



Defence Research and  
Development Canada

Recherche et développement  
pour la défense Canada



# Dynamic Plan Management in the Context of a Recognized Air Picture

*Aaron Hunter, Jens Happe, Melanie Dutkiewicz*

*MacDonald Dettwiler and Associates Ltd.  
13800 Commerce Parkway  
Richmond, BC, Canada V6V 2J3*

*Project Manager: Melanie Dutkiewicz  
Contract No.: W7701-6-3946NP*

*Contract Scientific Authority:  
Micheline Bélanger, (418) 844-4000 Ext.: 4734*

*MDA Project Title: JCDS-21 RAP  
MDA Document Number: RX-RP-52-5076*

The scientific or technical validity of this Contract Report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.
---

**Defence R&D Canada – Valcartier**

Contract Report

DRDC Valcartier CR 2007-446

December 2007

Canada



# **Dynamic Plan Management in the Context of a Recognized Air Picture**

## **Final Report**

**December 07, 2007**

**DRDC Valcartier CR2007-446**

MacDonald Dettwiler and Associates Ltd.

PWGSC Contract Title	Recognized Air Picture (RAP)
MDA Project Title:	JCDS-21 RAP
MDA Document Number:	RX-RP-52-5076
Contract No.:	W7701-6-3946NP
Project Duration:	JAN 07-NOV 07
DRDC Project Manager:	Melanie Dutkiewicz
DRDC Scientific Authority:	Micheline Bélanger
DRDC Document Number:	CR 2007-446

UNCLASSIFIED

Ref:  
Issue/Revision:  
Date:

RX-RP-52-5076  
1/3  
DEC. 07, 2007

**THIS PAGE INTENTIONALLY LEFT BLANK**

# **Dynamic Plan Management in the Context of a Recognized Air Picture**

## **Final Report**

**December 07, 2007**



**DRDC Valcartier CR2007-446**

**MacDonald Dettwiler and Associates Ltd.**

13800 Commerce Parkway  
Richmond, BC, Canada  
V6V 2J3

PWGSC Contract Title  
MDA Project Title:  
MDA Document Number:  
Contract No.:  
Project Duration:  
DRDC Project Manager:  
DRDC Scientific Authority:  
DRDC Document Number:

Recognized Air Picture (RAP)  
JCDS-21 RAP  
RX-RP-52-5076  
W7701-6-3946NP  
JAN 07-NOV 07  
Melanie Dutkiewicz  
Micheline Bélanger  
CR 2007-446


UNCLASSIFIED

Ref:  
Issue/Revision:  
Date:

RX-RP-52-5076  
1/3  
DEC. 07, 2007

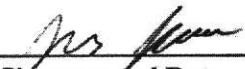
Prepared By:

Aaron Hunter

  
Signature and Date Dec 10, 2007


Project Engineer:

Jens Happe

  
Signature and Date

Project Manager:

Melanie Dutkiewicz

  
Signature and Date Dec 10, 2007

**CHANGE RECORD**

ISSUE	DATE	PAGE(S)	DESCRIPTION	RELEASE
1/0	Apr. 24, 2007	All	First Issue	
1/1	Apr. 25, 2007	All	Bibliography and references regenerated	
1/2	Nov. 16, 2007	All	First Issue, Second Revision	
1/3	Dec. 07, 2007	App B	First Issue, Third Revision	

RELEASED BY  
CONFIGURATION CONTROL

10 DEC 2007  
*J. L. Chapell*

**THIS PAGE INTENTIONALLY LEFT BLANK**



## TABLE OF CONTENTS

<b>1</b>	<b>INTRODUCTION.....</b>	<b>1-1</b>
<b>2</b>	<b>BACKGROUND .....</b>	<b>2-1</b>
2.1	Context of Study: AF Common Operational Picture .....	2-1
2.2	Dynamic Plan Management.....	2-2
2.3	Visualization Tools .....	2-3
2.4	Artificial Intelligence Planning .....	2-5
2.5	AF Plans.....	2-6
2.5.1	Plan Elements.....	2-6
2.5.2	Types of Plans.....	2-7
2.6	Summary .....	2-8
<b>3</b>	<b>LITERATURE REVIEW.....</b>	<b>3-1</b>
3.1	Plan Representation .....	3-1
3.1.1	Ontologies and Formal Languages .....	3-2
3.1.2	Probabilistic Approaches .....	3-7
3.1.3	Visualization .....	3-8
3.2	Plan Forecasting / Projection .....	3-9
3.3	Plan Analysis / Evaluation.....	3-12
3.4	Plan Monitoring .....	3-15
<b>4</b>	<b>SYNTHESIS .....</b>	<b>4-1</b>
4.1	Measures of Efficiency .....	4-1
4.1.1	Plan Representation .....	4-1
4.1.2	Plan Forecasting/Projection .....	4-2
4.1.3	Plan Analysis/Evaluation.....	4-2
4.1.4	Plan Monitoring .....	4-2
4.2	Description of Most Promising Approaches.....	4-3
4.2.1	Plan Representation .....	4-3
4.2.2	Plan Forecasting / Projection .....	4-18
4.2.3	Plan Analysis / Evaluation.....	4-24
4.2.4	Plan Monitoring .....	4-31
<b>5</b>	<b>INVESTIGATION OF TECHNIQUES AND APPROACHES.....</b>	<b>5-1</b>
5.1	Comparison of the Strengths and Weaknesses of Selected Approaches.....	5-1
5.1.1	Plan Representation .....	5-1
5.1.2	Plan Forecasting / Projection .....	5-2
5.1.3	Plan Analysis / Evaluation .....	5-3
5.1.4	Plan Monitoring .....	5-4
5.1.5	Plan Visualization .....	5-4
5.1.6	Conclusion .....	5-4
5.2	High-level Description of the Proposed Software .....	5-5

5.2.1	Identification of a Combined Approach .....	5-6
5.2.2	Plan Elements .....	5-8
5.2.3	Plan Abstraction .....	5-16
5.2.4	Dynamic Plan Management .....	5-17
5.2.5	Plan Visualization.....	5-24
5.3	Implementation Details .....	5-28
5.3.1	Plan Ontology.....	5-29
5.3.2	Validation .....	5-32
5.3.3	Visualization.....	5-32
5.3.4	Object-oriented representation of plan elements .....	5-33
5.3.5	Modelling Execution Data.....	5-41
5.4	Assessment .....	5-44
5.4.1	Achievements of the Software.....	5-44
5.4.2	Limitations of the Software .....	5-49
5.4.3	Overall Assessment .....	5-50
5.5	Avenues for further research and development.....	5-51
<b>6</b>	<b>SUMMARY.....</b>	<b>6-1</b>
<b>7</b>	<b>BIBLIOGRAPHY.....</b>	<b>7-1</b>
<b>A</b>	<b>APPENDIX .....</b>	<b>A-1</b>
A.1	Plan File Syntax .....	A-1
A.2	Document Type Definition file format.....	A-1
A.3	Plan files .....	A-3
A.4	Situation updates .....	A-4
A.5	Plan updates.....	A-4
<b>B</b>	<b>APPENDIX - IDENTIFICATION OF FOREGROUND INFORMATION .....</b>	<b>B-1</b>

## LIST OF FIGURES

Figure 4-1	Example hierarchy described by LOOM .....	4-7
Figure 4-2	Example CPN for representing task execution .....	4-11
Figure 4-3	A screenshot taken from NuSketch.....	4-16
Figure 4-4	Visual depiction of plan decomposition in three dimensions .....	4-23
Figure 4-5	A COA Evaluation Matrix, taken from the O-Plan web demo .....	4-30
Figure 4-6	A screenshot taken from CAAT-Xi. ....	4-34
Figure 5-1	Plan Representation .....	5-6
Figure 5-2	Proposed Approach.....	5-7
Figure 5-3	Element Hierarchy .....	5-9
Figure 5-4	Task status flow diagram .....	5-13
Figure 5-5	Plan View.....	5-26
Figure 5-6	Map View .....	5-27
Figure 5-7	An Invalid Plan .....	5-29
Figure 5-8	Map View of the North Atlantis Vignette.....	5-33
Figure 5-9	Generating Execution Data .....	5-42
Figure 5-10	XML Representation of Plan Updates and Situation Updates .....	5-44

## LIST OF TABLES

Table 4-1	Formal representation of the categories of entities used in SPAR .....	4-4
Table 5-1	Measures of efficiency applied to representation schemes.....	5-2
Table 5-2	Measures of efficiency applied to plan forecasting .....	5-3
Table 5-3	Measures of efficiency applied to plan analysis.....	5-3
Table 5-4	Measures of efficiency applied to monitoring.....	5-4
Table 5-5	Plan Elements .....	5-8
Table 5-6	Key Plan Elements in the Ontology .....	5-30
Table 5-7	Object Description Schema .....	5-34
Table 5-8	DPM Element Object .....	5-34
Table 5-9	Location and Time Objects .....	5-35
Table 5-10	Resource Objects .....	5-36
Table 5-11	Constraint Objects .....	5-37
Table 5-12	Effect Objects .....	5-38
Table 5-13	Action, Task and Plan Objects .....	5-39
Table 5-14	Plan Manager Objects .....	5-40
Table 5-15	Status Objects .....	5-40
Table 5-16	Measures of Efficiency for Plan Representation .....	5-47
Table 5-17	Measures of Efficiency for Plan Forecasting/Projection.....	5-48
Table 5-18	Measures of Efficiency for Plan Analysis/Evaluation.....	5-48
Table 5-19	Measures of Efficiency for Plan Monitoring.....	5-49
Table 5-20	Items for future research.....	5-52

## ACRONYMS AND ABBREVIATIONS

ACO	Airspace Control Order
ACPT	Air Campaign Planning Tool
AF	Air Force
AI	Artificial Intelligence
APSS	Anticipatory Planning Support System
ATM	Air Task Message
ATO	Air Tasking Order
BAM	Bird Avoidance Module
C2	Command and Control
C2PC	Command and Control PC
CAF	Canadian Air Force
CAPS	Combat Airdrop Planning Software
CAAT-Xi	Canadian ATO/ACO Tool XML Interpreter
CDR	Critical Design Review
CDS	Courseware Development Standards
CF	Canadian Forces
CFACC	Combined Force Air Component Commander
CFPS	Combat Flight Planning Software
CJCS	Chairman of the Joint Chiefs of Staff
C/JMTK	Commercial Joint Mapping Toolkit
COA	Course of Action
COAST	The Course of Action Scheduling Tool
CONOPS	Concept of Operations
CONPLAN	Contingency Operation Plan
COP	Common Operational Picture
CPG	Commander's Planning Guidance
CPN	Coloured Petri-nets

CTA	Cognitive Task Analysis
CTAPS	Contingency Theatre Automated Planning System
CTP	Common Tactical Picture
CTPS	Combat Target Planning System
CWDS	Combat Weapon Delivery Software
DART	Dynamic Analysis and Re-planning Tool
DIPART	Distributed, Interactive Planner's Assistant for Real-Time Transportation Planning
DP	Deliberate Planning
DZ	Drop Zone
FORMAT	Force Management and Analysis Tool
FOX-GA	Fox Genetic Algorithm
FT	Force-Oriented Task
GWTS	Guided Weapons Trajectory Software
HHQ	Higher Headquarters
ICC	Integrated Command and Control
IMPL O	Implementation Orders
INOVA	Issues-Nodes-Ordering/Variables/Auxiliary
INSPECT	Intelligent System for Air Campaign Plans Based on Expect
JADE	Joint Assistant for Development and Execution
JFACC	Joint Force Air Component Commander
JMTK	Joint Mapping Toolkit
LRTDP	Labeled Real-Time Dynamic Programming
LZ	Landing Zone
MCDA	Multiple Criteria Decision Analysis
MDP	Markov Decision Process
NORAD	North American Aerospace Defence Command
NATO	North Atlantic Treaty Organization
OP O	Operation Order or Op Order
OPLAN	Operations Plan

OPORD	Operations Order
OWL	Web Ontology Language
PC	Personal Computer
PDDL	Planning Domain Definition Language
PFPS	Portable Flight Planning Software
RAP	Recognized Air Picture
RAT	Route Analysis Tool
SA	Situational Awareness
SAA	Situational Awareness and Assessment
SDP	Standing Defence Plan
SIPE-2	System for Interactive Planning and Execution
SJS	Secretary, Joint Staff
SPAR	Shared Planning and Activity Representation
SPINS	Special Instructions
STRIPS	Stanford Research Institute Problem Solver
SUPLAN	Supporting Plan
TBD	To Be Determined
TBMCS	Theatre Battle Management Core Systems
TF	Task Formalism
TOI	Tracks of Interest
TOPFAS	Tool for Operational Planning, Force Activation and Simulation
TPEDIT	Time-Phased Force Deployment Data Editor
UML	Unified Modeling Language
XML	eXtensible Markup Language

**THIS PAGE INTENTIONALLY LEFT BLANK**



## ABSTRACT

This document surveys existing literature on dynamic plan management and describes the development of a prototype Air Force plan management system. The literature survey presents short summaries of a wide range of research papers, as well as a synthesis and analysis of existing approaches. The detailed comparison of existing approaches is used to formulate a specific methodology for the development of software for plan representation, plan forecasting/projection, plan analysis/evaluation and plan monitoring. The proposed methodology involves the development of a precise ontology of plan elements for plan representation. This representation removes ambiguity in the description of plans, facilitates automated analysis of plans, and also permits several different approaches to plan visualization. The implemented prototype software defines an ontology that provides a suitable internal representation of plans, along with basic plan validation capabilities. It also provides a map-based graphical user interface to visualize plans. The use of the software is demonstrated in the context of a combat search and rescue vignette. The findings conclude that the prototype demonstrates the overall utility of the approach, although further development is required to provide more detailed analysis, as well as additional visualization methods.

## RÉSUMÉ

Ce document examine la littérature existante au sujet de la gestion des plans et décrit le développement d'un prototype pour la gestion de plans de la Force Aérienne. L'enquête de littérature fournit de courts sommaires de plusieurs articles de recherches et une comparaison détaillée des approches existantes. Cette comparaison est employée pour formuler une méthodologie pour le développement de logiciel pour la représentation, la prévision, l'évaluation, et le suivi de plans. La méthodologie proposée requiert le développement d'une ontologie pour la représentation de plans. Cette représentation peut éliminer l'ambiguïté dans la description de plans, faciliter l'analyse automatisée de plans et permettre plusieurs approches à la visualisation. Le prototype implémenté définit une ontologie qui fournit une représentation interne appropriée des plans, avec en plus la capacité de validation des plans. Il fournit également un interface graphique pour visualiser des plans. L'utilisation du prototype est démontrée dans le contexte d'une vignette de recherche et sauvetage en combat. Les résultats expérimentaux obtenus avec le prototype démontre l'utilité de l'approche proposée, malgré que plus de développement et d'autres méthodes visualisation soient nécessaire.

## EXECUTIVE SUMMARY

In order to make informed decisions, Air Force commanders must be aware of the status of all plans scheduled for execution. Most existing software to support planning is focused on efficient plan generation. However, due to the sensitive nature of Air Force plans, it is preferable to develop decision support tools for the analysis of human-generated plans. This document presents an overview of the key challenges faced in Air Force plan management, as well as an approach to the development of an appropriate decision support solution.

Air Force plans tend to be complex, involving many tasks, objectives and constraints. In order to develop decision support software for dynamic plan management, plans must be *represented* in a precise manner suitable for automated analysis. Furthermore, given such a precise representation, three different temporal perspectives should be considered: First, prior to its execution, the outcome of a plan can be *projected* or *forecasted*. Second, following execution, the outcome of a plan can be *analyzed* or *evaluated*. Third, during execution, the current status of a plan can be *monitored*. From each perspective, plan visualization tools can aid a commander in gaining a deep understanding of plan structure and plan status.

Based on the existing literature, this document concludes that the best approach is to represent plans in terms of a formal taxonomy of plan elements. It is possible to develop automated plan management tools upon taxonomies of this form, and there is a large body of literature on this topic. In order to bridge the gap between the internal representation of plans and the military operator, several different visualization tools are proposed. The basic idea is to provide tools that allow a military commander to understand the structure and view the current status of a plan at any point in time. The authors have implemented a prototype plan management system following this approach. This document gives an overview of the system's design and implementation.

This work is significant because it combines the best features of several existing approaches to plan management. There has been relatively little literature explicitly on plan representation, plan forecasting/projection, plan analysis/evaluation and plan monitoring. Hence, one contribution of this document is that it collects, synthesizes and analyzes literature on this topic. The second contribution is the development of an explicit approach to developing dynamic plan management software. The third contribution is a proof-of-concept demonstration of the approach, in the form of a specific prototype.

Future work on this project could involve incorporating emerging trends from the planning research community, explicitly modelling plan abstraction for different military users, providing a more detailed analysis of plans and plan failures, and implementing additional visualization modes for the current prototype.

## SOMMAIRE

Afin de faire des décisions éclairées, les commandants de la Force Aérienne doivent savoir le statut de tous les plans programmés pour l'exécution. La plupart des logiciels existants pour aider la planification sont concentrés sur la génération efficace de plan. À cause de la sensibilité des plans de la Force Aérienne, il est préférable de développer des outils de support à la décision pour l'analyse des plans produits par un humain. Ce document présente les principaux défis de la gestion des plans de la Force Aérienne et une approche au développement de logiciel pour support à la décision.

Les plans de la Force Aérienne sont complexes, comportent plusieurs tâches, d'objectifs et de contraintes. Afin de développer un logiciel pour la gestion des plans, les plans doivent être représentés de façon précise pour permettre l'analyse automatisée. Si une représentation est précise, trois perspectives temporelles devraient être considérées: D'abord, avant l'exécution d'un plan, les résultats du plan peuvent être *prévus*. Deuxièmement, après l'exécution d'un plan, les résultats du plan peuvent être *analysés* ou *évalués*. Troisièmement, pendant l'exécution d'un plan, l'état actuel peut être *suivi*. De chaque perspective, les outils de visualisation de plan peuvent aider un commandant à comprendre la structure et l'état d'un plan.

Basé sur la littérature existante, ce document conclut que la meilleure approche est de représenter les plans avec une taxonomie formelle des composantes de plan. Cette représentation facilite le développement des outils de gestion automatisés de plans, et il y a beaucoup de littérature à ce sujet. Afin d'établir un lien entre la représentation interne des plans et l'opérateur militaire, nous proposons plusieurs outils de visualisation. L'idée fondamentale est de fournir des outils qui permettent à un commandant militaire de comprendre la structure et l'état actuel d'un plan. Les auteurs ont suivi cette approche pour développer un prototype de gestion de plans. Ce document décrit la conception et l'exécution du prototype.

Ce travail est significatif parce qu'il combine les meilleurs aspects de plusieurs approches existantes pour la gestion de plans. Il y a peu de littérature explicitement sur la représentation, la prévision, l'évaluation et le suivi de plans. Par conséquent, une contribution de ce document est l'assemblage et l'analyse de la littérature existante. La deuxième contribution est l'élaboration d'une approche de développement de logiciel de gestion de plans. La troisième contribution est une démonstration de l'approche, sous forme d'un prototype.

Les travaux futurs pourraient incorporer les tendances émergentes de la recherche sur la planification, modéliser explicitement l'abstraction de plans pour différents utilisateurs militaires, fournir une analyse plus détaillée des plans et de leurs échecs, et ajouter des modes additionnels de visualisation au prototype courant.

**THIS PAGE INTENTIONALLY LEFT BLANK**

# 1 INTRODUCTION

The focus of this document is to review the challenges associated with representing, maintaining and monitoring plans supporting Situational Awareness (SA) within the context of Canadian Air Force (CAF) operations, and to better understand the approaches and techniques being used by others to address these challenges.

This document is organized as follows:

- Section 2 provides the context for the report and gives some basic background material on military plans.
- Section 3 provides summaries for a wide range of published papers and documents on dynamic plan management, with a particular focus on plan representation, plan forecasting/projection, plan analysis/evaluation and plan monitoring.
- Section 4 discusses the most promising approaches in greater detail. It first describes the evaluation criteria to be used for each of the four areas of interest (plan representation, plan forecasting/projection, plan analysis/evaluation and plan monitoring), and then provides an analysis of the selected approaches, including examples, analysis, and explicit discussion in the context of the measures of efficiency provided.
- Section 5 compares the existing approaches, develops a specific recommendation for the development of plan management software, outlines the design of our prototype software and provides an assessment of the software.
- Section 6 provides a summary of the work done, including highlighting some of the key findings of the research.
- Finally, a bibliography of all the papers referenced in the document is provided at the end of the report.

Ref: RX-RP-52-5076  
Issue/Revision: 1/3  
Date: DEC. 07, 2007

UNCLASSIFIED



**THIS PAGE INTENTIONALLY LEFT BLANK**

## 2 BACKGROUND

The Common Operational Picture (COP) used by the Air Force (AF) to understand the situation at hand is of crucial importance to its success in carrying out and executing a mission. In order to increase the commanders' understanding of the situation, and support their ability to predict future states of a mission, this picture must provide them with all information of relevance to the operation in the most effective possible manner.

Plan representation, analysis, forecasting, and monitoring in a dynamic environment need to be supported in the COP. Visualization tools exist for planning and monitoring, but they do not have automated systems in place to improve this picture and therefore do not support all aspects of dynamic plan management, the focus of this work. However, planning research in the Artificial Intelligence community offers approaches to filling this missing link, so that the COP can give all information related to plans in such a way that plans can be understood, analyzed, and monitored.

### 2.1 Context of Study: AF Common Operational Picture

To manage operational activities and make informed decisions, commanders must develop a deep understanding of the situation at hand. So they need easy access to all information within their area of influence and responsibility. Potential sources include information regarding intelligence, tactical activity, the status and sustainability of their own forces, the mission environment, and the physical environment [JIIFC PD 2005]. In other words, they must develop Situational Awareness (SA).

In the context of military operations, SA has been interpreted as “the assimilation of current and historical information to form a mental image of what is going on and what may happen in order to inform decision making.” [JIIFC PD 2005] However, this mental image of the mission space will vary between commanders and staff dependent on their roles, differences in experience, and sources of information. A common framework for developing SA is therefore needed to ensure unity of effort and coordinated objectives.

A Common Operational Picture (COP) is a means to provide commanders with shared SA for operational and/or strategic initiatives. It is used to group operationally relevant information

into a manageable framework and to improve SA. The COP can be described as “a snapshot in time of friendly, neutral and adversary forces and of the battlespace environment” [Canadian Forces 2003].

In general, decision makers drive content. So although SA spans different levels of command, the COP needs to suit the SA needs of the operational and strategic levels of command. This could mean providing COP content in the form of templates and information layers, while providing the ability to modify the presented information in a flexible way so that the most relevant information is available to suit operational needs [JIIFC PD 2005].

For AF operations, the COP can be thought of as an abstracted version of a Common Tactical Picture (CTP), which would include information used to track, identify and monitor Tracks of Interest (TOIs). Whereas a CTP needs a near real-time representation of the battle space, a COP is considered non real-time [1 CDN Air Div A3 ASR 3 and AFCCIS 2005].

## 2.2 Dynamic Plan Management

In order to provide a snapshot of all forces in a battle environment, it is necessary to be aware of any relevant AF plans. As such, the AF COP includes information about AF plans that are currently being executed, as well as past and future AF plans. In order to support SA, this information must be regularly updated with relevant information. Making informed decisions involves checking the status of any plans that are currently executing. Moreover, if it appears that a plan is not going to meet some objective, it might be necessary to make the decision to modify the plan. Taken together, these facts illustrate that maintaining the AF COP involves *dynamic plan management*; that is, the management of changing plans in a changing environment.

Managing the execution of plans in a dynamic environment is a difficult problem, particularly with incomplete information. In order to manage plans effectively, one must first specify the elements of a plan that are considered relevant for decision making. Given a specific set of relevant elements, plan management involves representing plans, forecasting the outcome of plans, evaluating the performance of plans, and monitoring the execution of plans. We briefly define each of these components of plan management more precisely.

### Plan Representation

A representation of a plan is a complete, unambiguous description of a plan at the appropriate level of abstraction. The amount of detail included in the representation of a plan depends on the individual communicating the plan, as well as the intended use of the plan. For example, representations of plans must be very precise to be useful for computer-assisted plan management at the tactical level. On the other hand, a rough sketch of units overlaying a map might provide sufficient detail for strategic planning at a very high-level. Plan representation involves formulating conventions for specifying all aspects of a plan, including resources, assumptions, constraints, and objectives. It also includes approaches to plan visualization.



### **Plan Forecasting/Projection**

Any technique or approach to predict the outcome of a plan is referred to as plan forecasting. A plan is typically formulated to achieve some specific goals, but changes in the environment and actions of the adversary may alter the outcome of a plan. Plan forecasting involves a method that predicts the most likely state of affairs at a specific point in the execution of a plan. Since plans often have uncertain outcomes, such predictions may involve estimates or calculations of relevant probabilities.

### **Plan Analysis/Evaluation**

Plan analysis refers to the process of assessing the quality of a plan. One aspect of plan analysis is to look for flaws in the plan that will make the plan fail to achieve the desired goal. However, plan analysis may also involve comparisons between plans. Frequently more than one plan is developed to achieve a specific goal. In such cases, there needs to be some method for comparing the plans and determining which plan should be attempted. Plan analysis should consider geographic concerns, temporal concerns, financial concerns, risks, and many other factors. In the end, some well-defined performance measure(s) should be applied in assessing the value of the plan.

### **Plan Monitoring**

Observing the execution of a plan as it occurs is referred to as plan monitoring. Real-time monitoring of plan execution requires important physical resources, such as sensors for detecting unit locations and communications capabilities with active units. Plan monitoring involves obtaining this information and using it to determine any changes required to an active plan. Changing an active plan may have significant consequences that are difficult to predict. Plan monitoring involves assessing the current plan by comparing the current state of the plan with the state that would have been predicted at the same stage prior to execution. An important topic related to plan monitoring is the real-time visualization of plan execution.

## **2.3 Visualization Tools**

There are a number of visualization tools available for use by the Canadian Forces (CF). Note that the ones we were able to find information for have no automated planning facilities that will improve the operational picture used by the AF or identify the intent of an object [Boukhtouta, Bélanger et al. 2006]. Therefore, they lack the ability to support dynamic plan management, crucial to further increasing SA.

In this section, we provide a brief description of some of the tools that support different aspects of plan management. Paradigms for visualizing plan representation, forecasting, analysis, and monitoring will be discussed in Section 3.

### **C/JMTK**

The Commercial Joint Mapping Toolkit (C/JMTK) is a developer toolkit of geospatial applications for the management, analysis, and visualization of map and map-related

information [National Geospatial-Intelligence Agency 2007]. It replaces all JMTK functions with commercial off-the-shelf components, for application areas such as [GlobalSecurity.org 2007]:

- Visual: Presentation, overlays, symbology, event management.
- Non-visual: Terrain analysis, image analysis, threat envelopes, analytical tools.
- Spatial database: Raster, vector, text, imagery.
- Toolkit: Datum conversion, coordinate conversions, data optimization.

## **TBMCS**

Serving command and control of the air battle plan, the Theatre Battle Management Core Systems (TBMCS) provides tools to support air combat planning, coordination, and execution through the development and sharing of an operational picture [GlobalSecurity.org 2007]. Of particular interest to us is the Situational Awareness and Assessment (SAA) module, used to provide a configurable data display to support SA, and analysis tools to support the identification and intent of threats [Green 2001].

## **ICC**

Integrated Command and Control (ICC) is software for NATO air operation activities. It provides functional support for the most critical air command and control functions, such as planning and tasking, Airspace Control Order (ACO), Air Task Order (ATO) and Air Task Message (ATM) production, and current operations capability. It can also display tactical and operational pictures [NATO Programming Centre 2007].

## **FalconView.**

FalconView is a mapping system for the Portable Flight Planning Software (PFPS). PFPS is a suite of mission-planning tools that helps planners conduct effective and timely planning, as well as providing materials required for operations and post-mission debriefings. FalconView is used to display maps and geo-referenced overlays, along with information generated by other PFPS modules.

## **C2PC**

Command and Control PC (C2PC) is used to edit and share the COP across multiple service branches, using an open architecture based on Microsoft technologies so that it can easily integrate with desktop applications on a Windows system. It provides planning tools, decision aids, and display capabilities that can operate in tactical environments. The C2PC gateway can provide track data for multiple clients or gateways, to synchronize the COP, and the C2PC client provides the graphical interface to the COP [GlobalSecurity.org 2007].

## **CAAT-Xi**

The Canadian ATO/ACO Tool XML Interpreter (CAAT-Xi) is a light browser-based application to display Air Tasking Order (ATO) and Airspace Control Order (ACO) data in a

graphical and tabular format. It is capable of importing XML-formatted ATOs and ACOs, which it will parse and display along with other relevant mission information [Joint Warrior Interoperability Demonstration Management Office 2004].

### **CommandView**

CommandView is a system used by the Joint Staff to provide a high-level visual picture of military activity to be monitored [Hollands 2006]. The system displays a map with icons placed at locations of interest, and each icon has a different meaning. For example, there are icons to indicate the location of military exercises, operations and incidents. The user can zoom in on the map to get a more detailed view of the activities at a precise location. The system also includes various forms of textual information in menus surrounding the map; the textual information provides more detailed information about particular incidents or events.

## **2.4 Artificial Intelligence Planning**

There is a long tradition of planning research in the Artificial Intelligence (AI) community. Planning research in AI may provide the means to develop tools that can improve the operational picture through dynamic plan management, the focus of this research.

From the perspective of AI research, a planning problem can be stated as follows. Given a goal  $G$  and a set of possible actions, is it possible to find a sequence of actions  $A_1, A_2, \dots, A_n$  such that executing  $A_1, A_2, \dots, A_n$  will guarantee that  $G$  is satisfied? This simple statement leads to many natural questions about the nature of goals and actions. In order to make the problem more precise, AI planners normally use a formal representation language to describe planning problems.

Many approaches to AI planning are based on the language of the Stanford Research Institute Problem Solver (STRIPS) [Fikes and Nilsson 1971]. In this tradition, the state of the world is described by a set of sentences that are either true or false. A goal is also a set of sentences, describing some state(s) of the world that are considered desirable. Actions can then be represented as operators that make certain sentences true and certain sentences false. In this framework, a plan to achieve a goal is a sequence of actions that forces all the sentences in the goal set to be true. For a survey of approaches to AI planning, we refer the reader to [Ghallab, Nau et al. 2004].

AI planning is typically focused on general-purpose planners in relatively simple domains. As such, some problems of primary importance for AF planning are not treated in detail in approaches to AI planning. For example, many approaches do not draw a clear distinction between operational planning and strategic planning.

The main problem in AI planning is plan generation. Hence, the ultimate goal is to produce a system that automatically finds a plan to achieve a goal. A system that finds plans quickly is preferred over a system that takes a long time. In this document, we are not concerned with plan generation.

Regarding the suitability of AI planning in the context of this study, it is important to note that all approaches to AI planning require some formal approach to plan representation. The

approaches often address plan management at some level, and the underlying ideas are often applicable to AF plan management.

## 2.5 AF Plans

Plans provide information on the mission environment, which needs to be included in the COP for supporting SA [JIIFC PD 2005]. In order to effectively support dynamic plan management, it is important to consider the significant elements specific to an AF plan. However, academic literature does not generally discuss the specifics of AF plans and plan elements.

### 2.5.1 Plan Elements

From the most high-level perspective of AI planning, a plan simply consists of a set of goals or objectives, together with a sequence of actions. However, even for simple applications, this view is often too general. The actions in a plan are limited by temporal, spatial and/or resource-based constraints, among others. Similarly, goals may have important structure in terms of issues like sequencing or assets involved. In this section, we provide a high-level overview of the *plan elements* to be considered in this document. In Chapter 5, there is a detailed discussion of the plan elements relevant to the development of our proposed plan management software. These elements are comprehensively listed in Table 5-5, and their relationships are graphically represented in Figure 5-3.

In order to perform a literature review of documents relevant to AF plans, it is important to have an understanding of the significant elements specific to an AF plan. A precise specification of all plan elements can be abstracted from an Air Tasking Order (ATO), which is an AF document that details everything occurring a specific region at a specific point in time. The ATO includes, for example, every aircraft that is present at a location, the nationality of each aircraft, the mission that each aircraft is performing, and many other important pieces of information.

Rather than listing every piece of information that occurs in an ATO, we provide a list of the main categories of plan elements:

- Objectives: the desired outcomes of a plan.
- Tasks: the activities to be executed to achieve the objectives.
- Resources: resources required for plan execution, including personnel and aircraft.
- Constraints: precise restrictions on the execution of a plan.

We stress that this list is not intended to provide a precise specification of every component of a plan; it is simply intended to provide the most significant categories of elements. Elements in each of these categories have important components and structure. The level of detail considered and the amount of structure imposed on each component will vary depending on our plan representation. Note that our interpretation of the term “plan element” includes all components of a plan that are explicitly specified. There might be additional concerns that might be considered during plan management, without being elements of a plan. For example,

assumptions about the environment often play a role during planning, so assumptions must be represented at some level in plan management. However, for our purposes, assumptions are not considered elements of a plan.

Plan elements may have properties that are *static* or *dynamic*. Static properties are those that do not change for the entire execution of a plan, whereas dynamic properties change as the tasks are executed. For example, the nationality of a particular aircraft will not change during the execution of a plan. By contrast, the geographic location of an aircraft is likely to change repeatedly. For each element of a plan, it is necessary to specify the static and dynamic properties relevant for a given plan. An overview and discussion of the plan elements used in our proposed plan management software is given in Section 5.2.2.12.

Plan elements may be specified at different levels of abstraction. For example, a high-level description of a task might specify that a certain individual is to be transported to Ottawa. A lower-level description of the same task might specify that the individual is to be driven in a car to a particular military base, and then a particular plane is to fly them to Ottawa following a specific path. In later sections, we will discuss the *decomposition* of complex tasks into simpler tasks, and eventually into *actions*. However, this kind of decomposition is not only possible for tasks; it is possible for every category described above.

The specific plan elements that are represented can vary between different approaches to plan management. In the literature review, we present several different approaches to plan representation and we indicate the specific elements that are represented in each one. Based on this review, we then propose a specific set of precisely defined plan elements to be considered in the software component of this project. The choice of an approach to plan representation will directly impact the way in which plans can be forecasted, evaluated, and monitored, which will also influence which plan elements need to be considered for these categories of dynamic plan management. Even the approaches considered in each of these will change which plan elements are considered, at least to some extent. For example, different plan elements need to be taken into account when considering the rationale or objectives of a plan, which could be motivated by which of rationale-based or objective-based monitoring is used [Veloso, Pollack et al. 1998; Tate, Levine et al. 2000], among others.

## 2.5.2 Types of Plans

The CF define a plan to be “a proposal for executing a command decision or project”, prepared during the deliberate planning process [Department of National Defence 2004]. They define five particular types of plans:

- Operations Plan (OPLAN): Identifies the specific forces, functional support, and resources necessary to implement a defence mission that will be carried out within a specified time period.
- Contingency Operation Plan (CONPLAN): Reflects potential response options to address a defence mission that will be carried out within a specified time period, but without a set time of execution.
- Standing Defence Plan (SDP): A fully developed plan ready for execution, required by a commander to deal with a short/no-notice potential risk in order to fulfill their assigned role while in a peacetime stance.

- Support Plan (SUPLAN): A stand-alone document or annex within a main plan used to provide detailed direction and information on a particular aspect of the plan.
- Campaign Plan: A series of related plans or orders aimed at accomplishing a strategic or operational objective within a given time and space, prepared when military operations under consideration exceed the scope of a single major joint operation.

Of particular interest for this literature review are additional plans specific to the AF:

- Air Tasking Order (ATO): A tactical plan, associated with one single mission or several missions. Specifies what aircraft or unit is to carry out a specific mission, at what time, with what ordinance, and under what control agency. This is the primary method, when dealing with large numbers of aircraft, to control many events and avoid misunderstandings that could lead to a “blue on blue” engagement.
- Airspace Control Order (ACO): Provides overall direction for the conduct of air battle, specific direction on carrying out particular mission types, and puts into force the Airspace Control Plan and Airspace Control Measures which sketch out the altitude, geographic, and time limits of various tactical areas. It is an operational plan since more than one mission is covered.
- Course of Action (COA): A plan outline used to communicate one way to accomplish a mission. It includes a written description of the intent and desired outcome along with a diagram that relates a set of units and tasks to a geographic region. Normally, there are several different ways to accomplish a mission, and therefore several different COAs. Generally provides operational or strategic plan elements.
- Special Instructions (SPINS): Closely tied to the ACO, these provide the aircraft crew information concerning conduct of the mission, additional participants, and what COAs to take for a particular set of scenarios. Could apply to any of the AF plans listed.

## 2.6 Summary

The COP is a common framework for developing SA, providing commanders easy access to all information within their area of influence and interest. Plan representation, forecasting, analysis, and monitoring are a means by which to improve the COP in order to increase the SA of decision-makers. The goal is to provide a complete, unambiguous description of a plan, aid in predicting the possible outcomes of a plan, assess the quality and compare plans, and determine the appropriate reaction to change as plans evolve.

