MODELING WORKLOAD EFFECTIVENESS AND EFFICIENCY OF AIR FORCE WING COMMAND AND CONTROL

THESIS

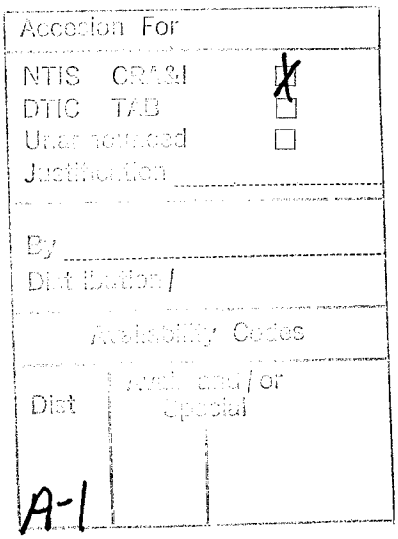
Michael D. Sarchet
Captain, USAF

AFIT/GCS/ENG/94D-22

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/94D-22



MODELING WORKLOAD EFFECTIVENESS AND EFFICIENCY OF AIR FORCE WING COMMAND AND CONTROL

THESIS

Michael D. Sarchet
Captain, USAF

AFIT/GCS/ENG/94D-22

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	Master's Thesis

4. TITLE AND SUBTITLE
Modeling Workload Effectiveness and Efficiency of Air Force Wing Command and Control

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael D. Sarchet

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/94D-22

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Capt Randel Langloss
HQ PACAF/SCE
25 E Street, Suite C-310
Hickam AFB, HI 96853

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This research investigated the feasibility of applying software engineering technology to the Air Force wing command and control (C2) domain. As part of this research, domain analysis and object-oriented techniques were investigated and a specific approach was chosen to analyze the domain. Analysis of the domain resulted in an object-oriented domain model that captured the key objects, operations, and associations of wing C2. The domain model was used to design and implement a prototype software tool that enables wing decision makers to make assessments about automation's impact on wing C2 operations.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software Engineering, Domain Analysis, Domain Modeling Command & Control (C2), Object-Oriented Modeling	77
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements*.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

*U.S.GPO:1990-0-273-271

AFIT/GCS/ENG/94D-22

MODELING WORKLOAD EFFECTIVENESS AND EFFICIENCY OF AIR FORCE WING COMMAND AND CONTROL

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Michael D. Sarchet, B.S.

Captain, USAF

December 1994

Table of Contents

List of Figures

Acknowledgements

Someone once said, "A wise man will not walk down the road of success with you. He'll simply point you in the right direction," and I cannot think of a statement that better characterizes "the three wise men" of my thesis committee. I owe a great deal of thanks to my thesis advisor, Dr Thomas Hartrum, for guiding me through the research process and for providing me the feedback I needed to complete this thesis. I'd also like to thank the readers on my thesis committee, Majors Paul Bailor and David Luginbuhl for helping me fine tune the actual thesis document.

I owe a special thanks to my research partner and good friend, Captain Rob Hunt, for being a great teammate, a hard worker, and an even better golfing companion!

Most of all, I want to thank my fianceé, best friend, and soul mate, Lisa Martinez. Her love and support got me through this Master's program, and I would not have made it without her.

Michael D. Sarchet

Abstract

This research investigated the feasibility of applying software engineering technology to the Air Force wing command and control (C2) domain. As part of this research, domain analysis and object-oriented techniques were investigated and a specific approach was chosen to analyze the domain. Analysis of the domain resulted in an object-oriented domain model that captured the key objects, operations, and associations of wing C2. The domain model was used to design and implement a prototype software tool that enables wing decision makers to make assessments about automation's impact on wing C2 operations.

MODELING WORKLOAD EFFECTIVENESS AND EFFICIENCY OF AIR FORCE WING COMMAND AND CONTROL

I. Introduction

1.1 Background

A combat commander's ability to effectively command and control his forces often means the difference between victory and defeat on the battlefield. Nowhere was this lesson more clear than in operation DESERT STORM. The Coalition forces, knowing the importance of command and control (C2) to combat operations, began the air war by targeting high priority Iraqi C2 nodes, and within a matter of hours Saddam Hussein had lost the ability to effectively command and control his forces. This decisive blow by the Coalition forces early in the campaign prevented Saddam Hussein and the Iraqi military from ever gaining the upper hand.

Today, battlefield commanders rely heavily on computer-based systems to maintain positive command and control of their combat forces. For example, the AWACS system provides commanders with a composite picture of the air battle, while Joint-STARS provides commanders with a picture of the ground battle. The situational awareness provided by these systems gives commanders the ability to control the air space and direct air strikes in near realtime.

One reason commanders have come to depend on C2 systems is because capability and performance have increased dramatically in recent years. C2 systems can now take multiple sources of data and correlate the information into a concise picture of the battle. Directly responsible for these robust systems are advances in computer technology like processing capability, networking capability, and user interfaces. However, as C2 systems have become more specialized and complex so has the effort required to develop new C2 systems and maintain existing systems. This, in turn, has led to increased development and maintenance costs. Also, battlefield commanders now have the responsibility to interpret and act upon the vast amount of information being produced by these modern C2 systems. Increasing costs and system complexity have led C2 experts to the realization that insufficient attention has been focused on how to integrate these highly specialized systems into a single battle management system. Without a mechanism to understand how a wing conducts its mission, it is difficult (if not impossible) to perform this kind of mission analysis or integrate new C2 systems into an overall battle management architecture.

1.2 Problem

Currently, Air Force wing and headquarters decision makers do not have a way to systematically represent how a wing goes about performing its mission. This deficiency means key personnel do not have the management and analysis tools necessary to assess the impact of C2 automation on wing operations or understand what requirements are best satisfied with computer-based C2 systems. Further, system developers and maintainers also do not have a knowledge base to use in the specification and design of new computer-based C2 systems or in modification of existing systems.

One way to capture the knowledge about key objects, operations, and relationships relevant to wing C2 is through domain analysis. Performing a domain analysis on the Air Force wing C2 domain would result in the production of a domain model that consists of taxonomic information to describe the domain structure and rule bases to capture domain knowledge. The domain model could then be used by wing and headquarters personnel to analyze the effects of C2 automation on unit readiness and effectiveness and best decide how to allocate scarce resources. The domain model would also help key decision makers in understanding what aspects of the mission are most suitable for automation. Furthermore, system developers and maintainers would have a definitive repository or knowledge base to establish the design of new C2 systems or justify the modification of existing systems. Thus, a domain model of wing C2 operations would serve as a management tool for wing and headquarters decision-makers while also serving as a reusable requirements repository for system developers and maintainers (see Figure 1).

1.2.1 Problem Statement. **Demonstrate the feasibility of applying software engineering technology to Air Force wing command and control.**

1.2.2 Research Objectives.

- Demonstrate how domain analysis techniques can be used to develop engineering models of Air Force wing C2 operations.
- Show how engineering models of Air Force wing C2 operations can be used to develop a software tool that determines the effects of automation.

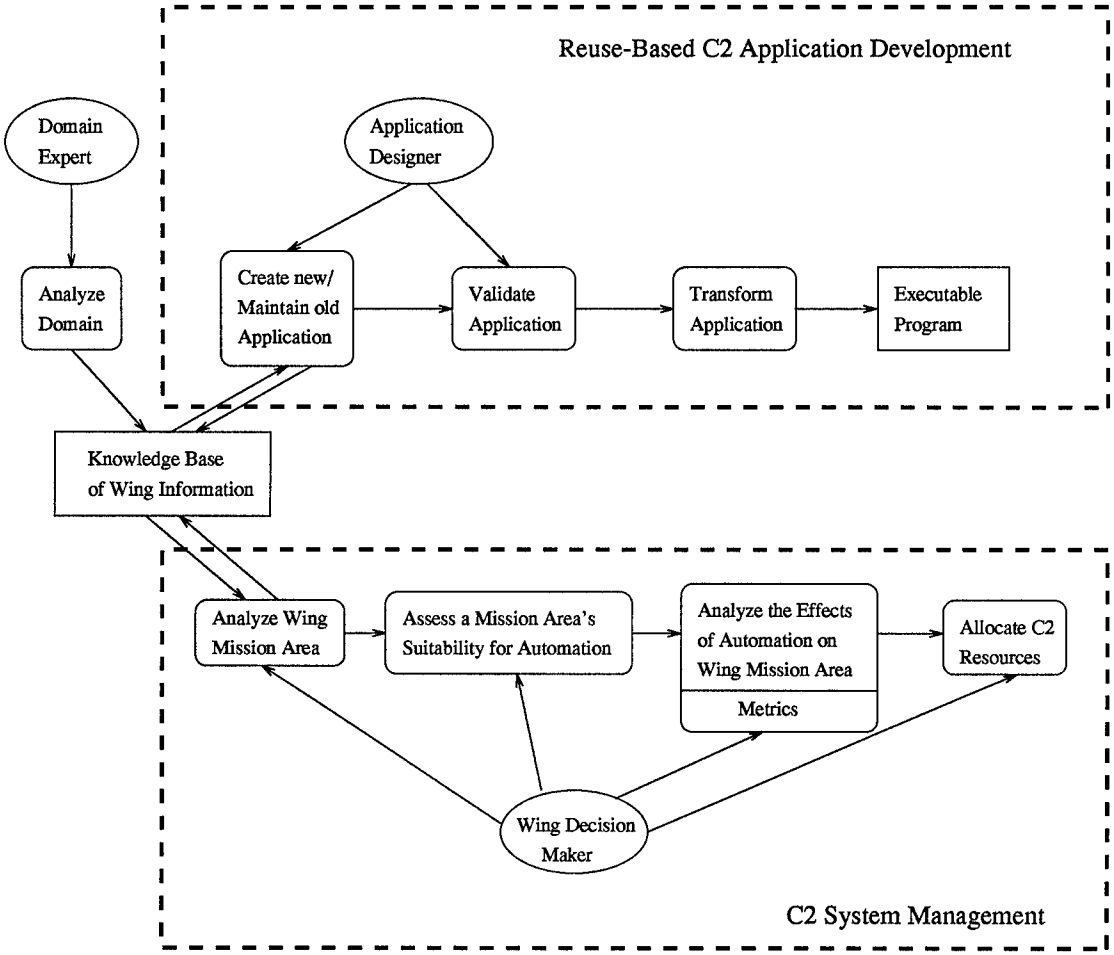


Figure 1. Formalized Model Development Process (21:1–2)

1.3 Scope

The domain analysis portion of this thesis effort focused on Air Tasking Order (ATO) processing, scheduling, and execution. As such, this research focused on the details of the Operations Group while the supporting groups were modeled more abstractly. Figure 2 shows the organization of a typical Air Force wing and the scope of this research effort.

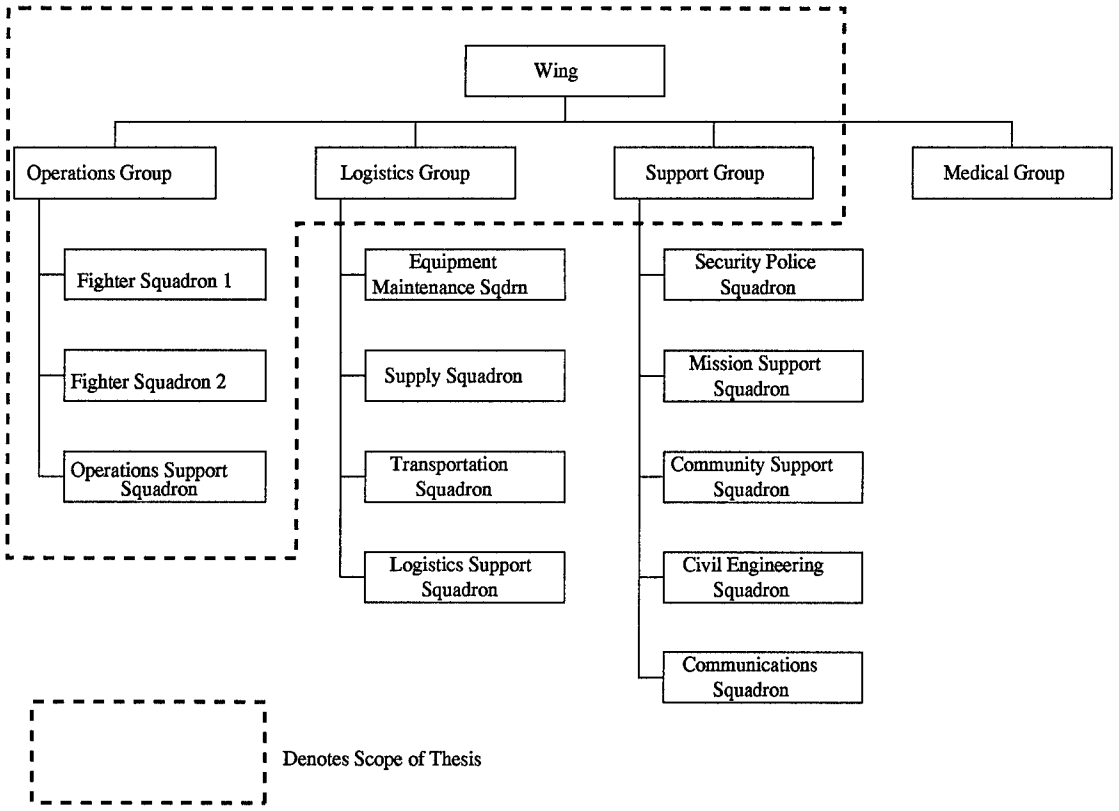


Figure 2. Typical Wing Organizational Chart (8:8)

1.4 Assumptions

Prior to conducting this research effort it was necessary to make a few assumptions. First, it was assumed that domain experts would be available to support the domain analysis process and verify domain model results. It was also assumed that the research

sponsor would provide access to valid and up-to-date information on past, present, and future C2 domain analysis activities.

1.5 Approach

To meet the challenges and research objectives described in Section 1.2, the following tasks were accomplished:

- *Gain an Understanding of Domain Analysis* - A survey was conducted of current software engineering literature, focusing primarily on domain analysis and object-oriented analysis/design.
- *Analyze Previous C2 Work* - Past, present, and future domain analysis work in the C2 domain was examined to determine applicability to this research project and avoid any duplication of effort.
- *Define a Domain Analysis Process* - Based on knowledge gained from the previous two tasks, a specific domain analysis process was defined for this research effort.
- *Perform Domain Analysis* - This task involved performing a domain analysis of Air Force wing C2 operations and producing a corresponding domain model. Sponsor C2 experts were consulted to verify the correctness of the domain analysis results and domain model.
- *Instantiate a Domain Model* - The domain model was used as a knowledge base to build a prototype software tool.
- *Prototype Testing* - The final phase of the research was to test the prototype software tool for correct behavior.

1.6 Sequence of Presentation

The remainder of the thesis is organized as follows: Chapter II reviews current software engineering literature. Chapter III describes the domain analysis process used to analyze the Air Force wing C2 domain and presents the domain model that resulted from the domain analysis. Chapter IV contains the analysis and design of the prototype tool.

Chapter V presents the testing and validation of the prototype tool. Finally, Chapter VI presents research conclusions and recommendations for future research.

II. Literature Review

2.1 Introduction

This literature review is a survey of current software engineering literature and research in areas of software engineering technology that can be applied to the Air Force C2 domain. In particular, Section 2.2 describes domain analysis and serves as the basis for defining the domain analysis approach used in this research project. Section 2.3 presents a brief overview of the object-oriented approach, which provided the framework for developing an object model of an Air Force wing.

2.2 Domain Analysis

Domain analysis is an important step in system development. In recent years, domain analysis has received considerable attention from the software engineering community in the form of research, conferences, workshops, and technical papers. One reason for the focus on domain analysis is the increasing reliance on software by today's computer systems and the fact that software design, development, and maintenance is becoming a more complex job. This need has led researchers to find better ways to engineer modern software systems.

