning well and have poor (or no) integration with simulation models and other services. In addition, simulators have been mostly developed in isolation from other systems.

This paper reports on continuing research within the UK Defence Evaluation and Research Agency (DERA) with the following key objectives:

- To explore the integration of planning systems with simulation models through the use of plan and simulation query languages providing guidance and validation of planning choices.
- To demonstrate the benefits of such integration in terms of more effective plans with greater accuracy, robustness and flexibility, hence accelerating the overall planning cycle.
- To make recommendations on improvements to description languages for plans and simulation models to better facilitate their integration.

Above all the integrated architecture should provide direct links between planning and command decisions made by military planners and validated simulation models that characterise and justify the military effectiveness for each decision, rather than just for the operations plan as a whole.

[†] Dr Jarvis now works at SRI International, Menlo Park, California

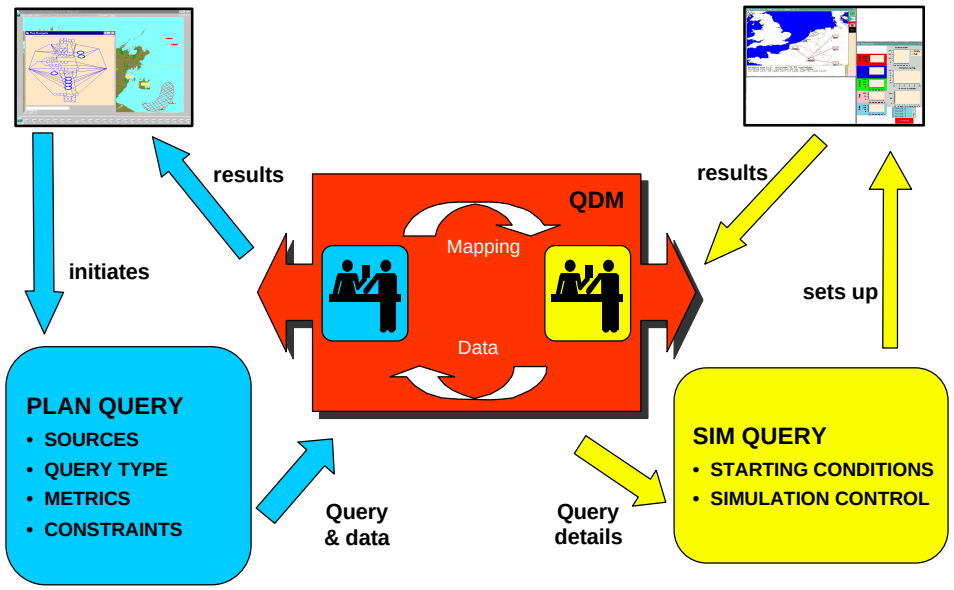

Figure 1. A high level view of the integrated planning & simulation systems architecture

This research started in April 1998 and will continue until April 2001 funded under the UK Ministry of Defence (MoD) Corporate Research Programme (CRP). Results of research from the first year of the project were reported last year and this paper presents results and lessons learned from the implementation of a prototype based on the proposed architecture for integrating planning and simulators. Leading directly from these initial results we present a revised and improved architecture which we believe addresses some of the main research issues.

2. Integrating planning systems and simulation models

A high level view of the concept is shown in Figure 1. A number of key research issues were presented last year [Desimone *et al*, 1999], which have been partially addressed in the current implementation in collaboration with University of Edinburgh[1]. In particular, the prototype now provides a direct link from initial plan query through simulator execution and feedback to the planner, thus achieving a complete loop that provides a loose-coupling between planning systems and simulation models via a query and data manager (QDM).

The current prototype comprises an initial implementation of the plan query language, the QDM and the simulation query language. The project has made use of an existing planning system (O-Plan) [Currie and Tate, 1991] from the University of Edinburgh and implemented specific simulators tailored to suit experimental purposes. Although the plan and simulator query languages are partially implemented, nevertheless, they are generic and not tied to specific planning or simulation systems.