None of the known visualization tools currently available for working with a COP provide automated ways in which to improve the operational picture used by the AF. They provide a framework for viewing, editing and sharing information, but improving the picture is left to manual processes involving human operators. However, providing SA support through dynamic plan management could increase the effectiveness of the COP by improving the operational picture in an automated way.

Planning research has been developed largely in the AI community. Although much of this research has focused on generating plans, all AI planning requires some formal approach to plan representation. Plan management is addressed at some level, and the underlying ideas are often

applicable to AF plan management. AF plans tend to be specific to tactical, operational, and/or strategic levels of abstraction, which makes it easier to group plan elements by these levels. However, specific plan elements that need to be used for the dynamic plan management categories we are considering are largely be determined by the approaches considered in Section 3.

Ref: RX-RP-52-5076  
Issue/Revision: 1/3  
Date: DEC. 07, 2007

UNCLASSIFIED



**THIS PAGE INTENTIONALLY LEFT BLANK**



## 3 LITERATURE REVIEW

We provide a review of the existing literature that is relevant to dynamic plan management in the context of the AF COP. Note that we are not concerned with automated plan generation in the tradition of AI planning. A survey of approaches to AI planning in the military domain can be found in [Boukerche and Rioux 2003]. Instead, this literature survey focuses on plan representation, plan forecasting, plan analysis, and plan monitoring. We consider both military and academic sources.

In this chapter, brief summaries are provided for a collection of papers relevant to dynamic plan management in the military domain. Analyses of the most promising approaches will be presented in the next chapter.

### 3.1 Plan Representation

The manner in which plans are represented is of fundamental concern for any approach to dynamic plan management. As argued in [English 2002], the CF Aerospace Doctrine manual uses expressions that are both imprecise and subject to interpretation. Moreover, this fact has hindered discussion and debate about Air Force command and control arrangements. Choosing a specific, clearly defined representation for plans will remove this kind of ambiguity.

In the course of this review, existing military planning systems will be introduced and discussed. A high-level overview of existing military planning software can be found in [Boukhtouta, Bouak et al. 2006]. Several well-known systems and approaches are covered, including the following:

- Fox Genetic Algorithm (FOX-GA)
- Contingency Theatre Automated Planning System (CTAPS)
- Joint Assistant for Development and Execution (JADE)
- Dynamic Analysis and Re-planning Tool (DART)
- Anticipatory Planning Support System (APSS)
- Time-Phased Force Deployment Data Editor (TPEDIT).

However, the focus in this section is on the method of representation, rather than the details of the implemented systems.

Many different approaches to plan representation have been developed. We briefly list some of the approaches that have been explored in [Ghallab, Nau et al. 2004].

- Propositional logic and extensions
  - Satisfiability planning
  - Temporal logic
  - Dynamic logic
- First-order logic and extensions
  - STRIPS
  - Situation Calculus
- Explicit representation of constraints (for constraint satisfaction)
- Graphical approaches
  - Planning graphs
  - Transition systems (for model checking)
  - Task networks (simple and hierarchical)
- Temporal planning
  - Temporal databases and temporal operators
  - Chronicles
- Planning under uncertainty
  - Markov Decision Processes
  - Conditional planning with observation functions

The preceding representation schemes have primarily been developed in the AI community for general purpose planning. Our focus in this document is on plan representation in the context of military planning. As such, in the following sections, we only present summaries for papers that are directly relevant to plan representation for military plans.

### 3.1.1 Ontologies and Formal Languages

#### **“Coalition Battle Management Language (CBML) Study in Support of Maritime Domain Awareness” [Davenport 2007]**

It is important to know about the CBML since, as this report suggests, it is most likely to be the formal language adopted by the CF as it builds on systems already used by NATO and its allies. The report identifies opportunities for CBML to help with the Recognized Maritime Picture,

which parallel those for the AF COP, applications of CBML that will have a positive impact on CF operations, and presents a vision of how CBML will look and act.

CBML will provide a grammar, a syntax, an ontology and an assistant system to communicate command intent. It will have abilities or provide support for: data fusion, directing assets, visualizing intent, enabling Internet searches, battlefield simulation, and tracking asset readiness. However, note that a full specification of CBML has not yet been published.

**“Ontological models for military courses of action.” [Boury-Brisset and Champagne 2006]**

Provides a survey of the use of ontologies for plan representation. The report concludes that an ontology should be supported by a set of representative scenarios to facilitate the task, as building COA ontology requires an analysis of military knowledge. Challenges arise when ontologies originally developed for Army operations are being adapted for use in Air Force operations. Several different existing ontologies are considered in this report.

**“A Knowledge Acquisition Tool for Course of Action Analysis.” [Barker, Blythe et al. 2003]**

Uses a general-purpose action description language called SHAKEN for the analysis of COAs. A COA problem is described by a map/sketch, together with a story, a mission specification, and an estimate of the situation. A COA is then an overlaid sketch that provides a sequence of actions to take, given the underlying problem.

The software described allows users to enter very high-level visual descriptions of a COA, but then translates the results into a classical planning formalism that is well-known and understood. Users enter visual information using the plan sketching software NuSketch [Forbus, Usher et al. 2003]. The sketched plan is then translated into a formal description of the COA in the SHAKEN action description language, which encodes tasks in a STRIPS-like language[Fikes and Nilsson 1971].

**“TOPFAS (Tool for Operational Planning, Force Activation and Simulation).” [Thuve 2001]**

Describes the Tool for Operational Planning, Force Activation and Simulation (TOPFAS). This tool does not rely on a sophisticated formal approach to knowledge representation, but instead on a collection of relational databases to store all information relevant to planning. The initial motivation for TOPFAS is the fact that a traditional COA map is difficult to represent on a computer, with issues surrounding the best user interface addressing problems of screen size and resolution.

TOPFAS is not only for COA planning; it is intended to aid at all stages of the planning process. The software attempts to use graphical user interfaces as much as possible, together with sequences of collapsing menus. Different kinds of tasks are differentiated, including implied tasks, assigned tasks and subordinate tasks. All of the tasks for a particular problem are defined in plain text, through a typical brainstorming session.

The software is intended for use with NATO, so it includes a force-oriented task (FT). FTs represent all of the tasks that NATO forces can be asked to do. The main task for a user of TOPFAS is to assign FTs to all of the brainstormed tasks, in a manner that accomplishes the goal. The way this is handled in the software is to drag-and-drop FTs to associate them with the standard task types. There are a lot of FTs, arranged in a tree-structure that indicates the type of task, the typical phases of the task, and the three levels of tasks. FTs are also labeled with a type: operational, strategic, or tactical.

After assigning FTs to all of the tasks, timelines for tasks are defined using a Gantt-like representation. Locations for tasks are drawn on screen using polygons on a map, and units are assigned to tasks either manually or automatically. TOPFAS has several key outputs, including a COA overview with units, times, tasks, locations; a statement of requirements; an allied force list with command relationships; and an allied disposition list.

Underlying TOPFAS is a collection of relational databases storing the following information: NATO tasks; available units; military equipment, including make/model for each unit. Getting the appropriate information from all of these databases would be difficult by hand. One of the main contributions of TOPFAS is that it helps users use all of this data in the representation of plans.

**“Building and (Re)Using an Ontology of Air Campaign Planning.” [Valente, Russ et al. 1999]**

Reports on the development of Loom—an ontology for working with air campaign plans. Loom has several contexts for different features of plans, including weapons, forces, fuel, and time. These contexts are all modular and the paper explicitly deals with re-using established ontologies when appropriate. For example, there are many applications where time is a factor, so there are several existing ontologies for representing time; the approach in this paper describes how to use these existing ontologies in the context of Air Force planning.

The paper concludes that it is often easier to merge two existing ontologies rather than creating one from scratch. Loom can also be used in connection with the Mastermind plan editor. However, Mastermind uses context-free grammar rules to represent plans. This paper illustrates how this representation can be translated into an ontology, providing a bridge between two approaches to plan representation.

**“A Representation and Library for Force Support Objectives in Air Campaign Plans.” [Valente, Swartout et al. 1998]**

Focuses on providing a rich, structured representation of objectives. An objective consists of several components, including: title, measures of merit, action, effect, location, percent complete, review status, and parent objectives. Some of these components could also be

understood to consist of components. For example, we could enforce a formal structure on the title, in which the title itself gave an indication of the associated action.

A formal grammar is introduced for giving precise representations of objectives. The paper also introduces a typology of action verbs, in order to further formalize the description of objectives. A preliminary version of the objective grammar can be found in [Valente, Gil et al. 1996], where the authors indicated that all required objectives could be represented as <verb>-<roles> pairs, using a set of approximately 30 basic verbs.

**“Multi-Perspective Planning: Using Domain Constraints to Support the Coordinated Development of Plans.” [Tate, Dalton et al. 1999]**

Reports on the O-plan project, which was run under various names and funding agencies from 1984 to 1998. The project investigates multi-agent mixed-initiative interaction between a task-assignment agent and a command agent. Each agent keeps track of a to-do list, and uses a common representation of plans as constraints on behaviour. This paper describes a web-based interface to O-plan with a comparison matrix for choosing between different COAs. The comparison matrix shows COAs together with elements of evaluation provided by plug-ins from plan evaluators. The matrix also shows the steps in the planning process and any outstanding issues. Plans are represented in the formal language of [Tate 1996].

**“JADE: A Tool for Rapid Crisis Action Planning.” [Mulvehill and Caroli 1999]**

Introduces the Joint Assistant for Development and Execution (JADE), developed by DARPA and the Air Force research lab, which was specifically designed to assist with force deployment. Data about the force units available is stored using PARKA, which is similar to a relational database, but it also allows for the natural representation of hierarchical data. Traditionally, planning force deployment could take days or weeks; with JADE the process takes less than an hour.

JADE integrates several existing tools into a unified package. The software uses the Force Management and Analysis Tool (FORMAT), which is an environment to build force units in a hierarchy. JADE also uses Prodigy, which is a general-purpose tool for solving planning problems [Veloso, Carbonell et al. 1995]. Prodigy uses a domain theory involving object classes that represent agents and operators that represent actions. Planning problems are represented in this state-space, and are being solved using a backward chaining approach. In order to solve common problems efficiently, Prodigy also has a case-based replay mode, where old plans can be saved and retrieved when faced with similar goals.

Essentially, JADE provides a front-end linking FORMAT with Prodigy, which allows plans to be generated in the context of military force deployment. The underlying idea is a mixed initiative approach to force deployment, so JADE simply provides recommendations to the user based on the goals and the forces available. One of the key features of JADE is that it provides a modular treatment of force modules and plans. The drag-drop interface allows simple interaction with the underlying software.

**“Roots of SPAR - Shared Planning and Activity Representation.” [Tate 1998]**

Describes the roots of the well-known Shared Planning and Activity Representation (SPAR) approach. The importance of sharing information about plans and activities between multiple groups is stressed. To make the planning problem precise, the paper provides a hierarchical breakdown of a plan into all of its elements. For example, a plan can be defined in terms of activities and objectives, which can in turn be defined in terms of lower-level concepts like states-of-affairs. In terms of the ontology-based approaches to planning, the SPAR approach is flexible in that it allows different plan ontologies or grammars to be plugged in.

**“A Framework for Argumentation-Based Negotiation.” [Noriega, Sierra et al. 1998]**

Describes a framework that can be used to model the negotiation between multiple agents. This paper does not explicitly discuss planning, but the connection between argumentation and planning has previously been covered in [Sycara 1989]. In the present framework, each agent is able to perform problem-solving tasks, and negotiation occurs when one agent tries to convince another agent to solve a particular problem. A simple ontology is introduced to give a precise specification of a negotiation. Among other things, the ontology includes roles for each participant, communicative actions that can be performed, and domain specific information. Roughly, a negotiation starts when an agent sends a *proposal* to another agent, where a proposal is either an offer to perform some task or a request to have some task performed. The negotiation then consists of a sequence of counter-offers exchanged between agents until an offer is accepted, or one participant withdraws from the negotiation. Although military applications are not considered, related applications in Business Process Management are discussed.

**“Representing Plans as a Set of Constraints - the <I-N-OVA> Model.” [Tate 1996]**

Provides an introduction to the Issues-Nodes-Ordering/Variables/Auxiliary (INOVA) model for representing military plans. The basic idea is to represent a plan as a set of constraints, where the constraints can be described in a formal language called Task Formalism (TF). The paper introduces a hierarchy of different kinds of constraints, ranging from high-level to-do lists to specific temporal constraints on tasks.

The framework presented is able to represent plans at different levels of abstraction, depending on the level of detail required. In particular, plans are described using a triangle model where the horizontal dimension represents time and the vertical dimension represents the activity decomposition. The different levels of planning are also represented in the syntax of the plan description language. One of the stated goals of this paper is to provide a bridge between theoretical and practical work.

**“Applying an AI planner to military operations planning.” [Wilkins and Desimone 1994]**

Describes the System for Interactive Planning and Execution (SIPE-2), an AI planning system that has been used for air campaign planning. It uses a sort hierarchy to represent different kinds of objects, and a world model consisting of predicates describing properties of objects. There is also a set of deductive rules. Actions are represented by operators, which can represent specific tasks or general strategies. The system has two different interfaces: one for engineers developing the system, and one for military planners.

The sort menu and action menu is displayed hierarchically, and plans are shown graphically on the screen. The program supports automatic planning or mixed-initiative assisted planning. One of the main advantages of the system is that the hierarchical representation of plans is well-suited for the different levels of abstraction of military plans. One of the main limitations is that the system is not able to manage resources effectively, because the appropriate use of resources normally involves some trade-offs that cannot be automated by the system.

**“Arguing About Plans: Plan Representation and Reasoning for Mixed-Initiative Planning.” [Ferguson and Allen 1994]**

Argues that the normal representation of plans in AI planning is not appropriate for plan communication in mixed-initiative planning. Typically a plan is given by a sequence of actions. However, empirical observation suggests that people do not describe plans to each other in this manner. Instead, plans are described in terms of goals, sub-goals, and motivation. The paper introduces a representation of arguments, where an argument step consists of a proposition together with a justification. An argument is a sequence of argument steps, and the paper illustrates that arguments can be used to describe plans. It is argued that this representation is well-suited for communicating about plans, particularly in mixed-initiative planning.

**“Semantics for Hierarchical Task-Network Planning.” [Erol, Hendler et al. 1994]**

Describes planning in a Hierarchical Task Network (HTN). The HTN framework is not based on ontologies, but it is based on a formal specification language for planning problems. This approach differs from STRIPS-type planning in that plans are not just sequences of actions. Instead, a plan is described as a network consisting of several goals and tasks, with an explicit structure and explicit constraints on the order tasks are performed. One important feature of the HTN framework is that it is possible to represent complex tasks and planning involves decomposing a complex task into primitive components. In this manner, plans can be represented at different levels of abstraction.

### **3.1.2 Probabilistic Approaches**

**“Model-based Development of a Course of Action Scheduling Tool.” [Kristensen, Mechlenborg et al. 2006]**

Discusses the Course of Action Scheduling Tool (COAST) and is mostly concerned with describing planning algorithms in COAST. However, we focus this summary solely on the portion of the paper dealing with plan representation. One of the goals of the COAST project is to illustrate how formal methods can be used to represent military plans in a manner that is transparent to the user.

The underlying formal representation employed by COAST is based on coloured Petri-nets (CPN), a formal approach to reasoning about concurrent events. A CPN consists of a network of nodes and arcs. Some of the nodes represent states of the system, and some of the nodes represent actions. The arcs indicate how the state of the system changes when actions are executed. The state of the system includes a set of “coloured” tokens; the notion of “colour” is actually very general—the colour of a token is simply an associated collection of data.

The COAST software provides a graphic user interface that allows users to define and manipulate plans, without knowing anything about the underlying CPN representation. For military planning, the initial state is a collection of facts represented by propositional variables, together with a collection of tokens representing tasks. The “colour” of the task-tokens contains: the task name, preconditions, postconditions, resources, triggers, probability of success, and task duration. Including the probability of task success is important in military planning, where tasks may fail.

**“Decision-Theoretic Military Operations Planning.” [Aberdeen, Thiébaux et al. 2004]**

Describes how Markov Decision Processes (MDPs) can be used for operational planning. An MDP consists of: a set of states, a set of actions, an initial state, a set of terminal states, probabilities  $\Pr(s'|s, a)$ , and a cost associated with each state. A solution is a policy function making non-terminal states to actions, and a great deal of work on MDPs is focused on finding minimal cost solutions.

One of the important features of an MDP in the context of Air Force planning is that it allows specifying the probability that a given task will fail. Another important contribution of this paper is the introduction of a schedule tree of contingency plans. Each branch in the schedule tree can be labeled with the probability that the contingency will be needed, the probability it will succeed, current facts and resource usage.

**A Fuzzy Logic Approach for Intelligence Analysis of Actual and Simulated Military Reconnaissance Missions [Ragsdale, Butler et al. 1997]**

Describes the use of fuzzy logic in military decision-making. Fuzzy logic is an extension of classical logic that can be used for modeling reasoning under uncertainty. Classic logic is restricted to sentences that can be true or false. By contrast, in fuzzy logic sentences can hold many different "truth" values. In this paper, fuzzy logic is used to assign various assessment values to different regions. For example, each region is assigned a value representing the risk associated with sending a unit to that region. A fuzzy logic inference engine is then used to perform analyses of reconnaissance missions.

### 3.1.3 Visualization

**“Observation of computer-supported, collaborative work tool usage during briefing and debriefing phases of coalition mission training research for Maple Skies 05-06.” [Bennett and Lamoureux 2006]**

This is actually a study of user behaviour in mission planning, using a specific set of technological tools. Specifically, the paper discusses planning in a computer-supported, collaborative work system by a group of geographically separated air force users. Planning is divided into mission tasking, detailed flight plan development and coordinated briefing.

The technology involved included interactive whiteboards, voice/video conferencing, and computer software (PowerPoint, FalconView, DCS mission log viewer). Video for analysis was collected during the planning process for a simulated operation. The most important



contribution of the paper is the discussion regarding the strengths and weaknesses of interactive whiteboards for planning.

The study found that interactive whiteboards displaying information were able to focus the users' attention, even more than a computer screen displaying the same information. Using a whiteboard for sketching certain aspects of a plan is an intuitively appealing approach. However, information was often lost when a whiteboard was used remotely, because the precise meaning of hand gestures was lost.

#### **“A Visualization Environment for Planning.” [Vrakas and Vlahavas 2005]**

Describes ViTAPlan-2, a tool for visualizing plans. The system is built on top of the PDDL planning language [Fox and Long 2002] and it allows a user to define and view plans using entity relationship diagrams. In this context, an entity relationship diagram is a system of nodes and arcs, where some nodes represent entities and some nodes represent relations. Arcs connect entities and relations, indicating how the entities are related. For example, there might be an entity representing a room, an entity representing a ball, and a location relation. Arcs between nodes could then illustrate that the ball is located in the room. The software includes a tool for defining maps that specify the relations between objects, and for specifying plans. The visually defined plans can then be automatically translated back in to the PDDL planning language.

#### **“Sketching for military courses of action diagrams.” [Forbus, Usher et al. 2003]**

Describes NuSketch, a piece of software that allows COA plans to be visually sketched on a screen. The basic idea is to mimic the manner in which COAs have traditionally been marked on paper. The software uses layers in the same manner that layered acetate has been used traditionally to separate different aspects types of units; for example, one layer might be used for friendly forces and a second layer for enemy forces.

There is an underlying, structured approach to knowledge representation and the glyph map used by NuSketch provides a simple visual interface. Units are represented by glyph symbols, and they are added to a sketch by selecting from a glyph map. The intent of different units is also represented visually, by sketching symbols from a simple grammar. The times allotted for various tasks are given through a Gantt-style representation of time.

The distinguishing feature of the NuSketch software is the emphasis on understanding what the things on a plan sketch actually mean; this is handled in the approach to knowledge representation underlying the user interface. In principle, the NuSketch software could be modified to support different ontologies.

## **3.2 Plan Forecasting / Projection**

Plan forecasting refers to the process in which we attempt to determine the outcome of a given plan. Since plans are normally defined to achieve or maintain some set of objectives, plan forecasting is a problem of central importance. In this section, we survey some existing work on forecasting. Note that some of the papers reviewed in this section also appear in other

sections of this literature review; the summaries in this section are distinguished by a focus on forecasting and projection.

Before continuing with the paper summaries, it is worth noting that the plan forecasting problem is related to the notion of effects-based operations [Williams 2002]. This is an approach to planning where a plan is designed to ensure a certain effect is obtained, rather than focusing on a broad objective. In the context of plan projection, effects-based operations require a planner to focus on the possible effects that can actually be obtained. Generally speaking, this should produce goals that are more specific and plans that are more targeted. Having made this observation, we remark the distinction between effects-based and objectives-based planning is usually not a serious concern in formal approaches to planning due to the fact that objectives are represented in a very flexible manner.

One of the main problems that must be addressed in plan forecasting is the representation of uncertainty. Many different formal approaches to reasoning about uncertainty have been explored, including non-monotonic logics, fuzzy logics, and belief networks. In this document, we focus primarily on quantitative representations of uncertainty, based on probabilistic methods. The remainder of this section includes a set of paper summaries.

**“Coordinated Plan Management Using Multiagent MDPs.” [Musliner, Durfee et al. 2006]**

Not concerned directly with military planning, but with planning in general, the paper uses MDPs to represent complex hierarchical task networks in a multi-agent environment. Every task has an assigned duration, along with multiple possible outcomes; each outcome is assigned a quantitative measure of desirability. In general, we are interested in finding a joint policy for all agents that will achieve some goal. A joint policy is a mapping from states to actions that indicates, for every state of the world, exactly what each agent should do.

Finding a joint policy that is maximally desirable is a difficult problem, and the paper focuses on the difficulties that arise in the coordination of agents. From the perspective of a single agent, forecasting the outcome of a plan is a difficult probabilistic exercise. The paper is relatively preliminary, and it does not get far beyond illustrating the basic problems that arise.

**“Decision-Theoretic Military Operations Planning.” [Aberdeen, Thiébaux et al. 2004]**

Notes that MDPs assign a probability of success with each task, and that the set of contingency plans can be stored as a tree. Each branch of the tree indicates the probability that the contingency will be needed, along with the probability of success.

Also discusses redundancy as an important feature that should be considered in military planning, and must be accounted for in plan projection. When tasks are able to fail, then plan projection may indicate that an objective is unlikely to be obtained. One way to guard against this is to introduce redundant tasks, to increase the probability that a specific sub-goal of a plan will be met.

**“A Coloured Petri-net Based Tool for Course of Action Development and Analysis.”  
[Zhang, Kristensen et al. 2002]**

Focused on the COAST system, but with an emphasis on determining the possible outcomes of a COA. The COAST system uses CPNs to represent plans, which permits events to have probabilistic effects. As such, each task in a plan may have several possible outcomes, each with an associated probability.

COAST uses an approach to plan forecasting that is particularly appropriate if we have accurate probabilistic assumptions about the effects of all tasks. To model plan execution, the COAST system generates every possible state-space graph that can be obtained from the underlying Petri net. The outcome of a plan can be determined by looking at all of the state-space graphs, and probabilities associated with each possible outcome of each event. In this manner, one is able to determine the probability that a given plan results in a given, desired outcome. If the probability of success is not high enough, the plan can be refined, by examining the collection of state-space graphs.

**“On the Role of Humans in Enterprise Control Systems: The Experience of INSPECT.”  
[Valente, Blythe et al. 1999]**

Discusses the Intelligent System for Air Campaign Plans Based on Expect (INSPECT), which combines the Loom ontology with a formal approach to knowledge acquisition. The paper stresses the importance of human involvement in evaluating and critiquing plans. It is mostly focused on plan analysis, so it will be discussed in detail in the Section 3.3.

It makes an important contribution in terms of plan forecasting, where the framework performs automated resource checking for a given plan. When a user defines a plan to achieve a particular objective, the system automatically checks to see if there are appropriate resources available for the required tasks. This is an important form of preliminary plan forecasting.

**“A Representation and Library for Force Support Objectives in Air Campaign Plans.”  
[Valente, Swartout et al. 1998]**

Main contribution, in the context of plan forecasting, is the presentation of a set of plan templates with known outcomes. Using the structured language for representing objectives, one can simply use the plan templates to define a new plan that achieves a formally defined objective. In a sense, this plan library allows a user to avoid the projection problem for a set of commonly required objectives.

**“Dynamic Visualization of Battle Simulations.” [Cohen, Davis et al. 2000]**

Introduces a visualization tool for simulated war games where task have probabilistic outcomes. The visualization allows users to look at the current state and project the most likely future states. The Abstract Force Simulator is used to define the available units and adversaries, and then hundreds of simulated battles can be simulated on multiple timelines. Each simulated battle is called a trial. Let  $x$  be a variable ranging over states and let  $t$  be a variable ranging over times. The probability of an outcome is computed through the following formula:

$$P r ( w i n \mid x, t ) =$$

number of winning trajectories through  $x, t$   
total number of trajectories through  $x, t$

This information can be visualized in a three dimensional map, representing states, times, and success probabilities. Pits in the graph represent unlikely outcome states at a particular point in time, and peaks represent likely outcomes. The paper suggests that military commanders are able to view such maps, and project the likelihood of various outcomes that will result from different sequences of actions.

**“AsbruView: Visualization of Time-Oriented, Skeletal Plans.” [Miksch, Kosara et al. 1998]**

Describes how temporal information can be incorporated in a skeletal plan. Skeletal plans are plan schemata that can be described at different levels of abstraction. Using skeletal plans allows a user to defined re-usable plans, and to allow plans to be filled in at the appropriate point in the planning process. Unfortunately, subject matter experts find skeletal plans difficult to read and understand.

In order to facilitate the use of skeletal plans, this paper attempts to provide some methods for the graphic display of plans over time. The system is called AsbruView, and it incorporates metaphors such as tracks and traffic to represent the evolution of a plan over time. Graphically, a plan is represented as a three dimensional block, where horizontal direction represents time, the vertical dimension represents plan decomposition, and the depth direction is used draw multiple plans on the same timeline. This representation facilitates forecasting the outcome of multiple plans at any point in time. It has been argued that AsbruView provides a better model for temporal projection than project management tools such as Gantt charts, at least in the management of medical treatment plans [Kosara and Miksch 2000].

### 3.3 Plan Analysis / Evaluation

Whereas plan forecasting is focused on determining plan outcomes, plan analysis is focused on assessing the quality of the plan. There are many senses in which one plan may be judged superior to another. For example, all other things being equal, a plan that costs less money to execute might be more desirable than an expensive plan. In more subtle cases, assessing the quality of a plan might require expert level knowledge about military planning domains.

**“Decision-Theoretic Military Operations Planning.” [Aberdeen, Thiébaux et al. 2004]**

Using MDPs to represent plans facilitates plan evaluation and, from the most general perspective, a solution to an MDP planning problem is a minimal cost policy. The cost of a solution depends on the application domain, but MDP theorists tend to prefer quantitative measures of cost.

The authors indicate that military experts do not like the idea of assigning quantitative cost values for different tasks, preferring instead to rank different costs. For example, a task that takes a long time might be understood to cost less than a task that is financially expensive.

The planning process is then performed repeatedly for each cost. This is a much more involved planning process that one would have with quantitative ranks, but the authors acknowledge the importance of expert opinion. In the paper, the ranking of different costs is faked by using exponentially increases quantitative costs. Algorithms, heuristics, and experiments are also discussed.

**“A Knowledge Acquisition Tool for Course of Action Analysis.” [Barker, Blythe et al. 2003]**

Describes a specific approach to plan analysis in which the authors designed a critiquing score for COAs. Critiques are entered by military planning experts, using SHAKEN as the input language. The expert critiques are then used to assign scores to different COAs, and these scores can be used to pick the best COA for a given objective. The initial feeling of the authors was that the experts would not need to learn a great deal in order to enter critiques, but they acknowledge that there were problems with the experts’ level of comfort and understanding with respect to SHAKEN.

**“A Mixed-initiative Framework for Robust Plan Sketching.” [Myers, Jarvis et al. 2003]**

In this paper, a plan sketch is essentially defined to be an incomplete plan. The basic idea is that a user should be able to provide a quick sketch of a plan, and then ask a system to fill in the details. Plans are represented by task networks, and plan sketches that are passed to the software will be filled-in or repaired to make complete plans. In terms of plan evaluation, the main contribution is the suggestion that, by dropping and adding actions, the software can automatically repair erroneous plan sketches.

**“TRELLIS: An Interactive Tool for Capturing Information Analysis and Decision Making.” [Gil and Ratnakar 2002]**

Introduces an interactive tool for capturing information analysis and decision-making. The tool presented is intended to support general-purpose decision-making, and it has had some limited application in military domains. The paper introduces a language for annotating plans to justify decisions that are made. The language consists of text-based statements, together with specialized connective terms such as “because” and “according to.” Using the formal language introduced in this paper allows a planner to provide a justification for all aspects of a plan, which facilitates any plan analysis being carried out by an independent party.

**“COA Advisory System Based on Multiple Criteria Decision Analysis.” [Bélanger, Guitouni et al. 2000] and “A Decision Support System for COA Selection.” [Bélanger and Guitouni 2000]**

Finding and evaluating various COAs is an important process, but time constraints make it difficult to do a comprehensive analysis in most cases. The traditional approach to evaluating COAs is through war-gaming, but this comparison does not always capture the differences one is interested in. This pair of papers suggests a different approach.

A set of criteria is introduced for the evaluation of a COA and these criteria are weighted to indicate the relative importance of each. The best plan is then determined through Multiple

Criteria Decision Analysis (MCDA), which is explained in the papers at a high level of abstraction. The COA advisory system is described, including a functional flow-chart of the decision process as well as a screenshot of the tool.

A graphic tool is used to show relative importance of criteria and MCDA is used to recommend the best COA. The authors stress that decision support systems must be integrated with the users' habitual tools and processes, and they point out that users of the COA advisory system indicated a key feature is being able to compare results with COAs used in the past.

**“INSPECT: an Intelligent System for Air Campaign Plan Evaluation based on EXPECT.” [Valente, Gil et al. 1996] and “On the role of humans in enterprise control systems: the experience of INSPECT.” [Valente, Blythe et al. 1999]**

Describes the INSPECT framework for evaluating air campaign plans, where humans define the plans and objectives; the only role played by the system is the evaluation of the given plans. Plans are entered using an extension of the Air Campaign Planning Tool (ACPT); the original ACPT allowed plans to be entered in natural language, but the version for INSPECT uses a case grammar. After plans are entered, users run INSPECT to check for errors. Since plans are typically complex, user-entered plans tend to be error-prone.

The authors describe INSPECT as being analogous to a spell-checker, in that it simply checks plans for a number of structural errors. A tool for entering classes of errors is provided, and any user can define new errors. Using this tool, a collection of military planning experts entered a library of common errors which INSPECT can now detect automatically. When an error is detected, the user is given a warning, a caution, or a note as a response; INSPECT finds the errors, but it asks the user to correct them.

It is significant to note that many of the defined errors are specific to the military domain. For example, one principle of military planning is that every objective must serve a higher-level objective. However, this is the kind of error that will be difficult for a user to detect when entering a large plan. Other errors that are detected include task sequencing problems, overly general objectives and resource-feasibility.

**“Evaluating Air Campaign Plan Quality in Operational Settings.” [Miller, Militello et al. 1996]**

Points out that plans generated by AI systems are often deemed to be inappropriate by experts in air campaign planning. Even when the terminology of AI planners and military planners coincides, it is not always the case that the interpretation of the terminology is the same. For example, a “robust” plan is understood to be one that works well under many different circumstances; however, AI planners and military planners do not necessarily agree on precisely what the different circumstances are.

The goal of this paper is to determine how to design systems that make “good” plans from an expert perspective. The paper describes how this might be done through Cognitive Task Analysis (CTA). Essentially, the authors interview a group of military experts to try and determine an inventory of plan features that are related to the assessed quality of a plan. Eight general features are identified, such as “a good plan can absorb changes in assumptions.” It is

suggested that it would be useful to incorporate these features when designing plans for air campaigns.

**“AI on the Battlefield: An Experimental Exploration.” [Rasch, Kott et al. 2002]**

Describes the results of experiments testing the suitability of AI based decision-aids, for the most part, used in COA planning and analysis. Specifically, the Integrated Course of Action Critiquing and Elaboration System (ICCES) was used in the US Army Brigade Military Decision Making Process to demonstrate the suitability of this technology.

The outcome of COA planning and analysis is usually visualized and recorded in a synchronization matrix, which is a type of Gantt chart. However, it was found that users had difficulty understanding a computer generated synchronization matrix, even though it is a familiar representation of information to military planners.

The authors believe such a matrix may be better at capturing an individual's reasoning for their own sake, rather than presenting someone else's results for the sake of understanding their reasoning. In other words, it would be better to support the visual processes familiar than summarize their results in a textual manner. They also raise concerns regarding the appropriate size of such a matrix.

**“Three-Dimensional Visualization of Hierarchical Task Network Plans.” [Kundu, Sessions et al. 2002]**

Describes a method to display hierarchical task network plans in a three-dimensional framework. The authors believe their method of visualizing hierarchical task networks allows users to analyze tasks and constraints generated by AI planners, without understanding AI planning. The visualization provides the user with a perspective of the flow of tasks, with the ability to investigate the details of the individual tasks. With three dimensions to work with, multiple levels of a plan can be visualized in an efficient manner.

Blocks are used for representing tasks, with subtasks placed inside parent tasks, in an attempt to reduce visual complexity. Subtasks are only shown when tasks are decomposed. Serial tasks are placed after each other, in a stair-like fashion, and constraints are shown as arced lines joining tasks. Image manipulation techniques are also available to users. However, the work is preliminary and only encompasses a fraction of the available data in a plan.

## 3.4 Plan Monitoring

Plan monitoring involves observing a plan as it unfolds, and checking to ensure that changes in the environment do not undermine the goals of the plan. As such, plan monitoring may involve monitoring of the environment as well as monitoring the interaction between several plans. Plan monitoring is covered in [Ghallab, Nau et al. 2004] with respect to several different the representation schemes. In this section, we present a preliminary review of some important papers on this topic.

**“Interactive Execution Monitoring of Agent Teams.” [Wilkins, Lee et al. 2003]**

Describes a domain-independent approach to plan monitoring which provides "alerts" to notify the user about any problems. Alerts are given a value, and the value of an alert is used to determine the appropriate reaction. This basic framework is used for mixed-initiative plan monitoring, by implementing executive assistants to help a human user. Several plans might run concurrently, each representing the activities of a particular team or of a planning agent. Individual planning agents do not communicate directly. Instead, the framework employs report-based monitoring where each planning agent applies plan-recognition techniques to determine the other agents' intentions based on observable actions. Although the framework is a general-purpose approach to plan monitoring, it is demonstrated by example that it is suitable for monitoring small unit army operations.

**“Using AI Planning Technology for Army Small Unit Operations.” [Tate, Levine et al. 2000]**

Focuses on the use of sensors and causal links for plan monitoring. Several different approaches are used for the underlying representation: objectives are represented with INOVA, actions are represented using Business Systems Method Design, and Oplan is used to find suitable actions to achieve a goal. No serious approach to plan checking is proposed, although it would be possible to plug-in an interface to INSPECT.

It is pointed out that a plan should also include goal-structure information, such as preconditions for later actions that must be satisfied by earlier actions. The goal-structure of a plan is represented by causal links [Tate 1977]. A causal link consists of a sequence of events, a condition  $p$  and an event with precondition  $p$ . A valid plan must ensure that all associated causal links are satisfied.

It is proposed that plans can be monitored during execution through sensors and feedback from units. The goal structure of a plan can be used to guide the placement of sensors, and to provide units guidance on the required feedback. The goal structure of a plan also allows a planner to see when a plan must be modified, because a precondition for some later action was not obtained. When a problem is encountered, the system tells the user which actions have been done and which actions remain to be done. The planner can then add new actions to ensure that the preconditions of the remaining actions will be met.

**“Continual Planning with Time-Oriented, Skeletal Plans.” [Miksch and Seyfang 2000]**

Suggests an approach to re-planning referred to as skeletal plan refinement. Skeletal plans are plan schemata that can be described at different levels of abstraction. A high level skeletal plan is intended to capture the essence of a plan, but the details are left open. In this manner, the details can be specified when required. Skeletal plans can be represented in the Asbru plan-representation language. The paper describes how such plans can be continually monitored and refined in the Aasgard planning framework.



**“There's More to Life than Making Plans: Plan Management in Dynamic, Multiagent Environments.” [Pollack and Horty 1999]**

Describes some of the main problems faced in plan monitoring, and outlines some potential solutions. For example, plan monitoring involves monitoring the environment, assessing alternatives to the current plan, changing the current plan, and co-coordinating with other agents.

This paper argues that an agent can not simply monitor the environment during plan execution; an agent must also keep track of the rationale for all actions executed and rejected. This capability has been tested in the Prodigy system [Veloso, Carbonell et al. 1995].