2.2.1 Definition of Domain Analysis. Examining the work of researchers produces a variety of definitions for domain analysis. Neighbors first introduced the concept in 1980 as "the activity of identifying objects and operations of a class of similar systems in a particular problem domain" (13). Similarly, Ogush and Prieto-Díaz define domain analysis as "the process by which information used in software systems is identified, captured and organized with the purpose of making it reusable when creating new systems"

(14, 16). Further, Lowry describes domain analysis as "a form of knowledge acquisition in which concepts and goals of an application domain are analyzed and then formalized in an application oriented language suitable for expressing software specifications" (9:648).

The common theme of these definitions is that domain analysis is a way to capture the knowledge about key objects, operations, and relationships relevant to a particular domain. Most often, this knowledge is maintained in a domain model which is an abstraction and encapsulation of domain information into a predefined knowledge structure.

2.2.2 Approaches to Domain Analysis. With definitions of domain analysis in hand, the next step is to understand the various approaches to applying domain analysis. Prieto-Díaz, in his paper *Domain Analysis for Reusability*, examined different domain analysis approaches. The results of his analysis are shown in Figure 3.

Raytheon	CAMP	McCain	Arango
Identify commom functions Group by classes Organize into library Analyze business system structures Identify common structures Abstract structures Build objects	Identify similar systems Decompose functionally Abstract functionally Define interfaces Encapsulate results	Define reusable entities Define resuable abstractions Classify abstractions Encapsulate results	Bound domain Collect examples Identify abstractions Formalize abstractions Classify abstractions Encapsulate results

Figure 3. Domain Analysis Methods Compared by Prieto-Díaz (22:8)

Using common elements from these various approaches Prieto-Díaz developed the following domain analysis process, which is also represented in Figure 4:

1. Pre-Domain Analysis
   (a) Define the Domain
   (b) Scope the Domain

(c) Identify Sources of Domain Knowledge

(d) Define Domain Analysis Goals and Guidelines

2. Domain Analysis

(a) Identify Objects and Operations

(b) Abstract the Objects and Operations

(c) Classify the Abstracted Objects and Operations

3. Post-Domain Analysis

(a) Encapsulate the Classified Objects and Operations

(b) Produce Reusability Guidelines (17)

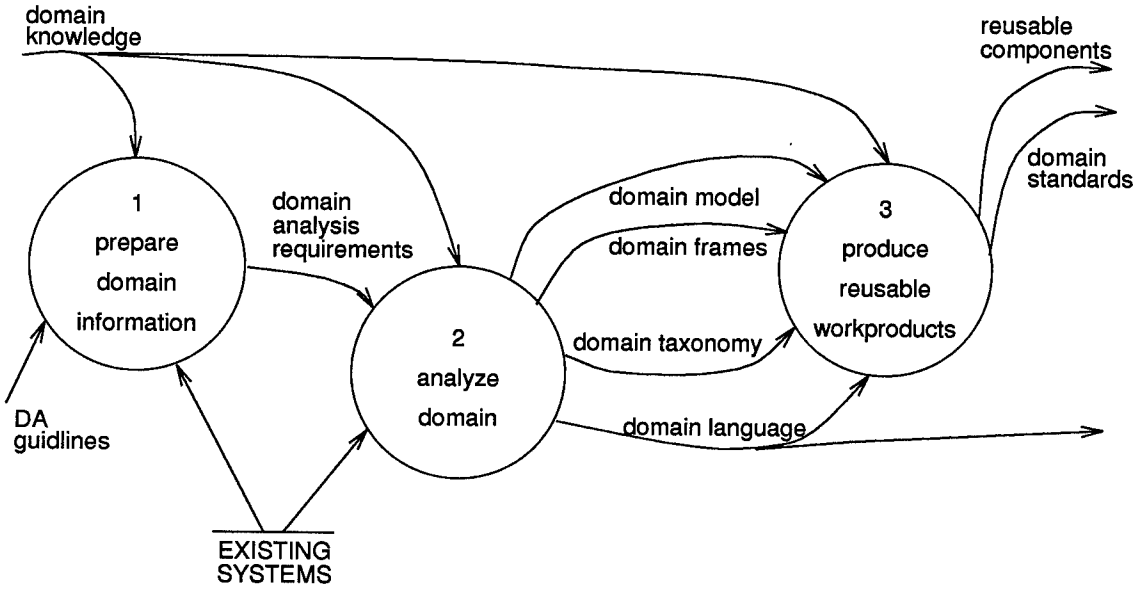


Figure 4. Domain Analysis Approach Proposed Prieto-Díaz (17:67)

The central activity of Prieto-Díaz's approach to domain analysis (Analyze Domain) is shown in Figure 5. In his approach, the domain analyst — with the assistance of domain experts — identifies and documents key pieces of domain knowledge. Sources of domain knowledge, for instance, include technical experts, documentation, requirements, and similar systems. The domain analyst organizes the collected domain knowledge into a usable form like a domain model, consisting of a domain taxonomy and a domain language that can be used as a specification language to construct new systems (17).

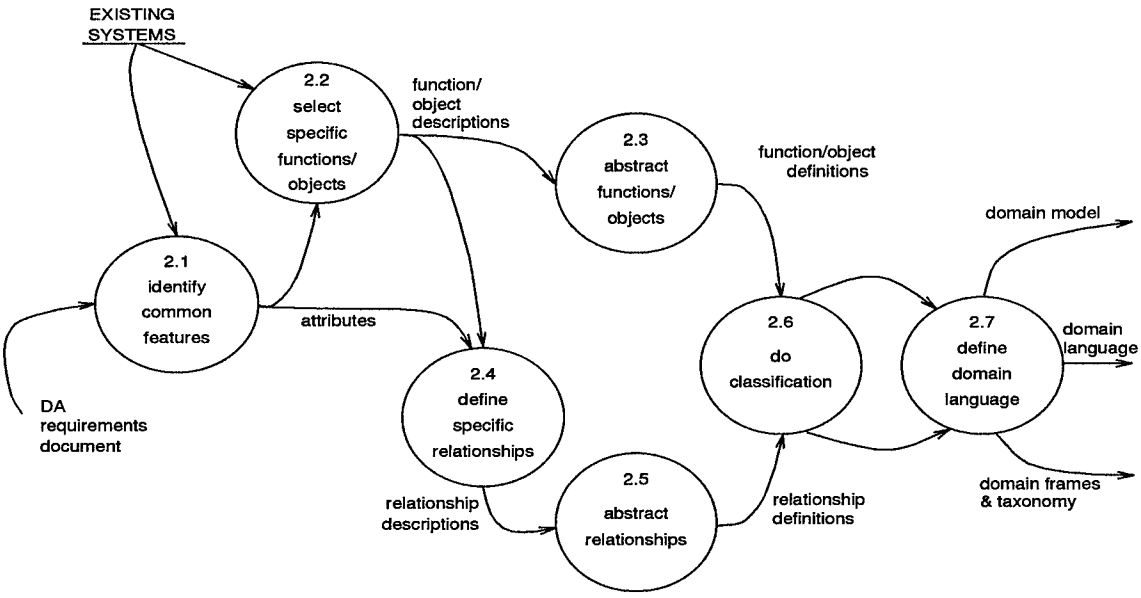


Figure 5. Prieto-Díaz's Concept to Analyze Domains (17:67)

McCain's approach to domain analysis consists of two stages. The first stage is an *application domain analysis*, which identifies the components to be implemented and their associated classification. This stage consists of the following tasks: define reusable entities, define reusable abstractions, and perform classification of reusable abstractions. The second stage is a *component domain analysis*, which is used to develop the component implementation requirements. This stage consists of the following tasks: define abstract interface specification, perform constraint analysis to minimize abstraction constraints, define applicable algorithms, and define customization requirements (10). McCain's approach is different from other approaches presented here because of his explicit inclusion of the component analysis stage (21).

Simos of the Domain Analysis Working Group proposes the following nine step domain analysis process:

1. Define domain analysis
2. Identify and scope the domain

3. Select a representative set of systems to study
4. Gather inputs for the domain analysis
5. Perform feature analysis at the requirements level
6. Analyze separability, selectability, and trade-offs of features
7. Select an implementation technology
8. Implement and validate products in phases
9. Deliver products of domain analysis (20)

Prieto-Díaz, at the same working group, proposed a somewhat different approach:

1. Acquire knowledge (define scope)
2. Conduct high-level functional analysis (top down)
3. Identify objects and operators (bottom up)
4. Define domain models (map objectives into architecture) (20)

The Working Group compared these two approaches with Wirfs-Brock's object-oriented design approach and Motta's knowledge elicitation process and found that all processes end up with a design model and are for the most part similar (20).

Iscoe, with a different approach to domain analysis, focuses on the product of domain analysis as opposed to the actual approach or its inputs. He suggests the problem is "to create a model for domain knowledge that is general enough to be instantiated in several domains" (6:299). His approach involves developing levels of "meta-models" that a domain analyst can use to capture the information of a particular application domain (6). Iscoe's approach to domain analysis is known as domain modeling (21).

2.3 Object-Oriented Approach and Modeling

An object-oriented modeling approach is based on the notion of an object. Objects consist of data structures (i.e., attributes) and behaviors (i.e., methods) in a single entity.

Object-oriented technology differs from conventional programming approaches where data structures and behaviors are only loosely connected. The main benefits of object-oriented analysis and design are that they promote software reuse, reduce the potential for errors, and simplify software maintenance (19). According to Pressman

> The unique nature of object-oriented design (OOD) lies in its ability to build upon three important software design concepts: abstraction, information hiding, and modularity. All design methods strive for software that exhibits these fundamental characteristics but only OOD provides a mechanism that enables the designer to achieve all three without complexity or compromise (15:334).

Other benefits of an object-oriented approach are the ability to better model a particular domain and to create models that include both physical and behavioral characteristics (2).

2.3.1 Definition of Terms. The following is a list of definitions for the major themes in object-oriented modeling.

- Abstraction – focusing on essential aspects of an object while disregarding levels of detail not pertinent to the particular problem. It also means focusing on what an object is and what it does without making premature design decisions (19:7).
- Encapsulation – Separating the external aspects of an object from internal implementation details of the object. This allows implementations of objects to be changed without affecting the applications that use it. Encapsulation is also referred to as information hiding (19:7).
- Object Model – Describes the static structure of objects and their associations/relationships with other objects. Object models are usually represented in a graphical form like an object diagram (19:6).
- Object Diagram – a formal graphical notation for modeling objects, object classes, and their relationships to one another (19:23).
- Object Class – A description of a group of objects with similar properties (attributes), common behavior (operations), common relationships to other objects, and common semantics (19:22).
- Instance – A single instantiation of an object. Each instance has its own attribute values and relationships to instances of other objects (1).

- Attribute – A data value (not an object) held by the objects in a class (e.g., *name, age,* and *weight* are attributes of *Person* objects, and *color, weight,* and *model-year* are attributes of *Car* objects) (19:23).
- Association – A physical or conceptual relationship between object classes. For example, a person *works-for* a company (19:27).
- Multiplicity – Specifies how many instances of one class may relate to each instance of another class in an association (See Figure 6) (19:30).
- Aggregation – The "*part-whole*" or "*a-part-of*" relationship in which objects representing components of something are associated with an object representing the entire assembly (19:36). This relationship is also referred to as an "*is-composed-of*" relationship (4).
- Inheritance – An abstraction for sharing similarities among classes while preserving their differences. It is the relationship between a class and one or more further refined versions of it. The class being refined is called the *superclass* and each refined version is called a *subclass.* Inheritance is sometimes called the "*is-a*" relationship because each instance of a subclass is an instance of the superclass as well. The most important use of inheritance is the conceptual simplification that comes from reducing the number of independent features in a system (19:38–43). For example, *Airplane* is the superclass of *F-16* and *B-52.* Each subclass *inherits* the features (attributes, operations, and relationships) of its superclass. For example, *B-52* inherits attributes *manufacturer, weight,* and *wing-span* from *Airplane* as well as the operations *takeoff* and *land* (1).

2.3.2 Object Model Notation. The object diagram notation selected for this thesis is Rumbaugh's Object Modeling Technique (OMT) (19). Figure 6 provides the subset of OMT notation used to develop the Air Force wing C2 object diagrams. Note that an object is represented as a rectangle containing the class name and optionally the attributes and/or operations defined for the class.

2.4 Summary

There is no single best approach to domain analysis or object-oriented modeling; each approach has its own merits. What is important is that the approaches to domain analysis and object-oriented modeling presented in this chapter provided the necessary

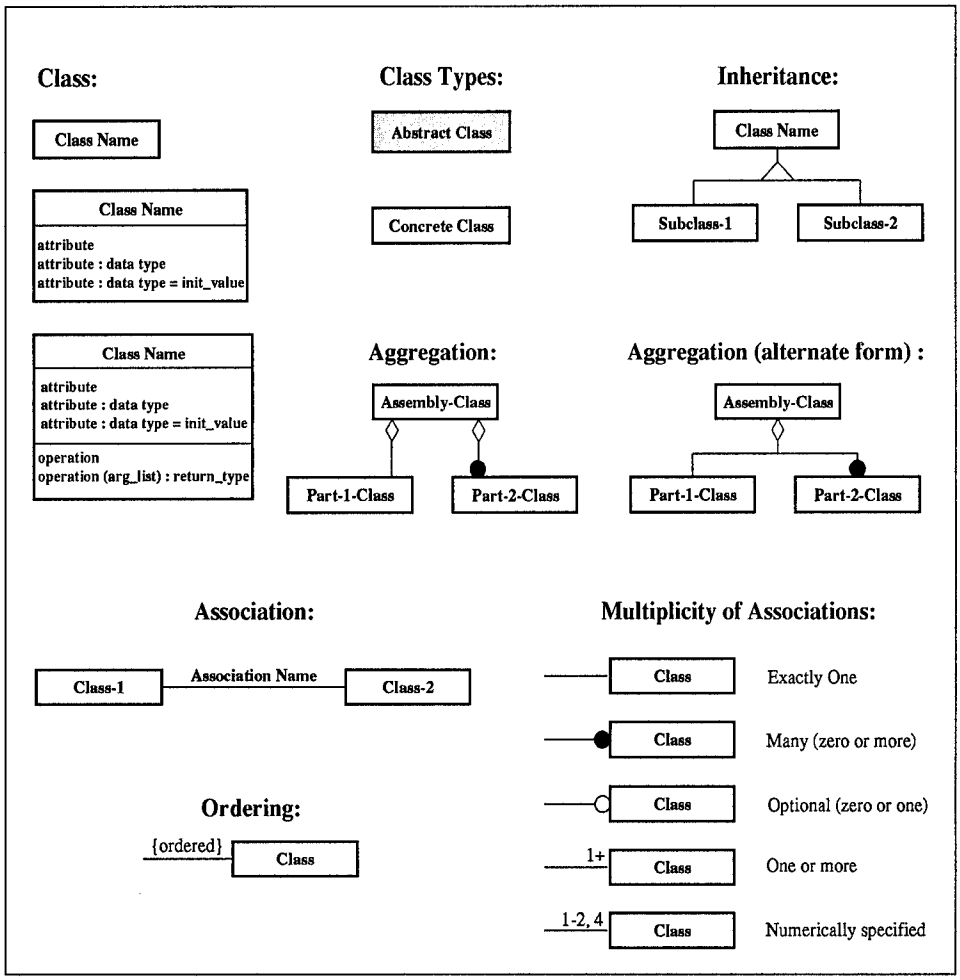


Figure 6. Object Modeling Notation (1:2–6)

foundation to begin the requirements analysis (i.e., domain analysis) of Air Force wing C2. This analysis is the topic of the next chapter.

III. Domain Analysis

3.1 Introduction

Chapter II characterized domain analysis as a way to capture the knowledge about key objects, operations, and relationships relevant to a particular domain. This chapter describes the process followed to conduct a domain analysis of Air Force wing C2, resulting in a domain model of Air Force wing C2 operations. As stated in Chapter II, there is no definitive approach to conducting domain analysis. In fact, even the same analysts, such as Prieto-Díaz, sometimes have different approaches. The approach chosen for this research effort combines techniques from the various approaches studied.