[1] Artificial Intelligence Applications Institute (AIAI), University of Edinburgh

The plan query must contain all the information necessary for the QDM and the simulation architecture to prepare, run and return results from simulations. There are four types of information referred to in the query:

- **Data Sources** –point to information used by the planning system.
- **Query Type** – specifies the plan query and highlights the parts of the plan to be simulated.
- **Metrics** –provide control and feedback on the use of resources within the simulation.
- **Constraints** –place boundaries on the plan query to be satisfied by the simulation.

The plan query is converted by the QDM into a simulation query, which has two parts: starting conditions and simulation control commands. The starting conditions include instructions on how to populate the initialisation files for the simulations required. The control commands instruct the architecture on how best to simulate the required action whilst taking account of the metrics and constraints imposed by the planner.

The QDM provides the bridge between planners and simulators as shown in figure 2. In the prototype implementation the plan query manager takes the input plan query, parses it, feeds it to the simulation script matcher and retriever, where it is matched with a set of simulation queries from a database. The appropriate simulation query is retrieved and passed on to the simulation query enactor, which feeds the query to the relevant simulator. During execution, the QDM provides feedback that allows the planner to assess progress and hooks are in place to permit further refinement of the query dynamically. At the end of execution complete data from the simulator is compiled into a summary by the QDM and returned to the planner whereupon the original decision can be made and the planning process continues.

3. Implementation issues

The previous section outlines broadly what has been implemented in the prototype software. In this section we examine in detail each of the components of the implementation and consider in each case to what extent the component fulfils or matches the concept. Later we shall describe what is absent from the prototype and how the project will build on this in the next prototype.

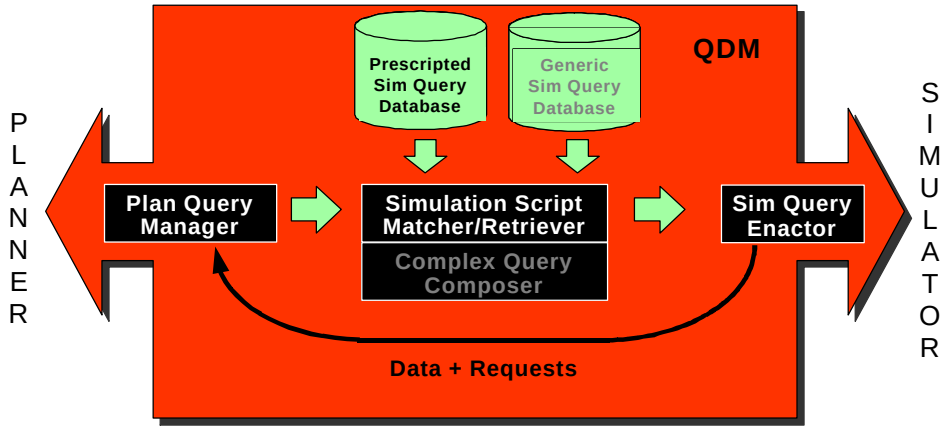

Figure 2. The QDM is the bridge between planners and simulators

Although O-Plan has been used as the planning system in this research we have deliberately worked to ensure that the coupling between O-Plan and the rest of the system is generic in nature and not driven by any features which are specific to the O-Plan system. Thus leaving it open in the future to link to other planning systems such as the *System for Interactive Planning Execution* (SIPE**)** [Wilkins, 1998], [Wilkins et al, 1995]. At present O-Plan, in common with any other candidate planning system, has no notion of a *plan query* and/or generating such information for external consumption. The solution to this is a free-standing interface module capable of translating the internal state of the planning system (O-Plan) into a generic Plan Query Language (PQL). The planner interface module should therefore be seen as an additional specific plug-in or extension necessary to allow any particular planner to generate generic plan queries and communicate with the rest of the system. In future systems it would be an integral part of purpose-built planning tools. In addition to generating and handling plan queries the interface module acts as the co-ordinating point for the results derived from execution of the plan query. These are posted dynamically and displayed textually and in graphical form. A compilation of the results is also posted after execution has completed.