It is also argued that a cost should be associated with each plan, so that minimal cost alternatives can be determined when plan monitoring suggests that the current plan is insufficient. These capabilities have been implemented in the Plan Management Agent (PMA) [Pollack, Tsamardinos et al. 1999].

**“Continuous Planning and Collaboration for Command and Control in Joint Synthetic Battlespaces.” [Gratch and Hill 1999]**

Addresses continual planning in a military domain, modeling Army Attack Helicopter Battalion operations. Introduces the notion of continual planning at a general level, and then describes how to implement a continual planning system in a system called Soar/CFOR. Plans are represented as sequences of tasks, and each task consists of preconditions, completion conditions, interruption conditions and a responsible agent.

Tasks can typically be decomposed into subtasks, allowing plans to be described at different levels of abstraction. The Soar/CFOR planner supports plan monitoring by continuously obtaining information from sensors and using the sensed information to define constraints on the plans being executed. The planner uses several basic operators for the modification of plans.

**“A Survey of Research in Distributed, Continual Planning.” [desJardins, Durfee et al. 1999]**

Of particular interest is the work on co-operative distributed planning, in which several planning agents work together to find a joint plan to achieve a common goal. Planning in real applications invariably involves a dynamic environment, and therefore continual planning becomes necessary. Distributed planning refers to planning in a context where the interaction between multiple plans must be considered. This paper provides a survey of some of the key work on distributed continual planning.

Much on co-operative distributed planning is in the framework of Hierarchical Task Networks (HTNs) [Erol, Nau et al. 1994], using an abstraction-based plan representation. The idea is that a plan is initially formulated at a relatively abstract level, and it is successively refined as more information is acquired regarding other agents plans. Two continual planning systems based on HTNs are discussed: NOAH [Sacerdoti 1977; Corkill 1979] and DSIPE [desJardins and Wolverton 1999].

**“Rationale-Based Monitoring for Planning in Dynamic Environments.” [Veloso, Pollack et al. 1998]**

A key problem is determining which information must be monitored and which information can be safely ignored. In plan monitoring, it is not advisable to monitor every piece of information available regarding the environment. Much of the information is not relevant to the execution of the current plan, and monitoring too much information makes plan management computationally difficult. This paper solves the problem of rationale-based monitoring.

The idea is to identify those features of the environment that are relevant to the execution of a given plan. When one of the designated features changes, the planning system must perform two steps. First, the system must determine if the detected change necessitates a change in the plan. Second, if the plan is to be changed, the system must determine a new plan.

The features monitored can be determined by analyzing the initial plan to determine sub-goals and resource constraints. Plan modifications can be defined in terms of simple operators that add or remove individual tasks from a plan. Implementation considerations are provided in the context of the Prodigy planning system.

**“Goal Transformations in Continuous Planning.” [Cox and Veloso 1998]**

Static plans executed in a dynamic environment generally will not succeed. One typical approach to plan management suggests that a plan needs to be monitored and refined during execution. In this paper, it is suggested that modifying the plan is not sufficient. In addition, the objectives of the plan must also be modified during plan monitoring.

A set of goal transformations is introduced, indicating precisely how and when goals should change. One case when goals must be modified occurs when sensors indicate that the environment has changed in a manner that makes the original goal insufficient. Another important case of goal change occurs when a lack of resources makes it impossible for a planner to achieve the current goal. A cost-benefit analysis is used to determine when goal transformations should occur.

**“Planning in Dynamic Environments: The DIPART System.” [Pollack 1996]**

Reports on the Distributed, Interactive Planner’s Assistant for Real-Time Transportation Planning (DIPART), which consists of a simulated environment together with a collection of nodes responsible for generating and managing a set of plans. The intention is that DIPART can assist a human planner with the management of plans in a dynamic environment, with each node essentially performing process scheduling. As the environment changes, some of the plans will become flawed; we say that a plan has become flawed when executing the plan no longer ensures that the desired outcome of the plan will obtain.

DIPART detects and repairs flawed plans, using several different algorithms. Emphasis is put on cost-based flaw-detection, where the cheapest flaws are corrected first. Emphasis is also put on earliest-deadline algorithms, where an external schedule indicates which plans must be successfully completed first. DIPART must deal with a lot of inconsistent information, including conflicting information from the human user and conflicting flaws at different nodes. The focus of this paper is mainly on efficiency and resource allocation.

**“CommandView User Interface Evaluation: Preliminary Heuristic Analysis Results.”  
[Hollands 2006]**

Presents an evaluation of the user interface for CommandView, which provides key information to military decision makers. CommandView is not intended as a tool for defining or representing COA plans. Instead, the approach taken by CommandView is to provide a map with labels indicating the location of all military units and events of importance. This view does not necessarily assist with monitoring a specific sequence of tasks. However, the software does provide a summary of global information at a higher level of abstraction. This global information may be relevant to specific courses of action, and CommandView provides a simple approach to monitoring this information.

The interface is evaluated with respect to several factors, including the visibility of system status and efficiency of use. The evaluation is performed using a well-known approach called heuristic analysis. Overall, the analysis suggests that CommandView provides a great deal of important information in a comprehensive format. However, it is pointed out that there are some problems regarding the consistency of organization of that information.

**“A Visualization Environment for Planning.” [Vrakas and Vlahavas 2005]**

Previously discussed in the section on plan representation, this paper also indicates that the ViTAPlan-2 framework can be used for visual plan monitoring. There are two different ways to monitor a plan. First, the system allows the user to view the entity-relationship diagram as it changes. Second, plans can be monitored on a timeline illustrating the entity relationship diagram associated with each step on a global clock.

**“Visual Correlation for Situational Awareness.” [Livnat, Agutter et al. 2005]**

Introduces VisAware, a conceptual framework for presenting visual information about events in a manner that effectively supports decision-making. In the context of plan monitoring, the events could be dynamic plan elements with What, When, and Where attributes, which are needed in this visualization framework. The focus in the paper is on describing how information about events should be visually displayed, and describing how correlations between events should be automatically determined by the system, in order to support SA. Event correlation aims to establish the confidence an agent should have in a particular event's occurrence.

The visual display of events in VisAware provides a sophisticated view for the purpose of increasing SA. However, the visualization relies on events remaining in a fixed position on the display. In other words, events relating to moving objects such as aircraft would not be appropriate to display with this system.

**“A Visualization Approach for Collaborative Planning Systems Based on Ontologies.”  
[Lino and Tate 2004]**

Plan monitoring would be facilitated by a visual display of the plan as it unfolds. However, this paper points out that existing planners tend to be weak in terms of visualization. It is proposed that plan visualization can be divided into a knowledge representation problem and a reasoning problem.

Representation can be addressed using an appropriate ontology, and then reasoning is the problem of mapping an ontology on to some visual display. The paper suggests several ontologies for representing different aspects of a plan using the Web Ontology Language (OWL). The paper is relatively preliminary, and it does not get into the details of how this can be translated into a picture.

## **4 SYNTHESIS**

In this chapter, the most promising approaches discovered in the literature review are revisited. We first discuss the measures of efficiency that will be applied during the synthesis. Then, with respect to the four major themes of this literature review, each approach is described in more depth including detailed descriptions, examples and analysis. The overall strengths and weaknesses are then considered, along with an identification of the approaches with the most potential for dynamic plan management.

### **4.1 Measures of Efficiency**

When choosing a particular approach to plan management, there are many factors that should be considered with respect to plan representation, plan analysis, plan forecasting, and plan monitoring. In this section, several important factors are proposed based on the findings of our literature review. They are given as yes-no questions to simplify the comparison of different approaches to plan management. However, the distinctions between approaches with respect to these features may be subtler than a yes-no answer. This will be clear in the discussion of strengths and weaknesses of the approaches investigated.

#### **4.1.1 Plan Representation**

- Plan abstraction: Can plans be represented at different levels of abstraction?
- Task specification: Is there a precise, unambiguous description of all possible tasks performed in an Air Force plan?
- Constraint specification: Is there a precise, unambiguous description of all personnel and equipment used in an Air Force plan? (including resource constraints and temporal constraints)
- Objective specification: Is there a precise, unambiguous description of the objectives of an Air Force plan?
- Definability: Can plans be defined or specified by military planning experts without excessive training?

- Readability: Can plans be read and understood by military planning experts without excessive training?
- Uncertainty: Can uncertainty about the outcome of a task be represented?

#### **4.1.2 Plan Forecasting/Projection**

- Resource checking: Are sufficient resources available to execute the plan?
- Pre-processed outcomes: Can the most likely outcomes of standard plans and tasks be projected in advance without duplicated effort?
- Probabilistic forecasting: Can outcomes be projected for plans involving tasks with uncertain outcomes?
- Complete on time: Will the plan be completed on time?
- Plan Status: Can the status of the plan be predicted at any point in time?
- Impacts of plan changes on future plans: If changes in the execution or outcome of a plan affect the future execution of a plan that depends on it, can this dependence be identified and highlighted?

#### **4.1.3 Plan Analysis/Evaluation**

- Military expertise: Can plans be evaluated according to criteria set out by military planning experts?
- Structural analysis: Does the definition of a plan contain errors in structure that make the plan invalid?
- Semantic analysis: Does the definition of a plan suggest a course of action that is unwise or inappropriate?
- Ranking: Can plans be ranked so as to choose the best one from a set of plans that achieve the same goal?

#### **4.1.4 Plan Monitoring**

- Feedback guidance: Does the monitoring system provide guidance regarding the information that must be obtained from field units during operation to ensure a plan succeeds?
- Assisted re-planning: Does the monitoring system provide guidance regarding changes to be made when original plans go awry?
- Violations: Does the system provide feedback on violated constraints, objectives, or assumptions?
- Interacting plans: Can multiple concurrent plans be monitored?

These issues will be revisited in the following sections.

## 4.2 Description of Most Promising Approaches

In this section, we consider the approaches reviewed in Chapter 3 in greater detail. We structure the discussion around three main categories of plan representation. This structure reflects the fact that representation is, in a sense, the first problem that must be addressed. After a particular approach to representation is selected, then it is possible to consider problems of forecasting, analysis, monitoring or visualization.

Since our focus is on approaches to plan representation, many general-purpose planning systems will not be discussed. For example, existing systems such as GraphPlan [Blum and Furst 1997] and Prodigy [Veloso, Carbonell et al. 1995] have not been considered in detail. These systems are focused primarily on plan generation. Planning systems of this nature have been discussed in this document, provided that the underlying representation of plans is useful in the context of military plan management. For example, the O-Plan system is actually a general-purpose planner, but it has been discussed in as much as it contributes to plan representation. Similarly, Prodigy is mentioned in passing, but only as it contributes to the architecture of a larger plan management system.

On a similar note, there has been little or no mention of many existing planning paradigms, such as satisfiability planning [Kautz and Selman 1992] or case-based reasoning [Bergmann, Munoz-Avila et al. 1998]. These approaches rely on relatively simple description languages. For example, satisfiability planning represents a planning domain in terms of propositional logic. In practice, it is not realistic to employ such a simple language for the representation of military planning domains.

One general-purpose planning paradigm that has been introduced in the present document is the HTN framework. This framework has been employed to solve planning problems in a wide range of applications, and there has also been work on visualizing HTN plans. While the representation of plans as networks is an interesting one, it is possible to define it in an ontology. As such, it is not clear that the HTN framework has an *expressive* advantage over an ontological model, in terms of plan representation. Certainly it is nice to exploit HTN algorithms in plan generation but, again, this is not our focus.

In the next section, we revisit the main approaches to plan representation. We provide a detailed description of each approach, along with an example and some analysis of the approach. In subsequent sections, we perform the same level of analysis with respect to plan forecasting, plan analysis, and plan monitoring.

### 4.2.1 Plan Representation

#### 4.2.1.1 Ontologies and Formal Languages

##### Overview of the Approach

An ontology is a formal language for representing entities and the relationships between entities, often based on a description logic [Baader, Calvanese et al. 2003]. An ontology may

consist of much more than a simple hierarchy of terms. However, precisely defining the notion of an ontology is beyond the scope of this report. In the discussion that follows, the important feature of an ontology is that it provides a precise, formal description of the entities that are significant in a given problem domain. Many different ontologies have been proposed for plan representation, including those in [Lenat 1995; Tate 1996; Tate 1996; Valente, Gil et al. 1996; Chaudhri, Farquhar et al. 1998; Tate 1998; Tate, Dalton et al. 1999; Gil and Blythe 2000]. Moreover, a previous survey paper has reported on the application of ontologies for representing military courses of action [Boury-Brisset and Champagne 2006].

A military objective consists of several components, including: title, measures of merit, and parent objectives. Some of these components can be defined in terms of lower level components. It is possible to define a formal language to provide a structured representation of military objectives [Valente, Swartout et al. 1998]. Providing a separate description of this objective-grammar illustrates the depth of detail that may be required for a complete description language for a military planning ontology.

The basic idea underlying ontological models is that describing entities in a hierarchical manner facilitates plan representation by making statements more precise. Moreover, using a formal language to represent ontologies makes it possible to develop algorithms for computing different relationships between entities. The entities that are represented might include: military units, time, positions, tasks, goals, or intentions. This is a non-exhaustive list; the class of entities represented is dependent on the ontology under consideration. Creating an ontology for a specific aspect of military planning can be a difficult task requiring a great deal of research. As such, some of the work on planning ontologies is focused on re-using or merging existing ontologies.

### **Detailed Description of the Approach**

To describe the ontological approach, we briefly describe two specific military ontologies.

#### *The Shared Planning and Activity Representation (SPAR)*

SPAR provides a formal representation of categories of entities, shown in Table 4-1 [Polyak and Tate 1999]:

**Table 4-1 Formal representation of the categories of entities used in SPAR**

Activities	General Structures	Resources/Objects
Agents	Goals, Requirements, Objectives, Mission, Tasks	States
Control Structures/ Execution/ Simulation	Organizational	Time/Space
Domain Knowledge	Plans/Schedule	Uncertainty/Ambiguity
Evaluations	Rationale	

The hierarchy of entities in SPAR is defined by a collection of statements that explicitly give the relationships between different categories of entities. The statements are given as sentences



of English language using a restricted vocabulary. Categories of entities are printed in uppercase, with descriptions of the relationships given in lower case. The core sentences of SPAR are given as follows:

- A PLAN relates an ACTIVITY-SPECIFICATION and an OBJECTIVE-SPECIFICATION.
- An ACTIVITY-SPECIFICATION describes ACTIVITY.
- ACTIVITY-SPECIFICATIONS can include ACTIVITY-CONSTRAINTS that impose restrictions over a set of ACTIVITIES and ACTIVITY-RELATABLE-OBJECTS.
- An OBJECTIVE-SPECIFICATION describes OBJECTIVES.
- OBJECTIVE-SPECIFICATIONS can include OBJECTIVE-CONSTRAINTS that impose restrictions over a set of WORLD-STATES or which specify required ACTIVITY.
- An OBJECTIVE may have one or more EVALUATION-CRITERIA which may be applied to one or more WORLD-STATES to create an EVALUATION.
- EXECUTION of an ACTIVITY can change the WORLD.
- An AGENT is an ACTIVITY-RELATABLE-OBJECT which can PERFORM ACTIVITIES and/or HOLD OBJECTIVES.
- An ACTIVITY takes place over a TIME-INTERVAL identified by its two TIMEPOINTS, the BEGIN-TIMEPOINT and the END-TIMEPOINT.

Additional collections of sentences are provided to further define the concepts introduced in the core sentences. For example, the following rules define some of the temporal notions that are part of the basic SPAR ontology.

- Any TIMEPOINT may be associated with one or more TIMELINES.
- A TIMELINE has a nominated BEGIN-TIMEPOINT and a TIME-UNIT and may have a nominated END-TIMEPOINT.

Taken together, the rules of SPAR define a concept space that specifies the relationships between all of the entities that can be described. The SPAR approach essentially defines a general-purpose knowledge representation formalism, with no representation of entities that are specifically military. By defining the rules in modules, such as the temporal module given above, the SPAR ontology maintains flexibility. In principle, different ontologies or grammars can be plugged-in to give more detailed relationships between restricted classes of entities.

#### *Intelligent System for Air Campaign Plans Evaluation Based on EXPECT(INSPECT)*

Air campaign objectives are of primary importance in planning. As such, it is useful to consider an ontology specifically defined to describe objectives. The INSPECT ontology introduced in [Valente, Swartout et al. 1998] defines objectives through a formal grammar written in Backus-Naur form [Marcotty and Ledgard 1986]. The grammar is specified using rules of the form

`item ::= component_1, component_2, ..., component_n.`

Such a rule states that an “item” consists of components 1 up to n. Each of the components, or *roles*, on the right is either assigned a value or it is defined by another rule. Grammars of this form are typically used to define the syntax of programming languages.

A formal grammar representing the objectives of an air campaign is defined as follows:

```
objective ::= objective-type
            objective-name
            action specification
            measures of merit
            parent-objective
            [sequencing restrictions]
            [cogs]
            [% complete]
            [review-status]
```

Roles that appear in brackets are optional components. Some of the roles in this definition are defined using additional grammar rules. Notably, a typology of action verbs is defined specifying the possible actions that might occur in an air campaign.

Using this grammar, a library of templates for common air campaign objectives is defined. The templates can be used to define specific objectives, which are then passed to the Air Campaign Planning Tool (ACPT) to generate plans.

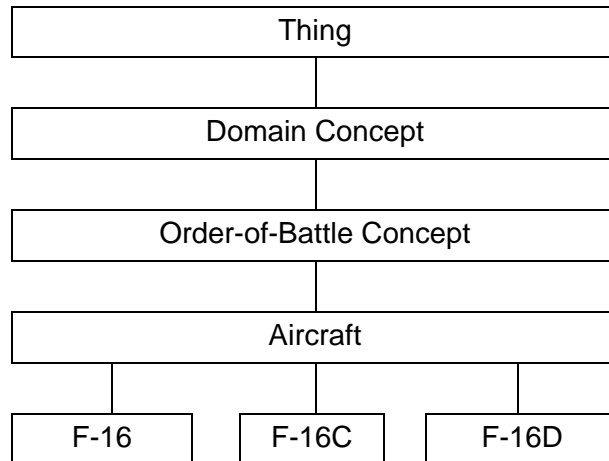
### **Additional Ontologies**

One ontology that is useful for representing Air Force plans is the Joint Forces Air Component Commander (JFACC) ontology [Chalupsky and MacGregor 2002]. This ontology contains specific information about Air Force equipment, including a hierarchy representing all Air Force vehicles. In order to represent Air Force plans in a general-purpose framework, it is useful to integrate the JFACC ontology. The difficulties encountered when performing this kind of integration are discussed in [Valente, Russ et al. 1999]. In terms of the entity relationships, this kind of integration is straightforward: the notion of a physical entity is simply refined by inserting a hierarchy of physical entities including things such as vehicles, aircraft, bombers, and F-16 bombers.

Note that ontologies vary greatly in the kinds of entities that are represented. The focus on SPAR in this section is intended to provide one simple example. However, there are aspects of plans that may be better represented in different ontologies. For example, the INOVA framework has the advantage that plans can be represented at different levels of abstraction [Tate 1996]. For the purposes of this report, the important point is that ontologies facilitate plan representation by using formal languages to describe entity relationships similar to the relationships typically represented in the well-known tradition of Unified Modeling Language (UML) diagrams [Booch and Rumbaugh 1998].

## Example

Ontologies such as SPAR, JFACC and LOOM [MacGregor 1991] are all used to define entity relationship diagrams. The level of detail differs depending on the underlying formal language. For example, the language of LOOM can be used to describe the hierarchy in Figure 4-1.



**Figure 4-1 Example hierarchy described by LOOM**

By contrast, the JFACC ontology defines a more refined hierarchy where Aircraft can be categorized as Fixed-Wing Aircraft or Combat Aircraft, which can in turn be categorized as Fighters or Bombers.

A simple example of an INSPECT objective modified from [Valente, Swartout et al. 1998]:

```

AIR OBJECTIVE 01
Maintain "air superiority" OVER "friendly forces"
measures of merit
    "No loss of allied aircraft to enemy forces"
AND
    "No enemy aircraft deliver weapons against allies"
parents
C1
    sequence-restrictions
        should occur before 02 and 03
  
```

The keyword "Maintain" is one of the action-verbs available for defining objectives.

## Analysis of the Approach

The main strength of this approach is that plans defined in an ontology have an unambiguous interpretation. Using an ontology to represent Air Force plans makes the meaning of all terms

precise and forces the plan author to define plans at the appropriate level of detail. This approach to plan representation is also appropriate for plan analysis, which will be discussed below.

When ontologies are formally defined in a logical language, it is possible to import the ontologies into new applications following [Valente, Russ et al. 1999]. Hence, this approach has the advantage of re-usability. This is clear when compared with approaches based on relational databases, such as [Thuve 2001]. Although the TOPFAS system itself has many nice features, the underlying representation of entities is not encapsulated in a manner that can easily be imported into new systems. The information about entities is essentially stored in a hierarchical manner using a relational database, and these databases can be re-used for new applications. However, the hierarchical structure cannot be modified by making simple additions or changes to a formal language.

There are many existing ontologies for representing different aspects of plans. In addition to the JFACC ontology for Air Force planning, there also exist ontologies for representing concepts related to time and space. Each of these ontologies requires significant effort to define, so it makes sense to combine existing ontologies to define an ontology for Air Force plans.

Although integrating two ontologies is conceptually straightforward, it can be difficult in practice to merge two different formal languages describing entities. For example, some languages focus on defining individual entities whereas others focus on defining concepts. These difficulties have been addressed in [Valente, Russ et al. 1999], where it is also suggested that combining multiple existing ontologies frequently leads to a better representation than one gets by defining a new ontology from scratch. It is even possible to translate approaches based on formal grammars, such as [Valente, Swartout et al. 1998], into an ontology in a systematic manner.

A bigger problem with the use of ontologies is that military experts normally do not have a background in formal logical languages. As a result, expert planners cannot define military plans directly in an ontology. However, there are existing tools that allow ontologies to be edited using familiar drag and drop techniques. There are also many libraries available to assist software developers in the creation of new ontology editing tools. Therefore, it is possible to develop plan editing software that displays plan components in an intuitive interface that is more accessible to a military expert. The effort required to build additional authoring tools would be justifiable, given all of the advantages offered by an ontology as an internal representation.

Using a formal grammar to represent objectives forces the user to explicitly consider the goals and sub-goals to be achieved. The library of objective templates provided does not supply fully defined objectives; the intention is that users should specify objectives specifically for a particular campaign. This feature of the system means that users must actually specify objectives using the grammar, which requires users to understand formal grammars. Again, grammars are relatively easy to understand and familiar to software developers, but it is not clear how appropriate this formalism will be for military personnel involved in planning. It seems that an interface is required to facilitate the definition of new templates by military experts.

Revisiting some of the work summarized in Chapter 3, one might notice that there has been no further discussion of systems in which the underlying representation of plans is based on a

relational database [Mulvehill and Caroli 1999; Thuve 2001]. These systems have not been explored because a relational database is simply not the ideal structure for storing representations of plans. The systems that are based on a relational database must define sophisticated sets of tables for storing complex relations among the entities comprising a plan. By contrast, ontologies are specifically defined for the representation of relationships between entities. Note that some aspects of systems based on a relational database may certainly be of interest. However, in focusing on approaches to plan representation, a table-based approach will not be stressed.

### **Efficiency Criteria**

Individual ontologies vary greatly with respect to the efficiency criteria set out for plan representation. For example, the JFACC ontology provides a very precise specification of military resources, but it does not address plan abstraction in a sophisticated manner. By contrast, the INSPECT ontology succeeds in providing a precise specification of objectives, while failing to accommodate any of the other criteria for plan representation. Following [Valente, Russ et al. 1999], it is possible to combine multiple existing ontologies to take advantage of complementary strengths.

As an overall approach to plan representation, the best ontology would be a “combined” ontology based on several of the formalisms introduced in this section. Such an approach would fare well in terms of plan abstraction, task specification, objective specification, and constraint specification. As a result, it will lead to a high-quality specification of plans and a deeper understanding of the interaction between plan elements, allowing for easier validation of plans and automatic detection of inconsistencies. Thus the advantages of a precise, machine-readable ontology of plan elements outweighs the additional effort required to build interface tools for military experts, to mitigate the difficult task of reading and authoring ontologies.

One limitation of the ontologies considered so far is related to the representation of actions with uncertain effects. In our proposed combined approach, we will address this limitation and present a formalism for representing uncertainty (see Section 5.2).

## **4.2.1.2 Probabilistic Approaches**

### **Overview of the Approach**

In many realistic planning domains, including military planning, the outcome of tasks and plans may be subject to uncertainty. One well-known approach to modeling uncertainty based on MDP algorithms has been developed in the machine learning community [Howard 1960]. Decision-theoretic planning refers to the approach to planning under uncertainty where MDPs are integrated with an AI planning formalism [Blythe 1999]. This approach has been explored in the domain of military planning [Aberdeen, Thiébaux et al. 2004]. MDP algorithms are used to generate plans that have a high probability of achieving some set of goals. The representation of the domain is given in a variation of the Planning Domain Definition Language (PDDL), a popular planning language that has become a standard for AI planning competitions [Fox and Long 2002]. For each task, there is an associated probability of success.

Coloured Petri-nets (CPNs) are another probabilistic formalism, one that has been used for the representation of plans in the Course of Action Scheduling Tool (COAST) [Zhang, Kristensen et al. 2002; Kristensen, Mechlenborg et al. 2006]. The COAST system is designed to hide the underlying representation of plans from the user. Hence, the user interacts with an intuitive graphical interface to define plans, which are then translated into the CPN framework.

This architecture allows plans to be represented in a precise manner suitable for formal analysis, but it does not require military planners to be concerned with the formal representation. The CPN framework consists of networks of nodes and arcs that represent the state changes that occur when events occur. CPNs have been used primarily for modeling systems involving concurrent events, which is certainly the case in the military planning domain. Traditional Petri-net models involve tokens on certain nodes, and the tokens are added, removed, or moved as events occur. In a CPN, the tokens contain data, represented by a “colour.” The tokens are understood to represent tasks, and the colours encode information such as task preconditions and probabilities of success. A CPN can encode all of the possible states in a military planning domain, as well as all of the possible tasks. By using this underlying representation, plans can be manipulated and analyzed using existing approaches to managing CPNs.

In the following sections we describe the MDP and CPN frameworks in detail, highlighting the similarities and differences.

### Detailed Description of the Approach

The basic unit for constructing plans is a “task,” which can range from logistical operations to strategic operations. Tasks may have durations, preconditions, and effects. A PDDL-like language is used to define tasks in the following format:

```
( :task-type NAME
    :parameter_1 (parameter specification)
    ...
    :parameter_n (parameter specification)
)
```

The parameters that must be specified depend on the type of task. For example, an action that takes place over an interval of time must include a parameter specifying the duration of the task. One parameter included for all tasks specifies the effects with a probability for each possible outcome.

In this approach, the effects of tasks are essentially modeled as state transitions. A state is a formal abstraction of the world that includes: the state’s time, a queue of events expected to occur, a list of possible tasks, the truth value of each fact, the available resources, and all costs associated with the state. A task described in the domain description language specifies mapping indicating the outcome state, when the given task is executed in any other state. Actually, every set of tasks defines a mapping on states; in this manner concurrent tasks can be naturally represented.

An MDP consists of the following:

- A finite state space  $S$ .

- A finite set of actions  $A$ .
- An initial state  $s_0$ .
- A set of terminal states  $T$ .
- A computable probability  $Pr(s' / s, a)$  that action  $a$  leads to state  $s'$  from state  $s$ .
- The cost of each state.

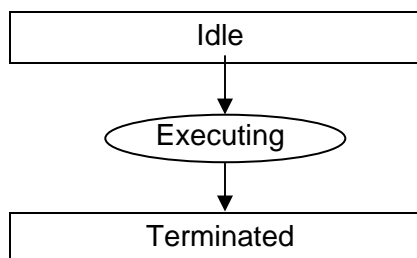
MDPs have been extensively studied, and many algorithms have been developed for computing minimal-cost plans. A military planning domain given in PDDL with probabilistic effects can be viewed as an MDP, with one minor complication. The problem is that states in military planning cannot necessarily be assigned a single numeric cost. Instead, there are several independent costs: 1) a failure cost if the goal becomes impossible, 2) the cost of resources used, and 3) the length of time required.

In terms of plan generation, several algorithms are required to find the “best” plan to achieve a goal: one algorithm computes the executable tasks at a given state, one algorithm computes the successor states, and another algorithm tries to find the minimal cost plans among the possible plans. MDP algorithms play a role in the last of these three algorithms. Since the focus of this section is on plan representation, these algorithms need not be considered in detail.

Turning to the CPN framework, the formal definition of a CPN can be found in [Jensen 1990] or [Kristensen, Christensen et al. 1998]. Roughly, a CPN consists of the following:

- A set of *colour sets* that informally represent different types, such as tasks or resources. Each colour set contains a set of *colours* that informally represent objects of the given type.
- A set of *nodes* that represent entities and events.
- A set of *arcs* that indicate how states change in response to events.

Schematically, a simple CPN for representing tasks execution can be given as shown in Figure 4-2. The complete CPN involves additional nodes for representing resources and other aspects. But this simple sub-net is sufficient for the present purposes. Notice that the nodes in a CPN differ from the nodes in typical state-transition approaches in that the nodes do not represent states. Some of the nodes represent entities and some of the nodes represent actions; the state of the system is given by the collection of all nodes, together with all coloured tokens.



**Figure 4-2 Example CPN for representing task execution**

The tokens representing tasks have a colour that represents all the attributes of the task, including: preconditions, postconditions, probability of failure, resources consumed and

duration. This is just a subset of the attributes: every token with the colour “task” has 22 attributes. There are also tokens representing resources, with a distinct set of attributes.

Initially, all tokens representing tasks are placed on the node that is labeled “Idle” and all tokens representing resources are placed on the node labeled “Resources.” When a task is executed, the corresponding token is moved to the “Executing” node. If the task consumes any resources, the tokens representing those resources are removed from the Resources node. After the task is completed, the task tokens are moved to the “Terminated” node.

The basic procedure of shifting and removing tokens is used to model all tasks executed. Since tokens can be shifted and removed simultaneously from multiple locations, it is easy to represent concurrent actions. Formally, the colour set for tokens is described using explicit colour declarations; syntactically, the declarations are essentially equivalent to the rules of a formal grammar.

The underlying CPN is invisible to the user. The user defines a plan by interacting with a series of collapsible menus. The user selects a sequence of tasks, and then asks COAST to analyze the sequence to determine the possible outcomes. In order to determine the possible outcomes, COAST expands the CPN into a set of state-transition graphs which allows the system to determine the probability that a goal is achieved. Again, this process is invisible to the user who simply receives text feedback in a readable format.

### Example (MDP)

A simple example might involve the representation of unit locations. In this context, a *fact* would be a true/false statement about the location of a unit. Suppose that there are 4 possible locations: *loc\_1*, *loc\_2*, *loc\_3*, *loc\_4*. If there are 2 military units *unit\_1* and *unit\_2*, then a fact would have the form *unit\_1(loc\_2)* or *unit\_2(loc\_4)*. Suppose that both units are initially in *loc\_1* and the goal is to get them into *loc\_2*. A fee of \$10 is required for every hour where a unit is in location *loc\_1*. A partial specification of an MDP for this example can be given as follows:

- The state space  $S$  is the set of pairs  $(unit\_1(loc\_i), unit\_2(loc\_j))$ .
- The initial state  $s_0$  is  $(unit\_1(loc\_1), unit\_2(loc\_1))$ .
- The set of terminal states  $T$  is  $(unit\_1(loc\_2), unit\_2(loc\_2))$ .
- The cost of  $(unit\_1(loc\_i), unit\_2(loc\_j))$  is 0 if  $i$  and  $j$  are greater than 1, 10 otherwise.

To complete the specification of the MDP, there is a *move* action that moves a unit between any two locations, with a .1 probability of failure.

The simple example involving 2 units and 4 locations illustrates the basic idea behind a MDP formalization. However, in real military examples, the state space will be much larger and the cost function may be much more complex. Moreover, the set of tasks will generally be more complex, requiring a precise specification language. The following example [Aberdeen, Thiébaux et al. 2004] is a more realistic illustration of a task description for military planning.

```
(:durative-action refuel-convoy
  :duration (= ?duration 10)
```



```

:condition      (and (at start convoy-needs-fuel)
                     (at start convoy-at-rendezvous)
                     (at start ( >=cash 100))
                     (at start ( >= tankers 1)))
:effects        (and (at start convoy-delayed)
                     (at start (decrease cash 100))
                     (at start (decrease tankers 1))
                     (at end
                       (probabilistic
                        0.95(and(not convoy-needs-fuel)
                               (increase tankers 1))
                        0.05(tanker destroyed))))))

```

Given a collection of tasks described in this format, MDP algorithms can be used to generate the minimal cost plans.

### Example (CPN)

The example in the description of the approach illustrates the simplest form of a CPN. Some basic colours are defined through the following explicit declarations:

```

colour Name = String;
colour Bool = bool;
colour NamexBool = product Name * Bool;
colour Condition = union CONDITION : NamexBool;
colour Conditions = list Condition;

```

The task colour is then defined in terms of these colours:

```

colour Task = name:Name, startcondition:Condition, ...;

```

Each component of the Task colour is an attribute defined in terms of another colour. This illustrates that the use of the term “colour” is really synonymous with “data” in this context. Arbitrarily complex colours can be defined using the given declarations.

### Analysis of the Approach

The main contribution of MDPs and CPNs is the treatment of probabilistic outcomes of tasks. This is a significant contribution, since outcomes will always be probabilistic in real military planning domains. The representation of probabilities is simple and flexible, providing a simple probability function with every task. Beyond this feature, each approach has its own minor advantages. For instance, by translating planning problems into MDPs, this approach is able to exploit the rich history of MDP algorithms for in the context of military planning. MDP

algorithms are normally used for determining the minimal cost plan achieving some goal; as such, representing plans as MDPs facilitates the analysis of plans. This point will be revisited in the section on plan analysis.

As noted above, the CPN framework has been used for the internal representation of plans in the COAST system. Separating the graphic user interface from the underlying plan representation is clearly a good idea. Even if the representation formalism is very simple to a software developer, it is likely to be unnatural to a military planning expert. The interface for the COAST system is not discussed in detail in the existing literature, although a few screenshots suggest that the interface is based on a series of drop down menus. The experience of the NuSketch developers suggests that this kind of approach can be frustrating for military planners. In terms of the internal representation of plans, the underlying CPN representation of plans is precise and sophisticated, but it has not been adopted beyond a relatively isolated community. The representation in terms of tokens appears to be very expressive for representing concurrent actions, and it is nice that tasks, resources, and entities are represented uniformly in a single formalism. It is not clear that the advantages of the CPN approach are sufficient to adopt it over the ontological models that are more plentiful, and supported by a larger number of research groups.

One limitation of both probabilistic approaches is the focus on the representation of tasks, with little discussion of the representation of resources, military units, or time. The only description of resources in the published descriptions of this work is given in the examples of task descriptions in PDDL. As such, it is not clear what constitutes a resource; similar comments apply to military units and temporal issues. The representation of probabilistic effects is such an important feature, that these frameworks should not be overlooked. However, this work provides an incomplete representation of military domain knowledge. Given a formal ontology of military planning concepts, one could integrate the key ideas from this approach by adding a straightforward treatment of probabilistic effects. The algorithms for finding minimal cost plans could then be integrated in a straightforward manner, which would essentially mean that the main contribution of this work would be incorporated.

### **Efficiency Criteria**

The main distinguishing feature of the MDP approach is that it makes it possible to represent uncertainty. The approach also provides a detailed specification language for tasks and objectives, comparable to the specifications available in an ontology. No precise specification is given for constraints, however. In particular, resource constraints are not represented in a precise manner. It is difficult to represent resource constraints, because there is no precise language for describing military resources.

Broadly, the problem with the MDP approach is that the focus is primarily on representing tasks rather than representing entities. By contrast the ontological models are focused on representing entities, and tasks are reified and viewed as one particular kind of entity. Finally, MDPs are mathematical constructions that cannot be defined or understood without sufficient training. The situation is arguably worse than it is for ontologies, because understanding an MDP representation involves some understanding of probabilities and utilities.

In terms of the efficiency criteria specified, CPNs are essentially equivalent to MDPs.