3.2 Domain Analysis of Air Force Wing C2 Operations

The approach used in this research combines ideas from Prieto-Díaz, Tracz, and others. The specific steps in this process are:

1. Define the Domain
2. Scope the Domain
3. Identify Sources of Domain Knowledge
4. Obtain Domain Knowledge
5. Choose Model Representation
6. Develop Domain Model

3.2.1 Define the Domain. The first step in performing domain analysis is defining and naming the domain of interest. Defining the domain can be accomplished by analyzing project requirements. One must understand the project goal before one can define the

domain of interest. It is important not to be too restrictive in defining the domain of interest. The next step will deal with scoping the domain defined here (17:67).

The domain of interest for this project has been defined by research requirements from the sponsor; the domain of interest is Air Force C2 activities. A typical Air Force wing is depicted in Figure 2. A top-level view of wing operations is depicted in Figure 7. These diagrams describe the initial domain of interest.

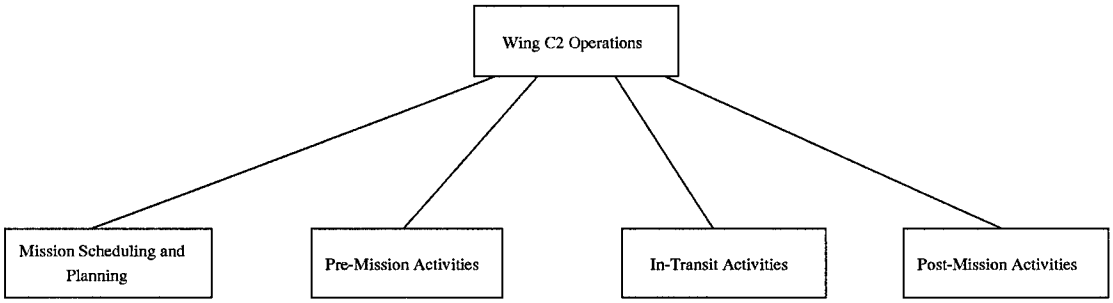


Figure 7. Breakdown of Wing Operations for ATO Processing

3.2.2 Scope the Domain. A domain can be defined at a great level of detail or it can be more abstract. In addition, it is often unclear where one domain ends and another begins. During this step, the domain analyst must place specific bounds on the domain of interest. The goals of domain analysis may place constraints on the domain and help the analyst find the proper scope of the domain (22:21).

For this research, the initial domain of wing C2 has been narrowed in several areas. The primary interest is in how the Air Tasking Order (ATO) affects the wing mission and what effects automation tools have on mission execution. Wings are typically tasked to do several missions, but this research is not concerned with those missions not related to the ATO. Also, as previously stated in Section 1.3 and illustrated by Figure 2, the domain

scope has been narrowed to just the Operations Group level, since the major portion of the wing mission tasked via the ATO is accomplished at this level. The other portions of a wing will either be modeled at a more abstract level or not at all.

Another scoping issue was that this research was to model the performance issues of the various tasks that comprise the processing of an ATO. Modeling or simulating the actual tasks in processing an ATO was outside the scope of this research.

3.2.3 Identify Sources of Domain Knowledge. The domain analyst is not always an expert in the domain at hand. In such a case, the analyst must find sources from which to obtain the knowledge necessary to reason about the subject domain. These resources may include domain experts, domain documentation, or previous domain research (17:67).

For this research, assistance has been provided by domain experts from HQ PACAF, the sponsor, as well as from the 89th Fighter Squadron (AFRES) at Wright-Patterson AFB, OH. Valuable domain information has also been obtained from many pieces of documentation, including (11), (12), and (18).

3.2.4 Obtain Domain Knowledge. Once sources of domain knowledge have been identified, one must start the process of collecting and analyzing domain knowledge. Whether from interviewing domain experts or studying domain documentation, the analyst must record the key information obtained about the domain and organize this information in a meaningful, logical way (22:23).

For this research, initial domain knowledge was obtained by studying the research proposal, which was provided by the research sponsor. Correspondence and meetings

with the sponsor helped clarify issues. After receiving and studying many pieces of C4I documentation, meetings were held with another domain expert. This was an iterative process and continued until adequate knowledge of the domain was obtained.

3.2.5 Choose Model Representation. The domain analysis process will result in a domain model. Therefore, it is important to choose a model representation up front. Modeling techniques range from informal to formal representations and can be object or functionally oriented. Models can be merely paper studies or can be fully automated.

An object-oriented approach for modeling the wing C2 domain was used for this research. The approach selected was based on Rumbaugh's OMT (19) and is briefly described in Section 2.3. The domain model developed from this analysis contains two components: the object model and the functional model. The development of a separate dynamic model as part of the overall domain model was not necessary because the event and state information pertinent to the wing C2 domain was incorporated into the functional model.

The object model is used to represent the static structure of the domain. In the C2 domain, for example, the object model describes how the wing and its components are organized as well as identifying the associations or relationships between these entities. The functional model, on the other hand, describes the processes, operations, or functions the objects from the object diagram perform. In the case of the Air Force C2 domain, the functional model shows the various steps in processing an ATO and the inputs, outputs, and data stores relevant to each task. From the functional model of the C2 domain an analyst can also understand the precedences and dependencies among the various tasks, providing insight into the issue of sequential vs. concurrent tasks.

3.2.6 Develop the Domain Model. As described in the previous section, a two part domain model was developed for this research. The following sections discuss the object model development and functional model development in greater detail.

3.2.6.1 Object Model. In object model development, the analyst looks for specific entities in the scoped domain and defines them as classes of objects. **Aircraft** is an example of an object class. The use of the term object is often ambiguous in the literature. For this research, unless otherwise noted, the term object will mean object class.

The analyst must give a description of the object, as well as define the attributes an object possesses. For example, the object **Aircraft** has attributes such as **Type**, **Tail Number**, and **Fuel Capacity**. The analyst must also determine the interrelationships or associations between objects. For example, **Maintains** is an association between a **Maintenance Squadron** class and an **Aircraft** class. The analyst must also define operations that are applied to or performed by the objects defined in the previous step. For example, **Refuel** is an operation for the **Aircraft** object. With knowledge of the objects, attributes, associations, and operations, the analyst can develop the object model.

For this research, object model development began with analysis of (11), (12), (7), (18), and (8). In particular, process diagrams and context diagrams were reviewed, noting all inputs, outputs, and controls. Additionally, the data dictionaries in (11) and (12) helped to further refine the attributes of the initial set of objects. Studying the documentation and interviewing the domain experts helped define the relationships or associations between various objects. Operations for each of the objects identified in the previous steps were defined by studying (11), (12), and (7), and the functions listed in (7) were mapped to the

objects performing the function. Similarly, processes from the context diagrams of (11) and (12) were mapped to specific objects.

The preliminary object model was drafted based on the knowledge and understanding of the objects, attributes, associations, and operations. Refinement of the object model was an iterative process, as with most analysis and design projects. Feedback from the research sponsor and domain experts was key to finalizing the object model.

Figures 8, 9, 10, and 11 are the results of the object model development and are discussed below. The reader is referred to the Data Dictionary in Appendix A for more detailed definitions of the objects shown in these figures.

Figure 8 shows the major object classes of the C2 operations domain — **C2 Task** and **Wing/Wing Operations Center (WOC)** — and how they fit together. The **ATO Task** and **Non-ATO Task** are specializations of the **C2 Task**. The reason for this distinction is because this research is only concerned with the tasks required to process an ATO (see Section 3.2.2). The **Wing/Wing Operations Center (WOC)** is composed of a **Battle Staff**, **Mission Planning Cell (MPC)**, and **SRC Cell**, which are composed of **Wing Personnel**. The **tasked** association indicates that a **Wing/Wing Operations Center (WOC)** can be assigned multiple **C2 Tasks** (i.e., ATO). The attributes of the **tasked** association, Time To Complete and Overall Accuracy, are defined as follows:

1. Time To Complete - A measure of how long it takes to complete processing of an ATO.
2. Overall Accuracy - A measure of how error-free the products of the ATO are.

As its name indicates, the association **assigned** shows that **Wing Personnel** can be assigned multiple **C2 Tasks**. The attributes of the association **assigned** — Time and Accuracy and — are defined as follows:

1. Time - A measure of how long it takes to complete a task.
2. Accuracy - A measure of how error-free the product of a task is.

The attributes for the **tasked** and **assigned** associations were selected after detailed analysis of the C2 domain and represent the key measures for quantifying **C2 Task** performance. The values of these attributes are derived from the characteristics of their associative objects.

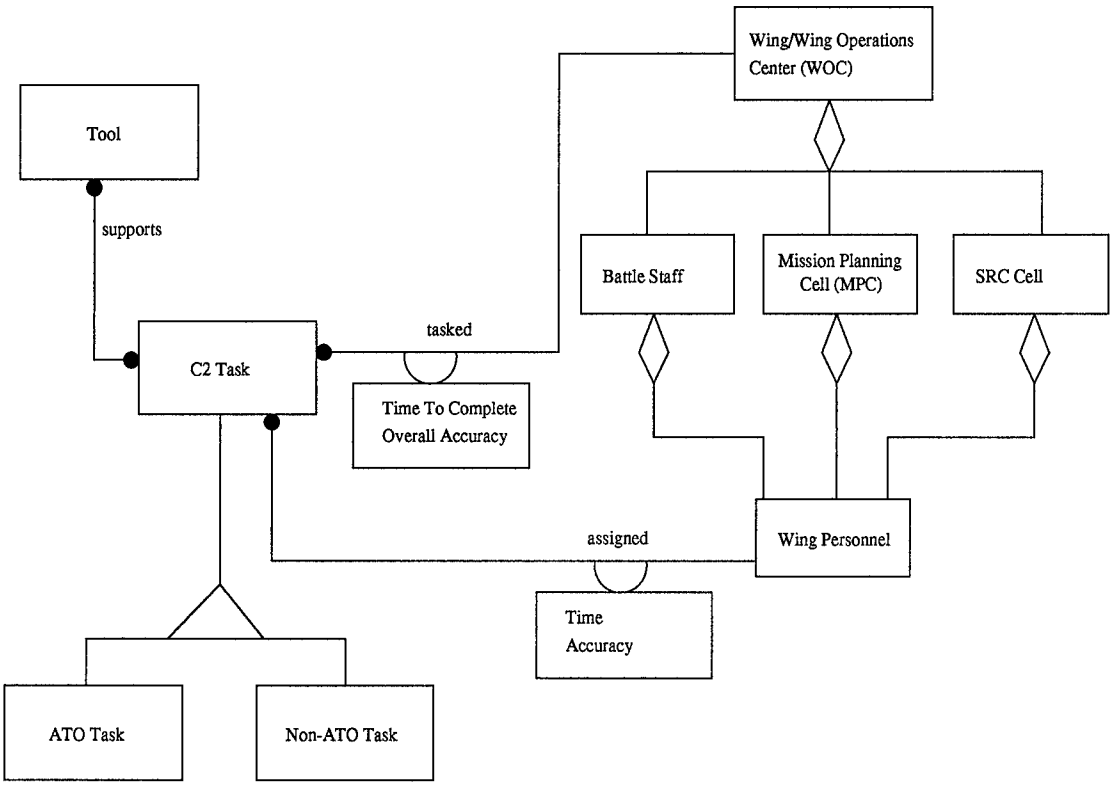


Figure 8. Top-Level Perspective of Object Model

Figure 9 shows the major sub-tasks that are specializations of an **ATO Task** and the **Mission Scheduling and Planning** task. Along with the hierarchy of tasks are the common attributes for all ATO tasks. The attributes were defined by abstracting out the key characteristics of the **ATO Task** class. These attributes are at the heart of understanding the details of the **ATO Task** class.

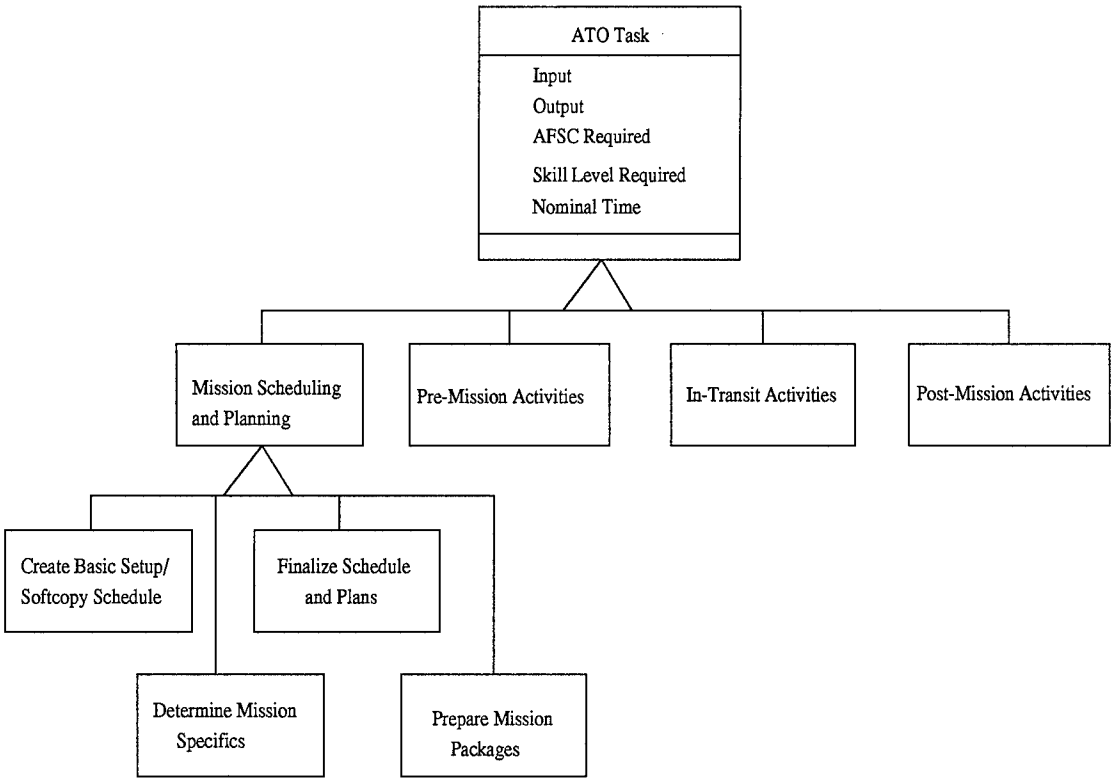


Figure 9. Task Portion of Object Model

Figures 10 and 11 are both object models of an Air Force wing but are developed from different points of view. First, Figure 10 shows the wing from an organizational perspective, more like an organizational chart, that is typical of peacetime operations. It shows that the wing is composed of groups, which are in turn composed of squadrons. Figure 11, on the other hand, shows how the wing is organized for its war-fighting mis-

sion. The people from the various organizational components join forces to staff a WOC — the hub for conducting all wing-level airborne operations. For example, Figure 10 indicates that **Weather**, **Weapons**, and **Intelligence** personnel are assigned to the **Support Squadron** of the **Operations Group** for peacetime activities. However, when the wing prepares for combat operations, these people become part of the **Mission Planning Cell** and focus on supporting wing-level activities as opposed to squadron-level activities.