Once a query has been formulated it can be submitted to a Query Data Manager service. To do this, it will identify and register with an instance of the QDM somewhere on the network. All subsequent queries will then be submitted for processing to that instance of the QDM. In this implementation multiple instances of the planner interface module and a QDM may co-exist on the network. At any one time, a planning system (via the interface module) may be connected to a single QDM, but any instance of QDM may receive queries from any number of planners anywhere on the network. Having received a plan query, the QDM manages the processing of it from acceptance through to delivery of the results. Currently, this can be an entirely automatic process or allow for manual intervention at any stage.

The QDM exists now as a single implementation module, this has been broken this down into a number of distinct components, each of which is described below and depicted in figure 3.

- **The Plan Query Manager**
  Newly arrived queries are stored in a queuing system and processed on a first-in-first-out basis. It is possible to browse any of the queries in the queue. At this point a query may be selected for rejection and a suitable message annotated to the reply explaining the reason for rejection. Otherwise the query is accepted and proceeds to next stage.

- **The Simulator Registry**
  A candidate simulator registers with a QDM on the network with a description set of its features. This allows the Matcher (below) to identify simulators (or more generally, services) capable of answering posted queries.

- **The Matcher**
  Accepted queries are matched to a simple database of simulator scripts capable of directing a simulator to carry out the specified tasks. Candidate scripts are matched to registered simulators. A combination of script and simulator must be selected before execution of the query can begin. The candidate scripts can be browsed before selection and in future implementations it is envisaged that dynamic modifications will be