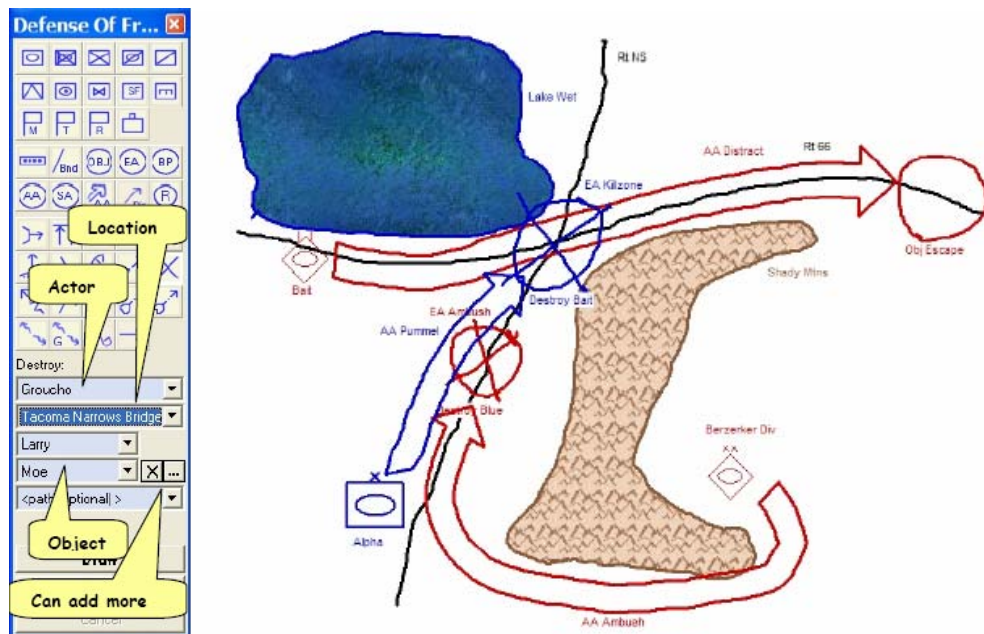
### 4.2.1.3 Visualization

#### Overview of the Approach

Several systems have been developed for assisting with the description of COAs, but none have been completely satisfactory for military users [McGee and Cohen 2001]. For example, we have already seen that the COAST system involves a visualization component. As noted, our focus in this document is on approaches rather than implemented systems. One of the main objections to existing approaches to visualization is the fact that using a computer mouse to select from menus is more difficult than sketching a COA using a pencil and paper. As such, our focus in this section is on one approach to visualization that does not rely solely on this manner of interaction. The approach under consideration is the visual sketching approach in the NuSketch software. As much as possible, we focus on the approach rather than the implementation details.

NuSketch was developed to mimic the traditional sketching system by allowing users to draw a COAs on a computer screen [Forbus, Usher et al. 2003]. The distinguishing feature of the software is the emphasis on understanding what the things on a plan sketch actually mean. This is handled in the approach to knowledge representation underlying the user interface. In particular, NuSketch can be modified to support different ontologies and formal languages. For example, sketched plans have been translated into the SHAKEN action description language [Barker, Blythe et al. 2003], which encodes tasks in a manner similar to the well-known language of STRIPS [Fikes and Nilsson 1971].

In the following subsections, we describe the NuSketch approach in detail. Our focus will be on two unique aspects of the approach. First, NuSketch allows users to sketch plans on screen rather than defining plans through a restricted set of commands or menus. Second, NuSketch can work with different ontologies for plan representation. These two features give NuSketch an advantage over related software for authoring plans. There are certainly additional features that would be desirable in NuSketch. For example, it would be desirable if plans could be viewed at different levels of abstraction and it would be desirable if plans could be monitored during execution. However, the advantages of NuSketch outweigh the disadvantages in terms of plan representation.



**Figure 4-3 A screenshot taken from NuSketch**

### Detailed Description

NuSketch attempts to follow the traditional approach of describing COAs through sketching with markers on layered sheets of acetate. For example, the software uses layers in the same manner that layered acetate has been used traditionally to separate different aspects types of units; one layer might be used for friendly forces and a second layer for enemy forces. Units are placed on the screen by selecting *glyph* symbols from a *glyph map*. By selecting unit symbols from a menu, the system avoids difficult recognition problems and it assures that the unit symbols will be clean and readable; users stated a clear preference for this approach.

Every sketch using this software has a precise interpretation that is fixed when the drawing process is complete. Information about the glyphs, such as the amount of geographic space occupied, is sketched on the map using standard conventions. Pressing a “draw” button starts the process of sketching a glyph, and pressing the button again finishes the process. In addition to representing units, glyphs are also used to represent regions, lines, paths, and even actions. Roughly, a glyph represents any information about a COA that has a geographic position on the map.

In addition to the information sketched on the screen, there is normally some textual information that accompanies a COA. For example, the intent of different units must be included. In order to represent intent in a uniform manner, the NuSketch system uses an intent dialogue box that asks the user to enter the intent of a unit in the following format: “<type> <actor> <operation> <target>.” This format was derived from extensive interviews with military experts. In both the sketching portion and the textual portion, the emphasis is on avoiding recognition problems by ensuring that everything entered has an unambiguous interpretation. Another component to a COA is a timeline for the actions. This is specified in a Gantt-style chart.

COAs defined in NuSketch can be translated into the action description language of the SHAKEN system [Barker, Blythe et al. 2003]. SHAKEN is a general-purpose tool that can be used by subject matter experts to enter domain knowledge in an intuitive manner [Clark, Thompson et al. 2001; Blythe 2002]. The domain knowledge is then translated into an underlying formalism for knowledge representation. The SHAKEN system includes a STRIPS-style action description language for describing events in a hierarchical manner. An event in SHAKEN is described in a structured format:

```
(every EVENT has
  (a SUBEVENT called ***** with ...)
  (an AGENT *****) ... (an OBJECT *****) ... (a NEXTEVENT)...) )
```

So events in SHAKEN include information about subevents, agents, objects, and event sequencing. Using the language of SHAKEN to represent plans is valuable, because subject matter experts can use the same language to enter critiques of plans which facilitates COA analysis.

### Example

There are 294 distinct friendly unit symbols. A friendly infantry unit can be placed on the map as follows:

- Select <friendly> from: <friendly, enemy>
- Select <infantry> from: <infantry, armor, etc.>
- Select <regular> from: <regular, minus, plus, etc.>
- Select <corps> from: <corps, squad, etc.>
- Use a single stroke on the map to indicate location and size of unit.

Since this glyph represents a unit, the size of the graphic representation on screen does not have any real meaning. The image will be drawn at the size specified by the user, but the only important feature of the graphic representation is the position. The center of the graphic representing the unit is taken to be the coordinates of the unit.

If this unit is intended to guard a certain region called Region1 represented on the map, this can be represented in the intent dialogue as follows:

- Select <maintain> as the type
- Select <friendly> as the actor
- Select <protect> as the operation
- Select <Region1> as the target

The important feature of this example is that it illustrates how NuSketch forces objects and intentions to have a uniform structure, suitable for the underlying knowledge representation.

## **Analysis of the Approach**

Experimental evidence suggests that military users can easily learn how to develop COAs with NuSketch, and that the COAs are no less imaginative than those developed using traditional approaches [Rasch, Kott et al. 2002]. The NuSketch approach also compares favorably with related COA design tools such as QuickSet [Cohen, Johnston et al. 1997], where a great deal of effort is required for the recognition of multi-modal input. Rather than focusing on recognition, NuSketch forces the user to enter COAs in a specific manner, which allows for a precise underlying formal representation of plans.

One negative feature of the approach is that users have difficulty comparing different COAs using the standard comparison matrix [Rasch, Kott et al. 2002]. This is surprising, because the comparison matrix in NuSketch is virtually identical to the paper-based matrix that is traditionally used. This difficulty illustrates that users still interpret information differently when it is presented on-screen, as opposed to paper. Moreover, the kind of screen used to display information is also significant. In particular, interactive whiteboards appear to focus the users' attention more than a standard computer screen [Bennett and Lamoureux 2006].

Overall, the NuSketch system appears very promising as a tool for entering COAs, mostly because it manages to mimic the traditional system very closely. Hence, military users can use the system without significantly modifying any standard procedures.

## **Efficiency Criteria**

The NuSketch software is easy for military users to understand, both for defining new plans and for reading existing plans. This is the major strength of the approach, as it allows military users to define plans with little additional training. However, with respect to most of our measure of efficiency, NuSketch does not fare well.

First, note that NuSketch only allows plans to be represented at the level of COAs. As such, the system is not able to represent plans at different levels of abstraction. Moreover, the system is not really suitable for representing earlier stages in the planning process, such as initiation or orientation.

Although the NuSketch glyph map provides a flexible approach to placing units on a map, it is not flexible enough to represent arbitrary constraints on a plan. Similarly, there is no precise representation of objectives built in to the system. Finally, NuSketch does not provide the user any way to represent probabilistic information.

## **4.2.2 Plan Forecasting / Projection**

In this section, the main approaches to plan forecasting and projection are revisited. Many of these approaches have been summarized in detail in the section on plan representation. In such cases, the focus in this section is only on the analysis of each approach for the purposes of plan forecasting. Hence, only the measures of efficiency introduced in 4.1.2 are considered in this section.

#### 4.2.2.1 Ontologies and Formal Languages

##### Overview of the Approach

The most important approach to plan forecasting based on ontologies and formal grammars is the development of the work on INSPECT: The Intelligent System for Air Campaign Plans Based on Expect [Valente, Swartout et al. 1998; Valente, Blythe et al. 1999]. The INSPECT system makes two main contributions towards plan forecasting. First, it provides a library of templates for standard objectives that allow forecasting to be done offline in advance. Second, it does some preliminary resource checking.

##### Detailed Description of the Approach

The INSPECT system uses the Loom ontology as well as the formal grammar for objectives defined in [Valente, Swartout et al. 1998]. By inspecting the objectives specified in applications using ACPT, one can specify a collection of approximately 30 verbs describing the basic action types. Moreover, each action type can only accept a limited selection of arguments. For example, the action “establish” generally takes the argument “air superiority” over some selected region. The actions that occur in past applications can be specified through templates of the form:

`objective-type (type-of-argument1, ... ,type-of-argument2)`

A collection of objective templates is defined by listing each of the 30 action verbs together with constraints on the permissible argument types. A partial typology of action verbs in these templates can be defined, by introducing abstract classes of actions. For example, verbs such as “damage” and “destroy” can easily be categorized together with verbs describing actions that directly affect enemy units. By contrast, verbs such as “maintain” or “gain” refer to higher-level strategic objectives.

The typology of action verbs does not provide a complete taxonomy of action-types, but it does help to further delimit the kinds of activities available. Some action verbs are explicitly excluded from the set of templates due to ambiguity. Specifying objectives based on the templates is intended to simplify forecasting, by ensuring that the objectives of a given plan are precise objectives with a clear meaning that has been useful in past planning.

After entering plans and objectives using plan editing software, one uses the INSPECT software to check the plan for errors. In general, error checking falls more accurately as a topic in plan analysis; so INSPECT will be discussed in more detail in the Section 4.2.3. However, one category of error that is directly relevant to plan forecasting is related to resource checking.

Many of the actions specified in a plan require minimum resources in order to be executed properly. For example, maintaining air superiority over a certain area may require a specific number of planes or a specific type. As a plan is executing, it is useful to constantly check that the resources required for future actions will be available. When a user runs INSPECT to check for errors in the plan, one of the checks performed is the number aircraft required to execute the plan.

The software returns a WARNING, CAUTION or NOTE to alert the user of any problems. A WARNING is the most severe feedback, indicating a problem that requires immediate attention. In terms of resource checking, a WARNING might indicate that there are not enough aircraft to execute some aspect of the plan. The INSPECT system will then suggest modifications to the plan that would make it executable, given the current resources.

The feedback of the INSPECT system is always mixed-initiative: no modification to the plan is made without user involvement. In terms of plan forecasting, the INSPECT system helps the user project whether or not a given plan can be executed given current resources.

### Example

Two of the templates given in the library of objectives are:

Destroy	DOB-object
Maintain	DOB-aspect

The action verbs on the left specify the kind of activity being described. The right column specifies the general format of the permissible arguments. A DOB-object argument must be filled by something that is of the type “object” as specified in the ontology of the system. In this case, an object is understood to be a physical object; this includes all forms of military unit that are represented in the ontology. By contrast, a DOB-aspect is a higher-level entity such as “air superiority.” Again, the underlying ontology specifies the permissible aspects.

An example of an action verb that does not occur in a template is the verb “attack.” The precise meaning of “attack” is not clear in terms of the objects that are affected. The exact meaning of an attack-objective should be specified in terms of more specific verbs that are specified in the objective library, such as “destroy” or “deploy.”

### Analysis of the Approach

The library of objectives does not give precise examples or specific predefined objectives. Instead, the templates require the user to fill in the details of the objective subject to some simple constraints. This approach encourages the mixed-initiative approach to planning by forcing the user to explicitly define objectives for every plan. However, it also limits the usefulness of the templates for plan forecasting. If the INSPECT system provided a collection of precise objectives and plans to achieve the objectives, then it would be possible to do more offline plan forecasting for modular plan development.

Forecasting the outcome of a plan is very difficult if the plan is specified using ambiguous or imprecise objectives. Hence, the library of objectives is useful for plan forecasting in that it forces users to specify objectives using unambiguous action-types that have been previously employed in past plans.

One reason why there are so few action-types is because the library of objectives is defined based on the problems that have previously been addressed using ACPT. However, ACPT is only concerned with force-application objectives. There are many Air Force plans that are not directly related to force-application; in order to represent these plans, new action-types must be introduced. At best, the library is useful for developing plans for a restricted class of objectives.



In terms of the resource checking aspect of INSPECT, this feature is very useful and it seems likely that some form of resource pre-processing should be performed for all Air Force plans. However, in the current format, INSPECT only checks for required aircraft. More comprehensive resource checking would be preferable.

### **Efficiency Criteria**

The INSPECT system performs resource checking, which is an important form of constraint checking. However, the system does not really allow any other constraints to be checked. The system allows pre-defined templates to be used, which means that projecting the outcome of common plans can be done in advance.

There is no mechanism for probabilistic forecasting in INSPECT, since the underlying ontology is not able to represent uncertain outcomes. The current version of the system does not check if a plan will complete on time. In fact, there is no way to determine how long a plan will take to complete. Perhaps the biggest weakness of INSPECT with respect to our criteria is that it does not allow the status of a plan to be projected to a particular point in time.

However, these weaknesses are not indicative of problems with the overall representation by ontologies. The INSPECT system is just one tool for automated reasoning about plans. Certainly a representation of a plan in an existing military ontology can be used to project to different points in time. Any military ontology with a temporal component could be used for this purpose.

## **4.2.2.2 Probabilistic Approaches**

### **Overview of the Approach**

MDPs and CPNs were discussed in Section 4.1.2 with respect to plan representation. One important feature of both these approaches is that tasks have probabilistic outcomes. These outcome probabilities can be used to forecast the result of a sequence of tasks.

### **Detailed Description of the Approach**

If the state of the world is represented by a set of facts that can be true or false, then the outcome of a task is to make certain facts true and certain facts false. If tasks have probabilistic outcomes, the outcome of a task is described in terms of the probability that it makes a fact true. This is the approach of both the MDP and CPN frameworks. The details are slightly different in that MDPs define tasks with an outcome that has a fixed probability of success, whereas CPNs define several possible outcomes with different probabilities.

In each case, the important fact is that predicting the outcome of a sequence of tasks involves computing joint probabilities. In particular, for any sequence of tasks and any fixed objective defined in terms of facts, it is straightforward to specify the probability that the given sequence of tasks will achieve the objective.

Rather than asking if a plan will achieve an objective, plan forecasting in the MDP framework asks how likely a plan is to achieve an objective. A lower bound can be set on the probability  $p$

such that an “acceptable” plan is one that achieves the required goal with probability greater than  $p$ . The details of this computation are slightly more involved in a CPN as opposed to an MDP. To model the execution of a sequence of tasks in a CPN, one must expand the net to generate every state-space graph that can be obtained. However, in both formal approaches, the plan forecasting involves basic calculations on probabilities.

### Example

The following example is modified from [Aberdeen, Thiébaux et al. 2004]. Suppose that there are two goal facts, represented as truth-valued variables  $F$  and  $G$ ; so *success* is defined as the simultaneous truth of  $F$  and  $G$ . Consider three tasks  $A$ ,  $B$ ,  $C$  where  $A$  succeeding causes  $F$  and  $G$  to both become true,  $B$  failing causes  $F$  to become true, and  $C$  succeeding causes  $G$  to become true. For facts, let  $Pr(F)$  be the probability that  $F$  is true. For tasks, let  $Pr(A)$  be the probability that  $A$  succeeds. If the tasks  $A$ ,  $B$ , and  $C$  are executed, then:

$$\begin{aligned} Pr(\text{success}) &= Pr(F \text{ and } G) \\ &= Pr(A \text{ or } (C \text{ and not } B)) \\ &= 1 - Pr((\text{not } A) \text{ and } B \text{ and } (\text{not } C)) \\ &= 1 - Pr(\text{not } A) Pr(B) Pr(\text{not } C) \end{aligned}$$

Note that each of the probabilities in the final term is explicitly given in the MDP representing the planning domain. Hence, this expression is sufficient for forecasting the likelihood that the goal is achieved.

### Analysis of the Approach

Using a probabilistic representation of task outcomes changes the underlying question faced in plan forecasting. Instead of asking if a plan will have a certain outcome, the question is now how likely it is that a plan will have a certain outcome. In practical planning domains, this is a more realistic question.

One problem with the probabilistic approach, however, is that it is not clear how probability distributions can be assigned to all tasks. If the probability assignments are not meaningful, then performing a series of calculations for plan forecasting seems unnecessary.

Moreover, it is not clear that the simple calculations suggested by the approach are actually appropriate for plan forecasting. The outcome of a task is very likely to depend on the situation, as well as previous tasks executed. As such, simply forecasting based on an assumption of independent probabilities may be naïve. Probabilistic forecasting may require a more sophisticated treatment of dependency.

### Efficiency Criteria

The only feature from Section 4.1.2 that is treated by the MDP and CPN approaches is probabilistic forecasting. In other words, based on the probabilities of various task outcomes, it is possible to predict the most likely outcome of a plan. The remaining criteria for forecasting are not treated in this framework. In particular, it is not possible to project the status of a plan to a particular point in time. At best, it is possible to look at all possible outcomes of a plan at a

particular point in time. Since tasks have probabilistic outcomes, projection will always be described in terms of a probability distribution over outcome states. In terms of checking if a plan completes on time, this is also impossible in a probabilistic setting. At best, one can indicate the likelihood that a given plan will complete on time.

### 4.2.2.3 Visualization

#### Overview of the Approach

Visualization for plan forecasting requires a graphical depiction of a timeline, together with some mechanism for projecting a plan to different points.

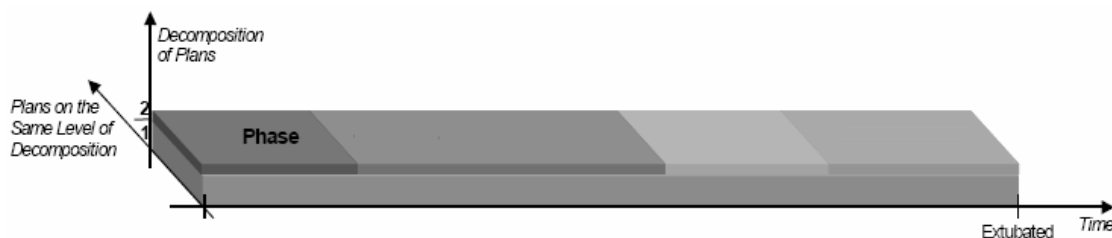
#### Detailed Description of the Approach

One of the standard techniques for representing timelines for plan management is the Gantt chart. In a Gantt chart, the horizontal dimension represents time. Each task is represented as a horizontal line on the timeline, and lines can overlap in terms of the time required. As noted previously, it has been suggested that Gantt chart representations are not adequate for representing military [Rasch, Kott et al. 2002].

The AsbruView software expands on the typical Gantt chart in a manner that is more suitable for forecasting military plans. The representation is three dimensional, with plan decomposition in the vertical dimension, time in the horizontal dimension, and plans at the same level of decomposition at the same depth in the image. This visualization improves on a typical Gantt chart, primarily by allowing different levels of abstraction to be considered in plan forecasting.

#### Example

A visual depiction of plan decomposition in three dimensions [Miksch, Kosara et al. 1998]:



**Figure 4-4 Visual depiction of plan decomposition in three dimensions**

#### Analysis of the Approach

Although this visualization improves on a typical Gantt chart, the approach is limited in that military applications have not been explored. AsbruView has been applied primarily in the medical domain, modeling patient treatment plans. However, the same kind of task decomposition is relevant to military planning and the timeline is appropriate for projection.

## **Efficiency Criteria**

In terms of the criteria in Section 4.1.2, this approach to visualizing plan forecasting fails with respect to the first three criteria. In particular, it does not provide any mechanism for resource checking, defining pre-processed templates, or performing probabilistic forecasting. Nevertheless, the visualization method provides a simple view of different plan phases unfolding over time. It is possible to look at a graphical depiction of a plan and automatically determine if the plan is scheduled to complete on time. The status of a plan at any point in time can be determined by decomposing all of the tasks occurring at that point in time.

## **4.2.3 Plan Analysis / Evaluation**

In this section, selected approaches are considered with respect to the measures of efficiency in Section 4.1.3.

### **4.2.3.1 Ontologies and Formal Languages**

#### **Overview of the Approach**

The basic assumption of the work described in this section is that military expertise is required to evaluate the suitability of a plan. When plans are represented in a formal language describing an ontology, it is difficult for military experts to help. As such, additional effort is required to obtain information from military experts and translate it into useful information about formal descriptions of plans.

#### **Detailed Description of the Approach**

The INSPECT system is a piece of software that can be run on a plan to detect errors [Valente, Gil et al. 1996]. A special tool is provided for defining new classes of errors. The developers of INSPECT had military planning experts use the tool provided to specify a collection of fundamental errors that users often make when defining plans. Four different kinds of error can be detected.

- **Completeness errors:** These are errors caused by some missing element of a plan, or some missing connection between elements. For example, every objective in a military plan must serve a higher-level objective. It is common in practice for users to specify intermediate goals that do not explicitly serve a higher-level purpose; such plans are not deemed to be acceptable by military experts.
- **Reasonable structure:** Plans must conform to certain structural limitations. For example, there are sequencing restrictions that specify that some objectives must be achieved before others.
- **Feasibility:** This kind of error is related to available resources. Hence a plan will be rejected if there are no aircraft available to execute the plan.

- **Plan contents:** The main error of this kind is related to incoherent objectives. In particular, high-level objectives should be more general than the objectives that serve them.

When any of these errors is detected, the user is alerted about the problem through a CAUTION, WARNING, or NOTE. The plan is not automatically changed to fix the problem; the INSPECT system simply suggests some possible solutions. Changing the plan to incorporate the suggestions is exclusively the domain of a human user.

The errors detected by INSPECT are errors in plan structure; the developers of the system suggest that it is like a spell-checker for plans. Plans will not be rejected based on financial cost, risk, or inefficiency. By contrast, the SHAKEN system incorporates input from military experts to analyze plans [Barker, Blythe et al. 2003].

In the approach used in the SHAKEN system, COA plans are authored using NuSketch and then translated into the SHAKEN language. Military experts have been solicited to enter critiques of plans based on simple plan patterns. A plan pattern indicates a set of antecedent conditions, together with some tasks executed under these conditions. Military experts provided critiques of patterns based on various dimensions such as risk, cost, simplicity, and resource use.

Given a COA, the SHAKEN system collects all patterns that apply to the COA and combines the expert critiques to assign a critiquing score to the COA. The critiques provided in this system are not based on the structure of the plan description. The plan patterns are not complete plan specifications, so the precise structure is not being tested. The critiques entered by military experts in this case are explicitly concerned with using military knowledge to react to determine the best task to execute in a given situation.

### Example

The following example modified from [Valente, Gil et al. 1996] provides an example of an error detected by INSPECT with the output of the system.

#### **Agenda Item**

Air Objective "Destroy enemy political control of armed forces" has no measure of merit.

#### **Explanation**

INSPECT is checking plan completeness. One step in doing that is to check objectives for assessment of completion. Objective "Destroy enemy political control of armed forces" has not assessment of completion because it has no measure of merit.

#### **POSSIBLE CAUSES AND FIXES TO CONSIDER:**

- A measure of merit for the objective has not been defined yet.

#### **POSSIBLE FIXES:**

- Add a measure of merit to the objective
- Dismiss item: Objective does not have a measure of merit.

In this example, an objective has been specified without any indication of how to determine if the objective has been accomplished. As indicated previously, INSPECT simply alerts the user to the problem and suggests solutions. Note that the second solution is essentially to disregard this advice. If the user chooses this option, the system will provide an additional warning and ask for a justification. This additional step is intended to make sure that users take the warnings and errors seriously.

For comparison, the critiques entered by experts in the SHAKEN system can be paraphrased as follows [Barker, Blythe et al. 2003]:

- If a COA secures a piece of terrain narrower than 50 meters, it makes good use of terrain.
- If an armored unit attacks a mechanized infantry unit outside a city, it is good for enemy maneuver engagement.

These critiques are not focused on the structure of a plan description; they are focused on critiquing behaviors occurring in a COA.

### **Analysis of the Approach**

Using military expertise is an important part of plan analysis, and the easiest way to incorporate military knowledge is to solicit information from military experts [Miller, Militello et al. 1996]. Typically the developers of planning formalisms and planning software do not have sufficient expertise to determine what exactly constitutes a good plan. The problem is further complicated by the fact that military planners and AI planners use some of the same vocabulary to describe plans, but with slightly different meanings.

In order to incorporate expert military knowledge for plan analysis, the approaches described in this section ask military experts to enter critiques directly for use in software. This is a good approach to improving critiques, but it introduces several challenges.

The first challenge that must be faced is that military experts do not always have the time required to assist with software development. For this reason, the experts used to develop these approaches are typically retired military officers. In the case of the SHAKEN approach, the experts were required to spend 15 days training and critiquing plans.

Even with 15 days of training and practice, the experts still made basic mistakes with using the formal language for entering critiques in the system. As such, one limitation of this approach is simply that it is difficult to get useful expert information represented in a format that is useful for plan analysis.

Each of the critiquing systems discussed is limited in the scope of the error detected. In the case of INSPECT, only structural problems are detected. In the case of SHAKEN, structural problems are not detected at all. By combining the advantages of both systems, one could obtain a more comprehensive tool for plan analysis. The SHAKEN system is particularly intriguing, since it allows COAs to be authored using the NuSketch system. This cuts out one complicating layer when asking for expert evaluations.

### **Efficiency Criteria**

The INSPECT system performs structural plan analysis, by checking that a plan does not contain any errors in terms of the manner in which it is defined. By contrast, the SHAKEN system performs semantic plan analysis. Therefore, a combined ontology could be defined that performs both.

Military experts have been consulted in both forms of analysis. Although neither INSPECT nor SHAKEN provides a ranking scheme for plans, it is possible to define a ranking scheme for choosing between “equivalent” plans in an ontology. One approach to ranking plans is given by the COA comparison matrices employed in the O-Plan framework [Tate, Dalton et al. 1999].

## **4.2.3.2 Probabilistic Approaches**

### **Overview of the Approach**

MDPs can be used to evaluate a plan, and compare the suitability of two different plans. As outlined above, a MDP associates costs with states and MDP algorithms are generally used to find minimal cost plans. In the context of military planning, many factors have to be balanced before choosing a plan.

### **Detailed Description of the Approach**

MDP algorithms usually assume that the cost of a task is given by a single numeric value. In Air Force planning, it is not clear that states can be ranked in this manner. Several different values need to be considered, including the monetary cost of a state, the duration of the state, and the risk associated with the state.

The output of a typical MDP algorithm is an optimal policy specifying the best task to perform in any given state. In standard MDP algorithms, multiple costs are combined by assigning weights representing the importance of each cost. Military planning experts are reluctant to combine these costs in to a single cost by simply applying weights.

Military planning experts suggest that costs should be ranked in order to ensure that lower ranked costs never take precedence over higher ranked costs. For example, a task that takes a long time might be understood to cost less than a task that is financially expensive.

The traditional approach to solving MDPs with ranked costs is to find all minimal cost policies for the most important cost, then repeat the process for the second cost, and continue until all costs have been considered in order. This requires a great deal of time and computation.

A simpler approach is used for military planning in which exponentially increasing weights are assigned to different costs. Hence, the cost of a state is just a linear combination of the component costs; but the weights are chosen in a way that simulates ranking of costs. The advantage of this approach is that minimal cost policies can be found using existing MDP algorithms. Specifically, the minimal cost policy is computed by using the labeled real-time dynamic programming (LRTDP) algorithm of [Bonet and Geffner 2003].

Computing the actual cost of a plan is unnecessary if the plan can be deemed unacceptable with less computational effort. Heuristics that can be useful in this regard are the probability of success. As noted in the section on plan forecasting, it is straightforward to compute the probability that a sequence of tasks will achieve a simple goal.

Before computing the cost of a plan, it is possible to determine the probability that the plan will succeed. If the probability is too low to be acceptable, then the plan can be disregarded without further computation. Similar heuristics can be defined based on the duration of a plan and the amount of resources required. Without determining the cost of a plan, it is possible to determine the maximum length of time that the plan will take as well as the minimum amount of resources. If the plan is too long or it requires more resources than available, the plan may be disregarded immediately.

### **Example**

The example given in the section on plan forecasting can be used to compute the probability of success of a plan, and similarly straightforward computations can compute resource and temporal bounds. Giving a detailed example illustrating the LRTDP algorithm is not necessary for the present report; it is sufficient to note that it is a greedy algorithm that successively chooses the best possible task at each step.

### **Analysis of the Approach**

As a methodology for plan analysis, the MDP approach appears to provide a comprehensive treatment of several factors. The process of assigning costs and comparing plans is not straightforward, but it can be computed and used as a helpful tool to choose among plans. There are two limitations of the methodology, however.

It is not at all clear that the various costs can be accurately measured. In other words, assigning a value to the “risk” associated with a state is a subjective process and it is not clear that the scale for this cost is at all related to the scale of a temporal or monetary cost. It is for this reason that military experts suggest that costs should be ranked rather than weighted. However, the developers of the MDP approach almost dismiss this suggestion by using a linear combination of weights anyway. While it is clear that this facilitates the computation of an optimal policy, it is not clear that the sacrifice in accuracy is justified.

An alternative approach to combining multiple independent criteria for a single evaluation is provided in [Bélanger and Guitouni 2000]. In this paper, a graphic tool is used to specify the relative importance of different criteria. After specifying the relative importance, the best plan is determined by using multiple criteria decision analysis.

The important point is that military experts can use the given tool to rank criteria, and the ranking is taken seriously in the formation of an overall ranking of plans. The approach to combining multiple criteria in the MDP approach might be too simplistic.

The second concern is that it does not appear that military experts have been consulted enough in the development of this approach. Many approaches to military plan analysis take expert opinion seriously, focusing mainly on translating expert evaluations into a particular formalism. By contrast, the MDP approach tries to force military plan evaluation into an existing decision-



theoretic framework. Although the system returns specific recommendations on plans, it is not clear that the recommendations match conventional military wisdom.

### **Efficiency Criteria**

The probabilistic analysis of plans is essentially a semantic analysis. The outcome of a plan is the state of the world that is deemed to be the most likely after the plan executes. The manner in which the plan is written is not considered at this level of analysis. Plans are ranked according to cost, duration, chance of success, etc. This ranking is an explicit tool for plan evaluation.

In the MDP approach, military experts are asked if a particular plan represents an appropriate course of action to achieve some goal. Hence, the military experts are considering whether or not the described sequence of actions is appropriate, rather than considering if the description contains errors. The CPN approach does not appear to consult military experts in the same manner.

The probabilistic approaches to plan analysis do not allow the user to perform structural analysis of a plan. This is particularly problematic because probabilistic models of plans are difficult to author without expert mathematical knowledge. The translation from military plans into probabilistic descriptions is likely to be an error-prone process, so structural errors may be introduced.

## **4.2.3.3 Visualization**

### **Overview of the Approach**

The standard visualization tool for evaluating a COA is a COA evaluation matrix. In this setting, all of the possible COAs to achieve a goal are placed on one side and all of the relevant evaluation criteria are placed on the opposing side. Traditionally, military planners would use this evaluation matrix to try and determine the best COA available. This approach can be automated in a naïve manner for display, providing a basic tool for visualization.

## Example

### O-Plan COA Evaluation Matrix

<a href="#">Help</a>   <a href="#">Map</a>   <a href="#">View TF file</a>   <a href="#">Select evaluations</a>			
Define task:	<a href="#">COA-1</a>	<a href="#">COA-2</a>	<a href="#">Add COA</a>
Set authority:	<a href="#">Auth</a>	<a href="#">Auth</a>	
Generate plan:	<a href="#">Plan</a>	<a href="#">Plan</a>	
nodes in plan	<a href="#">35</a>	<a href="#">17</a>	
longest path length	33	23	
minimum duration	4 hrs	3 hrs	
object types	<a href="#">3</a>	<a href="#">2</a>	
object values	<a href="#">4</a>	<a href="#">2</a>	
View plan:	<a href="#">View</a>	<a href="#">View</a>	

#### COA objectives

COA	Objective 1	Objective 2	Objective 3	Objective 4	Objective 5
1	evacuate injured abyss				
2	defuse terrorist bomb barnacle				

#### COA initial situations

COA	Weather	Road Delta Abyss	Road Abyss Barnacle	Road Barnacle Calypso	Road Calypso Delta
<a href="#">Default</a>	clear	open	open	open	open
1	clear	open	landslide	open	flooding
2	rain	flooding	open	open	open

**Figure 4-5 A COA Evaluation Matrix, taken from the O-Plan web demo**

## Analysis of the Approach

The O-Plan approach will be analyzed in detail in Section 4.2.4.1. Any approach to plan visualization can be used for plan analysis. However, some tools provide information that is more directly applicable to the analysis of a plan while it is being defined, or while it is being executed. For example, the NuSketch software can be used for visual evaluation of a plan. Similarly, the three-dimensional AsbruView model (see Section 4.2.2.3) can be used for evaluation. As noted in the paper summary section, this three dimensional view uses the horizontal direction to represent time, the vertical dimension to represent plan decomposition, and the depth direction to draw multiple plans on the same timeline. A similar visualization tool for plan evaluation has been described in [Kundu, Sessions et al. 2002]. As such, the approaches to visualization for plan representation, forecasting and monitoring can also be employed for plan analysis.

## Efficiency Criteria

Using visualization tools for plan analysis and evaluation is useful for catching problems that are nested deep within the plan structure and may not be easy to detect or even formally describe. However, if the underlying representation of plans is given in a formal ontology, then it would be better to use automated reasoning on this ontology whenever possible. In particular,

it would be valuable to have the system alert the user when a constraint, objective, or assumption cannot be satisfied. At the moment, the given approach to visual plan analysis does not provide this information automatically.

## **4.2.4 Plan Monitoring**

Monitoring the execution of a plan introduces new practical problems that need not be addressed in plan analysis or forecasting. In particular, there needs to be some kind of sensing infrastructure to provide feedback about plan execution. In the simplest case, this feedback could be provided by reports from field officers or by aircraft location reports. In a more sophisticated case, sensors with a view of the planning area might provide automatic updates. The measures of efficiency introduced in 4.1.4 are considered in this section.

The papers reviewed with respect to the two probabilistic approaches to plan representation (the MDP approach and the CPN approach) do not address plan monitoring. Therefore, this section does not contain a subsection discussing probabilistic approaches to plan monitoring. Certainly it would be possible to visually monitor the execution of a plan in the COAST system, but the software does not provide any special monitoring tools beyond the typical visual display. In principle, however, monitoring the effects of actions in a probabilistic setting is particularly important. The presence of uncertain outcomes is one of the main aspects that distinguishes plan monitoring from plan forecasting.

### **4.2.4.1 Ontologies and Formal Languages**

#### **Overview of the Approach**

Plan monitoring can be facilitated by an underlying representation of plans in a formal ontology. For instance, a representation of a plan in a formal ontology can be mapped to a visual display of plan execution [Lino and Tate 2004]. In order to actually monitor a plan as it unfolds, information from external sensors must be obtained and integrated. Using a formal ontology that encodes the goal-structure of a plan can be useful in determining where to place sensors, as well as which information must be monitored [Tate, Levine et al. 2000].

#### **Detailed Description of the approach**

The approach to plan monitoring described in [Tate, Levine et al. 2000] is focused on monitoring small unit army operations in an urban area. By consulting with subject matter experts, the authors try to determine the points during plan execution where additional information may be useful. It is suggested that there are many points where re-planning occurs, and decision aids may be useful.

Before a mission begins, the objectives of the mission are typically specified as an Operations Order (OPORD) using natural language. However, the structure of an OPORD is strictly defined and standard phrases and terminology are re-used. In this work, OPORDs are translated into the INOVA ontology language. Similarly, the actions executed in a plan are represented in a formal language; the details of the representation are not the focus of this section.

It is sufficient to note that objectives are entered through a sequence of menus, COAs are generated by O-Plan together with an evaluation matrix [Tate, Dalton et al. 1999], and then the user selects the plan for execution. The most important aspect of the representation language is that it encodes the goal-structure of a plan [Tate 1977]. The goal-structure indicates which tasks must be executed successfully in order for later tasks to be possible.

After the plan is selected, the plan monitoring process begins. The goal-structure of the plan is useful in plan monitoring for two reasons. First, an outcome of a task must be monitored closely if the task is a precondition to later tasks in a plan. This fact can help guide the placement of sensors, and it can also provide guidance to field units regarding the most important feedback to provide. Second, the goal-structure of a plan can inform plan repair. If a particular task fails to execute, then a repaired plan must address this failure by ensuring that any later task preconditions will still be satisfied.