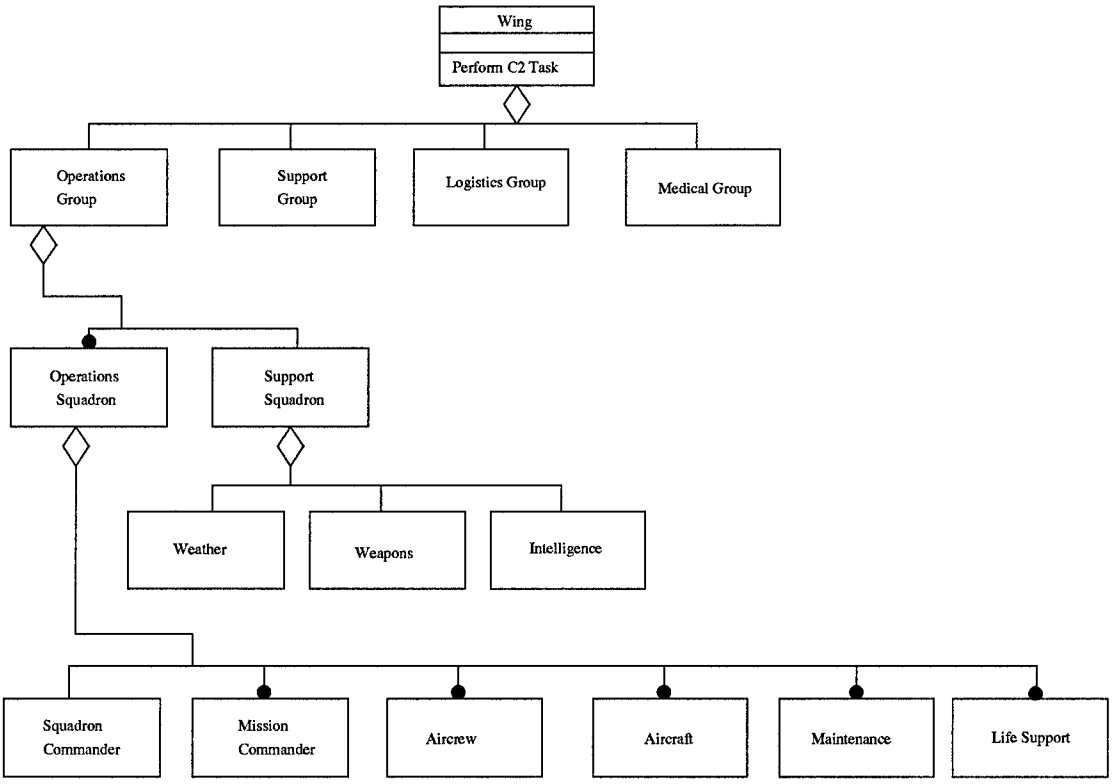


Figure 10. Wing Organization — Peacetime Perspective

3.2.6.2 Functional Model. This phase of the domain analysis process was devoted to documenting the behavior associated with Air Force wing C2 operations. Specifically, this research focused on the tasks associated with how a wing processes the ATO and who performs those tasks. The object model is an important piece of the domain

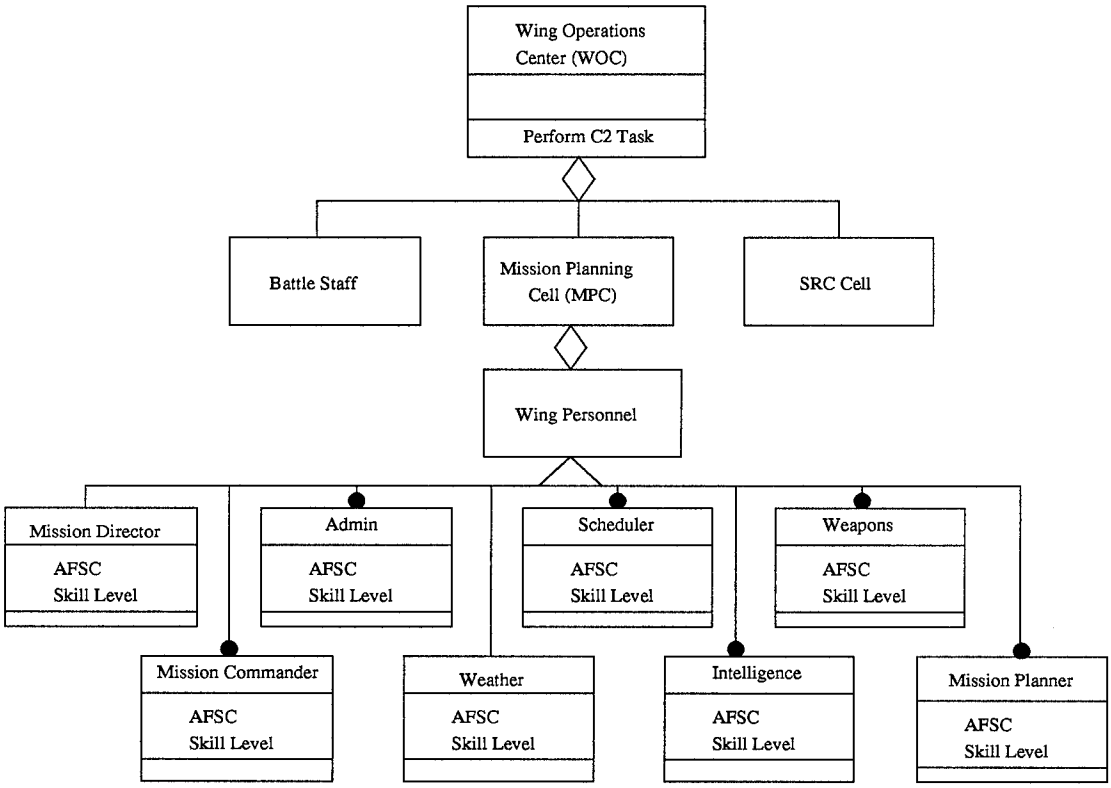


Figure 11. Wing Organization — Wartime Perspective

model because it represents who the players or "doers" are. However, without behaviors for the wing C2 objects the domain model would be inadequate for system development or any form of decision making support.

A major portion of the effort in this phase was the abstraction of various sources of mission-specific wing information (i.e., F-15 and F-16) into a more generic model of Air Force wing C2 operations. In the case of the F-15 and F-16 wings, the basic structure of tasks was the same (see Figure 7). However, further decomposition yielded numerous differences that needed to be resolved. For example, in the case of the F-16 wing, **Determine Mission Specifics** consists of the following tasks (12):

1. Obtain Intelligence
2. Obtain Weather Information
3. Analyze Threat to Mission
4. Coordinate With EW
5. Plan Flight Path/Tactics
    (a) Plot Route and Threats
    (b) Assess Air Support Assigned
    (c) Coordinate with Support Wings
    (d) Plan Route (Initial Point to Target Run)
    (e) Determine Fusing Delays

For the F-15 wing, **Determine Mission Specifics** consists of these tasks (11):

1. Determine Weapons Package
2. Coordinate with Support Wings
3. Determine Tactics
4. Obtain Weather Report

After discussions with the research sponsor and domain experts, the slight variations between the F-15 and F-16 wings were abstracted out to produce a generic model of **Determine Mission Specifics** (see Figure 15). This type of discrepancy (i.e., semantic differences, different interpretations, etc.) between various sources of domain information is quite common, and abstracting out the unnecessary details to produce a common, potentially reusable model is at the heart of domain analysis.

Another major goal in this phase of the analysis was to document the dependencies and precedences among the various tasks. The dependencies and precedences were based on the inputs and outputs of the individual tasks. This led to an understanding of required sequentiality or potential concurrency of the tasks in the scenario.

Figures 12, 13, 14, 15, and 16 depict the behavior associated with Air Force wing C2 operations and make up the functional model. The individual figures are discussed below. The reader is referred to the Data Dictionary in Appendix A for more detailed definitions of the data flows used in these figures.

Figure 12 shows wing C2 operations from the top level. The inputs are the **ATO FRAG** (which is the portion of the ATO that pertains directly to an individual wing) and **Airspace Information**, which includes intelligence information and weather information. The **ATO** and **Airspace Information** are modeled as data stores because they are passive (i.e., they respond to requests to access data) and generate no operations on their own. The **Wing** provides the resources (i.e., manpower, facilities, and equipment) to conduct C2 operations. The **Air Operations Center (AOC)** is responsible for all air operations

within the theater or campaign. As such, the **AOC** has multiple **Wings** it can task to perform specific portions of the **ATO**.

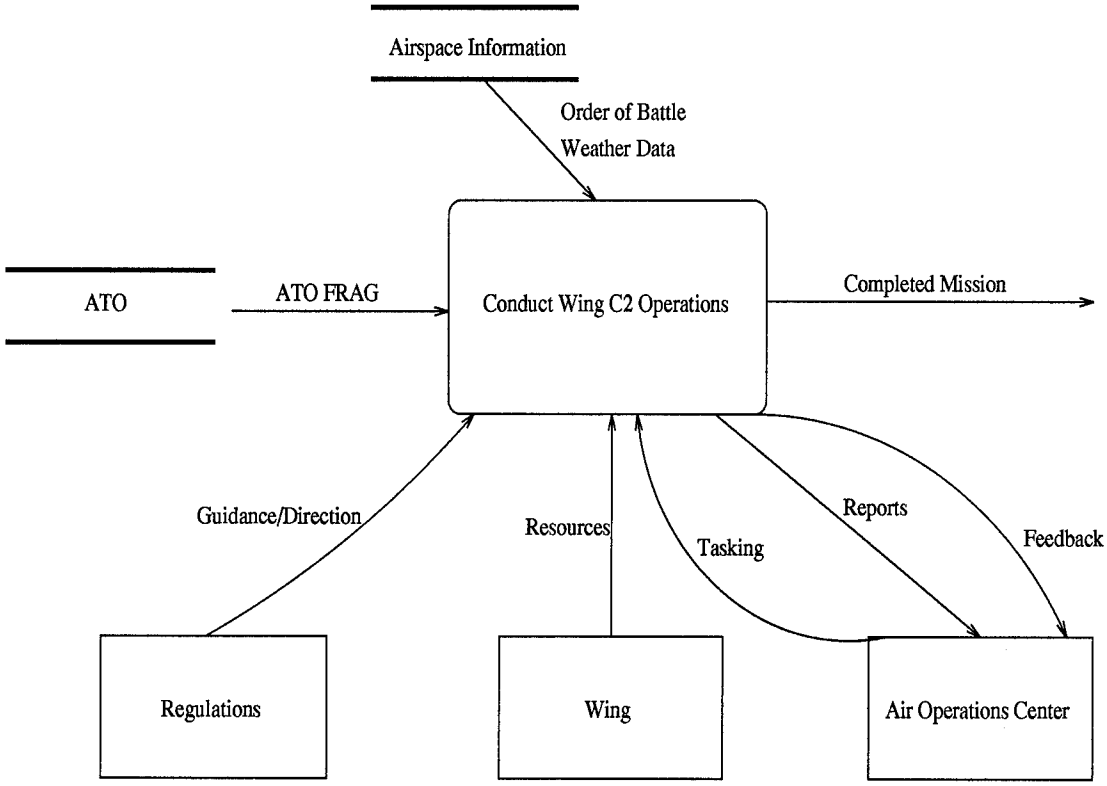


Figure 12. Functional Model — Air Force Wing C2 Operations

Figure 13 shows how **Conduct Wing C2 Operations** from the previous figure is broken down into the major elements required to process an ATO. **Prepare Mission Plans and Schedules** is the task where the majority of planning and scheduling takes place. This task is concerned with issues ranging from making sure the wing has the resources to complete the assigned tasking to planning/plotting the actual flight route and determining the tactics used to fly the mission. The outputs of this process are **Mission Packages**, which are packages of information containing every aspect of the mission from take off times to ingress/egress routes, and **Schedules**, which are detailed timelines of the

events to occur. **Conduct Pre-Mission Activities** is the activity of physically preparing the aircraft (i.e., loading munitions and fueling) and making sure the aircrews are fully briefed on the upcoming mission. The output of this process is a **Launched Mission. Manage Mission and Base Support** deals with all aspects of managing the mission once it's underway. For example, if friendly aircraft are shot down, a search-and-rescue mission must be launched. The output of this task is a **Recovered Mission** and occurs when the aircrews have returned to base. The final process, **Conduct Post-Mission Activities**, is concerned with debriefing the personnel involved in the mission and making sure the results of the mission are documented, reported to the AOC, and incorporated into future ATOs.

Through the functional model analysis, it was discovered that the majority of analysis, planning, and scheduling tasks reside in **Prepare Mission Plans and Schedules**. For this reason, it was decided that this research should concentrate here. The next paragraph discusses **Prepare Mission Plans and Schedules** in greater detail.

Figure 14 shows how **Prepare Mission Plans and Schedules** from Figure 13 is divided into more detailed sub-tasks. **Create Basic Setup/Softcopy Schedule** is the process that creates the initial mission schedule based on aircraft/aircrew availability and establishes the mission timeline. **Determine Mission Specifics** does exactly what its name implies. **Mission Specifics** include items like EW Guidance, Tactics, and Route Map, for example. **Finalize Schedules and Plans** transforms the **Softcopy Schedule** and **Mission Specifics** into the final **Schedules**. Minor adjustments to the **Softcopy Schedule** are made once the **Mission Specifics** have been determined. Another aspect of this process is the coordination of the final **Schedules** with the **AOC**. The final process

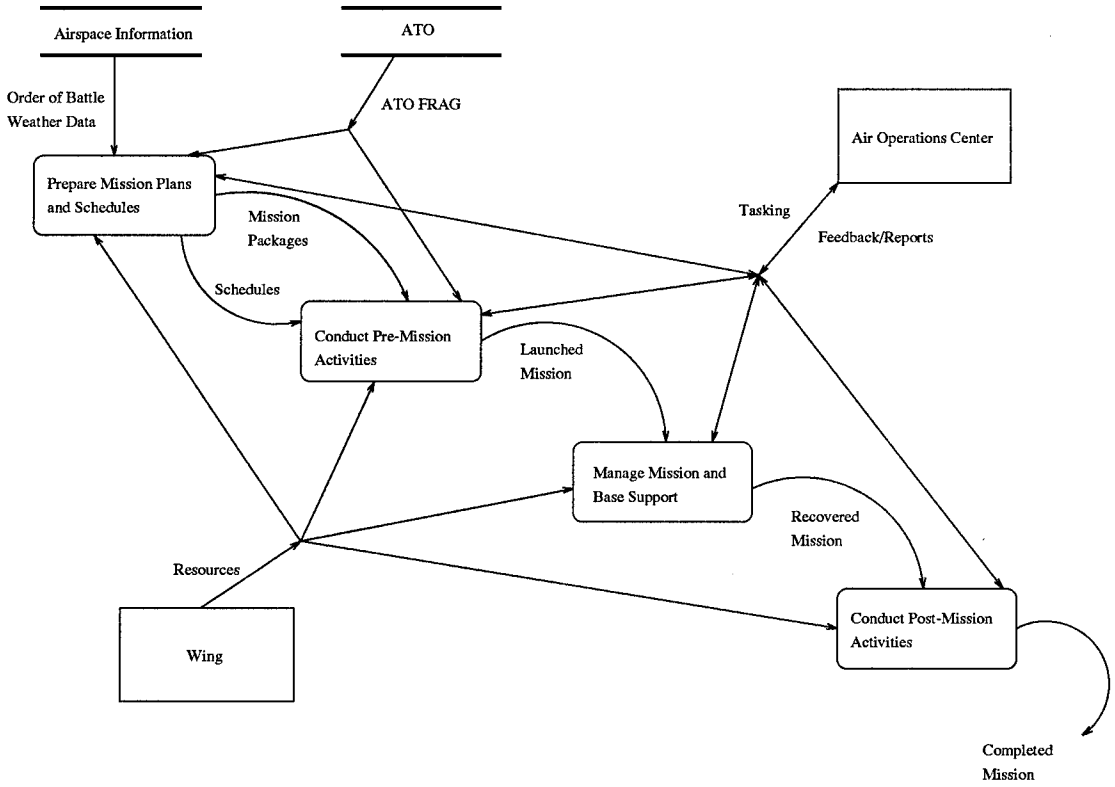


Figure 13. Functional Model — Conduct Wing C2 Operations

in Mission Scheduling and Planning is **Prepare Mission Packages** where the **Schedules** and **Mission Specifics** are used to build the **Mission Packages**. **Mission Packages** contain the information the pilot needs to fly the mission: maps, search-and-rescue information, communication frequencies, special instructions, code words, target imagery, altitudes, kill boxes, and the information for the flight computer, which is found in the Data Transfer Cartridge (DTC).

Similar to the reasons given above as to why this research focused on **Prepare Mission Plans and Schedule**, it was further discovered that the majority of analytical tasks reside in **Determine Mission Specifics**. For this reason, it was decided this research should further narrow in on this process. The next paragraph discusses **Determine Mission Specifics** in greater detail.