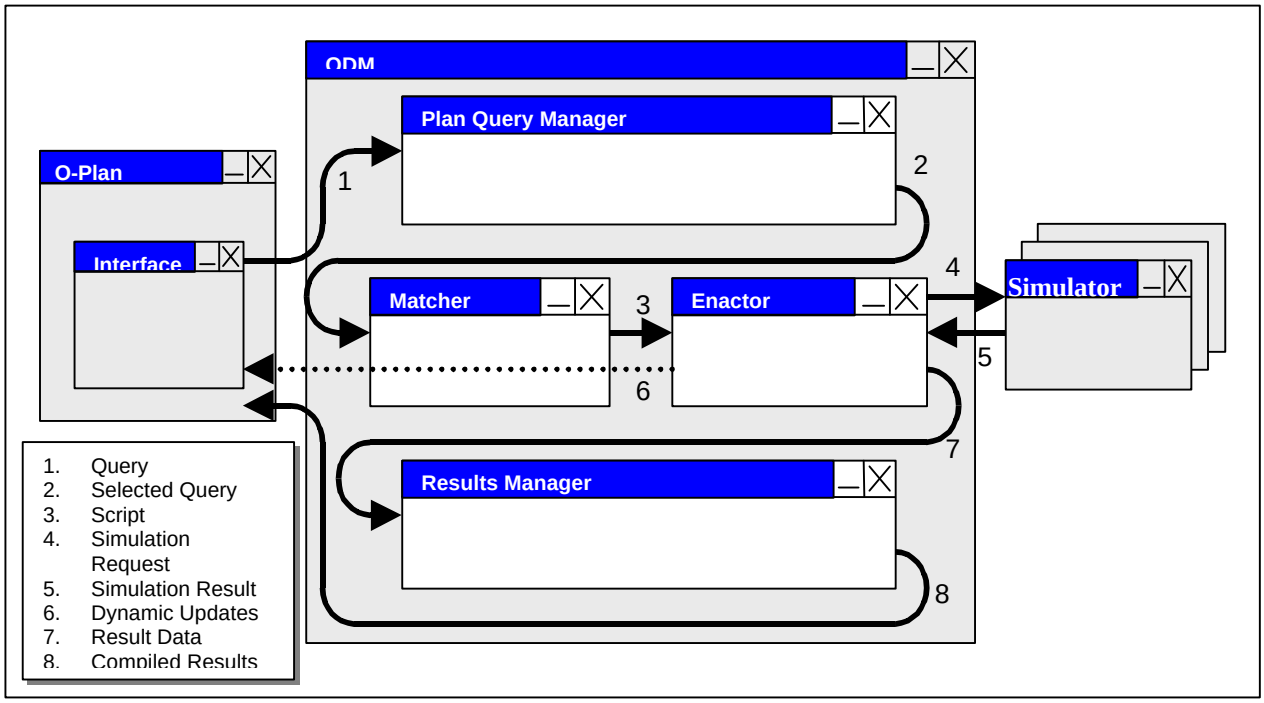

Figure 3: A schematic depiction of the prototype

supported. Again the query may be rejected at this stage probably because no suitable script and/or simulator is available.

- **The Enactor**
  The Enactor interprets the simulator query language (SimQL**)** script line by line and directs execution. It handles initialisation and simulation control issues and manages communications with the selected simulator. Execution continues to the end of the script or it can be paused or indeed aborted at any stage.

- **The Results Manager**
  This component is responsible for compiling the results and returning them in a summarised form to the originator. This includes details of specific metrics identified in the plan query, which may be derived from multiple runs of a simulation (e.g. averaging). Rejected queries are also posted via the Results Manager.

It was decided not to integrate with large-scale, well-established simulators that were available, as this would have required an unacceptable amount of effort. Instead, two simple but effective simulators were developed in-house: one a stochastic simulator capable of simulating ground vehicles (Sim2**)**, the other (Sim1**)** a deterministic equivalent of the former. These were developed very quickly and served the needs of the research objectives.

4. Scenario and worked example

This experimental prototype is based around a non-combatant evacuation operation (NEO) occurring on the island of *Pacifica* taken from the PRECiS (Planning, Reactive Execution and

Constraint-Satisfaction) environment[2] [Reece et al, 1994]. NEOs are undertaken to provide rapid response to a variety of circumstances, including natural disasters, requiring the evacuation of civilians from trouble zones. The PRECiS environment provides a readily available planning domain accessible through the O-Plan system and is rich enough in context to allow us to explore all of the relevant issues present in the problem of linking planning systems to simulators and to demonstrate the potential military benefits. For expediency in exploring the research issues within the prototype, Sim1 and Sim2 have been implemented to correspond with this domain.

A typical plan query will now be described in the context of the Pacifica scenario. A breadth-first approach to this implementation has been adopted with the emphasis on achieving an integrated set of services that completely close the loop from planner to simulator and back to the planner again. The next section discusses the outcomes of this approach and how a depth-first approach will be pursued for the next prototype to enhance and strengthen the completion of the loop.

The situation is as follows: due to local unrest it has been decided to evacuate a civilian population stranded in a Pacifica city. Stormy weather prohibits the use of helicopters in the evacuation and the planner has already made the decision that ground trucks will be used. Three different types of truck are available on the island immediately but there is a local shortage of diesel fuel. To ensure the safe evacuation of the civilian population the planner wishes to select the truck type requiring the minimum amount of fuel. To assist in the selection a plan query is generated in PQL so that the operation may be simulated for each truck type and advice provided to the planner.

The plan query is matched with candidate scripts. In this case there are two scripts available: one involving a deterministic simulator and the other utilising a stochastic simulator (see figure 4). This preliminary implementation of SimQL requires considerable further development but nonetheless those elements, which are considered to be necessary, are present:

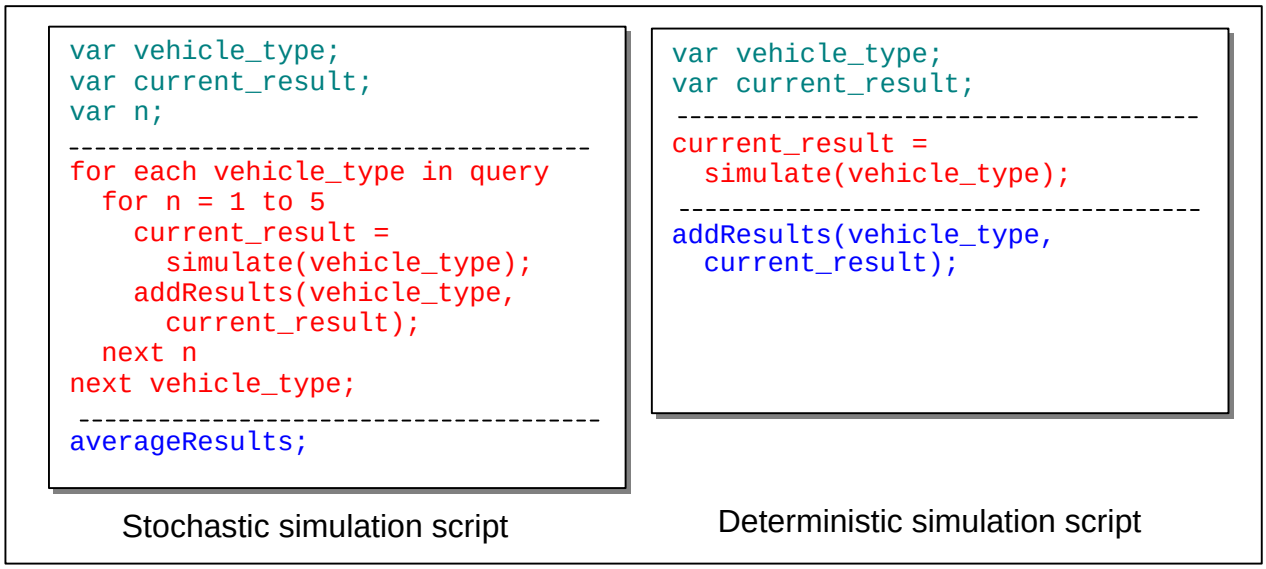

Figure 3. Examples of the simulation scripts

[2] http://www.isx.com/pub/ARPI/ARPI-pub/precis/

- The first section deals with initialisation (teal).
- The middle section with execution and control (red).
- The last section addresses the compilation of results (blue).

A script is selected together with a candidate simulator and the Enactor begins execution. Feedback is provided to the plan query owner via the Enactor during execution and upon completion all of the collected data is formatted into a summarised reply posted to the planner. The planner now has information indicating which truck performed best against the specified metric (fuel consumption) and is able to make his choice and continue planning. Of course, the human planner remains fully in the loop and may choose to override the query response perhaps because he has access to a wider picture.

5. Experimental results and lessons learned

The experimental results have validated the approach of using planning and simulation query languages to integrate planning and simulation systems, but further work is required to fully implement the language specifications. The project has deliberately adopted a breadth first approach to allow rapid development of a prototype supporting the complete process required for integrating planners with simulators. The next step is to address the major research issues in greater depth. The experimental results have largely confirmed our expectations about what the key research issues would be in attempting to link planners and simulations.

There are a number of issues concerning the Plan Query Language (PQL):

- The representation of data sources in a plan query has not been addressed. A URL style reference has potential for this purpose.
- Query type and task details are described in a language based on a predicate logic. Our experience with the prototype supports this approach and we believe it is suitable for expressing metrics also.
- The issue of supporting constraints in PQL has not been addressed.
- In future, more complex queries that include competing metrics and constraints need to be supported, rather than the simple queries so far.

This work has also allowed us to gain experience about the required features of a simulation query language (SimQL):

- As with the interfacing module on the planners' side, it is expected that there needs to be a specific interface module to translate generic simQL queries into elements, which may be understood by a specific simulator.
- The simulators used here were deliberately pre-configured to correspond with the Pacifica domain. Although detailed scenarios will necessarily be generated off-line, there is scope to improve the ability to initialise and control simulation execution.
- A review of the High Level Architecture (HLA**)** specification [DMSO, 1994] and available tools has been conducted with the intention of leveraging this technology in this area.