The plan monitoring system is tested in a simulated environment, where a user can simply enter world events during plan execution. The idea is that user-entered events simulate information obtained from sensors regarding obstacles to plan execution. The system is able to use the goal-structure of a plan to determine which obstacles require plan repair, and which obstacles do not. The system repairs plans locally. When an obstacle prevents an action from executing, the system tries to insert a sequence of actions that will ensure that the preconditions of all later actions will still hold. The user is involved in plan repair through the same menus and evaluation matrices that are used in plan development.

### **Example**

The following example is based on an example in [Tate, Levine et al. 2000]. Consider an army unit that is exploring a sequence of buildings that may contain enemy forces. This objective can be described in the given system, and an initial plan may suggest that the building be explored in order from closest to farthest away. After visiting the first two buildings, the user may encounter an obstacle in the form of a minefield in front of the third building. One precondition for exploring the third building is that the unit is not damaged. As such, the system detects that the plan needs to be repaired. The system suggests that the minefield may be traversed using mine detectors, or that the dangerous area may be avoided by taking a different route. The user is asked to determine the appropriate repair.

### **Analysis of the Approach**

It is clear that goal-structure is an important feature of plans in the context of plan monitoring. A repaired plan that does not respect the goal-structure is certainly not going to be a useful repair. However, the paper does not provide a precise specification for goal-structure. The original approach in [Tate 1977] does not translate directly into the domain of military planning without some modification, and it would be useful to know precisely how goal-structure is being represented.

While this approach appears to provide a useful step towards realistic plan monitoring, it avoids a difficult problem. The system simply has users enter obstacles and events in order to generate repaired plans. This is useful for testing, and it is also useful for integrating explicit reports from field units. However, a more realistic front-end for monitoring would involve a visual display of the environment, from which obstacles would need to be abstracted. The work in

question does not address this problem. Overall, the use of goal-structures to perform local repairs on damaged plans is a useful contribution to plan monitoring.

In Section 3.4, several paper summaries are provided for papers focusing on monitoring the interaction between several concurrent plans. This existing work on interacting plans addresses a relatively isolated problem in plan monitoring, so it has not been selected for further discussion. Our focus is on providing detailed descriptions of approaches to plan management that fare well with respect to plan representation, plan forecasting, plan analysis, and plan monitoring. Hence, although monitoring multiple concurrent plans is an important problem to be addressed in the development of our mock-up software, this problem has simply not been addressed substantively in the literature reviewed.

### **Efficiency Criteria**

Analyzing the goal-structure of a plan provides assistance with re-planning as well as guidance regarding required feedback. However, both of these points are related to setting up the plan to be monitored properly. While the plan is actually executing, a plan monitoring system should provide feedback when a constraint, objective or assumption is violated. Unfortunately, the proposed approach to plan monitoring does not provide any automated assistance with respect to this kind of feedback.

A major problem is that the approach outlined does not provide any specific details regarding interactions between concurrent plans.

## **4.2.4.2 Visualization**

### **Overview of the Approach**

A variety of tools have been proposed for visualization of plans and for graphic depiction of the current situation. These tools were introduced in the paper summaries of Chapter 3 with respect to each aspect of plan management. Using such tools may provide a starting point for monitoring a plan during execution.

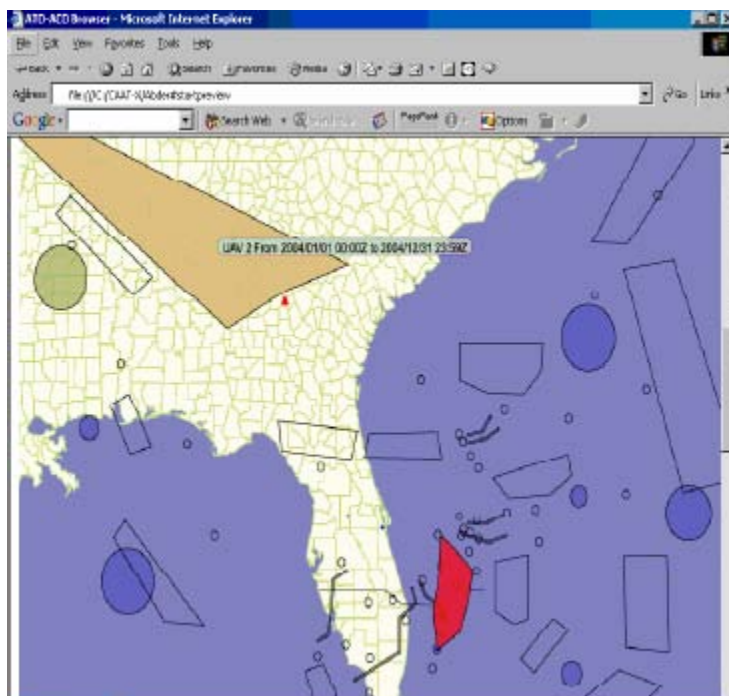
### **Detailed Description of the approach**

The highest level visualization tools, such as CommandView, simply provide maps with overlaid images indicating the presence of military units and activities [Hollands 2006]. This level of detail is not really sufficient for monitoring plans in real-time. Several visualization tools have previously been considered in [Boukhtouta, Bélanger et al. 2006]. Each tool is developed with a different purpose. For example, the FalconView system is primarily intended to assist with route planning, whereas the Situation Awareness module of the Theatre Battle Core Systems performs a fundamental role in preparing, planning and re-planning aircraft missions.

For the present report, one system of particular note is the Canadian ATO/ACO Tool XML Interpreter System (CAAT-Xi). This system is intended to provide mission updates in a graphic format, based on XML input. The XML input includes the Air Tasking Order (ATO), the Airspace Control Order (ACO), as well as Airport information. This input is used to display

mission updates on a map displayed in a web browser on a computer screen. Since XML files are simply text files, this provides a lightweight solution for providing graphical updates of mission status.

### Example



**Figure 4-6 A screenshot taken from CAAT-Xi.**

### Analysis of the Approach

None of the visualization tools considered in [Boukhtouta, Bélanger et al. 2006] provides a complete plan monitoring solution. The tools provide a mechanism for users to view the geographic locations of objects while a plan is executing, but little planning-specific information is provided. It is noted, for example, that the intent of an entity is not displayed on screen. Similarly, these visualization tools do not provide a mechanism for monitoring the achievement of objectives, or the interaction between plans. The communication between agents scheduling tasks to achieve multiple goals has previously been discussed in [Pollack 1996]. This kind of concern is not addressed by simply providing a graphic display of significant entities.

Although the existing visualization tools are not specifically tailored for plan monitoring, it is clear that visualizing the units executing a plan is a useful component of monitoring. As such, it may be useful to use an existing visualization tool in the development of a plan monitoring system.

The CAAT-Xi software appears to be a reasonable candidate system for development, due to the fact that it supports XML input. The XML language is a flexible tool for describing

ontologies, so it seems possible that the CAAT-Xi system could be combined with some of the previously discussed work on ontologies and formal grammars.

### **Efficiency Criteria**

For the most part, visualization tools do not provide assistance to a user regarding the most important feedback to monitor. Instead, visualization tools provide a complete picture of the environment, and the user must use external information to determine the particular areas to monitor. Some existing visualization tools, such as the Theatre Battle Core System, provide the user assistance with re-planning. Multiple concurrent entities can be visualized concurrently, although the existing tools do not explicitly allow a user to view the interactions between the plans these entities are executing.





## **5 INVESTIGATION OF TECHNIQUES AND APPROACHES**

Following the discussion in Chapter 4, it is clear that none of the approaches considered treats every aspect of planning effectively. This chapter aims to combine the favourable aspects of several approaches into a new, more powerful approach to dynamic plan management.

The chapter is structured as follows: Section 5.1 builds upon the synthesis description in the previous chapter and compares the strengths and weaknesses of the most promising approaches. Section 5.2 proceeds to combine these existing approaches into a high-level description of a system for plan representation, plan forecasting, plan analysis and plan monitoring. In Section 5.3, we will describe some important details of the system implementation. Section 5.4 provides an assessment of the strengths and weaknesses of the combined approach, and how well the objectives of dynamic plan management have been attained. The chapter concludes in Section 5.5 with an outline of future research directions.

### **5.1 Comparison of the Strengths and Weaknesses of Selected Approaches**

In this section, we re-capture and summarize the most promising systems in the literature, and present the contributions, strengths and weaknesses in tabular form.

#### **5.1.1 Plan Representation**

The manner in which plans are represented influences the manner in which plans can be forecasted, evaluated, and monitored. In Section 4.2, the most promising formal approaches to plan representation were presented in detail. If we consider MDPs and CPNs as separate approaches, then four distinct formal representations of plans are considered. These approaches have been chosen from the papers surveyed in the initial literature review survey of Section 3.

As noted previously, the term “ontology” refers to a wide range of formal languages for describing hierarchical categories of entities. Each ontology for military planning has different strengths and weaknesses. In principle, one could combine several ontologies to create one

large ontology that combines the best features of each of its components. In discussing the relative strengths and weaknesses of the ontological approach, it is reasonable to introduce a hypothetical “combined” planning ontology. Informally, the combined ontology is the most expressive ontology that can be created from the existing planning ontologies in the literature. In practice, defining this ontology would require some additional research effort.

Table 5-1 summarizes how each of the four representation schemes fares with respect to the measures of efficiency identified in Section 4.1. The left side of the table contains the measures provided, and the top of the table contains the four most promising approaches considered in Chapter 4. Each cell of the table contains one of two things. If a given measure is addressed by a given approach, then the corresponding cell provides a brief description of how the measure is addressed. In cases where one of the measures is not addressed, the corresponding cell contains the character ‘N’.

**Table 5-1 Measures of efficiency applied to representation schemes**

	<b>(Combined) Ontology</b>	<b>NuSketch</b>	<b>MDP</b>	<b>CPN</b>
Plan abstraction	Task decomposition, INOVA system	N	N	N
Task specification	Variety of formal languages	By visual sketching	Formal language	Graphic model
Constraint specification	INOVA uses constraints extensively	N	N	N
Objective specification	Objectives are explicitly specified entities (e.g. in SPAR)	Directive arrows and target glyphs	Specified as outcome states	Modelled by tokens on CPN
Definable by military users	N	Standard COA sketching	N	N
Readable by military users	N	Standard COA sketching	N	N
Uncertainty	N	N	Effect probability	Effect probability

A superficial examination of this table indicates that none of the four approaches addresses every one of the measures of efficiency. As such, none of the approaches is uniformly better than the others. In order to compare the approaches in more detail, we consider how each approach fares with respect to the measures of efficiency for plan forecasting, plan analysis and plan monitoring. Tables 5-2, 5-3, and 5-4 contain the results of these comparisons.

## 5.1.2 Plan Forecasting / Projection

For plan forecasting, the probabilistic approaches fare well due to the ease with which uncertainty can be represented. However, the ontological approach allows an explicit representation of time that is very useful for deterministic forecasting.

**Table 5-2 Measures of efficiency applied to plan forecasting**

	(Combined) Ontology	NuSketch	MDP	CPN
Resource checking	INSPECT system	N	Resource use given in task	Consumed tokens
Pre-processed outcomes	Templates for ACPT	N	N	N
Probabilistic forecasting	N	N	Joint probability	Joint probability
Complete on time	Explicit temporal ontologies	N	Likelihood of success based on actions and joint probability	Likelihood of success based on actions and joint probability
Plan status	Deterministic effects and explicit representation of time supports status check	N	N	N

### 5.1.3 Plan Analysis / Evaluation

For plan analysis, a fundamental concern is the integration of military expertise. Several approaches succeed in this regard.

**Table 5-3 Measures of efficiency applied to plan analysis**

	(Combined) Ontology	NuSketch	MDP	CPN
Incorporate military expertise	Experts interviewed	N	Experts interviewed	N
Structural analysis	INSPECT plan checker	N	N	N
Semantic Analysis	SHAKEN evaluator	N	Expert cost assignment, and probability of success	Analyse probability of success
Ranking of plans	SHAKEN ranking based on expert opinion	N	Assigning weights to different features, taking sums	Based on probability

## 5.1.4 Plan Monitoring

For plan monitoring, the ontological approach is the only approach that has been exploited in the literature surveyed. Note, in particular, that visual sketching systems like NuSketch have not been integrated with visual plan monitoring systems.

**Table 5-4 Measures of efficiency applied to monitoring**

	(Combined) Ontology	NuSketch	MDP	CPN
Feedback guidance	Use goal-structure to place sensors and monitor	N	N	N
Assisted Re-planning	In O-plan, exploiting goal structure	N	N	N
Violations	Detected by explicit monitoring of states	N	N	N
Interacting plans	N	N	N	N

## 5.1.5 Plan Visualization

Several different approaches to visualization were presented in the literature review. Roughly, these approaches can be categorized in terms of two broad categories. One natural form of visualization could be called *state visualization* or *world visualization*. This refers to visual depictions of the current state of the world while a plan is being executed or forecasted. This category includes map-based visualization tools where units are displayed over the map of a designated area, and this category also includes the display of evolving entity relationship diagrams. This is the main form of visualization employed for plan monitoring and plan forecasting.

The second form of visualization could be called *plan visualization*. This refers to visual depictions of the structure of a plan. This category includes graphic representations of HTNs, as well as most variations of a Gantt chart. This form of visualization is particularly appropriate for plan monitoring and plan representation. It can also be useful for representing sequencing constraints on tasks.

At a minimum, the literature suggests that plan management software should provide some approach to each of these categories of visualization.

## 5.1.6 Conclusion

These summarizing tables obscure some important differences between approaches. For example, the tables do not capture the subtle differences between MDPs and CPNs. However, the tables do effectively indicate some general trends.

The representation based on ontologies is strong in the sense that all of the concepts involved in planning have precise specifications. This is not surprising, since concept specification is exactly the kind of task for which ontologies are normally employed. What is perhaps more surprising is that an ontological representation supports nearly every desired feature related to forecasting, analysis, and monitoring. There are two reasons why ontologies fare so well in this comparison. First, there has simply been a great deal of research on the use of military planning ontologies; this volume of research naturally leads to a more complete treatment of the problem. Second, the comparison is performed with respect to a hypothetical “combined” ontology. If the grids were broken in to separate columns for SPAR, INSPECT, INOVA, etc. then each column would have different strengths and weaknesses. As noted previously, one of the main limitations of an ontological approach is that creating ontologies requires specialized technical knowledge. However, by packaging an ontology with appropriate authoring tools, this is not a problem. We also re-emphasize that ontologies describe the internal representation of plans, which is transparent to the military operator, who is using the system to design and evaluate plan instances.

Note that the NuSketch system has been included as an illustrative example of plan representation through visual plan sketches. The NuSketch system has been singled out as the representative example because it has performed well with respect to related software. However, our focus here is on approaches, and the inclusion of NuSketch should be understood as an approach rather than a specific software implementation of the approach. The main strength of the NuSketch approach is that military experts are able to author and read plans in the traditional manner. However, only COA plans are considered in the present version. The NuSketch software does not directly address plan forecasting, analysis, and monitoring.

The main strength of the MDP and CPN approaches is the representation of uncertain task outcomes. However, both formalisms are somewhat lacking with respect to the specification languages for military resources. The two approaches are relatively similar from a high-level perspective, but the MDP approach has two important advantages. First, the MDP plan-ranking scheme was designed based on expert critiques. Second, the MDP research community appears to be larger than the CPN research community. However, despite the fact that the CPN community is relatively small, the COAST system is a complete planning tool that is the subject of continued development funded by the Australian military.

It is worth noting that none of the representation schemes discussed in this section make any significant attempt to model multiple concurrent plans, nor do any of the representation schemes support visualization for plan monitoring. Both of these features would be valuable, and both of these features will be addressed in our proposed mock-up software.

## 5.2 High-level Description of the Proposed Software

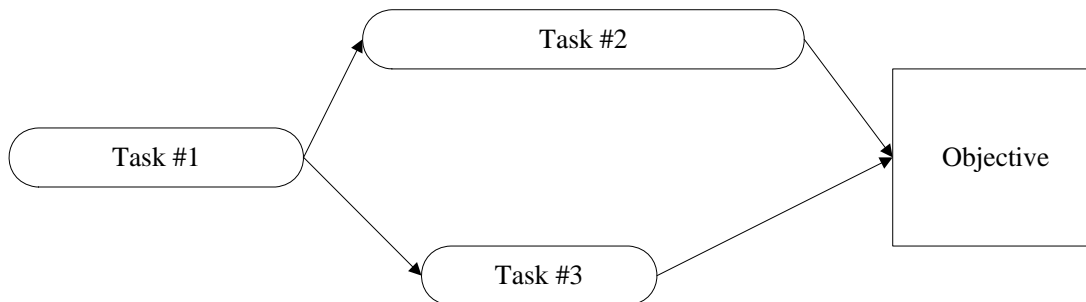
Knowing the strengths and weaknesses of existing systems, we now define a new approach that combines the respective strengths and overcomes the weaknesses. This section

- derives a combined approach from existing elements of other systems
- establishes an ontology of plan elements that will be used in the new system

- explains how these elements will be used in the system to perform dynamic plan management
- shows how the elements will be visualized.

## 5.2.1 Identification of a Combined Approach

As noted previously, none of the approaches reviewed addresses all of our measures of efficiency. It is therefore desirable to use a combined approach that exploits the best features of several different formalisms. As a starting point, we have seen that a formal ontology can provide a precise, unambiguous representation of plans. The literature review also suggests that ontological models are well suited for automated forecasting, analysis and monitoring. However, there are many different planning ontologies, and it is not immediately clear if one is clearly dominant over the others. Rather than choosing some particular ontology, we propose a simple planning ontology that captures the key aspects of plan representation identified in the literature. Given such an ontology, we could then revisit the existing ontological tools for automated forecasting, analysis and monitoring. As a starting point, consider the HTN representation of plans illustrated in Figure 5-1.



**Figure 5-1 Plan Representation**

This representation can be used to represent additional information that is not typically available in the HTN framework. In particular, if we view the horizontal dimension in terms of time, then the width of each task block can be understood to represent the duration of a task. The horizontal location of a task can be understood to represent the start time of a task on a fixed global timeline. Furthermore, the edges can be annotated with probabilities representing effects. This extended graphical representation of plans captures many of the features that are covered in the literature. However, this graphical representation does not necessarily lend itself to automated analysis or forecasting.

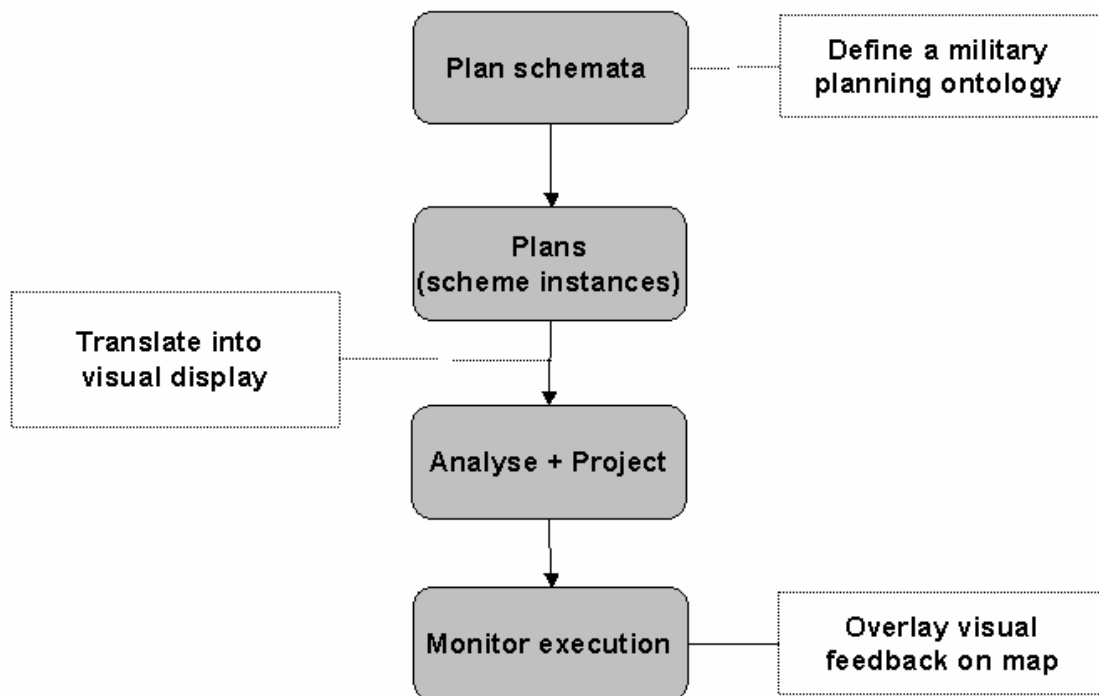
It is possible to define an ontology that captures the extended HTN framework illustrated in Figure 5-1. In particular, we have seen that it is possible to include sequencing constraints and decomposable tasks in an ontology. Similarly, following the specification language for MDPs, it is straightforward to specify probabilistic effects in a formal ontology specification language. The semantics of such effects may be complex; however, it is easy to simply append probabilities to different outcomes. As such, a new planning ontology can be defined in a manner that essentially combines the most desirable features of existing ontologies, together with the key features of HTNs and probabilistic formalisms.

The combined ontology described above would be very expressive, but it would still suffer from the main limitation of most ontological models. In particular, since ontologies are not easily understandable by military experts, an ontology is only appropriate as an internal representation of a plan. For plan authoring or plan visualization, a more usable interface is required. As such, the development of plan management software can be broken in to two main tasks. First, a suitable ontology for plan representation can be developed. Second, visualization and analysis tools can be developed as an interface to the ontology.

An ontology for representing plans can define *plan schemata* specifying precisely what constitutes a plan. Instances of the schemata can represent plans, which can then be analyzed formally and displayed visually. The NuSketch system and the COAST system both illustrate that an internal representation in an AI formalism can be successfully integrated with visualization software. This suggests the following basic approach:

1. Define a planning ontology that can represent the tasks, resources and objectives involved in an Air Force plan. This can be defined using freely available ontology editors, borrowing the structure of existing military ontologies.
2. Perform forecasting and analysis on formal plan descriptions, using ontological tools.
3. As the plan executes, translate the plan to a visual image for display. Several views can be defined to present different features of the plan.

The proposed approach can be pictured as follows:



**Figure 5-2 Proposed Approach**

Note that the military planning ontology can be prepared in advance, and instances of plan schemata can be obtained offline. Hence, the starting point for analysis and monitoring is a plan

provided in advance. This approach does not require or provide any mechanism for generating plan instances automatically.

There are also two main weaknesses to this approach:

1. It is not clear how plans should be authored. Following the NuSketch approach, it would be preferable to use a visual tool for plan authoring using some form of sketching on screen. Providing such an authoring tool is beyond the scope of the present project.
2. It is difficult to perform probabilistic forecasting in this framework.

However, both of these weaknesses can be addressed to some degree. For example, there are existing ontology authoring systems that can be used by subject matter experts [Noy, Grosso et al. 2000]. Regarding the second weakness, the development of ontological models of probability is an ongoing research problem. It may be the case that uncertain outcomes can be added. Therefore, the advantages of the ontological model outweigh the advantages of the MDP and CPN approaches.

One of the main problems to be addressed at a software level is the development of tools for the visualization of plan instances. To display a plan as it executes, one would need to take periodic snapshots of the ontological description of the plan. If these descriptions can be translated into XML, then it might be possible to use existing XML parsers to display the current state of affairs visually over a map. It might also be possible to re-use visualization tools, such as those employed in the CanCoastWatch (CCW) test-bed.

## 5.2.2 Plan Elements

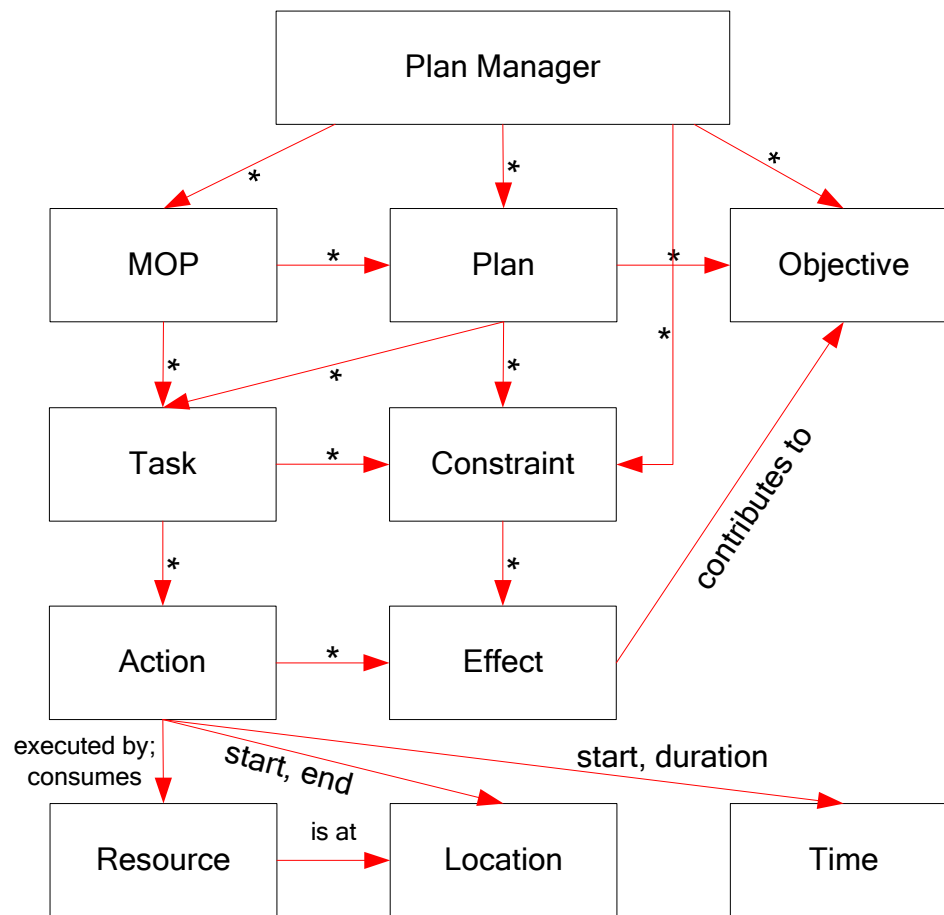
In the literature review, several existing military ontologies were mentioned. Developing a sophisticated ontology can be a major research effort, so it is important to specify the scope of the ontology in advance. In this section, we give a precise specification of the plan elements to be considered in the present project. The elements that we consider are a restricted set of the elements considered in the SPAR ontology. From a general perspective, we will consider the following general categories of elements:

**Table 5-5 Plan Elements**

Time	Resources	Plans	Effects	Objectives (goals)
Locations	Actions	Tasks	Constraints	Measures of Performance (MOP)

These elements will be specified in the object-level description of the proposed software. Given the literature reviewed, these categories of elements appear to be sufficiently expressive for managing and evaluating Air Force plans. The following diagram provides a rough object hierarchy representing the relationships between these elements.





**Figure 5-3 Element Hierarchy**

This object hierarchy has been defined using the Protégé ontology editor. It is strictly a component hierarchy, in the sense that all arrows represent a “has a” relationship. Arrows labeled with an asterisk represent a one-to-many, i.e. “has (possibly) multiple” relationship. Many of these elements have specializations, forming an “is-a” relationship, which for the sake of conciseness is not shown.

The elements in Figure 5-3 will be described in the following sections. The implementation of all elements and their specializations will be listed in detail in Section 5.3.4.

### 5.2.2.1 Time

We distinguish three different types of time elements:

- *Time points* are represented relative to a defined reference date, e.g. Julian Date.
- *Durations* mark an amount of elapsed time, measured e.g. in seconds. For example, an action has a duration.
- *Time intervals* are fixed in time, so their start and end time are time points. They are used in time window constraints.

### 5.2.2.2 Location

Locations are positions in space. We distinguish:

- *Point locations*, given in three-dimensional coordinates relative to some defined coordinate system, e.g. UTM
- *Trajectories*, consisting of a start and end point location
- *Area Locations*, which are specified by the vertices (point locations) of a bounding polygon. This allows one to specify approximate locations or the location of a group of resources.

### 5.2.2.3 Resources

Resources are entities that are used or consumed when actions are executed. We distinguish between

- *consumable* resources (e.g. fuel) and
- *allocatable* resources (e.g. personnel, platforms).

The fundamental difference between these two classes of resources is that allocatable resources are individually distinguishable. By contrast, consumable resources can only be specified in quantities. Both allocatable and consumable resources can be *attached* to other resources.

Attachment corresponds to any of the following physical activities:

- fuelling an airplane
- staffing an airplane with personnel
- grouping platforms together into a military unit (squadron, platoon etc.)

The last point allows grouping resources at incremental levels of detail. Allocatable resources at a higher level are groups of allocatable resources at lower levels. In principle, there is no limit to the number of levels or to the size of groups.

For an allocatable resource to be allocated to an action, the executing allocatable resource must have various (allocatable and consumable) resources attached to it, as specified by resource constraints. Attaching resources to an AllocatableResource is usually a separate action, but it can be implicitly defined as part of an action. (This is so that the plan need not contain separate actions for cumbersome details such as personnel boarding a plane.) Likewise, after the task is finished, resources can be disattached and become available again. Consumable resources are different, however, in that they are used up by a task and will no longer be available.

### 5.2.2.4 Constraints

Constraints are conditions that must be satisfied for a task or plan to execute successfully. For example, a sequencing constraint indicates that one task needs to be executed before another. One can distinguish *preconditions*, which must hold at the time the task is started, and *corequisites*, which must hold for the entire duration of the task. There are many different types

of constraints related to tasks, resources, situation information and environmental conditions. Additionally, the plan manager can also specify *global* constraints that must hold for all tasks.

### 5.2.2.5 Effects

Effects characterize the direct or indirect outcomes (changes to the environment) brought about by actions. We distinguish:

- *direct* effects, which are caused by the execution of some atomic action,
- *cumulative* effects, which are computed from simpler effects. Most often, a cumulative effect is attained when all its sub-effects are attained. But a cumulative effect can be computed in many other ways from its sub-effects.

The literature on effects-based planning also characterizes *indirect* effects, which are caused by actions in response to other effects. For instance, the enemy response to a surveillance action may be radar spoofing; its effect, a reduced sensing capability, is indirect, since it does not result from our own actions. Our simple model does not allow specifying indirect actions or other means by which a direct effect causes the indirect effect. However, an indirect effect can be modelled as a cumulative effect.

Our model obeys an important principle of causality, namely that each simple effect is caused by a single atomic action. Even so, we can compute and refer to a well-defined set of (simple, cumulative, and perhaps indirect) effects caused by the execution of a task or plan; namely, the set of all effects caused, directly or indirectly, by any action executed as part of the task/plan. Effects that seem to violate the causality principle, such as destruction by simultaneous fire from multiple platforms, can be modelled as composite effects resulting from the damage (simple effect) caused by each individual platform.

Effects can also be prerequisites for other actions. For this reason, they can be used as constraints as well.

### 5.2.2.6 Goals (Objectives)

A goal is used to describe the desired outcome of a plan. Similarly to cumulative effects, whether a goal is attained during the execution of that plan can be computed from lower-level effects. But goals are distinguished from other cumulative effects in that they are at the highest level in the composition and causation hierarchy, and in that they can be specified within plans and tracked for the purposes of plan evaluation and monitoring.

### 5.2.2.7 Actions

An *action* is an elementary activity that is *executed* by a single platform (allocatable resource) at a specified location (point, trajectory or area) for an explicitly defined duration (expected time to complete the task). (The duration of an action could sometimes be computed from the speed and other characteristics of the platform executing it, but we will not build this sophistication into the mock-up.) An action has certain *effects* on the environment at that location, as well as on the resource itself (change in location, consumption of consumable resources, amount of

time spent). Conversely, the effects of an action may be conditioned by the state of the environment at the time the action is executed:

- the action's actual duration may be longer or shorter than anticipated
- the action may consume more or less resources than anticipated
- the action may not have the desired effect (e.g. lost person found during the search task)
- the executing platform may be neutralized during execution

As a result of these influences, the action may succeed or fail. We make the simplifying assumption that actions cannot be interrupted. (Otherwise, we would need to define partial resource consumption and effects, as well as intermediate locations.)

### 5.2.2.8 Tasks

A *task* is a collection of actions subject to constraints. To allow specification of a structured task hierarchy, we distinguish:

- simple tasks, consisting of just one action
- composite tasks, consisting of several subtasks.

Many of the properties of tasks correspond to those of actions and are computed recursively:

- The *set of actions* represented by a composite task is the union of the sets of actions represented by the subtasks.
- We use the term *scheduled task* to refer to a task whose component actions have been recursively scheduled (assigned definite start times and possibly durations), obeying all specified constraints. The *start time*, *end time* and *duration* of the scheduled task then are implicitly determined from the point at which execution of the first action begins to the time point at which the last scheduled action completes.
- Execution of a task *succeeds* if all subtasks succeed (and no constraint has been violated during execution); otherwise it *fails*.

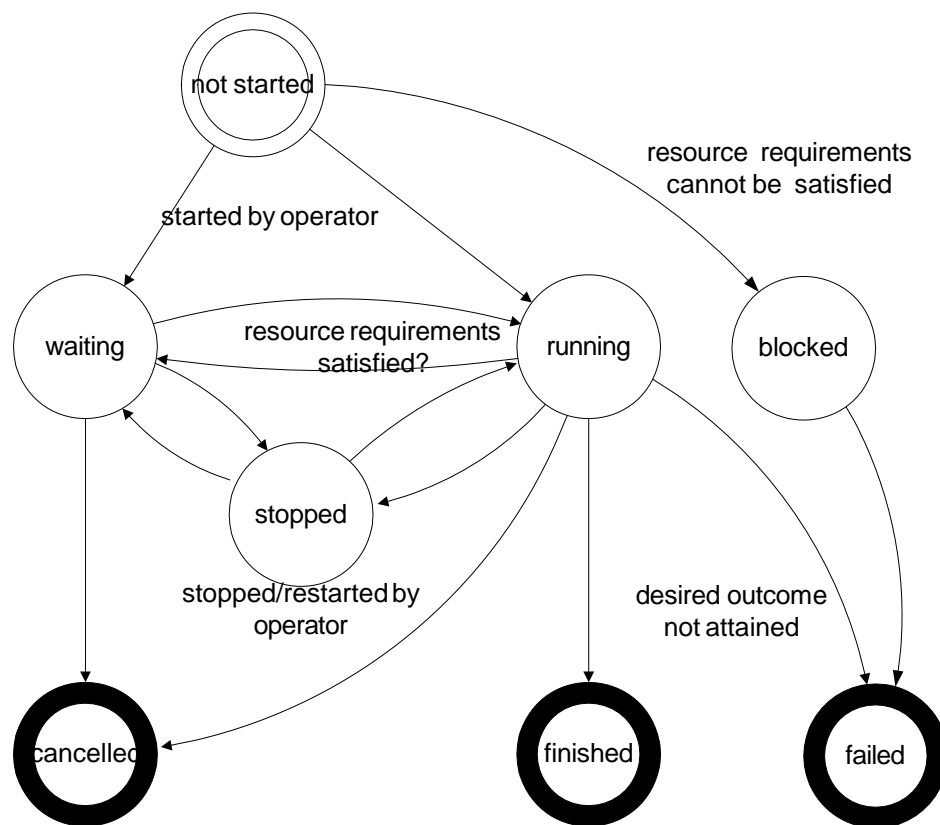
The *effects* of a task are the effects of its actions, as well as all cumulative effects resulting from these.

During plan execution, each task is assigned a *status*, which transitions over time as shown in Figure 5-4. The different status values have the following meaning:

- **notStarted**: not all resource constraints satisfied yet, so task cannot start
- **waiting**: task has started but is currently waiting for resources to become available
- **running**: task has started but not completed yet
- **stopped**: task is interrupted
- **blocked**: task cannot be executed because of constraint violations
- **finished**: last action of this task has completed successfully

- **cancelled:** task (instance) has been permanently cancelled
- **failed:** task cannot be completed within the given constraints, resources or time limits, or has finished and failed to produce the desired effect(s).

The execution of a task may fail for various reasons: First, just as for actions, the desired effect of the task may not be obtained. Secondly, a subtask may fail, and another subtask depends on the successful execution of this task. Thirdly, due to deviations (e.g. delays) in the execution of earlier subtasks or due to environment changes, constraints that were originally satisfied become violated, and a later subtask cannot execute. When it becomes evident that a task is going to fail, it makes sense to interrupt it and free its allocated resources, rather than continuing to execute useless subtasks; the mock-up task scheduler will allow this. Interrupting a task interrupts all its subtasks. All actions in progress are completed, but no new actions are scheduled or executed.