Figure 15 is a highly detailed view of **Determine Mission Specifics** and is the portion of the ATO process used to develop the prototype reasoning system, which is discussed in later chapters. The figure shows how the inputs are transformed into the various components that comprise Mission Specifics.

Figure 16 is a Activity on Node diagram of **Determine Mission Specifics** and shows the precedences and dependencies among the various sub-tasks more clearly than Figure 15.

3.3 Summary

This chapter has described the approach and results for the domain analysis of Air Force wing C2 operations. The domain model — consisting of the object model and

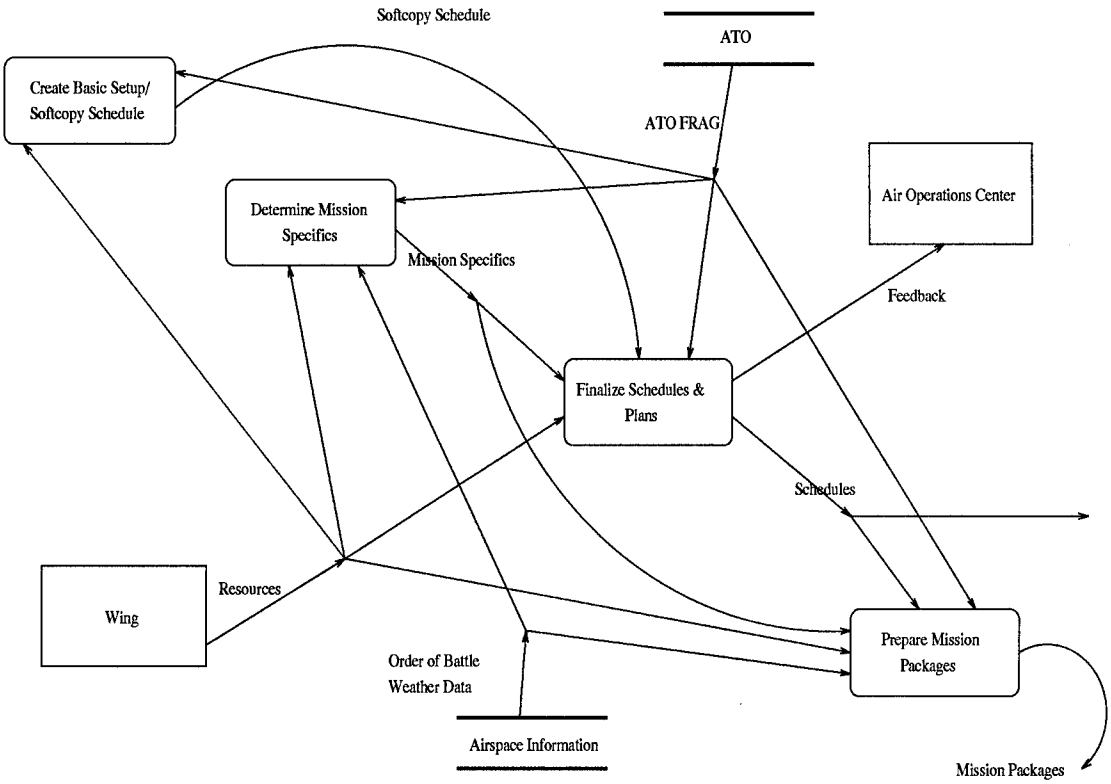


Figure 14. Functional Model — Prepare Mission Plans and Schedules

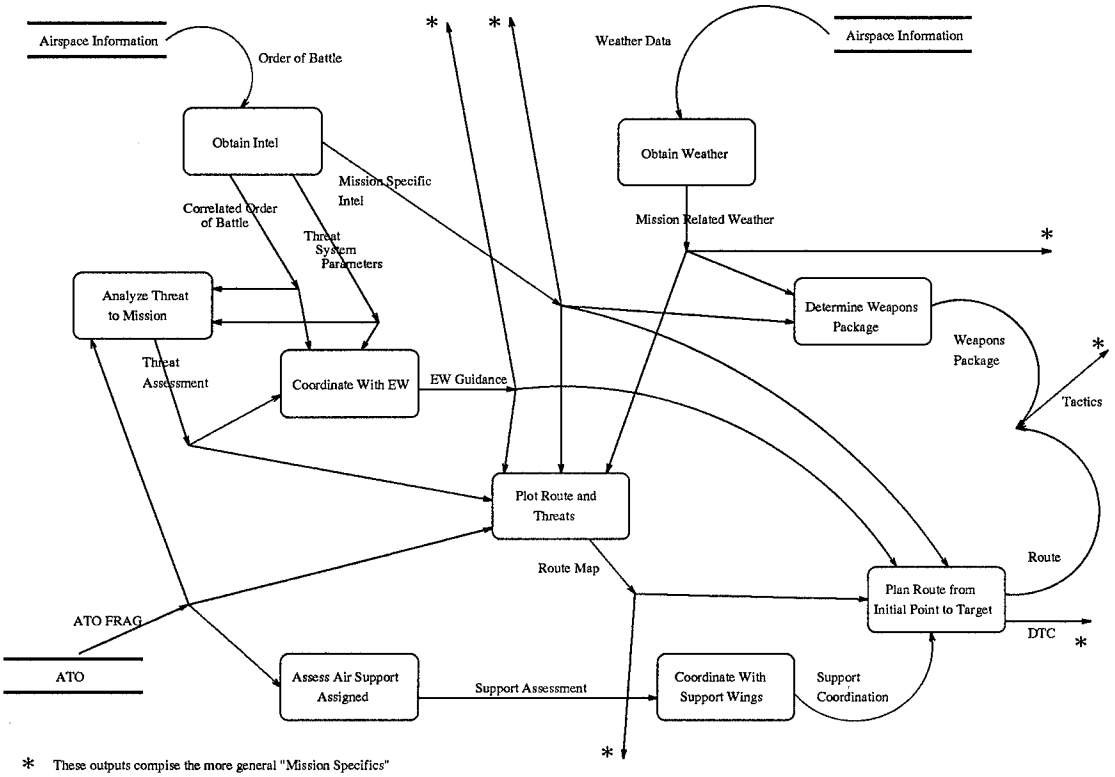


Figure 15. Functional Model — Determine Mission Specifics

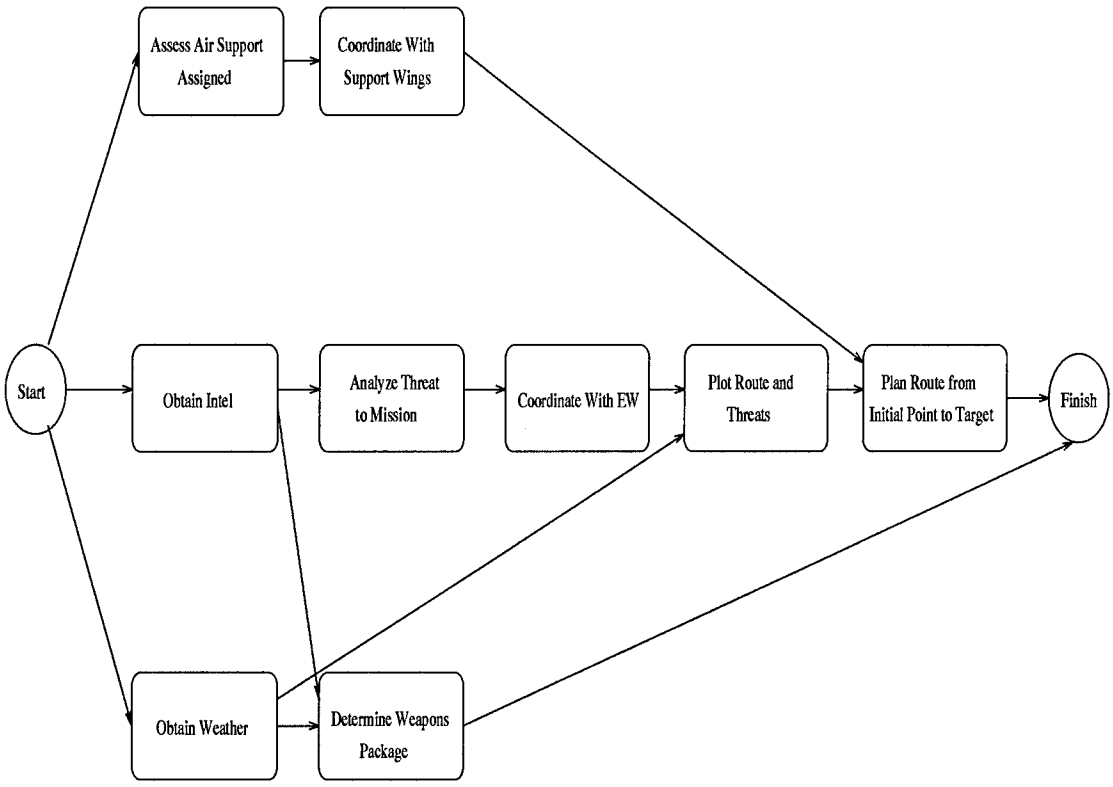


Figure 16. Dependency-Based Representation of Determine Mission Specifics

functional model — served as the knowledge base to analyze and design a prototype of the automated reasoning system, which is discussed in Chapter IV.

IV. Prototype Analysis and Design

4.1 Introduction

The domain model of Air Force wing C2 operations, discussed in Chapter III, along with sponsor research requirements provided the framework for developing the prototype software tool. As discussed in Section 1.2, the sponsor needed decision making support for resource allocation and determining the effects of automation. To that end, analysis of the prototype software tool yielded the object model shown in Figure 17.

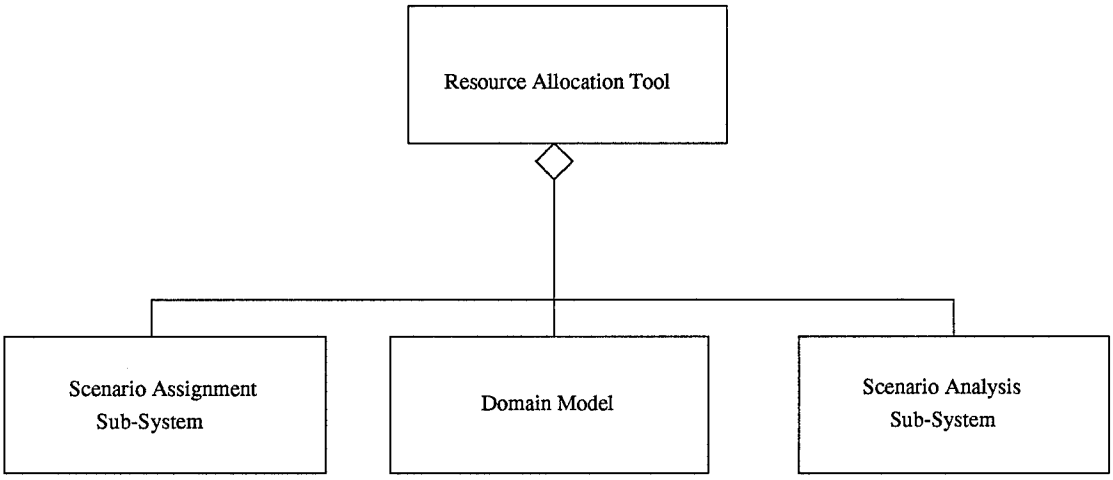


Figure 17. Object Model of Resource Allocation Tool

The Scenario Assignment Sub-System provides the wing user a means to assign personnel from the wing, as well as an automation tool, to the various tasks that comprise the ATO process, and to provide an assessment for each person/task pairing. The Scenario Analysis Sub-System, on the other hand, uses the person/task pairings and individual task assessments to produce an overall assessment and the necessary feedback to give the wing user insight into how well he allocated the resources and whether or not automation provided any value. Both sub-systems are based on the information from the domain

model. The design and analysis of the Scenario Assignment Sub-System is discussed in Hunt's thesis (5). The remainder of this chapter discusses the analysis and design of the Scenario Analysis Sub-System.

4.2 Analysis

The primary goal of this analysis was to determine the metrics necessary for the wing user to make a determination about his resource allocation and the impact of automation. This set of metrics would be the output of the Scenario Analysis Sub-System.

The first step was to determine what feedback measures were important to decision makers and necessary to determine how automation added value. With input from domain experts, the following broad categories of measures were determined to be necessary for the problem at hand:

1. Timing
2. Worker Efficiency
3. Tasks Required To Wait
4. Accuracy
5. Critical Path

The second step was to further decompose each of these broad categories into the individual metrics. This decomposition is discussed below.

4.2.1 Timing. The first area examined was timing. The wing user needs to know how long it would take to complete the scenario he created. This is important because processing the ATO, or a portion of it, might have to be finished by a particular point in

time. Not satisfying a time-to-complete requirement might indicate that the user needs to make more efficient person/task assignments or use additional automation tools to reduce the time to a point where the finish time requirement is met. Related to this time metric is knowing the total amount of work performed during the execution of the scenario. Because certain tasks can be performed concurrently, the total amount of work performed might be more than the time required to complete the scenario. In fact, this is the most desired case; wing personnel want to maximize the amount of work performed in the shortest amount of time (i.e., increase concurrency). This time metric tells the wing user how well he is using concurrency. For example, when the time to complete a scenario is twenty-five minutes and the total amount of work performed during that period is also twenty-five minutes, the wing user is not taking advantage of any concurrent tasks. Finally, the wing user needs to know what the shortest and longest tasks are in the scenario. A large difference between these two might tell the user that he may need to adjust the person/task pairings or use a different set of automation tools. For example, information about the shortest task might lead the wing user to conclude that he has assigned this task to someone who is overqualified or given that person an inappropriate automation tool. The wing user would then be able to assign this task to someone more evenly matched to the task and give that worker a more appropriate automation tool. The higher skilled person would now be available to perform a more difficult task. Similarly, the information about the longest task might lead the wing user to the conclusion that he has assigned this task to someone who is underqualified or given that worker an ineffective automation tool.

4.2.2 Worker Efficiency. Another important metric for the wing user is the efficiency of the people performing the tasks in the scenario. This metric is the ratio of the total amount of work each person performed to the time it took to complete the scenario. For example, a person who performed fifty minutes of work during a scenario that lasted 200 minutes would have an efficiency of 25%. This information tells the wing user how well or how poorly he has distributed the work.

4.2.3 Tasks Required to Wait. Making good use of concurrent tasks is one way a wing user can increase the efficiency of the wing. However, to make concurrency work the wing user needs to know where the bottlenecks are in the scenario so he can find ways to avoid them. This is where knowing how many times a task was ready to begin but the person was unavailable can help. For example, if during the execution of a scenario a person had multiple tasks that had to wait because this person was busy performing other tasks, the wing user could conclude that he has assigned too many tasks to this person and that he needs to redistribute the work to make better use of the concurrency within the scenario.

4.2.4 Accuracy. Accuracy (or quality) of the product is also a metric the wing user needs. The user needs to know what the least and most accurate tasks are as well as the overall accuracy of the entire scenario. This is one area where automation tools can have a big effect. For example, a task with a low accuracy might be because the assigned person was underqualified or had an inappropriate automation tool. At this point the user would be in the position to assign a more robust tool to that person and possibly increase

the accuracy of that task. The overall accuracy is important to give the user a feel of how the accuracies of the individual tasks affect the accuracy or quality of the entire scenario.

4.2.5 Critical Path. The final metric necessary for the user to properly assess his assignments is data about the critical path (or longest path). Critical path information tells the user what people are performing tasks that drive the time required to complete a scenario. From this the user knows that if he is to reduce the amount of time to complete the scenario, he must reduce the time spent performing at least one of the tasks along the critical path. Thus, wing user analysis can focus on the people performing tasks along the critical path.

This analysis led to the object model of metrics shown in Figure 18. In the next section, algorithms designed to compute these metrics are presented.

