NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ARCHITECTURAL DESIGN AND PROTOTYPING OF A WEB-BASED WAR GAME SIMULATION FOR CAMPAIGN PLANNING EXERCISES

by

Antonios Chalakatevakis

September 2000

Thesis Advisor:	Man-Tak Shing
Co - Advisor:	Leroy A. Jackson

Approved for public release; distribution is unlimited

DTIC QUALITY INSPECTED 4

20001026 148

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE: ARCHITERURAL DESIGN AND PROTOTYPING OF A WEB-BASED WAR GAME SIMULATION FOR CAMPAIGN PLANNING EXERCISES		5. FUNDING NUMBERS
6. AUTHOR(S) Antonios Chalakatevakis		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

ABSTRACT ***(maximum 200 words)***

The Campaign Planning Exercise (CAMPEX) War Game is being used for the training of the students of the Air War College in the area of the Air Campaign Planning and the Ground Forces Deployment. The CAMPEX life cycle started in 1986 and the last version was released in 1994. Microsoft Basic Version 7.10 Professional Development System was used for its development. CAMPEX was not designed or developed with the objected-oriented technique, so further extension and its use as component for Distributed Components Applications is not feasible.

TRADOC Analysis Center (TRAC) of Monterey plans to use a collection of old War Games as Components of a Distributed Embedded Application. The CAMPEX Employment Module is the first War Game that will form one of the components of this application, so the redesign and implementation of CAMPEX Employment Module with object-oriented technique is necessary. This thesis examines the distributed component architectures available to support the Distributed Embedded Application, re-engineers the CAMPEX Employment Module into an object-oriented design, and validates its requirements via a prototype developed using ACCESS2000. The new design will be the basis for reengineering the other war game planning software for the Air War College.

14. SUBJECT TERMS Battlespace Environments, Distributed Components Architecture, Object-Oriented Design, Modeling and Simulation.			15. NUMBER OF PAGES 195 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ARCHITECTURURAL DESIGN AND PROTOTYPING OF A WEB-BASED WAR GAME SIMULATION FOR CAMPAIGN PLANNING EXERCISES

Antonios Chalakatevakis
Major, Hellenic Army
Hellenic Military Academy, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2000

Author: _____
Antonios Chalakatevakis

Approved by: _____
Man-Tak Shing, Thesis Advisor

_____
Leroy A. Jackson, Co-Advisor

_____
Dan Boger, Chairman
Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Campaign Planning Exercise (CAMPEX) War Game is being used for the training of the students of the Air War College in the area of the Air Campaign Planning and the Ground Forces Deployment. The CAMPEX life cycle started in 1986 and the last version was released in 1994. Microsoft Basic Version 7.10 Professional Development System was used for its development. CAMPEX was not designed or developed with the objected-oriented technique, so further extension and its use as component for Distributed Components Applications is not feasible.

TRADOC Analysis Center (TRAC) of Monterey plans to use a collection of old War Games as Components of a Distributed Embedded Application. The CAMPEX Employment Module is the first War Game that will form one of the components of this application, so the redesign and implementation of CAMPEX Employment Module with object-oriented technique is necessary. This thesis examines the distributed component architectures available to support the Distributed Embedded Application, re-engineers the CAMPEX Employment Module into an object-oriented design, and validates its requirements via a prototype developed using ACCESS2000. The new design will be the basis for reengineering the other war game planning software for the Air War College.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1