**Figure 5-4 Task status flow diagram**

### 5.2.2.9 Plans

A *plan* consists of a collection of tasks, a collection of goals, and a collection of constraints. The constraints apply across all component tasks. The main distinguishing feature between tasks and plans in this setup is that plans have explicitly specified goals. A plan is executed by executing the component tasks in accordance with the component constraints. A completed

plan's attainment of these goals can be evaluated, or an ongoing plan's present progress towards them can be monitored or projected into the future.

### **5.2.2.10 Measures of Performance**

For plan evaluation, it is useful to define some Measures of Performance (MOPs). These measures can be used to determine the degree to which a plan is successful, as well as for graphing the performance of a plan over time. In the domain under consideration, there are three primary measures to take into account:

1. How many goals are satisfied upon plan completion?
2. How long did the plan take to complete?
3. How many resources were consumed during the plan?

MOPs are usually evaluated by aggregating the corresponding information (goal achieved, time/resource consumption) for the underlying tasks. Each type of MOP essentially consists of a target value that can be compared with actual performance. For example, a plan might be targeted for completion in 2 hours, but then it might actually take 2.5 hours. A natural quantitative measure of temporal performance would be  $-0.5$ . Similar remarks apply if a plan has a targeted number of objectives to achieve or a targeted amount of resources to consume.

It may be useful to aggregate the performance of several plans. For this reason, a MOP can refer to more than one plan.

### **5.2.2.11 Plan Manager**

The top-level element is the PlanManager. It maintains an overall time horizon, a set of (multiple) plans, a set of global constraints, a set of overall goals to be accomplished, and a set of performance measures against which the plans are evaluated. This allows the specification of multiple plans and the study of their interactions. We mention two interactions between plans in particular:

- A plan may compete with another plan for allocatable and consumable resources.
- A plan may depend on another plan that brings about the environment (effects) it needs to satisfy its own constraints.

### **5.2.2.12 Dynamic Plan Elements**

Dynamic plan elements are those elements that change during the execution of a plan. These elements are the focus of plan monitoring and plan forecasting. Note that not all the elements given in Table 5-5 are dynamic elements. Some are entirely static, whereas others contain some dynamic aspects or attributes. For the purposes of the present project, we identify the following dynamic elements and attributes:

- time (most obvious dynamic element). The current point in time changes at a constant rate when a plan is executed. As time elapses, it is possible to monitor the duration of actions.

- the action currently scheduled for a platform
- the location associated with each platform:
  - The location of controlled platforms changes in response to planned transit actions.
  - The location of neutral and enemy platforms also changes in response to transit actions, but these actions are not modeled.
- the amount associated with a consumable resource. For instance, a transit action causes the amount of fuel in a platform to decrease.
- the attachment of a platform to higher-level units. For instance, a fighter plane might execute one action in a formation with others and execute another action individually.
- whether the effects of actions were actually obtained
- task status — whether a task has been started, is running, or is completed
- measures of performance—which are continually updated

It is important to distinguish the *predicted* and the *actual* change that occurs to a dynamic element. Furthermore, it is useful to identify changes that can be *controlled* to some degree by the choice of actions in a plan. For example, changes in time are predictable but cannot be controlled. The position of an enemy unit is very difficult to predict and certainly cannot be controlled. Hence, it must be monitored by observation. On the other hand, the position of a friendly platform is directly affected by the actions in a plan and can be predicted with high accuracy. The probability of success of a search action can be controlled within certain bounds (by lengthening the search), but the outcome (whether or not successful, and after how much time) ultimately cannot be predicted.

To keep the mock-up implementation simple, we assume the following components of plan elements to be static during plan execution:

- the constraints on a plan
- the objectives of a plan
- the component tasks in a plan
- the actions of a plan
- the effects caused by actions (but not whether they are attained)
- the set of platforms available for use (but existing platforms may become unusable)

These assumptions are useful in the development of mock-up software, but clearly might not hold in real AF planning domains. In particular, one would want to introduce additional tasks and possibly resources to repair a failing plan.

The assumption that actions and action effects are static is particularly significant. This assumption allows a small set of actions to be specified at the outset of a plan, with effects given by a *transition function* that explicitly specifies the effect of each action. Roughly, the effect of an action is a specific change to the environment that occurs when the action is performed. Note that the effect of actions may or may not be obtained, so it will still be necessary to monitor the actual change in the environment that follows an action.

### 5.2.2.13 Situation Updates

The description of a plan generally does not include information about the evolution of the world. For example, a particular enemy platform may fly around in a designated airspace in an unpredictable manner; the specification of a plan cannot dictate this movement. In a deployed application, sensors would monitor the movement of enemy units in real-time while a plan was being executed. A plan might include conditional tasks that depend on uncontrolled movement, but the specification of a plan need not specify any predicted behaviour.

The approach taken in our software will be to provide two separate inputs. One input represents the structure of the plan, including all of the component objectives, tasks, and constraints. The second input is a set of *situation updates*, which represents information that might be obtained from sensors and units in the field. At the most basic level, situation updates are intended to model actual changes in the state of the world. In this case, the information provided at time  $n$  is understood to represent the actual state of the world at time  $n$ . However, it is also possible to use the same format to represent predictive information about future points in time.

We can think of the situation updates as consisting of two distinct components representing actual changes and predictions, respectively. However, this partition is actually too simplistic, since predictions change in response to the actual updates. Moreover, the situation updates must include information about the actual effects of actions, and this information depends on the structure of the plan. In order to address this subtlety, the proposed mock-up will employ a time-stamping syntax to specify actual situation updates as well as predicted situation updates. For the moment, it is sufficient to simply emphasize that two distinct inputs are provided to distinguish between the structure of a plan and the predictions about changes to the environment.

## 5.2.3 Plan Abstraction

Representing plans at different levels of abstraction is an important feature in Air Force planning. The most promising approach to abstraction that is employed in literature reviewed is the approach taken in the HTN framework. In this framework, abstraction is represented through *task decomposition*, where a task can be decomposed into sub-tasks. The proposed software mostly follows this approach: A plan will be composed of a set of tasks, each of which in turn is composed of a set of actions subject to constraints. However, this three-layer approach will not be fine-grained enough for modeling more complex military scenarios. For this reason, subsets of the actions of a task can be grouped together into a subtask, which also contains the relevant subset of constraints. The nesting of tasks allows an arbitrary number of levels of abstraction.

In a similar way, one could devise a subplan hierarchy for plans, whereas the subplans of a plan would accomplish subgoals contributing to the plan's goal. However, the proposed software will not take this approach. Instead, these subplans will be specified as tasks (the highest-level components of a plan), and the subgoals as effects, attained by the tasks. The definition of the plan's goal would specify how these effects contribute towards accomplishing the goal. In this way, the task hierarchy and the effects hierarchy eliminate the need for a separate plan hierarchy.



Beside the task and effects hierarchy, there is also a resource hierarchy for allocatable resources. This allows a high-level representation of groups of platforms, for instance, reducing the cluttering in map-based views. Groups of resources can be specified in resource constraints for high-level tasks and plans, to express the need for allocating a whole fleet of platforms.

The distinction between plans, tasks and actions on the one hand, and between objectives and effects on the other hand, provides a conceptual abstraction. For instance, a plan objective could be:

Objective: Get all agents out of the range of enemy fire.

So a plan can be viewed as an underspecified recipe for accomplishing a goal under constraints. The component tasks of a plan then specify (possibly several different) concrete ways of accomplishing that goal while satisfying the constraints. So a concrete task might have the following form:

Task: Send two helicopters from their home base to the crash site, extract stranded crew, and bring them to the military hospital at location  $(x,y)$

The actions contain a concrete, fully specified description of each step of a task, and the respective effects attained:

Action: Fly one helicopter in a straight line from the crash site to location  $(x,y)$ .

Effect: All passengers are at location  $(x,y)$ .

The last step is to *schedule* some of the actions, that is, to assign start times and allocate resources to them. This must be done so that each task's constraints are satisfied. A scheduled action is the most concrete description. The concrete effects of a scheduled action hold at the scheduled end time.

Now the effects attained by the set of scheduled actions propagate up in the effects hierarchy, entailing more and more abstract cumulative effects. Ultimately, the plan objective, which sits at the highest level of abstraction, is satisfied.

## 5.2.4 Dynamic Plan Management

In the following sections, we describe how the proposed software will handle plan representation, plan forecasting, plan analysis, plan monitoring, and plan visualization. For each aspect, we specify some goals as well as some high-level description or schematic examples to illustrate how the goals will be met.

### 5.2.4.1 Plan Representation

The goal of the proposed approach to plan representation is the following:

- Represent plans in a simple yet expressive and extensible ontology of plan elements, allowing multiple levels of abstraction.

- Make this internal representation transparent to the user through suitable visualization tools.
- Represent assumptions and predictions related to a plan.

The proposed software will satisfy these goals through the use of an ontology for the representation of plans. However, the goal is not to define a complex ontology competing with existing work. Instead, the goal is to define a simple ontology that captures all of the features of an AF plan that are important for dynamic plan management and plan visualization. At the implementation level, the structure of a plan is to be specified in an XML document. (In the proposed software, plans are input as XML documents because XML is well suited for the representation of an ontology.) Situation updates will also be included in an XML format.

Schematically, an XML representation of a single plan might take the following form:

```
<plan>
  <task>          ...    </task>
  <objective>     ...    </objective>
  <constraint>    ...    </constraint>
</plan>
```

Note that this schematic representation is for illustrative purposes only, and it does not provide a representation of the actual XML syntax that will be employed. This representation explicitly specifies all of the components of a plan. Moreover, several plans can be represented in a single XML file. Each plan specifies all of the relevant tasks, objectives and constraints independently. It is understood that each plan occurs concurrently over a single timeline. Therefore, the XML input may include several plans, with additional global parameters.

There are a few key points worth emphasizing:

- The input includes tasks and objectives, subject to constraints. However, the tasks are unscheduled. A given task can take place at any point in time satisfying any time window constraints, as well as any task-sequencing constraints. Given the plan-input, the proposed mockup will try to schedule the tasks accordingly through a straightforward greedy algorithm. However, the scheduling of tasks may also be influenced by changes in the environment and the outcome of actions.
- In reality, the input need not be limited to a single XML document. Several different XML sources could be merged into a single input, providing a modular representation of different aspects, such as objectives and resources.
- We do not provide any formal approach to merging multiple plans. Plans are merged on a single timeline using a straightforward scheduler that respects the time constraints in each plan, but it does not attempt to consolidate tasks or objectives.

Situation updates can be used to define predictions related to a plan, as well as information obtained from sensors. Schematically, situation updates can be represented in XML in the following manner:

```
<actual time=0>
```

```
<platform id=1 affiliation="enemy">
  <trajectory>
    <point time=0 x=-128.4 y=50.0>
    <point time=1 x=-128.5 y=50.0>
    <point time=2 x=-128.6 y=50.1>
    ...
  </trajectory>
</platform>
<platform id=2 affiliation="neutral">
  ...
</platform>
</actual>
...
<actual time=1>
  <platform id=1 affiliation="enemy">
    <trajectory>
      <point time=1 x=-128.5 y=50.01>
      <point time=2 x=-128.6 y=50.11>
      ...
    </trajectory>
  </platform>
  ...
</actual>
```

Hence, at each point on our timeline, it is possible to specify entities that are present at some particular location. Using this basic approach, it is possible to specify a wide range of conditions. The idea is simply to provide an XML representation of all observable information available at each point in time.

The format of the situation update file, therefore, will include timestamps in two places. One time stamp (inside the <actual> tag) indicates the time that information is obtained; the second time stamp (inside the <point> tag) includes the time at which the information is predicted to hold. Situation updates for the current point in time can be understood to represent information about the state of the world. Situation updates for future points in time can be understood to represent predictions. It is necessary to provide both predictive information and actual information in the situation updates, because predictions will change in response to observations. (And the proposed mockup tool contains no functionality to update predictions of the environment behaviour based on actual observations, which is something an actual simulation tool would provide.)

In order to keep the plans separate from environmental information, it might be appropriate to provide two separate XML documents as input. However, this is not strictly necessary. It is clear that a single document could include the entire input. Determining the number of XML documents required as input is a low-level implementation consideration to be addressed at a later stage. In the next section, it will be convenient to distinguish between the “plan” component of the input and the “environment” component of the input. Whether these are provided as separate documents is not important, provided that the input is modular.

#### 5.2.4.2 Plan Forecasting / Projection

The software has the following goals with respect to plan forecasting:

- Explicitly represent predicted changes in the state of the world.
  - Predicted outcomes for the actions executed in a plan
  - Predicted actions for enemy/uncontrolled platforms.
- Explicitly represent the structure of a plan completely separately from these predicted changes.
- Allow projecting conflicts and dependencies between multiple plans.

As noted previously, the situation updates file contains predictions regarding the future state of the environment. These predictions can be used in conjunction with the plan description to project the outcome of a plan. Combining this input with the representation of a plan allows the user of the software to project the state of the environment to any future point of plan execution.

As an illustration, suppose that a particular scheduled action specifies a friendly platform to travel to location (-128.6,50.2) on a given map. Suppose further that a particular enemy unit is expected to remain at location (-128.4,50.0) for the entire duration of a plan. In this case, the situation updates file has the following form:

```
<actual time=0>
  <platform id=1 affiliation="enemy">
    <trajectory>
      <point time=0 x=-128.4 y=50.0>
      <point time=2 x=-128.4 y=50.0>
      ...
    </trajectory>
  </platform>
  <platform id=2 affiliation="neutral">
  ...
</platform>
</actual>
```

...

This input indicates that at time 0 the enemy platform is predicted to be at  $(-128.4, 50.0)$  until at least time step 2. The proposed software will determine (by consulting information about the range of the available weapons or similar means) that it is safe for the friendly platform to traverse through  $(-128.6, 50.2)$  without enemy interference, and thus the specified effects of this action is predicted to hold; so are all cumulative effects depending on these effects.

### 5.2.4.3 Plan Analysis / Evaluation

In terms of plan analysis and evaluation, the software has the following goals:

- Allow a user to pick between alternate plans for achieving a given objective.
- Allow a user to perform post-analysis to determine if a plan was effective or not, using general measures of performance that aggregate the outcomes of tasks.
- Analyse the interaction between multiple plans. Which plans depend on each other for meeting their objectives? Which plans compete for resources?

The first goal can be achieved by comparing different situation update files. As stated in Section 5.2.2.13, the situation updates at the actual point in time can be understood to simulate real-time input from sensors. We refer to this portion of the situation updates file as *the actual-input*. Altering the environment input provides a reasonable approach for the comparison of different plans. When choosing between two different plans, one would normally like to choose the plan that is more likely to achieve its goals. However, since changes in the environment are by nature unpredictable, it will be hard to predict whether a plan will achieve its goals. One way to compare two different plans is to prepare a set of environment scripts representing different ways that enemy units might behave. The plan that succeeds against the widest range of enemy behaviour might then be deemed superior. Note that the proposed software will not include a true event simulator; different enemy behaviour will need to be simulated manually by generating appropriate XML input.

The second goal can be achieved by setting targets that a plan is intended to achieve. For example, a target might be set for the amount of fuel to be consumed by a particular platform. This target values can be used to define a basic Measures of Performance (MOP). After setting the target value, the plan can then be executed and the actual fuel consumption can be recorded. By comparing the actual performance with the target performance, one is able to determine if the plan was effective or not with respect to that particular measure. Fuel consumption is not the only possible MOP; targets can also be set, for example, for the duration of tasks and the number of objectives achieved. Together the list of MOPs can be used to generate a *performance vector* consisting of several values. Negative values indicate poor performance in some aspect, and positive values indicate good performance. In order to evaluate the plan, it would be advantageous to provide a weighting scheme for combining these individual measures. However, as indicated in the literature review, it is not always possible to combine several different measures into a single quantitative plan evaluation. As such, we maintain the vector of MOPs as independent measures. The MOPs will be visualized as a bar graph centered at zero, with a bar for each measure. In order to avoid artificial scaling, the horizontal length of each bar is displayed as a percentage relative to the target. The value of each MOP is computed

over the entire simulation, so the performance can be evaluated at the end of plan execution or at a specific time point.

#### 5.2.4.4 Plan Monitoring

For plan monitoring, the main goals are the following:

- Monitor the actual state of the world at any point during plan execution, and determine if the objectives are likely to be achieved.
- Monitor the interaction between multiple plans.

It is easy to see how the state of the world can be determined at any point in time from the XML representation of a plan together with the actual-input. Also, the interaction between multiple plans can be handled by allowing several plans to be specified in the XML input. The plans all occur on the same timeline, competing for resources. The present version of the software will not be able to optimize resource use, nor will it prioritize between plans.

Using the actual-input, the task view and the map view can both be used to monitor the execution of a plan. Continuing our previous example (a scheduled action in which a friendly platform travels to location (-128.6,50.2)), suppose that at time point 1 the enemy unit is observed at (-128.5,50.1) and, based on this observation, is predicted to move to (-128.6,50.2) at time 2. The situation updates file then includes the following information:

```
<actual time=0>
  <platform id=1 affiliation="enemy">
    <trajectory>
      <point time=0 x=-128.4 y=50.0>
      <point time=2 x=-128.4 y=50.0>
      ...
    </trajectory>
  </platform>

  <platform id=2 affiliation="neutral">
  ...
</platform>
</actual>
...
<actual time=1>
  <platform id=1 affiliation="enemy">
    <trajectory>
      <point time=1 x=-128.5 y=50.1>
```

```
<point time=2 x=-128.6 y=50.2>  
...  
</trajectory>  
</platform>  
...  
</actual>
```

In this case, the actual situation at time 1 differs from the predicted situation. Using this input, the proposed software will show the enemy platform and the traversing friendly platform in the same location at time 2. On the assumption that the enemy platform is armed to neutralize the friendly platform, it will show the action as failing due to the high threat level. In other words, the action's effects will not be attained. This is the result of monitoring the actual execution of the plan.

Plan monitoring and plan forecasting are closely related. Plan monitoring covers the impact of environment changes on the current task, whereas plan projection describes how both the present and the predicted changes in the environment affect the preconditions for future tasks and plans to be scheduled and executed. In principle, the map view could provide two timelines: One would represent the actual execution of a plan, and the other would represent the predicted execution. The actual timeline would move forward at a constant rate, while the predicted timeline could be moved back and forth by the user. In this manner, the actual execution and the predicted execution could be overlaid on a single map. However, this approach will not be pursued in this version of the mock-up software because it introduces difficult problems for plan visualization that are not in the scope of this project, but could be addressed in future versions.

We include most of the dynamic plan candidates to be monitored. The elements that are directly affected by changes are:

- effects (= task outcomes)
- task execution time
- task resource consumption

To avoid information overload, however, only those elements that are caused by recent changes and that have a significant impact on plan execution will be displayed:

- unsatisfied constraints that cause tasks to fail
- unavailable resources
- actions and tasks affected
- goals that are no longer attainable
- scheduled actions and tasks that become useless because the original goal cannot be satisfied (e.g. transfer to a rendezvous point if one of the participants fails to arrive at the scheduled time)

The list does not include MOPs, which are also dynamic. But we only consider these for the purpose of plan evaluation.

## 5.2.5 Plan Visualization

The software is to provide visualization tools for displaying information about plan execution in a comprehensive format. The main criteria for effective plan visualization are:

- Provide incremental level of detail
- Track plan history over time, and project plan outcomes into the future
- Visualize dependencies between several plans
- Alert the user to critical changes and their impact on plans
- Allow comprehensive evaluation of plans along multiple criteria.

In this section we highlight three important visualization modes that will be implemented: *plan view*, *map view*, and *MOP view*. Each view has a time slider to study (analyze, monitor, project) plans over time. A fourth visualization mode will be available for debugging purposes: it simply displays all the raw data that is available.

### 5.2.5.1 Plan view

This will present a plan graphically as a network of tasks, similar to a Hierarchical Task Network. The edges between tasks will indicate sequencing restrictions. In addition to sequencing restrictions, the tasks will be placed on a timeline in a Gantt-like format. The current point in time will be clearly marked on the timeline. Delays in a plan will be depicted on the same timeline. Such delays may occur, for example, if a certain resource is not available to a particular task.

The following plan elements will be represented in lines (bars) parallel to the time line:

- tasks: bar with different colours corresponding to task status
- platforms: bar with different colours corresponding to platform status
- consumable resources: graph showing the amount of resources left
- effects/goals: solid bar from the point at which the effect/goal is achieved
- time window constraints: shown on the corresponding task or platform bar
- precedence constraints: shown as diagonal lines between tasks

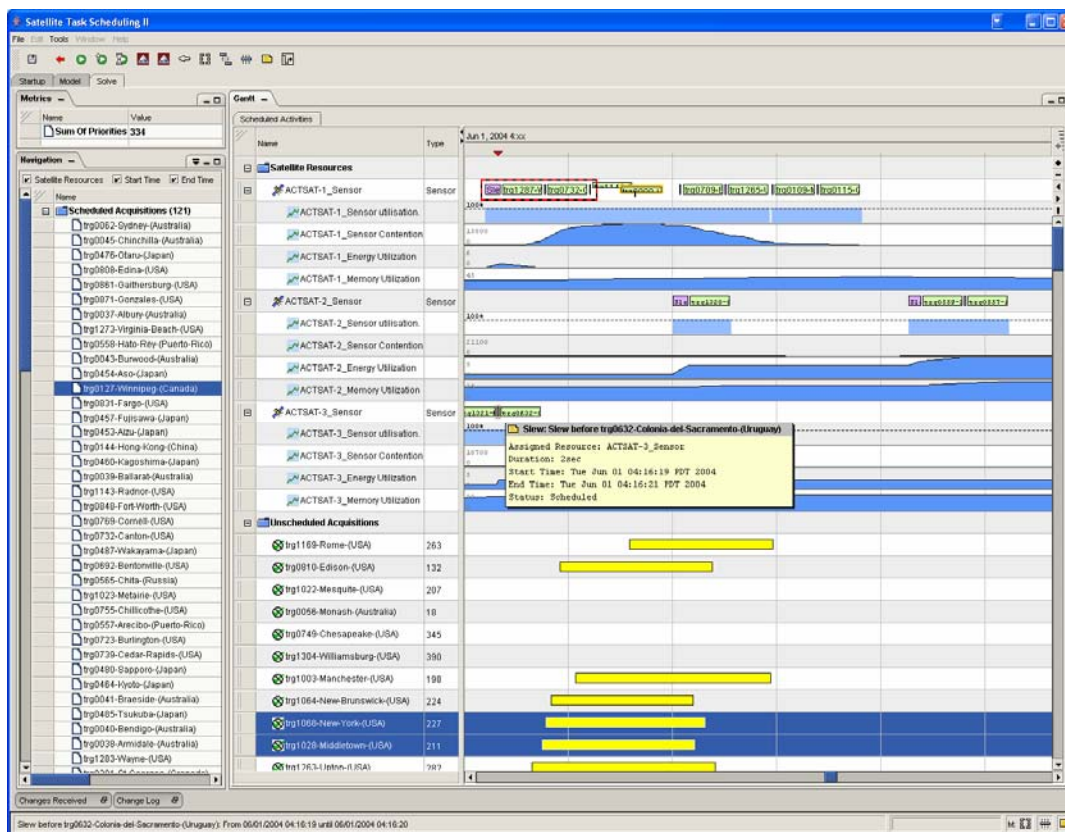
The bar to the left of the current time will display past *actual* plan execution data, whereas the bar to the right of the current time will represent the *projected* plan execution data.

The following display and navigation functionalities will aid in analyzing plan dependencies, causes of plan failure, and reduction of uninteresting or irrelevant information. For most of these, a task must be selected first:



- Select maximum level of plan detail  
1 = plans, 2 = plans and their subplans, etc. Only tasks up to that level are shown
- Show (only) critical plans/tasks/resources  
Critical resources are those below a specified minimum level; critical tasks are failed tasks.
- Show all tasks competing with the selected task  
Task B competes with A if both use a resource of the same type.
- Show all tasks blocking/blocked by selected task  
Task B blocks A if A is blocked but could execute if B was stopped.
- Show all tasks depending on selected task  
These are tasks that have a time constraint with, or depend on effects set by, the selected task
- Show all tasks the selected task depends on.  
If any of these tasks fails or delays, so does the selected task.
- Show all elements blocking a task
- Show all elements the selected task depends on
- Show all goals the task contributes to

The plan view shown in Figure 5-5 has been taken from the MICROS satellite scheduling system, which was a previous project at MDA that has some parallels to the current project. In the top half, it shows resource allocations and consumptions, and in the bottom half (yellow bars), it shows the scheduled tasks. Note that the MICROS project does not define hierarchies of tasks, and the figure does not show precedence or time constraints of tasks.

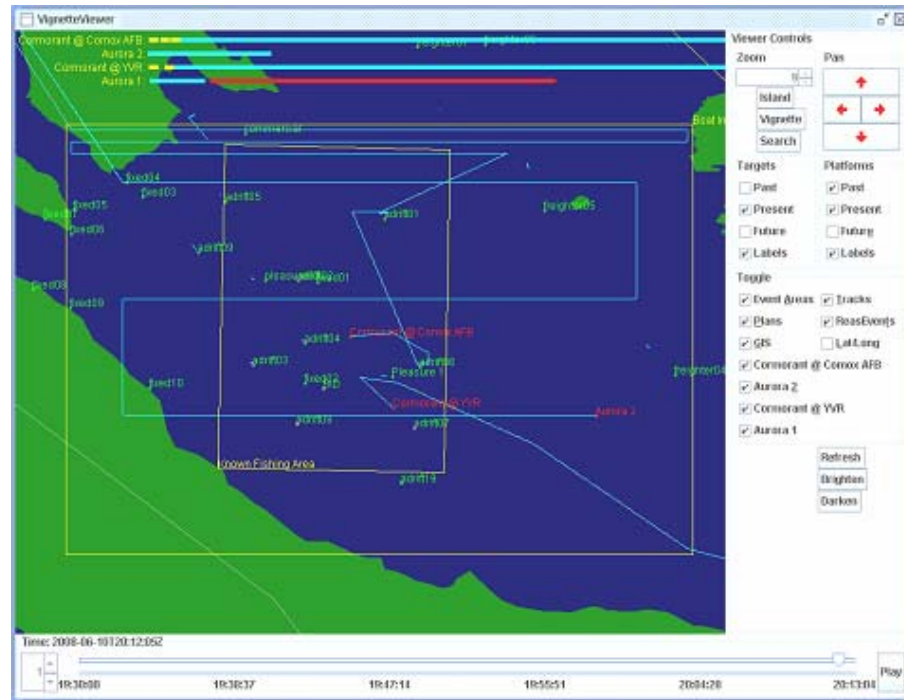


**Figure 5-5 Plan View**

### 5.2.5.2 Map view:

Shows a map with all geospatial information (platform locations, target locations, flight paths). The user can step through time using a slider bar (time line) located along the bottom of the view window. Sliding the bar forward will project the outcome of the plan with respect to the predicted changes in the state of the world. For post-analysis, the actual changes in the state of the world can be displayed. Clicking on an entity provides all information from the internal representation in a floating text box. The map view tries to mirror pen and paper representations of plans.

The map view shown in Figure 5-6 has been taken from the CanCoastWatch (CCW) ISR simulation and visualization project done for DRDC-Valcartier. This is another project with useful parallels to the current project; it is expected that the existing visualization components in CCW will be reusable to expedite development of the mock-up.



**Figure 5-6 Map View**

The following plan elements will be represented in this view:

- Platforms: as icons
- Tasks: as trajectories or points
- Significant locations: as yellow diamonds or area boundaries
- Location constraints: by showing enemy tracks indicating “unsafe” flying areas
- Attached resources: by showing the labels of attached resources underneath the label of the resource they are attached to, with an upward or downward arrow indicating which resource is attached to which
- Goals: by representing the desired positions of all units at the conclusion of a plan’s execution
- plan updates and anticipated completion time of plans: using red markers on the time line.

The map view makes extensive use of colours, shading and icons to distinguish:

- past and future tasks (histories)
- tasks that succeed and those that fail to achieve their goals/effects
- tasks that are in progress and those that fail to schedule/execute
- unscheduled tasks.

We will incorporate plan abstraction in two ways: First, at a high level view several units may be combined in to a single dot; upon zooming into the dot, the details become explicit. Second, some menus will be offered for providing more abstract views of time or views for specific units. It may be possible to implement the map view by using visualization tools built for CCW. We will adopt this approach if it is deemed to be feasible and appropriate for our needs.

Using the map view, it will be possible to monitor how close a set of plans comes to achieving all of the relevant goals.

### **5.2.5.3 MOP view**

This is a specific view to track and compare different measures of performance, including aggregated goal attainment, duration, and consumption of resources. The view will allow plan evaluation, as described in Section 5.2.4.3. The MOPs will be visualized as bar graphs centered at zero, with a bar for each measure. This view can be brought up for the current time, or projected to the end of the plan.

### **5.2.5.4 Data view**

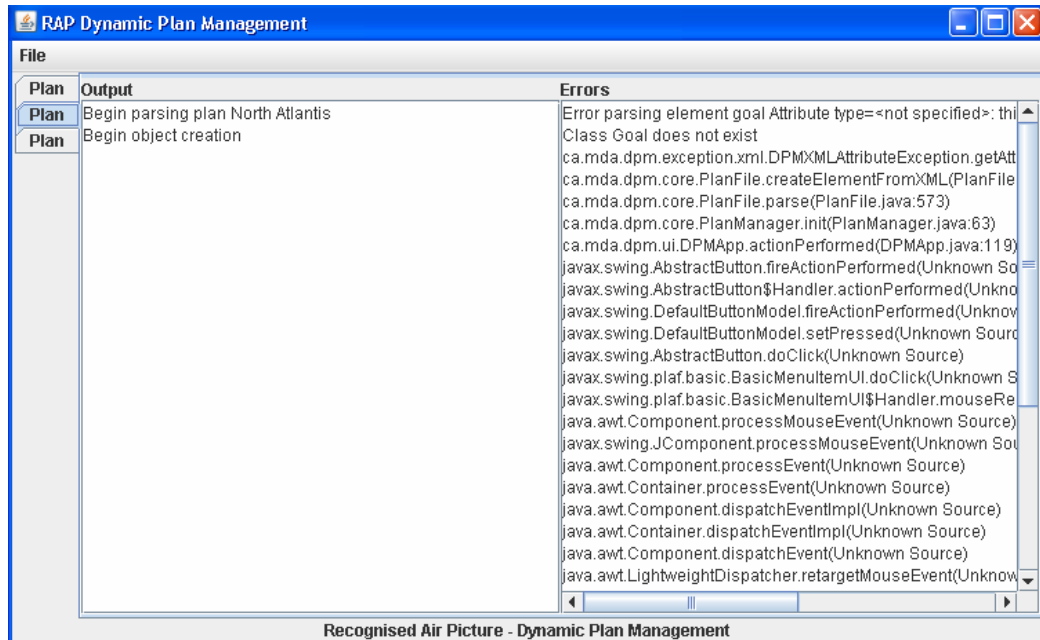
The data view is a simple browser-type viewer that displays all of the raw data available. This view is not intended to provide a sophisticated visualization tool for military planning. It is primarily included for debugging, system evaluation and for users who need the ability to track data in a completely unrestricted manner.

## **5.3 Implementation Details**

In this section, we provide a detailed description of the proposed software. In short, the system operates as follows:

1. Plans are represented as an ontology (plan representation).
2. The system reads plan data (plans, objectives, constraints) as an XML file.
3. The system checks plans for consistency and correctness (plan analysis).
4. The user can view plan execution over a map (plan monitoring) and project time forward to see future outcomes (plan projection).
5. The user can interact with a plan execution by loading plan updates and situation updates.
6. The user can re-schedule the tasks to repair failed plans.

At a practical level, when the system is launched from the command line, the main interface is displayed. The user is able to open a plan configuration file through a standard file dialog. The plan configuration file is an XML file to be described below. If the file contains an invalid plan description, then the user is given an error message (Figure 5-7). If the file contains a valid plan description, then the user is informed that the plan is valid through a short message.



**Figure 5-7 An Invalid Plan**

After loading a valid plan file, the user may click on map view tab. Switching to the map view, the user will see a map of the area where the defined plan is to be executed. The user can zoom in, move the displayed area, use a slider bar to move through time, and view the trajectories flown by all platforms during the execution of the plan. A more detailed description of the map view component will be provided in Section 5.3.3.

To facilitate plan representation, monitoring, forecasting and evaluation, completely scheduled plan executions are furnished as input to the system. Hence, the representation of plans includes a sequence of scheduled actions to be executed. An INSPECT-style checker then analyzes the plan input for constraint violations, doubly allocated resources and other contradictions. This answers the question: "Is this plan consistent?"

The initial input is primarily concerned with the representation of plans. Additionally, the system can be given a situation update file containing information about the environment over the course of the all plans executed. (This is because automated prediction based on current observation is too complex for the scope of this project.) The plan can be modified through a third kind of input, called a plan update.

### 5.3.1 Plan Ontology

The first step in the development of the software was to develop an expressive planning ontology. Our ontology is loosely based on SPAR, incorporating the plan elements from Section 5.2.1. We have defined our ontology of plan elements in a Document Type Definition (DTD) file. Each element of the ontology is explicitly specified in the DTD file, along with all of the components. A specific plan or set of plans can then be specified as XML documents including all of the components specified in the DTD file. This basic setup provides a

lightweight representation of plans, which allows plans to be analysed through text processing on the underlying XML representation.

The top-level element in our ontology is the PlanManager, which is an abstract element that wraps several interacting plans. A PlanManager element may include several components, such as constraints, locations or platforms. The key plan elements required for representing the plans in the Atlantis vignette are listed in Table 5-6.

**Table 5-6 Key Plan Elements in the Ontology**

Plan Element	(Optional) Sub-Elements
PlanManager	Constraint, Goal, Location, Plan, Platform, Schedule, AllocatableResource, VignetteBounds
Plan	Goal, Constraint, Task
Goal	Effect
Task	Action, Constraint, Task
Action	EffectAttainment, ResourceAmount, Trajectory
Schedule	ScheduledAction

In addition to specifying components of each plan element, the DTD file also specifies the attributes of each element. For example, a ScheduledAction consists of a “start time” attribute, together with an Action. The complete DTD file used for the North Atlantis vignette is included in Appendix A.

One feature of the ontology that is worth noting is that Effects could not be modelled as constraints. Instead, a new subclass of Constraint called Precondition was defined. An Effect object can be wrapped into a Precondition constraint indicating an effect that must hold at the time the task is scheduled to start. Different kinds of constraints are used for several different purposes. For example, a new class of AllenXXXConstraints have been introduced as special cases of PrecedenceConstraint, where the precedence between tasks is dictated by Allen relations between intervals.

Note that the software does not provide a fixed list of plan elements. The plan elements are specified in the DTD file, so it is easy to extend the ontology by new types of elements without modifying the existing software. For this purpose, an *ontology manager* is provided, which lets one flexibly add new types of plan elements. We describe its functionality using the following example:

Assume a new type of aerial platform, say, a Heron UAV, has been purchased. We would like to use it in a plan specification as follows:

```
<allocatableResource name="UAV squad">  
    <platform name="bigbird" type="Heron UAV"/>  
</allocatableResource>
```

In this example, `platform` denotes the element category and `Heron UAV` denotes the specific element type; `allocatableResource` denotes the category of the parent element. To create and register the new type of element, the user must perform the following two or three steps:

1. Write a Java class `Heron_UAV.java` describing specific properties of the UAV. This class must be a direct or indirect subclass of the `Platform` class. (If an existing, similar platform type is subclassed, the Heron UAV can inherit properties from that platform type, which will minimize development effort and programming skills needed.)
2. Compile the class, and place the new class `Heron_UAV.class` inside the software directory.
3. If the element requires specific validations to determine consistency, the user can optionally override the `validate()` method in the `Heron_UAV` class.

When the software starts up, the ontology manager creates a class database, consisting of all the classes inside the software directory. (These include in particular all existing and user-defined classes belonging to the ontology of plan elements.) Furthermore, the ontology manager parses the elements described in the `plan.dtd` file.

When a plan is loaded and the above XML element is parsed in, the ontology manager performs the following checks to determine syntactic correctness:

1. It verifies whether the element category `platform` is defined in the `plan.dtd` file, and whether it is a valid child element of an `allocatableResource` parent element.
2. It verifies whether the specified element type `Heron_UAV` exists in the class database, and whether it is indeed a subclass of the `Platform` class.
3. If any of these verifications fail, the ontology manager rejects the plan file and points to the specific XML element that violated the plan structure. Otherwise, it reads any attributes that the `Heron_UAV` element might specify, creates an internal representation of this plan element, and attaches it as a subelement of the allocatable resource (here: a squadron) it belongs to.

Note that the ontology manager automatically capitalizes class names and inserts underscore characters into the XML tags, which allows a standards-compliant and more user-friendly representation of plan elements in the XML file, while ensuring that Java's class naming conventions are not violated.

Note also that it is not necessary to include the new element type in the `plan.dtd` file. This is necessary only if the user defines a new *category* of elements.

### 5.3.2 Validation

One of the main tasks performed by the mock-up software is plan validation. When a plan XML file is loaded, the software performs four steps.