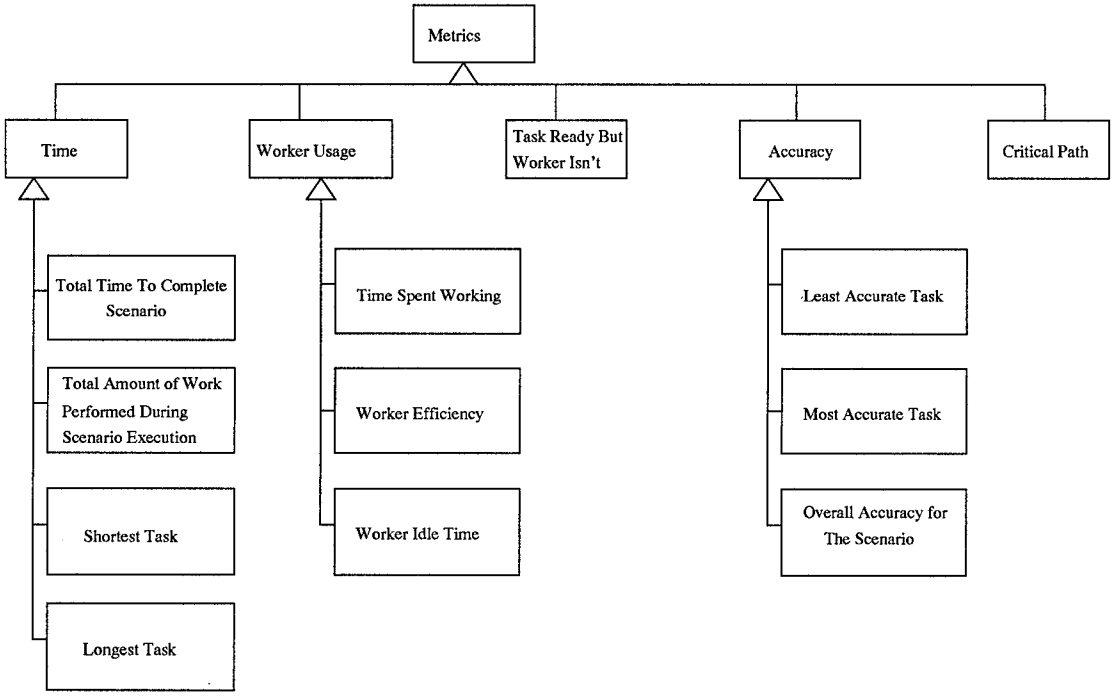


Figure 18. Object Model of Metrics

4.3 Design

The primary goal of the prototype design phase was to develop an approach to calculating the metrics described in the previous section. The individual algorithms described below are later transformed into the methods of the Scenario Analysis Sub-System object.

4.3.1 Interface To The Scenario Assignment Sub-System. Before presenting the design, it is important to describe the interface between the Scenario Assignment Sub-System and the Scenario Analysis Sub-System. This interface is specified as follows and defines the input data to the Scenario Analysis Sub-System:

1. A set of wing personnel. Each person has an AFSC and skill level.
2. A set of tasks that make up the scenario. For each task there is a required AFSC, required skill level, set of predecessor tasks, automation tool (if applicable), and nominal time.
3. An assessment for each task/person pairing. The individual assessments include a measure for task accuracy and time needed to complete the task.

4.3.2 Approach to Calculating The Timing Metrics. Analysis of the timing measures discussed in the previous section yielded the following timing metrics, each of which are described in greater detail:

1. Shortest Task
2. Longest Task
3. Total Time to Complete Scenario
4. Total Amount of Work Performed During Execution of the Scenario

The approach to finding the Shortest Task is to search through the set of task/person pairings and find the task that completes in the shortest amount of time. The following is pseudocode for making this calculation:

```
Shortest_Task <- the first task
while not end of tasks loop
   if the Current_Task.Time < the Shortest_Task.Time then
      Shortest_Task <- Current_Task
   end if
   Current_Task <- Next_Task
end loop
```

Similarly, finding the Longest Task requires searching through the set of task/person pairings and finding the task that completes in the longest amount of time. The pseudocode for this operation is defined as follows:

```
Longest_Task <- the first task
while not end of tasks loop
   if the Current_Task.Time > the Longest_Task.Time then
      Longest_Task <- Current_Task
   end if
   Current_Task <- Next_Task
end loop
```

The approach for calculating the Total Time To Complete The Scenario is to trace through execution of the scenario keeping track of the order of individual task execution and the associated times. The algorithm for computing this metric is defined as follows:

```
Scenario_Time <- 0
while not Scenario_Complete(Tasks) loop
   loop from 1 .. Num_Of_Task
      if Predecessors_Complete(Current_Task) and
         Worker_Available(Selected_Worker)   and
         Tool_Available(Selected_Tool)       and
         Current_Task.Status /= Complete     then
               Current_Task <- In_Progress
               Selected_Worker.Available <- false
               Selected_Tool.Available <- false
      end if
      Current_Task <- Next_Task
   end loop
```

```
   Scenario_Time <- Scenario_Time + Shortest_Task.Time
   Shortest_Task <- Complete
   Selected_Worker.Available <- true
   Selected_Tool.Available <- true

   loop from 1 .. Num_Of_Tasks
      if Current_Task.Status = In_Progress then
         Current_Task.Time = Current_Task.Time - Shortest_Task.Time
      end if
      Current_Task <- Next_Task
   end loop
end loop
```

The approach to computing the Total Amount of Work Performed During Execution of the Scenario is to accumulate the times for each individual task. Pseudocode for calculating this metric is defined as follows:

```
Total_Amount_Of_Work <- 0
loop from 1 .. Num_Of_Tasks
   Total_Amount_Of_Work <- Total_Amount_Of_Work + Current_Task.Time
   Current_Task <- Next_Task
end loop
```

4.3.3 Approach to Calculating Worker Efficiency. Analysis of the Worker Efficiency measures discussed in the Section 4.2 yielded the following efficiency-related metrics, each of which is described in further detail:

1. Time Spent Working
2. Worker Efficiency
3. Idle Time

Time Spent Working is calculated for each worker by accumulating the individual times he spent performing tasks. Worker Efficiency is the ratio between his Time Spent Working and the amount of time required to complete the scenario. Worker Idle Time is simply the difference between 100% and Worker Efficiency expressed as a percent. The pseudocode for these calculations is described below. This pseudocode assumes the time required to complete scenario, called Scenario_Time below, has already been computed:

```
loop from 1 .. Num_Of_Tasks
   Current_Task.Selected_Worker.Busy_Time <-
              Current_Task.Selected_Worker.Busy_Time + Current_Task.Time
   Current_Task <- Next_Task
end loop

loop from 1 .. Num_Of_Workers
   Current_Worker.Efficiency <- Current_Worker.Busy_Time/Scenario_Time
   Current_Worker.Idle_Time <- 1.0 * Current_Worker.Efficiency
   Current_Worker <- Next_Worker
end loop
```

4.3.4 Approach to Finding How Many Tasks Had to Wait For a Worker. Analysis of the Waiting Task measure discussed in the Section 4.2 led to the following pseudocode:

```
while not Scenario_Complete(Tasks) loop
   loop from 1 .. Num_Of_Task
      if Predecessors_Complete(Current_Task) and
         Worker_Available(Selected_Worker)   and
         Tool_Available(Selected_Tool)       and
         Current_Task.Status /= Complete     then
               Current_Task <- In_Progress
               Selected_Worker.Available <- false
               Selected_Tool.Available <- false
      elsif not Worker_Available(Selected_Worker) then
         Selected_Worker.Num_Waits <- Selected_Worker.Num_Waits + 1
      end if
      Current_Task <- Next_Task
   end loop

   Shortest_Task <- Complete
   Selected_Doer.Available <- true
   Selected_Tool.Available <- true

end loop
```

4.3.5 Approach to Calculating the Accuracy Metrics. Analysis of the accuracy measures discussed in the Section 4.2 yielded the following accuracy-related metrics, each of which will be described in further detail:

1. Least Accurate Task
2. Most Accurate Task
3. Overall Accuracy for Scenario

The approach to calculating the Least Accurate Task is to search through the set of task/person pairings and find the task that has the lowest individual accuracy measure. The following is pseudocode for making this calculation:

```
Least_Accurate_Task <- the first task
while not end of tasks loop
   if the Current_Task.Accuracy < the Least_Accurate_Task.Accuracy then
      Least_Accurate_Task <- Current_Task
   end if
   Current_Task <- Next_Task
end loop
```

Similarly, determining the Most Accurate Task requires searching through the set of task/person pairings and finding the task that has the highest individual accuracy measure. The pseudocode for this operation is defined as follows:

```
Most_Accurate_Task <- the first task
while not end of tasks loop
   if the Current_Task.Time > the Most_Accurate_Task.Time then
      Most_Accurate_Task <- Current_Task
   end if
   Current_Task <- Next_Task
end loop
```

The Overall Accuracy of the Scenario is computed by finding the average accuracy of the scenario and is defined by the following pseudocode:

```
Total_Accuracy <- 0
loop from 1 .. Num_Of_Tasks
   Total_Accuracy <- Total_Accuracy + Current_Task.Accuracy
   Current_Task <- Next_Task
end loop
Overall_Accuracy <- Total_Accuracy/Num_Of_Tasks
```

4.3.6 Approach to Calculating the Critical Path. Analysis of the Critical Path measure discussed in the Section 4.2 led to the following pseudocode:

```
Scenario_Time <- 0
while not Scenario_Complete(Tasks) loop
   loop from 1 .. Num_Of_Task
```

```
      if Predecessors_Complete(Current_Task) and
         Worker_Available(Selected_Worker)   and
         Tool_Available(Selected_Tool)       and
         Current_Task.Status /= Complete     then
               Current_Task <- In_Progress
               Selected_Worker.Available <- false
               Selected_Tool.Available <- false
      end if
      Current_Task <- Next_Task
   end loop

   Selected_Worker.CP_Contribution <- Selected_Worker.CP_Contribution + Shortest_Task.Time

   Shortest_Task <- Complete
   Selected_Worker.Available <- true
   Selected_Tool.Available <- true

   loop from 1 .. Num_Of_Tasks
      if Current_Task.Status = In_Progress then
         Current_Task.Time = Current_Task.Time - Shortest_Task.Time
      end if
      Current_Task <- Next_Task
   end loop
end loop
```

4.4 Summary

This chapter has described the assessments to be made by the Scenario Analysis Sub-System and an approach to calculating those assessments. The next chapter will discuss how the Scenario Analysis Sub-System was implemented in software, tested, and validated.

V. Prototype Testing and Validation

5.1 Introduction

This chapter begins by briefly describing the implementation of the Scenario Analysis Sub-System. It then describes how the Scenario Analysis Sub-System was tested, the results of that testing, and an analysis of the results.

5.2 Implementation

The pseudocode for calculating the metrics presented in Chapter IV was transformed into the following Ada procedures:

1. Find_Shortest_Task
2. Find_Longest_Task
3. Find_Least_Accurate_Task
4. Find_Most_Accurate_Task
5. Compute_Scenario_Accuracy
6. Compute_Total_Work
7. Compute_Scenario_Time
8. Compute_Worker_Efficiency
9. Compute_Num_Waiting_Tasks
10. Compute_Critical_Path

These procedures were placed in a single Ada package, and Figure 19 shows how this package is connected to the other components of the resource allocation tool. Details on how to run the Resource Allocation tool are contained in Appendix B, Configuration Management.

The following is a sample output run from the Scenario Analysis Sub-System where workers were selected but given no automation tools:

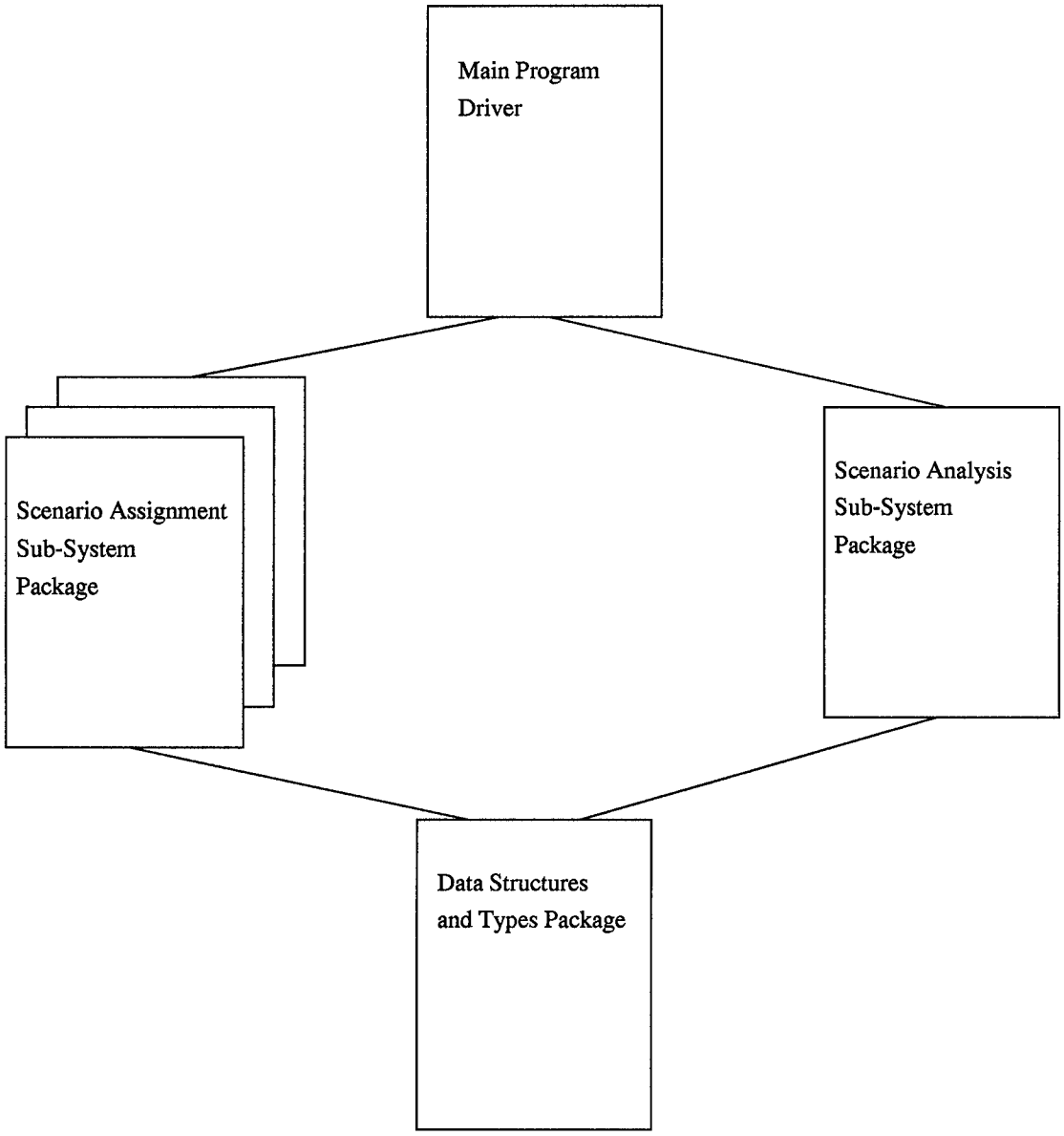


Figure 19. Package Organization of The Resource Allocation Tool

```
THE SHORTEST TASK:  Plan Route      ; VALUE =   8
THE LONGEST TASK:  Assess Air Sup ; VALUE = 143
THE TOTAL AMOUNT OF WORK PERFORMED WITH THIS SCENARIO = 486
THE TOTAL TIME REQUIRED FOR THIS SCENARIO = 215


THE LEAST ACCURATE TASK:  Assess Air Sup ; VALUE =   1
THE MOST ACCURATE TASK:  Anlze Thrt-Msn ; VALUE =   4
THE OVERALL ACCURACY FOR THIS SCENARIO =  2


WORKER ANALYSIS:
                      AMOUNT OF        WORKER BUSY
WORKER              WORK PERFORMED  (WORKER EFFICIENCY)    WORKER IDLE
---------------     --------------  ------------------    -----------
Msn Planner 1            174              80.9%              19.1%
Msn Planner 2             45              20.9%              79.1%
F-16 Pilot 1             132              61.4%              38.6%
F-16 Pilot 2              15               7.0%              93.0%
F-16 Pilot 3             120              55.8%              44.2%

NUMBER OF FAILED ATTEMPTS TO BEGIN A TASK WHEN THE WORKER WASN'T AVAILABLE:
Msn Planner 1       15
Msn Planner 2        2
F-16 Pilot 1         3
F-16 Pilot 2         0
F-16 Pilot 3         3


CRITICAL PATH ANALYSIS:
                        TIME CONTRIBUTED      PERCENTAGE OF
WORKER                TOWARDS CRITICAL PATH   CRITICAL PATH
---------------       ---------------------   -------------
Msn Planner 1                  42                 19.5%
Msn Planner 2                  11                  5.1%
F-16 Pilot 1                   95                 44.2%
F-16 Pilot 2                   15                  7.0%
F-16 Pilot 3                   52                 24.2%
```