HLA offers an environment that allows multiple distributed simulators to interact together. Most importantly, HLA-compliant implementations provide a hierarchical data model (the 'Object Model Template') which could also be adopted by planning systems.

Other recent work on collaborative planning [Ferguson et al, 1996] has explored the mixed-initiative planning paradigm which emphasises the importance of humans as agents within the planning process.

This work has confirmed that the integration of planning systems with simulators can improve the planning process and support the planner in his task. It has been found that:

- The human planner remains a key element in the loop.
- The use of dynamic reporting of results is useful.
- The human planner has a key role in controlling, monitoring and refining plan queries during execution.

Results have also confirmed that service-based architectures provide the necessary framework for linking planning and simulation systems. This prototype has been implemented with a number of complementary services within the QDM including management of queries, simulator execution and results and simulator registry. Eventually the QDM should comprise a set of coherent services that can be configured flexibly to adapt to the current plan query, following network-centric principles.

6. **Further work**

During the year a number of key areas for further work have been identified. These will be focussed on next year and are outlined below.

Most importantly, the PQL and SimQL languages will be developed further to allow us to improve the integration achieved so far. Specifically, the representation of data sources in a plan query and a wider range of query type will feature in the next implementation. While there is scope for enhancing plan queries in terms of more complex metrics and competing constraints it is expected that significant development will be left for a future project.

Work has now commenced on a second prototype, which will leverage a service-based architecture providing scaleable, flexible and adaptable systems. This new implementation will exploit suitable standards and candidate technologies include:

- Java
- Networking and distributed computing (e.g. TCP/IP and XML).
- Shared Plan & Activity Representation (SPAR) specification[3] [Tate, 1998].
- First order predicate logic languages (e.g. Description Logic).
- Software agent technology (e.g. KQML).

[3] http://www.aiai.ed.ac.uk/~arpi/spar

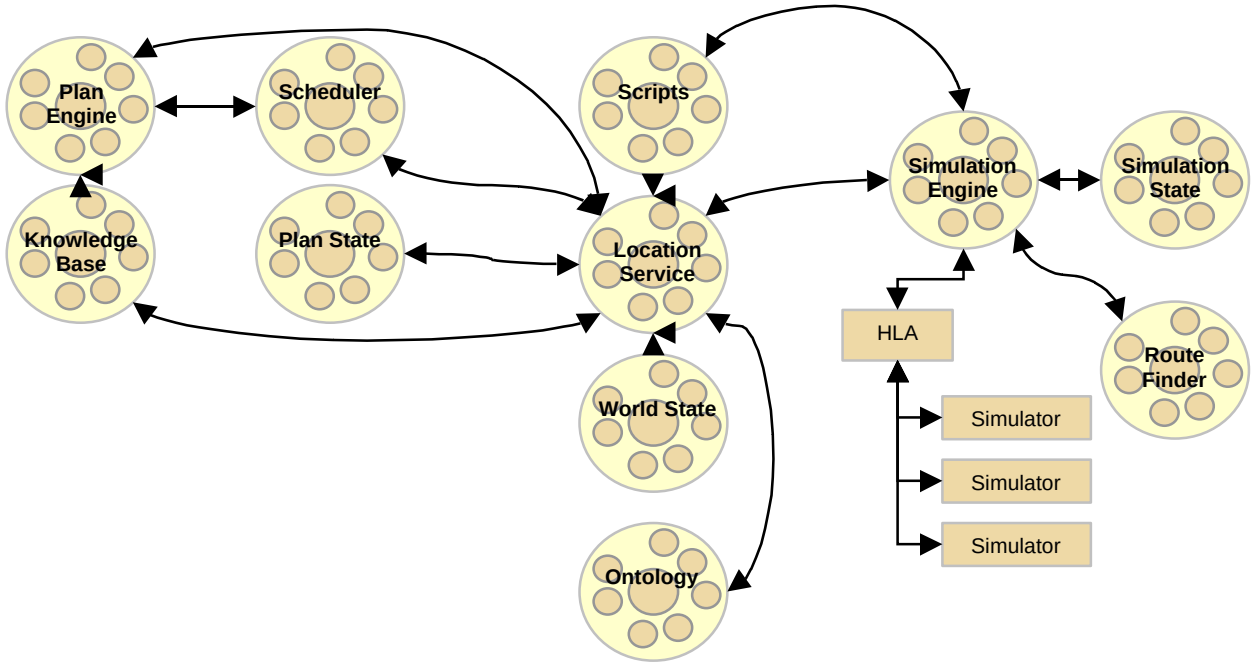

Figure 5. Typical services required in a service-based architecture.