A. GENERAL 1

B. MOTIVATION 3

C. OBJECTIVES 4

D. THESIS ORGANIZATION 4

II. DISTRIBUTED COMPONENTS ARCHITECTURE 7

A. THE PROBLEM DESCRIPTION 7

B. REQUIREMENTS AND CONSTRAINTS 7

C. DISTRIBUTED COMPONENTS SERVICES 9

D. EXISTING PARTIAL SOLUTIONS 10

E. EXISTING SOLUTIONS 12

F. COMPARISON AMONG THREE ARCHITECTURES 29

III. SOFTWARE REQUIREMENTS SPECIFICATION 31

A. OVERVIEW 31

B. CUSTOMERS 31

C. GOALS 31

D. USER CHARACTERISTICS 32

E. GENERAL CONSTRAINTS 32

F. ASUUMPTIONS AND DEPENDENCIES 33

G. SYSTEM FUNCTIONS 33

H. SYSTEM ATTRIBUTES 41

I. USE CASES 42
1. High Level Use Cases 42
2. CAMPEX Employment Module Use Case Diagram 49

J. RANKING USE CASES 50

K. CONCEPTUAL MODEL 52

IV. SOFTWARE DESIGN SPECIFICATION 53

A. INTRODUCTION 53
1. Purpose 53
2. Scope 53
3. Definitions, Acronyms, and Abbreviations 53

B. THE SYSTEM ARCHITECTURE 54
1. The Detailed Architecture Diagram 54

C. Object/Class Diagrams 60
1. Object Diagram 60
2. Classes-Objects Attributes and Operations 61

V. PROTOTYPE 79

A. PURPOSE 79

B. PROTOTYPE IMPLEMENTATION 79
1. General 79
2. Database Design 80

C. User Manual 91
1. Installing CAMPEX Employment Module Prototype 91
2. Running the CAMPEX Employment Module Prototype 92
3. CAMPEX Employment Module Initial Screen 92
4. "New Student" 93
5. "Select Student" 94
6. "Start Employment Module" 95
7. "Select to See the Program Reports" 96
8. "Select to Continue with Program" 97
9. "ATO Selection" 97
10. "Select an ATO" 98
11. "New ATO " 98
12. "Main Menu Screen" 99
13. "Options" 100
14. "Report Management" 101

15. "Area Map" 102
16. "Change ATO" 103
17. "Blue Basing Summary" 103
18. "Sorties Available" 103
19. "Analysis" 104
20. "ATO Planning" 104
21. "20 Top Priority Targets" 106
22. "Plan Edit Missions" 106
23. "Enter or Edit Flight" 108
24. "Flight Data" 109
25. "Estimated Results Choices" 110
26. "Estimated Results" 111
27. "Flights without Sorties" 112
28. "Daily Summaries" 112
29. "Cancel Mission or Package" 113
30. "Logistic Requirements" 114
31. "Fly ATO Missions" 114
32. "Ground Forces Deployed" 114
33. "Intel and Results" 115

VI. CONCLUSIONS 117

APPENDIX A – USE CASES 119

USE CASE (U1): Start Employment Module 119

USE CASE (U2): STUDENT INFO 122

USE CASE (U3): LOAD AN ATO 125

USE CASE (U4): Manage an ATO 126

USE CASE (U5): DESCRIBE THE 20 TGTS WITH HIGHEST PRIORITY .. 129

USE CASE (U6): PLAN AN ATO 130

USE CASE (U7): FLY AN ATO 133

USE CASE (U8): INITIAL INFORMATION 134

USE CASE (U9): ESTIMATED RESULTS 136

USE CASE (U10): ACTUAL RESULTS 138

USE CASE (U11): MAP 140

USE CASE (U12): SEND EXERCISE 141

APPENDIX B – SYSTEM SEQUENCE DIAGRAMS 143

Select Student 143

Add Student 144

Start Employ 145

Select ATO 146

New ATO 147

Copy an Existing ATO 148

Erase ATO 149

20 Targets with Highest Priority 150

Plan ATO Enter a New Mission 151

Cancel Mission 152

Cancel Missions of a Package 152

Fly ATO 153

Estimated Results 154

Missions Without Sorties 154

Initial Information - Daily Summary 155

Initial Information – Logistics Requirements 155

Initial Information – Blue Basing Summary 156

Initial Information – "Recce for Targets" 156

Initial Information – "Analysis" 157

Initial Information – "Sorties Available" 157

Actual Results – "Cumulative Summary" 158

Actual Results – "Enemy Over Blue Bases" 159

Actual Results – "Ground War Summary" 159

Actual Results – "Measures of Merit" 160

Actual Results – "Yesterday Losses By Aircraft Type" 161

Actual Results – "Yesterday Losses By Task Type" 162

Map 162

Send ATO 163

APPENDIX C ABBREVIATIONS ACRONYMS DEFINITIONS 164

BIBLIOGRAPHY 168

INITIAL DISTRIBUTION LIST 171

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Client Using COM Object through an Interface Pointer 14