The following is a sample output run from the Scenario Analysis Sub-System where the same set of workers were selected but given automation tools, if they existed:

```
THE SHORTEST TASK:  Det Weapons Pkg; VALUE =   5
```

```
THE LONGEST TASK:  Obtain Weather ; VALUE =  60
THE TOTAL AMOUNT OF WORK PERFORMED WITH THIS SCENARIO = 267
THE TOTAL TIME REQUIRED FOR THIS SCENARIO = 131


THE LEAST ACCURATE TASK:  Assess Air Sup ; VALUE =   2
THE MOST ACCURATE TASK:  Obtain Weather ; VALUE =   4
THE OVERALL ACCURACY FOR THIS SCENARIO =  3


WORKER ANALYSIS:
                      AMOUNT OF        WORKER BUSY
WORKER              WORK PERFORMED  (WORKER EFFICIENCY)    WORKER IDLE
---------------     --------------  ------------------    -----------
Msn Planner 1             79              60.3%              39.7%
Msn Planner 2             15              11.5%              88.5%
F-16 Pilot 1              88              67.2%              32.8%
F-16 Pilot 2               5               3.8%              96.2%
F-16 Pilot 3              80              61.1%              38.9%

NUMBER OF FAILED ATTEMPTS TO BEGIN A TASK WHEN THE WORKER WASN'T AVAILABLE:
Msn Planner 1       10
Msn Planner 2        0
F-16 Pilot 1         5
F-16 Pilot 2         0
F-16 Pilot 3         5


CRITICAL PATH ANALYSIS:
                        TIME CONTRIBUTED      PERCENTAGE OF
WORKER                TOWARDS CRITICAL PATH   CRITICAL PATH
---------------       ---------------------   -------------
Msn Planner 1                 17                  13.0%
Msn Planner 2                 15                  11.5%
F-16 Pilot 1                  62                  47.3%
F-16 Pilot 2                   5                   3.8%
F-16 Pilot 3                  32                  24.4%
```

Ada was the chosen implementation language because of the researcher's familiararity with it and because Ada's notions of packages, encapsulation, and information hiding were well suited for rapid prototyping of resource allocation tool.

5.3 Methodology

Testing of the Scenario Analysis Sub-System was divided into two phases. First, testing was performed to ensure the output produced by the Sub-System was correct for the input it was given (verification). Second, the Scenario Analysis Sub-System was tested to determine if it provided the necessary information for a user to make an assessment about automation upon the ATO process (validation). Both phases are discussed below.

5.3.1 Verification. The objective for verification testing was to demonstrate that the Sub-System performed correctly. To satisfy this objective a variety of test cases were prepared:

- All tasks were assigned to a single person.
- Each task was assigned to an available person (i.e., starting a task never had to wait on a worker to become available).
- Concurrent tasks were assigned to a single worker.

5.3.2 Validation. The objective of validation was to demonstrate that the Scenario Analysis Sub-System provided the metrics necessary to determine the impacts of automation, which was also one of the research objectives (see Chapter I). Complete validation of the Scenario Analysis Sub-System should include feedback from domain experts. However, due to a lack of access to domain experts validation was limited to trial runs and feedback from fellow researchers (see Section 6.4). Fellow researchers were asked to review a series of output runs (one output run showed the use of no automation tools. The second output run showed that same scenario with the incorporation of automation tools) and identify where automation had an effect, positive or negative.

5.4 Results

The results from the Scenario Analysis Sub-System verification testing were compared to manual computations. In all cases the two sets of computations were the same.

From the Scenario Analysis Sub-System validation phase, the responses from the graduate student testers were tallied and compared. The consensus was that output from

the Scenario Analysis Sub-System provided the necessary information to determine if automation played a role in processing the ATO.

5.5 Analysis of Results

The results from the verification testing were conclusive; the Sub-System was operating as expected. One reason this phase of testing went as smoothly as it did was because of the thorough object-oriented domain analysis work done up front.

The results for the validation testing were also conclusive, given the level of testing possible. The output from the Scenario Analysis Sub-System provided the user with the information he needed to determine if automation had an effect on processing the ATO.

5.6 Summary

This chapter has described how the Scenario Analysis Sub-System was implemented and tested. In the next chapter, the overall research results are provided and recommendations for future research are offered.

VI. Conclusions and Recommendations

6.1 Introduction

This chapter provides a summary of research accomplishments. It reviews the original problem statement and research objectives presented in Chapter I which is followed by a discussion of the research results. It is then shown that the results of this research can be mapped to another problem domain with known solution techniques. Finally, recommendations for future research are offered.

6.2 Results

The primary objectives of this research were the following (see Section 1.2.2):

- Show how domain analysis could be used to develop engineering models of wing C2 operations.
- Show how an engineering model of wing C2 could be used to develop a software tool for determining the effects of automation.

First, this research has successfully shown how domain analysis can be applied to an Air Force domain of interest and used to develop a domain model (i.e., engineering model) that captures information about the key objects, operations, and relationships relevant to wing C2 operations.

Second, this research has successfully shown how an engineering model of wing C2 operations can be used to develop a prototype software tool that provides insight into the effects of automation on wing C2. The prototype software tool gives the user an abstract model of wing C2 operations that can be used to analyze different ways to process an ATO. The user has the ability to try out different configurations of workers, tools, and tasks. In time, the user will select a configuration best suited for the particular constraints at hand. Throughout that process, he is able to quantitatively see where automated tools have value and where they do not.

6.3 Comparison With Scheduling Theory

The approach followed in this research was to apply software engineering domain analysis techniques to wing C2 operations. However, the implementation of the prototype software tool developed from this research has also shown how sequencing and scheduling theory can be applied to wing C2. In particular, the prototype software tool is an application of the classic job-shop scheduling problem. Job-shop scheduling problems are concerned with processing jobs through a set of machines or processors (3). In the case of wing C2 the jobs are the tasks necessary to process an ATO, and the machines or processors are the various wing personnel concerned with completing an ATO.

Like job-shop scheduling, ATO processing is concerned with performance measurement and many of the performance measures for job-shop scheduling apply directly to the processing of an ATO. French lists the following set of metrics as general performance measures (3), several of which were implemented in the prototype software tool:

- due date
- waiting time
- total waiting time
- completion time
- flow time
- lateness
- earliness

The bottom line is that processing an ATO is really concerned with the effective and efficient assignment of resources, which is the same as one objective of job-shop scheduling. Thus, the problem of assigning resources to process an ATO is best categorized as a job-shop scheduling type of problem, and properly classifying a problem is the most important — and often the hardest — part of solving a particular problem. While the current prototype tool primarily performs an analysis of a specified schedule, scheduling theory also addresses algorithms for creating an optimal schedule.

6.4 *Recommendations for Future Research*

The following are recommended enhancements for the Scenario Analysis Sub-System and merit additional research:

1. *Prototype Validation.* Before making any enhancements to the tool or determining the direction for future research, the current prototype software tool should be validated to determine what value the tool adds to wing C2 operations. The recommended approach to validation is to give the tool to domain experts to use, then conduct interviews with them to get user feedback.
2. *Investigate the Application of Job-Shop Scheduling Techniques.* Since the wing C2 problem maps directly to a scheduling problem, many of the potential improvements to the Scenario Analysis Sub-System (such as, generating the optimal set of assignments or calculating the optimal time to complete the scenario) relate directly to the field of job-shop scheduling. Further research into the application of scheduling theory should be pursued.
3. *Investigate The Use of Additional Metrics.* Currently, complexity/difficulty is not used to provide the overall assessment picture to the user. Further research might discover that complexity/difficulty is an attribute of a task or that it might be an attribute of a task/worker/tool assignment. Also, researching the area of effectiveness and efficiency might provide valuable information for the user. Another metric that might be of value to the user deals with the shortest and longest tasks. Here, the user might find it useful to know the difference between the nominal time and the shortest/longest tasks.

6.5 *Summary*

This research successfully developed an object-oriented domain model of wing C2 operations and a prototype software tool to help wing decision makers assess the impacts of automation on processing an ATO. These two products have demonstrated that software engineering can provide benefit to the wing C2 domain. Furthermore, the prototype tool has been shown to be an application of job-shop scheduling theory, which opens up a whole new approach to solving wing C2 problems. These results form the foundation for future software engineering projects to support wing C2 operations.

Appendix A. Data Dictionary

This is the Data Dictionary for the Air Force wing C2 domain model from Chapter III. It is a subset of the Data Dictionaries found in (11) and (12).

1. **Accuracy** is an attribute of the Assigned association and is a numeric ranking between 1 and 5 (1 is low and 5 is high) that indicates the accuracy (or quality) of the products from the particular task.
2. **Admin** is a general reference to a class of people who perform administrative tasks in support of mission execution.
3. **Aircraft** is the plane found in the wing, and for this research is an F-15 or F-16.
4. **Aircrew** is a pilot, and depending upon the aircraft, a pilot and weapons officer.
5. **Air Force Specialty Code (AFSC)** is an alphanumeric job specialty code assigned to Air Force personnel.
6. **AFSC Required** is an attribute of an ATO Task and is the job specialty best suited to perform the task.
7. **Air Operations Center (AOC)** is the theater level air operations management facility. The AOC monitors all air activity within the theater and is responsible for generation and dissemination of the daily ATO.
8. **Airspace Information** is a reference to a broad category of information dealing with the air combat environment.
9. **Air Tasking Order (ATO)** directs all planned airspace activity for the day. Missions are identified by wing and include mission number, name, and type, number of aircraft, time on target, and flying altitude among other things. The ATO is issued daily by the AOC.
10. **Analyze Threat to Mission** is an activity that involves comparing and analyzing correlated OB and threat system parameters with mission information to determine which threats will affect the mission. The threat assessment is used to plan the flight path and tactics. In addition, Intel produces a picture of the area threat environment that is primarily used for briefing purposes.
11. **Assigned** is the association (or relationship) between C2 Task and Wing/Wing Operations Center.
12. **Assess Air Support Assigned** is an activity where planners must identify the other wings that are to provide support to the mission and coordinate with them to ensure a smooth operation. If insufficient support is given to the wing, planners may call the AOC to request more support.
13. **ATO FRAG** is the portion of the ATO that specifically corresponds to the wing and the special instructions that accompany it. The ATO FRAG is needed before detailed mission planning can begin.

14. **ATO Task** is a category of C2 Task that is required to plan and execute an ATO.
15. **Battle Staff** is the senior staff function within the WOC.
16. **C2 Task** is a general category of tasks that refers to tasks that are necessary to perform wing operations.
17. **Conduct Pre-Mission Activities** is a sub-task of Conduct Wing C2 Operations and refers to conducting the pre-mission activities (i.e., preparing the aircraft, preparing the aircrews, etc.).
18. **Conduct Post-Mission Activities** is a sub-task of Conduct Wing C2 Operations and refers to conducting the post-mission activities (i.e., regenerating the aircraft, debriefing the aircrews, etc.).
19. **Conduct Wing C2 Operations** is process that encompasses all activity associated with command and control of fighter wing operations in a sustained deployed wartime environment.
20. **Coordinate With EW** is an activity where electronic warfare (EW) coordinates with mission planners to ensure that enemy threat systems have been appropriately considered during planning of the mission. In addition, EW provides guidance to personnel responsibility for carrying out the action. Examples of guidance include tactics, chaff and flare use, and loading/reprogramming directives for the electronic countermeasures.
21. **Coordinate With Support Wings** is an activity where planners must coordinate rendezvous point and lead times with other wings involved in the mission to ensure efficient mission execution.
22. **Create Basic Setup/Softcopy Schedule** is an activity where the schedulers build the softcopy schedule. The softcopy schedule must be approved by the wing commander, operations group commander, or a representative. At any time the softcopy schedule can be updated to reflect changes related to the mission.
23. **Correlated Order of Battle** is a correlation of multi-source threat data into a single comprehensive Order of Battle (OB).
24. **Determine Mission Specifics** is an activity where more detailed planning must be done, after general information has been extracted from the ATO. Further mission details include weather an threat information, communications and navigation requirements, tactics, fusing, and support wing coordination. Many experts on these topics are assigned to the MPC to help plan the missions.
25. **Determine Weapons Package** is an activity where the weapons officer determines which bombs to put on the targets and the type of fuse to be used. Target size/type, munitions availability, and changing weather and intelligence have an impact on which weapons are selected.
26. **Data Transfer Cartridge (DTC)** is a plug-in module that is used to program the flight computer with flight path information.

27. **EW Guidance** is guidance that is given on electronic warfare issues used in preparations of mission planning packages, aircrews, and aircraft.

28. **Feedback** refers to periodic updates of how the the mission is going.

29. **Finalize Schedule and Plans** is an activity where the softcopy schedule is modified (if necessary) and finalized after the mission specifics have been determined. The schedule will be maintained at the squadron. Later, squadron maintenance personnel will assign aircraft to the missions. The wing commander or his representative often check the schedule before it becomes final.

30. **Guidance/Direction** refers to the guidance and direction that Regulations provide to the Conduct Wing C2 Operations task.

31. **Input** is an attribute of an ATO Task and refers to the pieces of information required to begin the task.

32. **Intelligence** is a general reference to a class of people who perform intelligence-related tasks.

33. **Intel/Targeteer** is a person within the MPC who analyzes intel information and provides assessments to the mission planners in the MPC.

34. **In-Transit Activities** are activities associated with managing an on-going mission. Mission monitoring, reporting base and threat status, briefing higher authorities, and setting up alternate WOC facilities, and processing transient aircraft (if required) are examples. Many of these activities are the responsibility of the Mission Director, maintenance, disaster preparedness, and the battle staff who serve as facilitators.

35. **Launched Mission** is the actual departure of mission aircraft.

36. **Life Support** is a general reference to a class of people that provide life support to the aircrews.

37. **Logistics Group** is a group within the wing that is responsible for the logistics of sustaining a wing: equipment maintenance, supply, transportation, and logistics.

38. **Maintenance** is a general reference to a class of people within the wing that perform maintenance tasks on aircraft or equipment.

39. **Manage Mission and Base Support** is a sub-task of Conduct Wing C2 Operations and refers to managing the on-going mission and associated base support functions.

40. **Medical Group** is a group within the wing that provides medical services for wing personnel. In general, the Medical Group plays little role in planning and executing the ATO.

41. **Mission Commander** is the pilot in charge of a particular mission or strike package.

42. **Mission Director** is the senior person normally in charge of the MPC and a senior member of the WOC staff.

43. **Mission Packages** are folders containing all the information a pilot needs to fly the mission including: line-up card, maps with threat rings, search and rescue information, communications frequencies, special instructions, weapons and airplane checklists, code words, altitudes, ground spots, target imagery, kill boxes, and an Army information book for close air support missions.

44. **Mission Planner** is a person who performs mission planning tasks.

45. **Mission Planning Cell (MPC)** is the cell in the WOC responsible for all planning and scheduling activities.

46. **Mission Related Weather** is current weather conditions that affect the mission.