1. Parse the file and create all of the java objects, as specified in Section 5.3.5.
2. Populate the objects with all required attributes and properties.
3. Schedule the actions according to the Schedule tag.
4. Check the plan for internal consistency. In particular, the validation procedures checks for the following:
  - Reference to an object that is not defined.
  - Reference to a location or time that is not in the bounds of the software.
  - Constraint mentioning a task that is not part of the given plan.
  - One resource specified at two different places at the same time.
  - Two resources specified at the same place at the same time.

These problems are all issues that can be checked by simply looking at the specification of a single plan. The software also checks for any objects that have been defined in the XML file without specifying all required attributes.

In addition to checking for problems with the structure and XML encoding of a plan, the software also checks if all constraints in a plan have been verified and the software performs some rudimentary evaluation of a plan with respect to enemy tracks. In particular, the plan will fail if one of the resources in a plan is going to encounter an enemy presence that will prevent the execution of a task.

Every time that a new plan update or a new situation update is loaded, the plan is re-validated.

### 5.3.3 Visualization

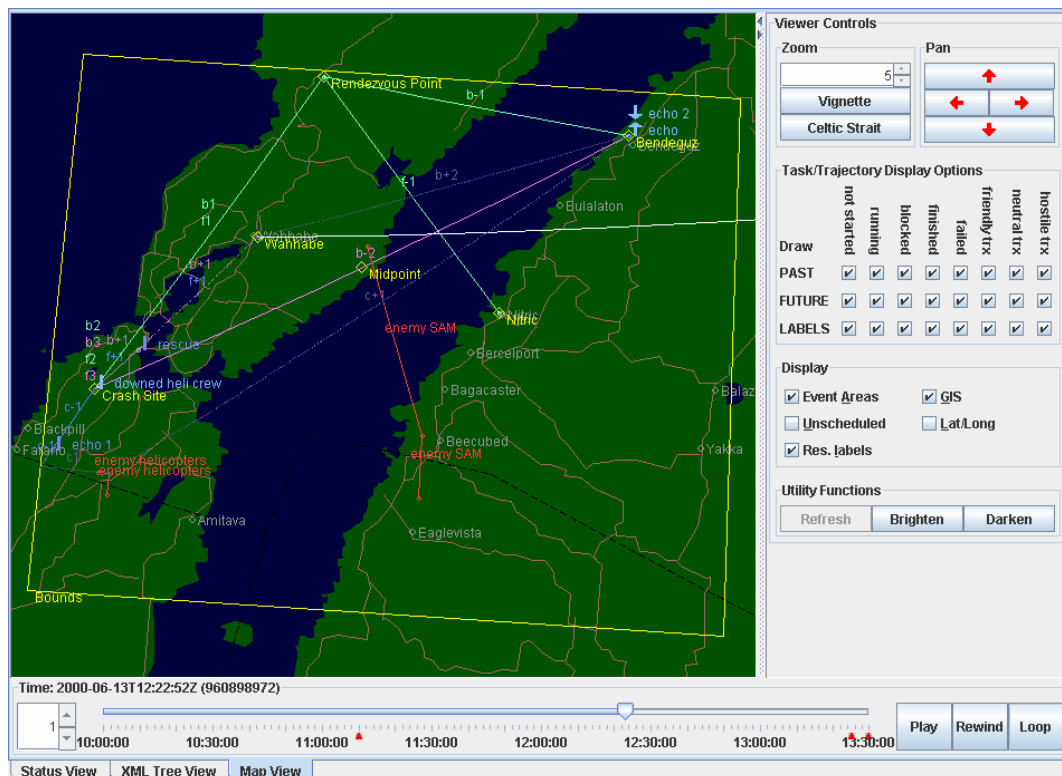
The mock-up software offers two of the visualization modes that were suggested in Section 5.2. In particular, the software provides a Map View and a Data View. The Map View illustrates the locations of all platforms over a pre-defined map. The viewer includes several significant features for plan forecasting and plan monitoring:

- A slider bar for visualizing different points in time. By sliding the bar into the future, it is possible to forecast the outcome of all plans. Red markers on the slider bar show significant events, e.g. the time point of plan or situation updates, and the time at which each plan finishes successfully.
- Visual depictions of the past trajectories of all platforms. These can be toggled on or off.
- Visual depictions of the future trajectories of all platforms, according to the input plans. These can also be toggled on or off.



- Labels for all actions and the resources executing them. These can also be toggled on or off.
- By dragging the mouse, the area in the display can be moved.
- By using the mouse scroll wheel, the user may zoom in or out in the map.

Figure 5-8 illustrates a map view showing a representative extract from the North Atlantis Vignette. The objective of this vignette is to rescue a stranded helicopter crew. This vignette has been completely defined in the XML format of the mock-up software. Moreover, the software is bundled with a GIS map of the North Atlantis theatre of operations.



**Figure 5-8 Map View of the North Atlantis Vignette**

In addition to the Map View, the software also incorporates a Data View mode that allows the user to see the XML representation of the plans under consideration. Some knowledge of markup languages is useful for viewing and understanding this view. However, the simple XML format used in the software is essentially human-readable.

### 5.3.4 Object-oriented representation of plan elements

As noted previously, a set of plans is represented as an XML file that contains only the elements and attributes specified in the DTD file. At present, these XML files must be authored by hand. However, in the future it would be possible to create a plan editor that would essentially function as an interface to the XML representation. Given an XML file representing a set of

plans, a parser will import the corresponding plan and generate an appropriate hierarchy of objects for the software to manipulate and process. This data structure forms the basis for analysis, forecasting, monitoring and visualization. In this section, we outline the object model our software employs for the representation of plans.

To illustrate the objectives of this research without being absorbed by too much detail, the domain is simple: The resources are aerial platforms consuming fuel, staffed with personnel and equipped with sensing capabilities. The only activities considered will be equipping planes with fuel, flying them to particular locations and performing search tasks at these locations. The objective is to find a given object, such as a stranded soldier, with a certain minimum probability. For each category of plan elements, we propose a number of types (classes) with attributes and selected methods as shown below.

In the following sections, we give a preliminary sketch of the classes used in our mock-up software. The classes are listed in tables of the following form.

**Table 5-7 Object Description Schema**

Plan Element	Attributes/Methods	Comments
ClassName: Superclass	parameter_1: Type_1	A text explanation of this parameter
	...	
	parameter_n: Type_n	A text explanation of this parameter
	function_1(): Return Type_1	A text explanation of this function
	...	
	function_m(): Return Type_m	A text explanation of this function

Arrays of a particular type are indicated by using the Java-like syntax *Type[]*. Also, enumerated types are specified using the syntax *enum Type*, followed by a sequence of values *Value1*, ..., *ValueN*. This syntax indicates that *Type* can take any of the given values, treated as atomic symbols.

#### 5.3.4.1 Basic element

The DPMElement class defines two fields common to all plan elements:

**Table 5-8 DPM Element Object**

Plan Element	Attributes/Methods	Comments
DPMElement	name: String	Unique name of this object
	description: String	A textual description

In the visualization (all views), the name is used as a label identifying this object. The description is shown when the user's mouse hovers over the object.

### 5.3.4.2 Location and time

Coordinates are defined relative to some defined coordinate system, e.g. UTM. Locations are specified using lists of coordinates, denoting the vertices of a bounding polygon. This allows one to specify approximate locations or the location of a group of resources.

Time points are represented by an integer, relative to a defined reference date, e.g. Julian Date. Unlike time points, durations mark an amount of elapsed time, measured e.g. in seconds. An unscheduled task has a duration, for instance. Intervals, on the other hand, are fixed in time, so their start and end time are time points. They are used in TimeWindow constraints.

**Table 5-9 Location and Time Objects**

Plan Element	Attributes/Methods	Comments
Coordinate	x,y,z: double	UTM Coordinates (as in CCW)
Location: DPMElement	coordinates: Coordinate[]	coordinates of the bounding area
TimePoint	value:int	Need not be a separate object
Duration	value:int	
Interval	start: TimePoint duration: Duration getEnd(): TimePoint	start time length of the interval computed as start time + duration

### 5.3.4.3 Resources

We distinguish between consumable resources (e.g. fuel) and allocatable resources (e.g. personnel, platforms). The fundamental difference between these two classes of resources is that each AllocatableResource object is individually distinguishable. By contrast, consumable resources must be specified in quantities. Both allocatable resources and consumable resources can be *attached* to other resources. Attachment corresponds to any of the following physical activities:

- fueling an airplane
- staffing an airplane with personnel
- grouping platforms together into a military unit (squadron, platoon etc.)

The last point allows grouping resources at incremental levels of detail. Allocatable resources at a higher level are groups of allocatable resources at lower levels. In principle, there is no limit to the number of levels or to the size of groups.

A task usually requires that the executing AllocatableResource has resources of various types attached to it; these are specified by the task's resource constraints. Attaching resources to an AllocatableResource is usually a separate action, but it can be implicitly defined as part of a task. (This is so that the plan need not contain separate actions for cumbersome details such as personnel boarding a plane.) Likewise, after the task is finished, resources can be disattached

and become available again. Consumable resources are different, however, in that they are used up by a task and will no longer be available.

A type hierarchy of allocatable resources can be defined by subclassing the AllocatableResource class. For illustrative purposes, our mock-up distinguishes two subclasses: Personnel and Platform. Resource constraints can then refer to a number of resources of each specific type by referring to the class name. Note that it would be straightforward to represent different kinds of platforms by creating subclasses of AllocatableResource. In this manner, a particular type of platform can be defined with specific attributes such as speed, fuel capacity and capabilities. The present mock-up does not define a complex type hierarchy of allocatable resources, but the system is flexible enough to incorporate additional kinds in a straightforward manner.

**Table 5-10 Resource Objects**

Plan Element	Attributes/Methods	Comments
Resource: DPMElement	attachTo(AllocatableResource parent)	if parent=null: resource becomes "independent"
ConsumableResource: Resource	quantity: double	the amount of resource
Fuel: ConsumableResource		marker class for fuel
AllocatableResource: Resource	attachedResources: Resource[] current_location: Location getLocation(): Location getStatus(TimePoint): AllocatableResourceStatus	either current_location, or the parent's location  Get AllocatableResource status at given time (see below)
Personnel: AllocatableResource		marker class for personnel
Platform: AllocatableResource		marker class for platforms

#### 5.3.4.4 Constraints

Constraints are properties (Boolean predicates) that must hold for a task or plan to be executable. There are many different types of constraints related to tasks, resources, situation information and environmental conditions. All prerequisites for a task can be specified as constraints. But one can also specify global constraints that must hold for all tasks. For the mock-up, we employ the following types of constraints:

**Table 5-11 Constraint Objects**

Plan Element	Attributes/Methods	Comments
Constraint: DPMElement	holds(): boolean	Is constraint satisfied?
ConsumableResourceConstraint: Constraint	resourceType: Class quantity: double	the type of resource (e.g. Fuel) the amount of resource needed and consumed
AllocatableResourceConstraint: Constraint	resourceType: Class quantity: int	the type of resource (e.g. AllocatableResource, Platform) the number of individual resources needed
TimeWindowConstraint: Constraint	task: Task interval: Interval	Task must start and end within the given interval.
PrecedenceConstraint: Constraint	predecessor: Task successor: Task lagTime: Duration	First task Second Task Number of secs between tasks
ResourceAvailabilityConstraint: Constraint	resource: Resource interval: Interval	This resource (e.g. personnel) is only available between the two given time points.

### 5.3.4.5 Effects and goals

Effects characterize the direct or indirect outcomes of tasks. We distinguish:

- direct effects, which are caused by the execution of some atomic action. Among these, we have SimpleEffects, which are explicitly set by tasks (example: BridgeDestroyed), and effects that are computed from the parameters of the action (e.g. the action of flying plane P from A to B.)
- CumulativeEffects, which are computed from simpler effects. In the deterministic case, a cumulative effect holds if all its sub-effects hold..

Our model obeys an important principle of causality, namely that each simple effect is caused by a single atomic action. Even so, we can compute and refer to a well-defined set of (simple, cumulative, and perhaps indirect) effects caused by the execution of a composite task; namely, the set of all effects caused, directly or indirectly, by any action executed as part of the task. Effects that seem to violate the causality principle, such as destruction by simultaneous fire from multiple platforms, can be modelled as composite effects resulting from the damage (simple effect) caused by each individual platform.

Effects can also be prerequisites for another task. For this reason, they are modelled as subclasses of constraints. The holds() predicate for effects specifies the outcome under which the constraint is considered satisfied.

A Goal is used to describe the desired outcome of a plan. Goals are distinguished from other cumulative effects only in that a goal is the highest-level effect in the composition and causation hierarchy.

**Table 5-12 Effect Objects**

Plan Element	Attributes/Methods	Comments
Effect: Constraint	holds(TimePoint): boolean	Effects can be specified as constraints for tasks. abstract; defined by subclasses
SimpleEffect: Effect	duration: Duration holds(TimePoint): boolean	the time for which the effect holds true at the end of an action that succeeds, for the specified duration.
AllocatableResourceLocationEffect: Effect	resource: AllocatableResource location: Location holds(TimePoint): boolean	true if the given AllocatableResource is at the given location
ResourceAllocationEffect: Effect	source: AllocatableResource resource: Resource holds(TimePoint): boolean	true if the given AllocatableResource has the given (amount of ) resource attached (e.g. after refuelling)
AreaSurveillanceEffect: Effect	location: Location holds(TimePoint): boolean	true if the given location has been observed for the given amount of time.
CumulativeEffect: Effect	subeffects: Effect[] holds(TimePoint): boolean	true if all subeffects are true
Goal: CumulativeEffect		the desired effect (outcome) of a plan

#### 5.3.4.6 Actions, tasks and plans

An *action* is an elementary activity that is *executed* by a single platform (AllocatableResource) and has certain *effects* on the environment, as well as the resource itself (change in location, resource consumption, amount of time spent). Conversely, the effects of an action may be conditioned by the state of the environment at the time the action is executed: originally estimated by a degree of probability, they will hold after execution of the action with degree 1.0 (success) or 0.0 (failure). Actions have a defined start and end location as well as an explicitly defined duration (expected time to complete the task). (The duration of an action could sometimes be computed from the speed and other characteristics of the platform executing it, but we did not build this sophistication into the mock-up.) We make the simplifying assumption

that actions cannot be interrupted. (Otherwise, we would need to define partial resource consumption and effects, as well as intermediate locations.)

A *task* is a collection of actions subject to constraints. The constraints can give structure to the actions in terms of time or priority. Every subset of the component actions can be understood to represent a *subtask*. The actions are then the lowest-level subtasks. Before a task can be executed, its subtasks must be scheduled so that all constraints are satisfied. In the mock-up, a *ScheduledAction* object is created for each subtask scheduled for execution; this object contains the start time, the allocated resources, and the task status at any given time.

The execution of a task may fail for various reasons: First, just as for actions, the desired effect of the task may not be obtained. Secondly, a subtask may fail, and another subtask depends on the successful execution of this task. Thirdly, due to deviations (e.g. delays) in the execution of earlier subtasks or due to environment changes, constraints that were originally satisfied become violated, and a later subtask cannot execute. When it becomes evident that a task is going to fail, it makes sense to interrupt it and free its allocated resources, rather than continuing to execute useless subtasks; the mock-up task scheduler will allow this. Interrupting a task interrupts all its subtasks. All actions in progress are completed, but no new actions are scheduled or executed.

A *plan* consists of a collection of tasks, a collection of goals, and a collection of constraints. The constraints are global constraints that apply across all component tasks, which might themselves be subject to constraints. Hence, the main distinguishing feature between tasks and plans in this setup is that plans have explicitly specified goals. A plan is executed by executing the component tasks, in accordance with the component constraints. A completed plan's attainment of these goals can be evaluated, or an ongoing plan's present progress towards them can be monitored or projected into the future.

**Table 5-13 Action, Task and Plan Objects**

Plan Element	Attributes/Methods	Comments
Task: DPMElement	subtasks: Task[] constraints: Constraint[] get_duration(): Duration getStatus (TimePoint): TaskStatus getEndTime (): TimePoint	computed from duration of actions task status at the given time time at which task is expected to complete
Action: DPMElement	startLocation: Location endLocation: Location resourceConsumption: ConsumableResource[] duration: Duration effects: Effect[]	The Effects produced by this action

Plan Element	Attributes/Methods	Comments
ScheduledAction	action: Action startTime: TimePoint allocatedResources: AllocatableResource getEndTime (): TimePoint	the action scheduled start time resource (usu. platform) allocated to this task expected time of completion
Plan: DPMElement	tasks: Task[] goals: Goal[] constraints: Constraint[]	Plans consist of a set of tasks, a set of goals, and a set of constraints

#### 5.3.4.7 Plan Manager

As noted previously, the top-level element is the PlanManager. It defines an overall time horizon, a set of (multiple) plans, a set of global (plan-independent) constraints, and possibly a set of overall goals the plans are to accomplish, and a set of performance measures against which the plans are evaluated.

**Table 5-14 Plan Manager Objects**

Plan Element	Attributes/Methods	Comments
PlanManager: DPMElement	timeHorizon: Interval plans: Plan[] constraints: Constraint[] goals: Goal[] mops: MOP[]	see description above

#### 5.3.4.8 Resource and Task Status

These are enumerations describing the status of resources and tasks at any given time:

**Table 5-15 Status Objects**

Plan Element	Attributes/Methods	Comments
enum TaskStatus	notStarted waiting running stopped finished	the plan containing this task has not started not all resource constraints satisfied task has started but not completed yet task is interrupted



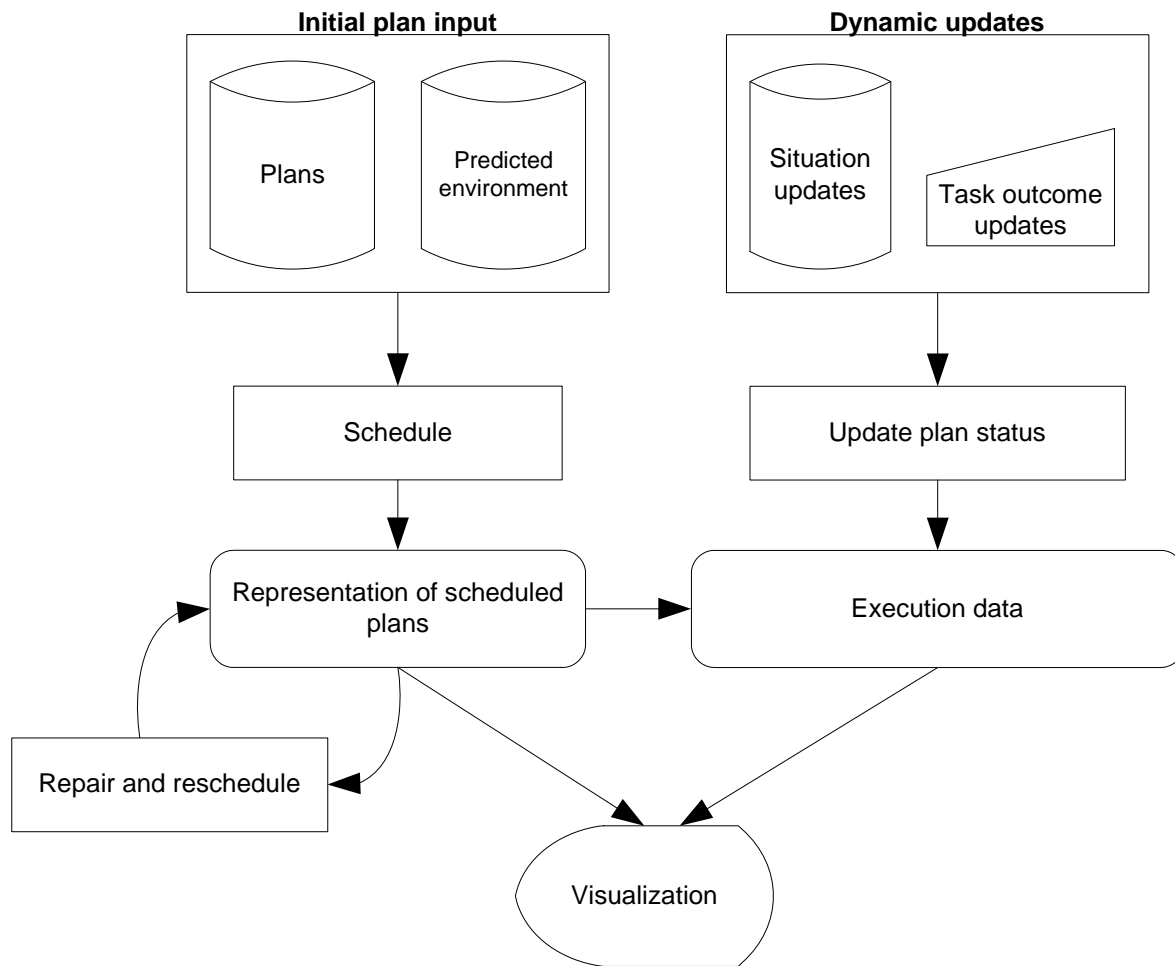
Plan Element	Attributes/Methods	Comments
	cancelled failed	task has completed successfully task (instance) has been permanently cancelled task cannot be completed with given resources, time etc., or failed to produce desired effect(s)
enum AllocatableResourceStatus	available allocated unavailable neutralized	AllocatableResource currently not scheduled AllocatableResource currently scheduled to a task AllocatableResource currently unavailable (outside its shift etc.) AllocatableResource permanently unavailable (destroyed).

### 5.3.5 Modelling Execution Data

There are two initial inputs required to run the proposed software. The first input is the DTD file and the second is an XML file representing a set of plans. Together, we refer to this initial input as the *plan-input*. The software also needs to generate and maintain execution data that captures the execution of the plans, subject to the changes in environment given in the situation updates. Among other things, the execution data will be used as input to the various visualization tools. Broadly speaking, execution data at any point in time consists of the following:

- The status of all tasks at that time
- The positions, status and available resources for all platforms at that time
- The degree to which effects and constraints are satisfied.

Thus execution data, scheduled plans and situation updates together provide a precise specification of the state of the world at every point in time. This information is provided in a format that is appropriate for visualization and also for formal analysis. An idealized picture of the process for generating and updating execution data is pictured in Figure 5-9.



**Figure 5-9 Generating Execution Data**

In our software, this basic framework is realized with some simplifying restrictions. In particular, we have not included a scheduler for actions. Instead, the schedule of actions is included as part of the plan input. If the schedule of actions provided in a plan schedules two actions at the same time, then the software will indicate that the plan cannot be carried out.

Note that Figure 5-9 requires both situation updates and plan updates in order to provide a realistic model of plan execution. In the remainder of this section, we describe our treatment of each of these topics.

### 5.3.5.1 Situation Updates

Over the course of the plan, however, the predicted outcome of tasks should be updated with their actual outcome, as the tasks complete. There are two possible ways to provide task outcomes. First, the outcomes could be randomly determined by the system, respecting appropriate probabilities. A more desirable solution, however, is to prompt the user to specify task outcomes at the end of each task through a dialog. In this manner the user is able to play different what-if scenarios.

The software allows the user to dynamically generate and load new situation updates at any point in time. In this manner, real-time observations can be simulated and input into the system. Dynamic situation updates are represented as XML documents containing new conditions at a given point in time. Task outcomes and situation updates in turn influence the current and future execution data. In particular, some tasks originally scheduled may not execute because constraints are violated. As a consequence, plans may no longer succeed.

Every situation update contains information about one or more tracks. A track is understood to represent the trajectory followed by an aerial platform. The trajectories are specified by giving a duration, along with coordinates for the start and end points. These documents are selected as input from the main interface, and then the XML is parsed to generate appropriate objects. Each time a new plan update or a new situation update is opened, the display window opens a new tab displaying appropriate text output regarding the parsed information.

The plan execution could also affect the situation updates (e.g. a platform moves to a location and makes an observation; an enemy target is shot down). It is entirely up to the user to represent these changes in the XML input for the next situation update. Trying to compute changes to situation updates automatically would add another dimension to the problem and is outside of the scope of this project.

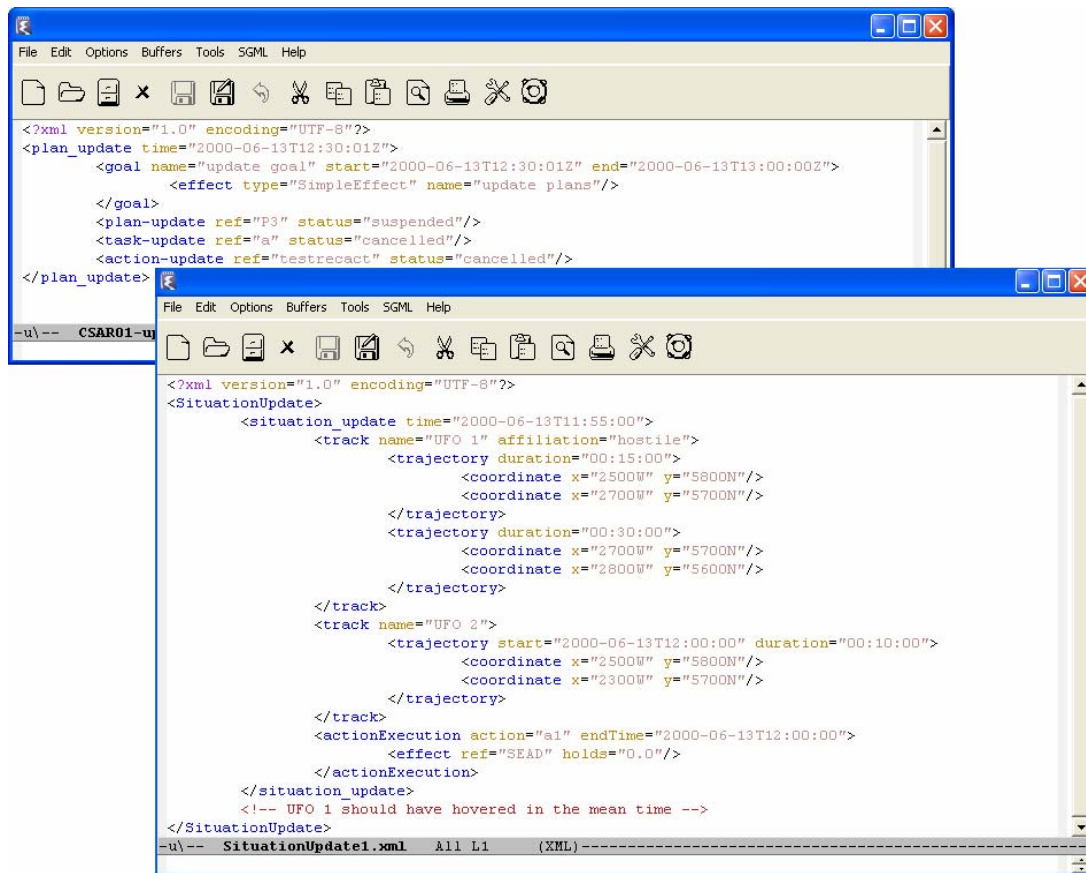
### 5.3.5.2 Plan Updates

The initial input to the software provides one static, pre-computed set of scheduled plans, and simply update the impact of changes and unexpected task outcomes on the execution data. If this was the only possible input, the user would be limited in several respects:

- The user would be unable to execute variations of the original plan.
- Even very minor changes in task outcomes may cause plans to fail.
- Even if a plan cannot succeed because one of its tasks cannot be executed, its other tasks will still be executed and use up resources.

In order to address these problems, we allow the user to load plan updates. A plan update specifies that certain tasks or plans should be cancelled or suspended. However, certain changes are not permitted, mostly related to attributes of plan elements that relate to the page. For example, it is not possible to modify resource amounts at earlier points in time.

Internally, plan updates and situation updates are both represented by XML files of the format depicted in Figure 5-10.



**Figure 5-10 XML Representation of Plan Updates and Situation Updates**

## 5.4 Assessment

In this section, we present an assessment of the mock-up software. The initial goal was to develop software addressing plan representation, plan forecasting/projection, plan analysis/evaluation and plan monitoring. Moreover, a complete treatment of the problem was expected to involve visualization in each dimension. However, it became clear during the implementation phase that a complete solution involving visualization was beyond the time scope allowed by the present project. As a result, the mock-up software has focused primarily on plan representation with one main visualization mode, as well as automated analysis of plan descriptions.

### 5.4.1 Achievements of the Software

In this section, we highlight some of the main technical accomplishments of our mock-up software for dynamic plan management. In particular, the software provides the following significant features:

- An expressive representation of plans.
- Expressive enough to represent all elements of the North Atlantis vignette.
- A flexible and customizable representation of plan elements.
- Implemented through an *Ontology Manager*.
- Allows new plan elements to be added quickly and easily.
- Driven by a DTD file.
- An extensive plan validation procedure.
- Checks for syntactic correctness and consistency of plans.
- Checks for consistency of plan updates.
- An appropriate model of effects.
- Changes in effects are handed up in the goal/effect hierarchy.
- Determines whether a goal is attained at any point in time.
- An extensive selection of constraints.
- Emphasizes task precedence, modeling all Allen relations.
- A representation of tasks that are part of multiple plans.
- Illustrated in the North Atlantis vignette.
- A representation of resource locations in terms of a co-ordinate system.
- Handles resource attachments correctly.
- A quantitative model of consumable resource amounts.
- Captures consumption through ordinary tasks.
- Handles errors when a resource is exhausted.
- Performs linear interpolation of resource amounts over time.
- A resource availability checker.
- Decides if a resource is available to be scheduled over an interval.
- Checks if the resource is attached to an available resource.

In the following sections, we expand on these points and included specific achievements with respect to plan representation, plan projection, plan analysis and plan monitoring. For each aspect of plan management, we revisit the measures of efficiency introduced in Section 4.1.

#### 5.4.1.1 Plan Representation

The mock-up software succeeds in developing an expressive plan representation, expressive enough to represent complex scenarios like the North Atlantis vignette. The representation is defined through a precise ontology of plan elements, which can be represented in an XML file.

In order to assess this representation of plans, recall the essential categories of plan elements specified in Section 2.5.1:

- Objectives
- Tasks
- Resources
- Constraints.

Our plan representation covers each of these categories and specifies an extensive variety of element types in each category. In fact, we identified an independent fifth category, namely *effects*. Moreover, the representation of plan elements is flexible and extendable. The software includes an ontology manager that allows a user to add new plan elements.

We briefly comment on each of the general categories of plan elements, and then conclude the section with a table summarizing how the software performs with respect to the specified measures of efficiency.

### **Objectives and Tasks**

The connection between tasks and objectives is explicit in our representation of plans, and each is closely connected to the notion of an *effect*. One of the achievements of the mock-up software is that we have developed a precise understanding of the nature of effects. Objectives are represented as desired effects, and the effects of each task are explicitly specified. This makes it explicit when a particular objective has been met.

Additionally, the modular representation of tasks and plans makes it possible for a single task to be a component of several different plans. This is a required feature for the North Atlantis vignette.

### **Resources**

The software uses a precise representation of consumable resources and allocatable resources, which allows each consumable resource to be associated with a particular allocatable resource. The locations of all resources are represented and tracked over time. In the case of consumable resources, consumption is also tracked over time and ordinary tasks such as fueling are treated appropriately. The software successfully determines whether a resource is available for the duration of a task, and also determines if the quantity of a consumable resource is sufficient for a task.

### **Constraints**

The software uses an extensive selection of constraints. In particular, tasks precedence constraints are treated in a manner that models all of the Allen relations, with minimal lag times.

## Effects

As stated, effects form an independent category of elements. Effects are the outcome of tasks, and they contribute to the attainment of goals. Moreover, the attainment of effects can be used as a precondition inside constraints for follow-up tasks. Thus effects form an important link between the other categories.

**Table 5-16 Measures of Efficiency for Plan Representation**

Measure of Efficiency	Assessment
Plan abstraction	No – although tasks are composed of sub-tasks, the present representation requires a set of scheduled actions in the representation of a plan.
Task specification	Yes – the ontology includes a precise and flexible representation of tasks.
Constraint specification	Yes – extensive representation of constraints.
Objective specification	Yes – objectives represented as desired effects.
Definability	No – specifying a plan requires specialized technical knowledge of XML format.
Readability	Yes (limited) – using the map view, it is possible to for a non-expert to understand the representation of a plan.
Uncertainty	Yes – the representation is flexible enough to allow uncertain effects to be added.

### 5.4.1.2 Plan Projection/Forecasting

Plan projection/forecasting is performed based on the initial description of the input plan. The validation of the plan checks that all of the constraints of the plan are satisfied. In terms of projection, this involves projecting the outcome of all tasks in order to ensure that the precedence constraints will be met. Since it is possible to determine if an objective is met at any point in time, the validation procedure also checks if the goals of a given plan will be satisfied when the plan is complete.

The current representation specifies that a plan *succeeds* if all constraints are satisfied and the desired goals are attained. Note that it is possible for a plan to succeed without all of its subtasks succeeding. This allows for alternative courses of action, out of which only one needs to succeed for the plan to succeed. It would be desirable to require that a minimum number of tasks must succeed in order for the plan to succeed. However, such success conditions for a plan can always be represented by a mixture of constraints on the tasks that must succeed, and plan goals that are attained as cumulative effects of the effects of these tasks.

Since plan updates and situation updates are stated over discrete intervals, the projection component of the software performs linear interpolation on consumable resource amounts and resource trajectories within these time intervals. If a consumable resource is going to be insufficient at some projected point in time, then the user will be alerted. Similarly, conflicts over resource availability will be detected.

Using the map view, it is also possible to visualize plan projection. Using the time slider, it is possible to visualize the location of all platforms at any point in time. Hence, by sliding the bar forward, it is possible to view the manner in which a plan is expected to unfold.

**Table 5-17 Measures of Efficiency for Plan Forecasting/Projection**

Measure of Efficiency	Assessment
Resource checking	Yes – resource availability checked during validation.
Pre-processed outcomes	No – predicting outcomes is done for each plan with no pre-processing.
Probabilistic forecasting	Possible – forecasting is currently deterministic, but it is possible to represent uncertain effects and incorporate probabilistic forecasting.
Complete on time	Yes – use precedence constraints.
Plan status	Yes – the software permits forecasting to any point in time.

#### 5.4.1.3 Plan Analysis/Evaluation

The simplest form of plan analysis is simply checking for plan consistency. This involves checking if the specification of a plan actually describes an executable plan. The mock-up software performs extensive consistency checking on the initial plan description, to determine if there are syntactic or logical flaws. Validation is also performed on plan updates and situation updates. For instance, if a plan update or a situation update has a time stamp in the past, then it will be flagged as inconsistent and rejected.

It is possible to use the software to perform post-analysis as follows. The initial description of a plan can be used to project whether the objectives will be achieved as well as the final amounts of all resources. Following several plan updates or situation updates, the results might actually be different. The actual final result of a plan can then be compared with the initial projected results. This comparison is not automated, but it is facilitated by the software.

**Table 5-18 Measures of Efficiency for Plan Analysis/Evaluation**

Measure of Efficiency	Assessment
Military Expertise	Possible – military experts have not been consulted at this stage, but this feature could be incorporated.
Structural Analysis	Yes – extensive syntactic consistency checking.
Semantic Analysis	Yes (limited) – resource attachments, availability and consumption are checked.
Ranking	No – no comparison between plans.



#### 5.4.1.4 Plan Monitoring

In real planning domains, plan monitoring involves monitoring the state of the world. Typically, field units will supply information about changes that might impact plan execution. In the mock-up software, this information is captured through situation updates. The software supports plan monitoring in conjunction with situation updates, thereby providing a realistic model of the manner in which actual military plans are monitored.

As noted previously, it is possible to track resource amounts and locations, as well as the status of all objectives, as situation updates come in. The updated plan can be compared with initial predictions, using the map view as follows: When a situation update is loaded, a new tab is opened, containing a map view with a new time slider pointing at the current time (the time of the situation update). This map view shows the plan execution in the context of the new situation information. The tab containing the original plan execution remains in the viewer and can be opened. Sliding the time bar in the original tab to the same point in time gives us a view of the original prediction of the state of the plans at that time. By switching between the old tab and the new tab, the predicted state and the actual state can be visually compared to monitor the differences between predicted and actual plan execution.

**Table 5-19 Measures of Efficiency for Plan Monitoring**

Measure of Efficiency	Assessment
Feedback Guidance	No – the information received is all in the plan updates and situation updates.
Assisted Re-planning	No – re-planning must be done manually.
Violations	Yes – all violations are reported.
Interacting Plans	Yes – multiple plans may be executed simultaneously.

#### 5.4.2 Limitations of the Software

The representation of plans in the software does have some limitations. For example, specifying even a single plan in an XML document requires significant effort, as well as specialized knowledge. This process could be simplified by implementing a visual plan editor.

The details of our XML implementation also impose limitations on the representation. For example, co-ordinates cannot be referenced; instead references to co-ordinates must be made through named Location objects. Similarly, it is not possible to explicitly represent a trajectory that is followed by two platforms together.