Figure 5 shows a typical set of services capable of interacting together to support linked planning and simulation activities. There are core services supporting the whole community (e.g. yellow pages, ontology server) and there are highly specialised services (e.g. plan engine, router, simulator, scheduler). Data sources represented as specialised information services can be shared across the network (i.e. both the planners and the simulators). Each of the services is wrapped with a common intermediate layer that enables communications, networking and task planning and scheduling. Coherent services 'make friends' with each other to enable the kind of fast, robust, distributed communication, which would be necessary in a deployed system. Elements of the QDM are therefore present in each wrapped component, removing the necessity for a single central interface mechanism between planning and simulation.

One simple yet powerful concept explored in the implementation is the expression of capability by the simulators. This was in the form of simple text capabilities that were registered with the QDM along with a description and location of the simulator itself. Such a scheme will need to be developed so that all participating components can express their capabilities perhaps through a location service or facilitator.

7. Conclusions

These results will be exploited within relevant UK C4I programmes that might influence future operation UK joint command systems, especially within the MoD applied research programme for joint & strategic C2I applications. There is also potential for exploitation within the UK Joint Training & Warfighting Initiative (JTWI) in supporting the integration of campaign planning & mission rehearsal.

The results presented here confirm that the approach of linking planning systems and simulation models with the aid of planning and simulation query languages is very promising and should be researched further.

Acknowledgements

This work was carried out as part of Technology Group (TG10) of the UK MoD Corporate Research Programme. The authors wish to express thanks to the AI Applications Institute (AIAI) at the University of Edinburgh for their considerable expertise in AI planning research and technology. The authors are grateful for the support of the rest of the project team.

References

[Desimone et al, 1999], R. Desimone, M. Round, M. Illingworth. *Enhancing operational effectiveness through the integration of planning & simulation technology.* Proceedings of 1999 Command & Control Research & Technology Symposium (CCRTS), Newport Beach, RI, June 1999.

[Currie and Tate, 1991] K. Currie and A. Tate. *O-Plan: the Open Planning Architecture*. Artificial Intelligence, Vol. 52, pp49-86, 1991.

[Wilkins, 1998] D.E. Wilkins. *Practical Planning: Extending the classical AI planning paradigm.* San Mateo, CA. Morgan Kaufman. 1998.

[Wilkins et al, 1995] D.E. Wilkins, K. Myers, J. Lawrance, and L. Wesley. *Planning and reacting in uncertain and dynamic environments.* Journal of Experimental and Theoretical Studies in AI (JETAI), Vol. 7, pp 121-152, 1995.

[Reece et al, 1994] G. Reece, A. Tate, D.I. Brown, M. Hoffman, and R.E. Burnard. *The PRECiS Environment*. University of Edinburgh, March 1994.

[DMSO, 1994] Defense Modeling and Simulation Office. *High Level Architecture Overview*. **http://www.dmso.mil**. June 1997.

[Ferguson et al, 1996] G. Ferguson, J. Allen, and B. Miller. *TRAINS-95: Towards a mixed-initiative planning assistant.* Proceedings of the Third International Conference on AI Planning Systems. Edinburgh, UK. 1996.

[Tate, 1998] A. Tate. *Roots of SPAR –Shared Planning and Activity Representation*. The Knowledge Engineering Review, Vol. 13(1), pp 121-128. Special Issue on Putting Ontologies to Use. Cambridge University Press, 1998.