47. **Mission Scheduling and Planning** is an activity for preparing the schedules and plans for the next day's air operations. Planners match the taskings with available resources to meet mission requirements. This is accomplished by personnel in the MPC which can operate up to twenty-four hours per day. In the MPC, representatives from the squadrons including scheduling, intelligence, maintenance, munitions, electronic combat, weather, and command and control work together to determine the safest and most efficient route to accomplish the mission. The end result of this activity is the mission package and associated schedules for all missions to be flown.

48. **Mission Specific Intel** is information including SAM sites, enemy force location, imagery, minimum risk routing, search and rescue info, electronic order of battle, outside threats, threat airfields, threat CAPs, target type, makeup, location, and protection, and ingress/egress procedures that directly impact the mission.

49. **Mission Specifics** is a broad scope term including a variety of information not contained in the ATO (i.e., intel, weather, tactics, etc.) but necessary to perform the mission.

50. **Nominal Time** is an attribute of an ATO Task and represents the time it would take to complete the task for a person with an AFSC and skill level matching AFSC Required and Skill Level Required.

51. **Non-ATO Task** is a category of C2 Task that does not pertain to execution of the ATO.

52. **Obtain Intel** is an activity where mission planners use intel to provide them with the latest information concerning location and capability of enemy forces to plan the safest routes for mission completion. Intelligence includes SAM sites, enemy force locations, enemy ground and air order of battle, minimum risk routing, imagery, electronic order of battle, outside threats, threat airfields, threat CAPS, and egress procedures.

53. **Obtain Weather** is an activity where the weather representative assigned to the MPC coordinates with the Weather Shop and gets the weather information needed to plan and schedule the mission.

54. **Order of Battle (OB)** is intelligence information that reflects the battlefield situation. Information is collected from several sources including debriefings, intelligence systems, and force-level intelligence.

55. **Operations Group** is the Group within the wing that is responsible for all aircraft operations.

56. **Operations Squadron** is the operational part of the Operations Group. There are generally two Operations Squadrons per Operations Group.

57. **Output** is an attribute on an ATO Task and refers to the products of that task.

58. **Perform C2 Task** is an operation for the Wing and Wing Operations Center.

59. **Plan Route from Initial Point to Target** is the activity where the specific details of the route must be planned for. Mission planners determine way points, fuel requirements, take off times, location of allied and enemy ground forces, and target approach. The initial point (IP) is the last point on the route before the approach to the target. The IP to target run specifies the load delivery tactic and sequence, altitude, ingress and egress paths, countermeasure employment, and target and IP imagery if possible. In addition, planners consider tactics options so more than one tactic can be employed for multiple target passes.

60. **Plot Route and Threats** is an activity where planners are responsible for plotting the routes and threats on maps for aircrews to follow. Planners use the latest weather, intelligence and terrain information, threat information, rules of engagement, and guidance from the AOC to plot the routes.

61. **Post-Mission Activities** are those activities occurring from the time the aircraft return from the mission to the time the aircraft are re-generated and readied for another mission.

62. **Pre-Mission Activities** are those activities that are required to ensure the mission is properly prepared. This refers primarily to making sure the aircrews are rested, briefed, and outfitted. Also included is making sure the aircraft is properly fueled, maintained, and loaded to execute the mission.

63. **Prepare Mission Plans and Schedules** is a sub-task of Conduct Wing C2 Operations and refers to preparing the mission plan and associated schedules.

64. **Prepare Mission Packages** is an activity where mission planners produce the mission packages; this is the ultimate goal of mission planning.

65. **Recovered Mission** is the actual arrival and recovery of mission aircraft.

66. **Regulations** are documents that govern wing operations.

67. **Reports** is an output from the Conduct Wing C2 Operations task that gives the AOC an impression of how the mission went.

68. **Resources** are what the wing provides in support of performing the Conduct Wing C2 Operations (i.e., personnel, facilities, equipment, etc.).

69. **Route** is the planned path for the particular mission.

70. **Route Map** is the map showing the route to the target that mission planners and intel have determined to be the safest and most efficient.

71. **Scheduler** is a person who performs scheduling tasks.

72. **Schedules** are mission taskings that have been converted into the local sequence of events for aircraft departure/arrival. The schedule typically contains information about the weapons load, fuel, takeoff times, vulnerability times, return time, and pilots. The schedule is distributed to all base agencies concerned with the takeoff and arrival of aircraft.

73. **Skill Level** is a numeric ranking between 1 (low) and 5 (high) indicating how proficient the operator is at his or her job specialty.

74. **Skill Level Required** is an attribute of an ATO Task and is a numeric ranking between 1 and 5 indicating the skill level best suited to perform the task. This relates closely to the AFSC Required.

75. **Softcopy Schedule** is an initial copy of the schedule depicting number of sorties, mission flow, and turnaround times. Detailed mission planning must be performed before the schedule can be approved by the wing commander and finalized.

76. **Squadron Commander** is the first level authority of squadron operations.

77. **SRC Cell** is a cell within the WOC that is responsible for all aspects of the mission other than planning, scheduling, and executing the ATO.

78. **Support Assessment** is an assessment of the support required to successfully complete a mission. MPC personnel will contact other wings involved to ensure a safe and successful mission.

79. **Support Coordination** is an agreement between wings involved in the mission concerning the support required to complete the mission.

80. **Support Group** is a group within the wing that provides support to the Operations Group and the rest of the wing: security police, civil engineering, communications, and personnel.

81. **Support Squadron** is a squadron within the Operations Group that provides operations-oriented support to the Operations Squadrons: weather, weapons, and intelligence.

82. **Tactics** are techniques to be used to fly the mission and drop munitions in the most effective manner. Included are flight patterns, evasion techniques, countermeasure employment, weapons delivery, fusing, etc.

83. **Tasking** is what the AOC gives to the Conduct Wing C2 Operations task.

84. **Threat Assessment** is correlated threat data collected from multiple sources that serves as an assessment of the threat as it relates to the missions.

85. **Threat System Parameters** are parametric information on threat system performance and jammer capabilities. This information changes as systems are modified or new characteristics are observed such as war reserve modes.

86. **Time** is an attribute of the Assigned association and is the time required to complete the task. The amount of time required to complete a task varies depending upon the operator's AFSC, skill level, and use of an automated tool.

87. **Tool** is a generic reference to an automation tool that may be used by wing personnel to increase the accuracy and efficiency of a task.

88. **Weapons** is a general reference to a class of people who perform weapons related tasks.

89. **Weapons Officer** is a person who resides within the MPC and makes decisions about weapons use for given missions.

90. **Weapons Package** is a weapons configuration and fusing information that the weapons officer in the MPC has selected for the mission.

91. **Weather Data** is alphanumeric and graphical weather information.

92. **Weather** is a general reference to a class of people who perform weather related tasks.

93. **Weather Officer** is a person who resides within the MPC and is principally responsible for providing the weather support needed by the other people within the MPC.

94. **Wing** is the main fighting component of the Air Force. Wings are responsible for performing assigned tasking as spelled out in the ATO. To accomplish this, the wing performs a variety of tasks that are generally referred to as command and control tasks.

95. **Wing Operations Center** is the functional organization within the wing and is how the wing organizes itself to plan and execute a mission. The WOC serves as the hub for all wing aircraft operations (i.e., command post). The WOC provides several reports to the AOC to include aircraft, aircrew, and airfield status, mission scheduling, maintenance status, and intelligence reports.

Appendix B. Configuration Management

This appendix describes the configuration management and file conventions for the Resource Allocation Tool.

B.1 Source Code and Executable Code

The source code files and executable files for the Resource Allocation Tool can be found on AFIT's Hawkeye system under the directory **hartrum/wing/tool**. The program has been compiled using the Meridian compiler's **amake** utility. Thus, if changes are made to any of the program specifications or bodies, simply type **amake**. All necessary recompiles and linking will be carried out automatically. There is no need to compile via the **ada** command nor to link via the **bamp** command. In order to execute the program, simply type **ato_system**.

The Resource Allocation Tool is composed of the following files:

Package Specification files:

- analyze-utils.ads
- analyze.ads
- assigncon.ads
- assto.ads
- doer.ads
- doercon.ads
- support.ads
- supportcon.ads
- task.ads
- taskcon.ads
- tool.ads
- toolcon.ads
- ui.ads

Package Body files:

- analyze-utils.adb
- analyze.adb
- assigncon.adb
- assto.adb
- data.adb
- doer.adb
- doercon.adb
- support.adb
- supportcon.adb
- task.adb
- taskcon.adb
- tool.adb
- toolcon.adb
- ui.adb

Driver procedure file:

- ato.ada

Testing and Debugging package files:

- test.adb
- testass.adb
- makeass.adb

Data files:

- accuracy.dat
- doers.dat
- tools.dat
- tlevel.dat
- tasks.dat

B.2 Data File Conventions

Five data files are needed in order to run the Resource Allocation Tool. These data files are described as follows:

1. accuracy.dat

   This file is a table which holds the accuracy figures and is composed of integers ranging from one to five. The file consists of five rows of five integers. In each of the five rows, an integer is at column 1, 7, 13, 19, and 25.

2. doers.dat

   This file contains personnel information. Each record is on a separate line, and contains these fields:

Field Name	Type	Length	Starting Column
AFSC	alphanumeric	5	1
Position	alphanumeric	15	11
Skill Level	integer	1	28

3. tools.dat

   This file contains names of support tools. Each record is on a separate line and is composed of a single field:

Field Name	Type	Length	Starting Column
Name	alphanumeric	15	1

4. tlevel.dat

   This file contains tool support information. Each record is on a separate line and contains these fields:

Field Name	Type	Length	Starting Column
Task Ptr	integer	2	1
Tool Ptr	integer	2	any (but at least one space between fields)
Support Level	integer	1	any (but at least one space between fields)

5. tasks.dat

This file contains information on the tasks in the scenario. Each record may contain several lines of data. Records are separated by a blank line. The first line of each record contains these fields:

Field Name	Type	Length	Starting Column
Name	alphanumeric	15	1
AFSC	alphanumeric	5	17
Skill Level	integer	1	25
Nominal Time	integer	2	31

The second line of each record contains the word **Inputs:**. Zero or more lines following this line contain individual input fields which are alphanumeric, length 30, beginning in column 1. After all the input fields, there is a line containing the word **Outputs:**. Zero or more lines following this line contain individual output fields which are alphanumeric, length 30, beginning in column 1. The final output field is followed by a blank line which indicates the end of the record.

Bibliography

1. Cecil, Danny A. and Joseph A. Fullenkamp. *Using Database Technology to Support Domain-Oriented Application Composition Systems.* MS thesis, AFIT/GCS/ENG/93D-03, Graduate School of Engineering, Air Force Institute of Technology(AETC), Wright-Patterson AFB, OH, December 1993 (AD-A274091).
2. Colley, Grant. "Retain the Object Model and Retain the Benefits," *Object Magazine* (May 1992).
3. French, Simon. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.* New York: John Wiley & Sons, 1982.
4. Hartrum, Thomas C., "CSCE 594 - Object-Oriented Design and Analysis." Class Notes, 1993.
5. Hunt, Robert. *Modeling Operational Task Assignment in Air Force Wing Command and Control.* MS thesis, AFIT/GCS/ENG/94D-09, Graduate School of Engineering, Air Force Institute of Technology(AETC), Wright-Patterson AFB, OH, December 1994.
6. Iscoe, Neil. "Domain-Specific Reuse: An Object-Oriented and Knowledge-Based Approach." *Tutorial from Software Reuse: Emerging Technology* edited by Will Tracz, 299–308, IEEE Computer Society Press, 1989.
7. Langloss, Randel K. *Execution Function Air Operations, Unit.* Technical Report, HQ PACAF/SCC, July 15, 1994.
8. Langloss, Randel K. *Knowledge Based Software Engineering (KBSE) Support: A Formal Model of Wing-Level C2 Applied to the 432nd Fighter Wing Misawa AB, Japan.* Technical Report, HQ PACAF/SCC, June 14, 1993.
9. Lowry, Michael R. "Software Engineering in the Twenty-first Century." *Automating Software Design*, edited by Michael R. Lowry and Robert D. McCartney. 627–654. Menlo Park, CA: AAAI Press, 1991.
10. McCain, Ron. "Reusable Software Component Construction: A Product-Oriented Paradigm." *AIAA/ACM/NASA/IEEE Computers in Aerospace V Conference.* 125–135. AIAA, October 1985.
11. MITRE. *Theater Battle Management Command, Control, Communications, Computer, and Intelligence (TBM C4I) Architecture Air to Air (F-15) Wing Operations Center (WOC).* Technical Report, HQ ACC/DRI, 1 November 1993.
12. MITRE. *Theater Battle Management Command, Control, Communications, Computer, and Intelligence (TBM C4I) Architecture Air to Ground (F-16) Wing Operations Center (WOC).* Technical Report, HQ ACC/DRI, 1 November 1993.
13. Neighbors, James M. *Software Construction Using Components.* PhD dissertation, University of California, Irvine, 1981.
14. Ogush, Mike. "A Software Reuse Lexicon," *CrossTalk*, 41–45 (December 1992).

15. Pressman, Roger S. *Software Engineering: A Practioner's Approach.* New York: McGraw Hill, 1987.

16. Prieto-Díaz, Rubén. "Domain Analysis: An Introduction," *ACM SIGSOFT Software Engineering Notes*, *15*:47–54 (April 1990).

17. Prieto-Díaz, Rubén. "Domain Analysis for Reusability." *Domain Analysis and Software Systems Modeling* edited by Guillermo Arango and Rubén Prieto-Díaz, 63–69, IEEE Computer Society Press, 1991.

18. RCA, Government Systems Division. *TAF Mission, Level, Functions Analysis: Volume I - Mission Related Functional Analysis.* Technical Report, TAC, February 1978.

19. Rumbaugh, James, et al. *Object-Oriented Modeling and Design.* Englewood Cliffs, NJ: Prentice Hall, 1991.

20. Tracz, Will. "Domain Analysis Working Group Report - First International Workshop on Software Reusability," *ACM SIGSOFT Software Engineering Notes*, *17*:27–34 (July 1992).

21. Warner, Russel M. *A Method for Populating the Knowledge Base of AFIT's Domain Oriented Application Composition System.* MS thesis, AFIT/GCS/ENG/93D-24, Graduate School of Engineering, Air Force Institute of Technology(AETC), Wright-Patterson AFB, OH, December 1993 (AD-A274128).

22. Welgan, Robert L. *Domain Analysis and Modeling of a Model-Based Software Executive.* MS thesis, AFIT/GCS/ENG/93D-25, Graduate School of Engineering, Air Force Institute of Technology(AETC), Wright-Patterson AFB, OH, December 1993 (AD-A274087).

Vita

Captain Michael D. Sarchet, the eldest son of David and Linda Sarchet, was born November 30, 1962, in Dayton, Ohio, and graduated from Centerville High School, Centerville, Ohio, in May 1981. In September 1981, Mike entered Bowling Green State University (BGSU), Bowling Green, Ohio, to pursue a Bachelor of Science degree in Computer Science. Mike's acceptance of a 3-year AFROTC scholarship in 1982 was the beginning of his military career. In May 1985, Mike graduated from BGSU and was commissioned a Second Lieutenant in the US Air Force.

Upon graduation from the Information Systems Officer Core course at Keesler AFB, Mississippi, in February 1986, then Second Lieutenant Sarchet was assigned to the Air Force Electronic Warfare Center, Kelly AFB, Texas. His duties there included the design, development, and implementation of new hardware/software systems to support the Air Force electronic combat community. In September 1989, Captain Sarchet moved to the Office of Special Projects, Los Angeles AFB, California. Mike was a systems engineer for a major space program and lead engineer for translating warfighter requirements into new program office initiatives. In May 1993, Captain Sarchet entered the Air Force Institute of Technology (AFIT) at Wright-Patterson AFB, Ohio, to pursue a Master of Science degree with a Computer Science major and concentration in Software Engineering. Upon graduation from AFIT in December 1994, Captain Sarchet was re-assigned to the Office of Special Projects, Washington, DC.

Permanent address: 214 Ridge Rd
Lake Mary, FL 32746

sarchetm@sgate.com