We made the design decision that each resource can only execute one action at a time. This is a reasonable assumption, and it is made in most applications that handle scheduling problems. However, in the Air Force world, one may want to represent a platform that simultaneously flies from location A to B, drops bombs on a target, and observes the situation. Simultaneous actions can currently be handled as follows:

- Each action has an associated trajectory, so a location change can always be modeled within an action.
- Actions are generic, and their “semantics” are described in terms of effects. Therefore, it is possible to define a *single* action “observe and drop bombs” with two effects, SituationEvidenceGained and TargetDestroyed. However, bundling too many or unrelated actions together is an indication of a poor modeling approach.
- The actions are not executed by the platform itself, but rather by the pilot or by components or payloads on the platform. We can define these as allocatable resources and attach them to the platform. Allocating the different actions to the subresources allows us to execute the actions in parallel and still satisfy the constraint that each resource can only execute one action at a time. This comes at the cost of slightly higher modeling complexity.

If it was required that a resource perform multiple actions at once, we could drop the implicit constraint above. However, we would then have to determine the types of actions that *cannot* be executed simultaneously on the same resource (e.g. two motion actions) and define new implicit constraints accordingly. We may also have to state explicit mutex (mutual exclusion) clauses between pairs or sets of actions, which would usually result in a more complex plan definition. Finally, verifying these constraints also becomes more complex, since different actions can start and end at different times and overlap only partially.

The representation of plans is also rigid with respect to the representation of tasks and effects. At present, tasks have fixed durations, when it would be more reasonable to specify minimum durations. In terms of effects, it is currently not possible to represent cumulative effects of the form “effect1 or effect2.” Similarly, it is not possible to represent negations of effects such as “SAM not destroyed” implying “enemy air superiority”. However, these can be modelled using suitable aggregation functions.

Finally, the software is limited with respect to the information that can be visualized. In many cases, the internal representation includes information that can not effectively be visualized. Consider, for example, a single platform moving across the screen. Internally, the software keeps track of exactly how much fuel the platform contains at every point in time. However, the map view does not currently display this information. In general, the internal representation of plan execution contains a great deal of information that can not easily be viewed by a user.

### 5.4.3 Overall Assessment

The mock-up software successfully illustrates how an expressive planning ontology can be used for plan representation, plan forecasting, plan analysis, and plan monitoring. The basic flow of information presented in Figure 5-2 has essentially been realized, and the utility of the approach has been illustrated by a successful representation of the North Atlantis vignette. Therefore, the mockup has been succeeded as an implementation of prototype software illustrating our approach to dynamic plan management.

The area where the software should be improved is in visualization. The original intention was that several visualization modes would be provided, explicitly addressing each area of dynamic plan management. At present, only a single visualization mode is provided. A visual plan

editor in the tradition of NuSketch would also facilitate the authoring of plans. Note, however, that our mockup software does allow a user to view a visual representation of a plan described in an ontology. This addresses one of the main limitations of the ontology approach that we identified; namely, the fact that plans represented in an ontology often can not be understood by a non-specialist.

Overall, the approach taken in the mockup software has been to focus on developing an appropriate internal representation of plans, because it is straightforward to build new visualization tools on top of existing data structures. Thus, our mockup provides a solid framework that could be extended to address the limited visualization capabilities. The actual development of visualization software is a time-consuming task that was beyond the scope of the present implementation, because our emphasis was on providing an accurate model of plans suitable for dynamic plan management.

## **5.5 Avenues for further research and development**

This section contains suggestions for future work. These are distinguished between work on representation and reasoning with plans, and work on components to visualize plans and plan elements, and categorized as follows:

- highly important: work that is deemed critical for an effective plan management system
- promising: work that is expected to significantly enhance the functionality of a system
- interesting for future research: work that will explore and address certain aspects of plan management and thus opens up a new research topic.

**Table 5-20 Items for future research**

	<b>Plan representation and reasoning</b>	<b>Plan visualization</b>
<b>highly important</b>	Resources executing multiple actions simultaneously Evaluation of plans along performance measures Automatic tracing of plan failures to their original causes Automated analysis of competing/mutually exclusive plans	Task view
<b>promising items with high expected benefit</b>		MOP view Browser view to navigate through the plan hierarchy
<b>Interesting for future research</b>	Identification of bottlenecks (e.g. Plan A crucially depends on resource R) Intelligent advisor on resources (e.g. list how buying an additional resource would improve plan performance) High-level, assisted plan editing + repair	

## 6 SUMMARY

The first part of this document provides an overview of the existing literature on dynamic plan management with a particular focus on plan representation, plan forecasting/projection, plan analysis/evaluation and plan monitoring. After presenting some basic background material on military plans, short paper summaries are provided for a wide range of documents on dynamic plan management.

After presenting the paper summaries, several basic approaches to plan management are identified. Each approach is then considered in detail, providing specific details about the strengths of weakness of each approach. Based on this analysis, a set of efficiency criteria is introduced to compare various approaches to dynamic plan management. The most promising approaches are then discussed in greater detail, including examples, analysis, and explicit discussion in the context of the measures of efficiency provided.

The analysis suggests that none of the existing approaches to dynamic plan management provide a complete solution, but a combination of existing approaches is very promising. We suggest an approach that combines an ontology with a consistency checker, reasoning with uncertainty, and a visualization component. Following up on this suggestion, we have implemented a proof-of-concept system using our basic approach. The software is based on a precise ontology, with plans described by XML files. It provides detailed consistency checking for plans, as well as a visualization tool for plan monitoring and plan forecasting.

Assessing our experience in designing, implementing and testing the software, we find that it demonstrates the utility of our approach. However, the intricate interactions between plan elements are very hard to capture and require a very careful approach to plan validation, analysis and projection. This indicates that these tasks cannot (yet) be performed by an off-the-shelf ontology reasoner and justifies our approach of implementing special-purpose reasoning for each category, and sometimes for each type, of plan elements.

We determined that effects are a key category of plan elements, since they provide an important link between actions, plans, constraints and goals. The specific design we used for computing and propagating effects allows us to reason about uncertain and probabilistic outcomes of actions. This suggests that further studies into effects-based reasoning will bring about advances in plan management as well.

Finally, visualization capabilities are a key feature of any plan management system that is to be used in military operations. Validating and projecting a plan involves many recursive steps through the hierarchy of plan elements, so the output of a validation run on even a simple plan is too long to be read by an operator. Hence it is important to highlight the important information and to provide tools for the operator to trace dependencies and causal links, such as the reasons for a plan failure. We find that a graphical, map-based representation of plans considerably eases the task of monitoring plan execution. The map viewer is less suitable for plan analysis, however; a similar viewer for the task hierarchy and the links between plans and tasks, which has been proposed in this report but not yet implemented, is strongly desirable. Other interesting items for future development include a comparative view of performance measures and a browser view of the plan element hierarchy.

## 7 BIBLIOGRAPHY

1 CDN Air Div A3 ASR 3 and AFCCIS (2005). 1 CDN Air Div / AFCCIS Command and Control (C2) Situational Awareness and Assessment Systems: Capabilities, Structure, and Management. 32398-313-0012.

Aberdeen, D., S. Thiébaux, et al. (2004). Decision-Theoretic Military Operations Planning. International Conference on Automated Planning and Scheduling, Whistler, British Columbia, AAAI Press. 402-412

Baader, F., D. Calvanese, et al. (2003). The Description Logic Handbook, Cambridge University Press.

Barker, K., J. Blythe, et al. (2003). A Knowledge Acquisition Tool for Course of Action Analysis. Fifteenth Innovative Applications of Artificial Intelligence Conference Acapulco, Mexico, AAAI Press. 43-50.

Bélanger, M. and A. Guitouni (2000). A Decision Support System for CoA Selection. International Command and Control Research and Technology Symposium. Canberra, Australia.

Bélanger, M., A. Guitouni, et al. (2000). COA Advisory System Based on Multiple Criteria Decision Analysis. TN 2000-012.

Bennett, J. and T. M. Lamoureux (2006). Observation of computer-supported, collaborative work tool usage during briefing and debriefing phases of coalition mission training research for Maple Skies 05-06. CR2006-191.

Bergmann, R., H. Munoz-Avila, et al. (1998). Case-based reasoning applied to planning

Case-Based Reasoning Technology from Foundations to Applications. M. Lenz, B. Bartsch-Sporl, H. D. Burkhard and S. Wess, Springer: 169-200.

Blum, A. and M. Furst (1997). "Fast Planning Through Planning Graph Analysis." Artificial Intelligence Journal 90: 281-300.

- Blythe, J. (1999). "Decision-Theoretic Planning." AI Magazine 1(20).
- Blythe, J. (2002). "The SHAKEN action description language, using KM situations (SADL-S)." from <http://www.isi.edu/expect/rkf/sadl/>.
- Bonet, B. and H. Geffner (2003). Labelled RTDP: Improving the convergence of real-time dynamic programming. International Conference on Automated Planning and Scheduling.
- Booch, G. and Rumbaugh (1998). Unified Modeling Language User Guide, Addison-Wesley.
- Boukerche, K. and M. Rioux (2003). Investigation des planificateurs intelligents. W7701-1-2489.
- Boukhtouta, A., M. Bélanger, et al. (2006). Review of Visualisation Tools for Potential Use in the Recognized Air Picture. Valcartier, Québec. TM 2006-765.
- Boukhtouta, A., F. Bouak, et al. (2006). Description and analysis of military planning systems. TR-2004-320.
- Boury-Brisset, A. C. and C. Champagne (2006). Ontological models for military courses of action. TM-2003-320.
- Canadian Forces (2003). Joint Intelligence Doctrine. J7 Doctrine. Ottawa. B-GJ-005-200/FP-000.
- Chalupsky, H. and R. MacGregor (2002). Ontologies, Knowledge Bases and Knowledge Management. AFRL-IF-TS-TR-2002-162.
- Chaudhri, V., A. Farquhar, et al. (1998). OKBC: A Programmatic Foundation for Knowledge Base Interoperability. National Conference on Artificial Intelligence, Madison, WI, AAAI Press.
- Clark, P., J. Thompson, et al. (2001). Knowledge Entry as the Graphical Assembly of Components. International Conference on Knowledge Capture, Victoria, BC.
- Cohen, P., J. Davis, et al. (2000). Dynamic Visualization of Battle Simulations, University of Massachusetts Department of Computer Science. 00-16.
- Cohen, P., M. Johnston, et al. (1997). QuickSet:Multimodal Interaction for Distributed Applications. ACM Multimedia Conference, Seattle, WA, ACM Press. 31-40.
- Corkill, D. D. (1979). Hierarchical Planning in a Distributed Environment. International Joint Conference on Artificial Intelligence, IJCAI Press. 168–175.
- Cox, M. and M. Veloso (1998). Goal transformations in continuous planning. AAAI Fall Symposium on Distributed Continual Planning, Menlo Park, CA, AAAI Press / The MIT Press. 23-30.
- Davenport, M. (2007). Coalition Battle Management Language (CBML) Study in Support of Maritime Domain Awareness. Richmond, B.C., MacDonald Dettwiler and Associates. Draft.



Department of National Defence (2004). Canadian Forces Operations. J7 Doctrine. Ottawa. B-GJ-005-300/FP-000.

desJardins, M., E. Durfee, et al. (1999). "A Survey of Research in Distributed, Continual Planning." AI Magazine 20(4): 13-22.

desJardins, M. and M. Wolverton (1999). "Coordinating a Distributed Planning System." AI Magazine 20(4): 45-53.

English, A. (2002). Centralized Command and Decentralized Execution. Air Force Command and Control. D. Eriandson and A. English.

Erol, K., J. Hendler, et al. (1994). Semantics for Hierarchical Task-Network Planning. College Park, Maryland, University of Maryland at College Park: 28. CS-TR-3239.

Erol, K., D. Nau, et al. (1994). HTN Planning: Complexity and Expressivity. National Conference on Artificial Intelligence, AAAI Press. 1123-1128.

Ferguson, G. and J. F. Allen (1994). Arguing About Plans: Plan Representation and Reasoning for Mixed-Initiative Planning. Artificial Intelligence Planning Systems, Chicago, Illinois, AAAI Press. 43-48.

Fikes, R. and N. J. Nilsson (1971). "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." Artificial Intelligence Journal 2(3/4): 189-208

Forbus, K., J. Usher, et al. (2003). Sketching for military courses of action diagrams. International Conference on Intelligent User Interfaces, Miami, FL, USA, ACM. 61-68.

Fox, M. and D. Long (2002). "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains." Journal of Artificial Intelligence Research 20: 61-124.

Ghallab, M., D. Nau, et al. (2004). Automated Planning: Theory and Practice, Morgan Kaufman.

Gil, Y. and J. Blythe (2000). PLANET: A Shareable and Reusable Ontology for Representing Plan. AAAI Workshop on Representational Issues for Real-world Planning Systems.

Gil, Y. and V. Ratnakar (2002). TRELLIS: An Interactive Tool for Capturing Information Analysis and Decision Making. Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, Siguenza, Spain, Springer. 37-42.

GlobalSecurity.org. (2007). "C2PC (Command and Control for the PC)." Retrieved 2007-03, from <http://www.globalsecurity.org/intell/library/reports/2001/compendium/c2pc.htm>.

GlobalSecurity.org. (2007). "Joint Mapping Tool Kit (JMTK)." Retrieved 2007-03, from <http://www.globalsecurity.org/intell/systems/jmtk.htm>.

GlobalSecurity.org. (2007). "Theater Battle Management Core Systems (TMBCS)." Retrieved 2007-03, from <http://www.globalsecurity.org/military/systems/aircraft/systems/tbmcs.htm>.

Gratch, J. and R. Hill (1999). Continuous Planning and Collaboration for Command and Control in Joint Synthetic Battlespaces. The 8th Conference on Computer Generated Forces and Behavioural Representation, Orlando, FL.

Green, M. (2001). "Modeling & Simulation Web Links." Retrieved 2007-03, from <http://www.strategypage.com/prowg/MAND21.HTM>.

Hollands, J. G. (2006). CommandView User Interface Evaluation: Preliminary Heuristic Analysis Results, DRDC. TM 2006-039.

Howard, R. A. (1960). Dynamic Programming and Markov Processes. Cambridge, MA, MIT Press.

Jensen, K. (1990). Coloured Petri Nets: A high-level Language for System Design and Analysis, Springer Verlag.

JIIFC PD (2005). JIIFC Definition Paper--The Common Operations Picture.

Joint Warrior Interoperability Demonstration Management Office. (2004). "Joint Warrior Interoperability Demonstration 2004 Final Report." Retrieved 2007-03, from <http://www.cwid.js.mil/public/cwid05fr/htmlfiles/c218intr.html>.

Kautz, H. and B. Selman (1992). Planning as Satisfiability. European Conference on Artificial Intelligence. 359-363.

Kosara, R. and S. Miksch (2000). A Visualization of Medical Therapy Plans Compared to Gantt and PERT Charts. TIME. 173-182.

Kristensen, L. M., S. Christensen, et al. (1998). "The Practitioner's Guide to Coloured Petri Nets." International Journal on Software Tools for Technology Transfer 2(2): 98-132.

Kristensen, L. M., P. Mechlenborg, et al. (2006). Model-based Development of a Course of Action Scheduling Tool. Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools. University of Aarhus, Denmark.

Kundu, K., C. Sessions, et al. (2002). Three-Dimensional Visualization of Hierarchical Task Network Plans. The 3rd International NASA Workshop on Planning and Scheduling for Space, Houston, Texas.

Lenat, D. (1995). "CYC: A Large-Scale Investment in Knowledge Infrastructure." Communications of the ACM 38(11).

Lino, N. Q. and A. Tate (2004). A Visualization Approach for Collaborative Planning Systems Based on Ontologies. International Conference on Information Visualisation, London, UK, IEEE Computer Society 2004. 807-811.

Livnat, Y., J. Agutter, et al. (2005). Visual Correlation for Situational Awareness. IEEE Symposium on Information Visualization. Minneapolis, MN.

MacGregor, R. (1991). "Inside the LOOM Classifier." SIGART Bulletin 2(3): 70-76.

- Marcotty, M. and H. Ledgard (1986). The World of Programming Languages. Berlin Springer-Verlag.
- McGee, D. R. and P. Cohen (2001). Creating Tangible Interfaces by Augmenting Physical Objects with Multimodal Language. International Conference on Intelligent User Interfaces, Sante Fe, NM, ACM Press. 113-119.
- Miksch, S., R. Kosara, et al. (1998). AsbruView: Visualization of Time-Oriented, Skeletal Plans. The International Conference on Artificial Intelligence Planning Systems, AAAI Press.
- Miksch, S. and A. Seyfang (2000). Continual Planning with Time-Oriented, Skeletal Plans. The 14th European Conference on Artificial Intelligence, IOS Press. 511-515.
- Miller, T. E., L. G. Militello, et al. (1996). Evaluating Air Campaign Plan Quality in Operational Settings. Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative. A. Tate. Menlo Park, California, AAAI Press: 195-200.
- Mulvehill, A. and J. Caroli (1999). JADE: A Tool for Rapid Crisis Action Planning. International Command and Control Research and Technology Symposium. Newport, RI.
- Musliner, D. J., E. H. Durfee, et al. (2006). Coordinated Plan Management Using Multiagent MDPs. AAAI Spring Symposium on Distributed Plan and Schedule Management, Stanford University, AAAI Press.
- Myers, K. L., P. A. Jarvis, et al. (2003). A Mixed-initiative Framework for Robust Plan Sketching. International Conference on Automated Planning and Scheduling, Trento, Italy, AAAI Press. 256-266
- National Geospatial-Intelligence Agency. (2007). "Commercial Joint Mapping Toolkit." Retrieved 2007-03, from <http://www.cjmtk.com/>.
- NATO Programming Centre. (2007). "NATO-wide Integrated Command and Control Software for Air Operations (ICC)." Retrieved 2007-03, from <http://www.npc.nato.int/hm/icc.htm?tsfsg=2d0b477ebd8294862b1bc28ad73c4c21>.
- Noriega, P., C. Sierra, et al. (1998). A Framework for Argumentation-Based Negotiation. Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages, Berlin, Springer-Verlag. 177-192.
- Noy, N., W. Grosso, et al. (2000). Knowledge-Acquisition Interfaces for Domain Experts: An Empirical Evaluation of Protege-2000. Twelfth International Conference on Software Engineering and Knowledge Engineering, Chicago, IL.
- Pollack, M. (1996). Planning in Dynamic Environments: The DIPART System. Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative. A. Tate. Menlo Park, California, AAAI Press: 218-225.
- Pollack, M. and J. Horta (1999). "There's More to Life than Making Plans: Plan Management in Dynamic, Multiagent Environments." AI Magazine 20(4): 71-83.

Pollack, M., I. Tsamardinos, et al. (1999). Adjustable Autonomy for a Plan Management Agent. AAAI Spring Symposium on Agents with Adjustable Autonomy, Stanford, CA.

Polyak, S. and A. Tate. (1999). "Planning Initiative: Shared Planning and Activity Representation - SPAR Version 0.2." from <http://www.aiai.ed.ac.uk/~arpi/spar/spar-doc-02.html>.

Ragsdale, D. J., C. D. C. Butler, B.A. , et al. (1997). A Fuzzy Logic Approach for Intelligence Analysis of Actual and Simulated Military Reconnaissance Missions. IEEE International Conference on Systems, Man, and Cybernetics, Orlando, FL. 2590-2595.

Rasch, R., A. Kott, et al. (2002). AI on the Battlefield: An Experimental Exploration. Innovative Applications of Artificial Intelligence, Edmonton, AB, AAAI Press.

Rasch, R., A. Kott, et al. (2002). AI on the Battlefield: An Experimental Exploration. The Fourteenth Conference on Innovative Applications of Artificial Intelligence, Edmonton, Alberta, AAAI Press. 906-912.

Sacerdoti, E. D. (1977). A Structure for Plans and Behavior, New York: American Elsevier.

Sycara, K. (1989). Argumentation: Planning other Agents' Plans. Eleventh International Joint Conference on Artificial Intelligence. 517-523.

Tate, A. (1977). Generating Project Networks. Proceedings of the 5th International Joint Conference on Artificial Intelligence, Cambridge, MA, USA. 888-893

Tate, A. (1996). Representing Plans as a Set of Constraints - the <I-N-OVA> Model. International Conference on Artificial Intelligence Planning Systems, Edinburgh, Scotland, AAAI Press. 221-228.

Tate, A. (1996). "Towards a Plan Ontology." AI-IA Newsletter (Quarterly Publication of teh Associazione per l'Intelligenza Artificiale) 9(1): 19-26.

Tate, A. (1998). "Roots of SPAR - Shared Planning and Activity Representation." The Knowledge Engineering Review 13(1): 121-128.

Tate, A., J. Dalton, et al. (1999). Multi-Perspective Planning: Using Domain Constraints to Support the Coordinated Development of Plans, University of Edinburgh and DARPA Technical Report 238. AFRL-IF-RS-TR-1999-60.

Tate, A., J. Levine, et al. (2000). Using AI Planning Technology for Army Small Unit Operations. International Conference on Artificial Intelligence Planning Systems, Breckenridge, Colorado, AAAI Press. 379-387.

Thuve, H. (2001). TOPFAS (Tool for Operational Planning, Force Activation and Simulation). International Command and Control Research and Technology Symposium. Annapolis, Maryland.

Valente, A., J. Blythe, et al. (1999). On the role of humans in enterprise control systems: the experience of INSPECT. DARPA-JFACC Symposium on Advanced in Enterprise Control. San Diego, California, DARPA.

Valente, A., Y. Gil, et al. (1996). INSPECT: an Intelligent System for Air Campaign Plan Evaluation based on EXPECT, USC/ISI

Valente, A., T. Russ, et al. (1999). "Building and (Re)Using an Ontology of Air Campaign Planning." IEEE Intelligent Systems: 27-36.

Valente, A., W. Swartout, et al. (1998). A Representation and Library for Force Support Objectives in Air Campaign Plans. USC/ISI Technical Report, University of Southern California.

Veloso, M., J. Carbonell, et al. (1995). "Integrating Planning and Learning: The PRODIGY Architecture." Journal of Theoretical and Experimental Artificial Intelligence 7(1).

Veloso, M., M. Pollack, et al. (1998). Rationale-Based Monitoring for Planning in Dynamic Environments. International Conference on AI Planning Systems.

Vrakas, D. and I. Vlahavas (2005). "A Visualization Environment for Planning." International Journal on Artificial Intelligence Tools 14(6): 975-998.

Wilkins, D. E. and R. V. Desimone (1994). Applying an AI planner to military operations planning. Intelligent Scheduling. M. Fox and M. Zweben. San Francisco, CA, Morgan Kaufmann Publishers Inc.: 685-709.

Wilkins, D. E., T. Lee, et al. (2003). "Interactive Execution Monitoring of Agent Teams." Journal of Artificial Intelligence Research 18: 217-261.

Williams, B. T. (2002). Effects-Based Operations: Theory, Application and the Role of Airpower, Defence Technical Information Centre. ADA400990.

Zhang, L., L. M. Kristensen, et al. (2002). A Coloured Petri-net based tool for course of action development and analysis. Workshop on Formal Methods Applied to Defence Systems, Adelaide, Australia, Australian Computer Society. 125-134.

Ref: RX-RP-52-5076  
Issue/Revision: 1/3  
Date: DEC. 07, 2007

UNCLASSIFIED



**THIS PAGE INTENTIONALLY LEFT BLANK**

## A APPENDIX

### A.1 Plan File Syntax

This appendix describes the syntax of plans, plan updates and situation updates. All files are in XML format, and they all use a subset of the same XML element structure. This appendix first introduces the common element structure as defined in the `plan.dtd` file; then it introduces the specific subset for each type of file.

### A.2 Document Type Definition file format

The DTD file defines the syntax of plans, plan updates and situation updates by stating the categories of plan elements and what subelements they may contain. Its use is two-fold:

- It can be used (optionally) to validate the XML structure of a plan syntactically, using a standard, off-the-shelf XML validator.
- It is used by the Ontology Manager to ensure that only plans with a correct syntax are loaded.

The file is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated from CSAR01-valid.xml by XMLBuddy -->
<!ELEMENT PlanManager
(#PCDATA|constraint|goal|location|plan|platform|allocatableResource|schedule|vignetteBounds)*>
<!-- ATTLIST PlanManager
    end NMTOKEN #REQUIRED
    name CDATA #REQUIRED
    start NMTOKEN #REQUIRED
-->
<!ELEMENT VignetteBounds (coordinate*)>
<!ELEMENT ScheduledAction EMPTY>
```

```
<!--ATTLIST ScheduledAction
    action NMTOKEN #REQUIRED
    allocated CDATA #REQUIRED
    endTime NMTOKEN #REQUIRED
    startTime NMTOKEN #REQUIRED
>
<!--ELEMENT action
(#PCDATA|effectAttainment|resourceAmount|trajectory)*>
<!--ATTLIST action
    description CDATA #REQUIRED
    name NMTOKEN #REQUIRED
>

<!--ELEMENT actionExecution (#PCDATA|effectAttainment)*>
<!--ATTLIST actionExecution
    action NMTOKEN #REQUIRED
    duration CDATA #IMPLIED
    endTime NMTOKEN #IMPLIED
>

<!--ELEMENT constraint (effect?)>
<!--ATTLIST constraint
    name CDATA #IMPLIED
    predTask NMTOKEN #IMPLIED
    quantity (1|2|4) #IMPLIED
    resource CDATA #IMPLIED
    resource_type CDATA #IMPLIED
    start NMTOKEN #IMPLIED
    succTask NMTOKEN #IMPLIED
    task (this) #IMPLIED
    type
(AllenDuringConstraint|AllocatableResourceConstraint|TimeWindowConstraint) #REQUIRED
>
<!--ELEMENT coordinate EMPTY>
<!--ATTLIST coordinate
    x NMTOKEN #REQUIRED
    y NMTOKEN #REQUIRED
>
<!--ELEMENT effectAttainment (effect*)>
<!--ATTLIST effectAttainment
    effect NMTOKEN #IMPLIED
    value CDATA #IMPLIED
>
<!--ELEMENT effect (effect?)>
<!--ATTLIST effect
    location NMTOKEN #IMPLIED
    name CDATA #IMPLIED
    ref NMTOKEN #IMPLIED
    resource CDATA #IMPLIED
    source NMTOKEN #IMPLIED
    type NMTOKEN #IMPLIED
>
<!--ELEMENT goal (effect+)>
<!--ATTLIST goal name CDATA #REQUIRED>

<!--ELEMENT location (coordinate*)>
<!--ATTLIST location
```



```

        name CDATA #IMPLIED
        ref CDATA #IMPLIED
    >
<!ELEMENT plan (#PCDATA|goal|constraint|task)*>
<!--ATTLIST plan name ID #REQUIRED-->
<!ELEMENT platform (location|resource)*>
<!--ATTLIST platform
        description CDATA #IMPLIED
        name CDATA #REQUIRED
        type CDATA #REQUIRED
    -->
<!ELEMENT resource (#PCDATA|location|platform|resource)*>
<!--ATTLIST resource
        description CDATA #REQUIRED
        name CDATA #REQUIRED
        type (AllocatableResource|personnel) #REQUIRED
    -->
<!ELEMENT allocatableResource
        (#PCDATA|location|platform|resource)*>
<!--ATTLIST allocatableResource
        description CDATA #REQUIRED
        name CDATA #REQUIRED
        type CDATA #IMPLIED
    -->
<!--ELEMENT resourceConsumption EMPTY-->
<!--ATTLIST resourceConsumption
        amount NMTOKEN #REQUIRED
        type NMTOKEN #REQUIRED
    -->
<!--ELEMENT schedule (#PCDATA|ScheduledAction)*>
<!--ELEMENT task (action|constraint|task)*>
<!--ATTLIST task
        description CDATA #IMPLIED
        name ID #IMPLIED
        ref (a|e|f) #IMPLIED
    -->

<!--ELEMENT track (trajectory)*>
<!--ATTLIST track
        name CDATA #IMPLIED
        affiliation (friendly|neutral|hostile) #IMPLIED
    -->

<!--ELEMENT trajectory (coordinate+)>
<!--ATTLIST trajectory duration NMTOKEN #REQUIRED-->

```

## A.3 Plan files

A plan file may contain all the elements above, with the exception of `actionExecution`, which is reserved for situation updates. In particular, it may contain `track` elements to describe the initial and projected situation as of the mission start time.

## A.4 Situation updates

A situation update file may only contain `track` and `actionExecution` elements, describing

- the updated current and projected tracks,
- the success (in terms of attainment of effects) of actions that have finished since the last update, or the (updated) projected success of actions scheduled to finish in the future, respectively.

## A.5 Plan updates

A plan update file takes essentially the same structure as a plan. However, the information in the file is processed differently:

- Any new elements are added to the plan. However, it is an error to define a new plan element with the same name as an existing element.
- When referencing an existing element (using the attribute `ref=element_name`), any information (attributes, child elements) will be added to the element or, if previously specified, will replace the old information.
- If the additional attribute `remove=true` (which is valid only in plan updates) is specified, the referenced element will be deleted from the plan.

There are certain restrictions on plan updates to preserve the integrity of the ongoing plan execution. Specifically:

- Plan elements that refer to a time in the past (before the plan update time stamp) cannot be modified.
- Some attributes cannot be modified because they describe the initial setting of the element and hence are deemed to occur in the past. An example of this is the initial attachment of a resource to another. Changes in resource attachment must be specified as effects instead.
- Elements cannot be deleted if this would destroy the consistency of the plan. For instance, a scheduled action can be deleted (which effectively removes this action from the schedule), but an allocatable resource cannot (since it may be allocated to future actions). A resource can only be *disabled* by modifying its status (which will then prevent the future actions from being successfully scheduled).

## **B APPENDIX - IDENTIFICATION OF FOREGROUND INFORMATION**

The Foreground Information delivered in fulfilment of Contract W7701-63946NP “Identification of techniques/approaches for dynamic plan management in the context of a Recognized Air Picture” consists of the following artefacts:

- a final report presenting/explaining/describing the work done in this contract (this report)
- a bibliographic database containing all references that have been consulted during this contract
- a mock-up of an interactive, intuitive and adaptive human-computer interface to support multi-dimensional COA/plan visualization (Java source code and executables), with the exceptions listed below.

The following pieces of software are Background Information and remain Copyright © 2007 MacDonald, Dettwiler & Associates Ltd.:

In DPM.jar:

- All interfaces and classes in CCW.jar
- The files down.gif, left.gif, right.gif, up.gif, marker.gif

In the src folder:

- The files ca.mda.dpm.util.IndexedMap.java, ca.mda.dpm.util.IndexedMapData.java and ca.mda.dpm.util.IndexedMapTest.java
- ca.mda.dpm.core.PlanFile.java: all parts stated in the source header
- ca.mda.dpm.ui.map.MapPanel.java: entire file, except those parts stated in the source header
- ca.mda.dpm.ui.map.MapViewer.java: entire file, except those parts stated in the source header

In DPM.jar and main.jar:

- All .class files derived from the above source files

The contents of the following jar files, as well as the classes and interfaces com.simontuffs.onejar.\* are open-source tools under the following licences, which remain unaffected:

dtdparser121.jar	LGPL ( <a href="http://www.fsf.org/licenses/licenses/lgpl.html">www.fsf.org/licenses/licenses/lgpl.html</a> )
jcoord.jar	GPL ( <a href="http://www.gnu.org/copyleft/gpl.html">http://www.gnu.org/copyleft/gpl.html</a> )
collections-generic-4.01.jar	Apache License, Version 2.0 ( <a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a> )
dateparser.jar	W3C Software License ( <a href="http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231">http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231</a> )
jts-1.8.jar	LGPL
nanoxml-2.2.3.jar	zlib/libpng License ( <a href="http://www.opensource.org/licenses/zlib-license.html">http://www.opensource.org/licenses/zlib-license.html</a> )
swing-worker-1.1.jar	LGPL
junit.jar	IBM Common Public License ( <a href="http://www-128.ibm.com/developerworks/library/os-cpl.html">http://www-128.ibm.com/developerworks/library/os-cpl.html</a> )
com.simontuffs.onejar	( <a href="http://one-jar.sourceforge.net/one-jar-license.txt">http://one-jar.sourceforge.net/one-jar-license.txt</a> )



UNCLASSIFIED

Ref: RX-RP-52-5076  
Issue/Revision: 1/3  
Date: DEC. 07, 2007

## DISTRIBUTION LIST

NAME	COPY	NAME	COPY

UNCLASSIFIED  
SECURITY CLASSIFICATION OF FORM  
(Highest Classification of Title, Abstract, Keywords)

<b>DOCUMENT CONTROL DATA</b>		
<b>1. ORIGINATOR (name and address)</b> MacDonald Dettwiler and Associates Ltd 13800 Commerce Parkway Richmond, BC, Canada V6V 2J3		<b>2. SECURITY CLASSIFICATION</b> (Including special warning terms if applicable) Unclassified
<b>3. TITLE</b> (Its classification should be indicated by the appropriate abbreviation (S, C, R or U)) Dynamic Plan Management in the Context of a Recognized Air Picture		
<b>4. AUTHORS</b> (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.) Aaron Hunter, Jens Happe, Melanie Dutkiewicz		
<b>5. DATE OF PUBLICATION</b> (month and year) December 2007	<b>6a. NO. OF PAGES</b> 154	<b>6b. NO. OF REFERENCES</b> 98
<b>7. DESCRIPTIVE NOTES</b> (the category of the document, e.g. technical report, technical note or memorandum. Give the inclusive dates when a specific reporting period is covered.) Contractor report		
<b>8. SPONSORING ACTIVITY</b> (name and address)		
<b>9a. PROJECT OR GRANT NO.</b> (Please specify whether project or grant) 13dy	<b>9b. CONTRACT NO.</b> W7701-6-3946	
<b>10a. ORIGINATOR'S DOCUMENT NUMBER</b> RX-RP-52-5076	<b>10b. OTHER DOCUMENT NOS</b>  N/A	
<b>11. DOCUMENT AVAILABILITY</b> (any limitations on further dissemination of the document, other than those imposed by security classification)  <div style="display: flex; align-items: flex-start;"><div style="margin-right: 10px;"><input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/></div><div><div>Unlimited distribution</div><div>Restricted to contractors in approved countries (specify)</div><div>Restricted to Canadian contractors (with need-to-know)</div><div>Restricted to Government (with need-to-know)</div><div>Restricted to Defense departments</div><div>Others</div></div></div>		
<b>12. DOCUMENT ANNOUNCEMENT</b> (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.) No limitation		

UNCLASSIFIED  
SECURITY CLASSIFICATION OF FORM  
(Highest Classification of Title, Abstract, Keywords)

UNCLASSIFIED  
SECURITY CLASSIFICATION OF FORM  
(Highest Classification of Title, Abstract, Keywords)

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This document surveys existing literature on dynamic plan management and describes the development of a prototype Air Force plan management system. The literature survey presents short summaries of a wide range of research papers, as well as a synthesis and analysis of existing approaches. The detailed comparison of existing approaches is used to formulate a specific methodology for the development of software for plan representation, plan forecasting/projection, plan analysis/evaluation and plan monitoring. The proposed methodology involves the development of a precise ontology of plan elements for plan representation. This representation removes ambiguity in the description of plans, facilitates automated analysis of plans, and also permits several different approaches to plan visualization. The implemented prototype software defines an ontology that provides a suitable internal representation of plans, along with basic plan validation capabilities. It also provides a map-based graphical user interface to visualize plans. The use of the software is demonstrated in the context of a combat search and rescue vignette. The findings conclude that the prototype demonstrates the overall utility of the approach, although further development is required to provide more detailed analysis, as well as additional visualization methods.

Ce document examine la littérature existante au sujet de la gestion des plans et décrit le développement d'un prototype pour la gestion de plans de la Force Aérienne. L'enquête de littérature fournit de courts sommaires de plusieurs articles de recherches et une comparaison détaillée des approches existantes. Cette comparaison est employée pour formuler une méthodologie pour le développement de logiciel pour la représentation, la prévision, l'évaluation, et le suivi de plans. La méthodologie proposée requiert le développement d'une ontologie pour la représentation de plans. Cette représentation peut éliminer l'ambiguïté dans la description de plans, faciliter l'analyse automatisée de plans et permettre plusieurs approches à la visualisation. Le prototype implémenté définit une ontologie qui fournit une représentation interne appropriée des plans, avec en plus la capacité de validation des plans. Il fournit également un interface graphique pour visualiser des plans. L'utilisation du prototype est démontrée dans le contexte d'une vignette de recherche et sauvetage en combat. Les résultats expérimentaux obtenus avec le prototype démontre l'utilité de l'approche proposée, malgré que plus de développement et d'autres méthodes visualisation soient nécessaire.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Plan Management, plan representation, plan forecasting, plan analysis, plan monitoring, Recognized Air Picture

UNCLASSIFIED  
SECURITY CLASSIFICATION OF FORM  
(Highest Classification of Title, Abstract, Keywords)





## **Defence R&D Canada**

Canada's Leader in Defence  
and National Security  
Science and Technology

## **R & D pour la défense Canada**

Chef de file au Canada en matière  
de science et de technologie pour  
la défense et la sécurité nationale



[www.drdc-rddc.gc.ca](http://www.drdc-rddc.gc.ca)