Figure 2. Three Methods for Accessing COM Objects 15

Figure 3. Cross-Process Communication in COM 17

Figure 4. Creating a COM Object Pointer 18

Figure 5. CORBA Architecture Layers 20

Figure 6. Detailed CORBA Architecture 21

Figure 7. JINI Architecture Segmentation 28

Figure 8. Use Cases Diagram 50

Figure 9. Conceptual Model 52

Figure 10. Detailed Architecture Diagram 54

Figure 11. CAMPEX Employment Module Object 60

Figure 12. Entity-Relation Diagram 80

Figure 13. Return and Exit 92

Figure 14. Initial Screen 93

Figure 15. New Student 94

Figure 16. Select Student 95

Figure 17. Choose to see Reports or Not 95

Figure 18. Introductory Report 96

Figure 19. Ground Combat Units 97

Figure 20. ATO Selection 98

Figure 21. Enter a New ATO 99

Figure 22. Main Menu 100

Figure 23. Options Menu 101

Figure 24. Report Management 102

Figure 25. Area Map 102

Figure 26. Blue Basing Summary 103

Figure 27. Sorties Available 103

Figure 28. Analysis 104

Figure 29. Menu "ATO Planning" 104

Figure 30. List of 20 Targets with Highest Priority 106

Figure 31. Flight Categories 107

Figure 32. Assigned Flights By Category 108

Figure 33. Enter or Edit a Flight 109

Figure 34. Input or Update Flight Data 110

Figure 35. Estimated Results Choices 111

Figure 36. Estimated Results 111

Figure 37. Flights without Sorties 112

Figure 38. Daily Summary 112

Figure 39. Cancel Mission or Missions' Package 113

Figure 40. Logistics Requirements 114

Figure 41. Menu Intel Results 115

Figure 42. Sample Intel Report "Measures of Merit" 116

Figure 43. Sequence Diagram "Select a Student" 143

Figure 44. Sequence Diagram "Add Student" 144

Figure 45. Sequence Diagram "Start an ATO" 145

Figure 46. Sequence Diagram "Select an ATO" 146

Figure 47. Sequence Diagram "New ATO" 147

Figure 48. Sequence Diagram "Copy an Existing ATO" 148

Figure 49. Sequence Diagram of "Erase an ATO" 149

Figure 50. Sequence Diagram "20 Highest Priority Targets" 150

Figure 51. Sequence Diagram "Plan or Edit Mission" 151

Figure 52. Sequence Diagram "Cancel Mission" 152

Figure 53. Sequence Diagram "Cancel Package of Missions" 152

Figure 54. Sequence Diagram "Fly an ATO" 153

Figure 55. Sequence Diagram "Estimated Results" 154

Figure 56. Sequence Diagram "Flights Without Sorties" 154

Figure 57. Sequence Diagram "Daily Summary" 155

Figure 58. Sequence Diagram "Logistic Requirements" 155

Figure 59. Sequence Diagram "Blue Basing Summary" 156

Figure 60. Sequence Diagram "Recce for Targets" 156

Figure 61. Sequence Diagram "Analysis" 157

Figure 62. Sequence Diagram "Sorties Available" 157

Figure 63. Sequence Diagram "Cumulative Summary" 158

Figure 64. Sequence Diagram "Enemy over Blue Bases" 159

Figure 65. Sequence Diagram "Ground War Summary" 159

Figure 66. Sequence Diagram "Measures of Merit" 160

Figure 67. Sequence Diagram "Yesterday Losses By Aircraft Type" 161

Figure 68. Sequence Diagram "Yesterday Losses By Task Type" 162

Figure 69. Sequence Diagram of "Map" 162

Figure 70. Sequence Diagram "Send ATO" 163

LIST OF TABLES

Table 1. System Functions....41

Table 2. System Attributes....41

Table 3. Ranking Use Cases....51

Table 4. Class Employ–Attributes....62

Table 5. Class Employ–Operations....66

Table 6. Class Student–Attributes....66

Table 7. Class Student–Operations....66

Table 8. Class ATO–Attributes....67

Table 9. Class ATO–Operations....67

Table 10. Class ATODay–Attributes....67

Table 11. Class ATODay–Operations....68

Table 12. Class Mission–Attributes....69

Table 13. Class Mission–Operations....70

Table 14. Class Target–Attributes....71

Table 15. Class Target–Operations....71

Table 16. Class Category–Attributes....71

Table 17. Class Subcategory–Attributes....72

Table 18. Class Group–Attributes....72

Table 19. Class Group-Operations....72

Table 20. Class Aircraft–Attributes....73

Table 21. Class Aircraft–Operations....73

Table 22. Class Sector–Attributes.....74

Table 23. Class Sector–Operations.....74

Table 24. Class Task Type–Attributes..... 75

Table 25. Class Task Type–Operations.....75

Table 26. Bomb Type–Attributes.....75

Table 27. Class Map–Attributes..... 76

Table 28. Class Map–Operations.....76

Table 29. Class Report–Attributes.....76

Table 30. Class Report–Operations.....77

Table 31. Entities – Relations Attributes.....90

I. INTRODUCTION

A. GENERAL

There has been continuous progress in the development of computers over the past four decades. The results are more impressive by any measure in the state of hardware than in the state of software. From the beginning hardware developers have had a theoretical basis that comes from other sciences, like mathematics, physic, mechanics, etc. Also, the hardware developers borrowed and applied standard methods from other engineering disciplines for design and manufacturing. In contrast, software developers initially relied primarily upon human imagination, invention, and ingenuity. This lack of an engineering approach has produced legacy software applications that cannot be supported any longer.

As the science of software development slowly evolved into a distinct engineering discipline, software engineers established the processes, the techniques and the rules that govern the development of software systems.

The process to develop a software system usually consists of the following phases[1]: Requirements Analysis, Functional Specification, Architectural Design, Implementation and Evolution. In the nineties the object-oriented methodology has emerged as the most popular method because it supports the rule that can be described by the maxim, "reduce, reuse recycle". The object-oriented methodology increases efficiency, reduces development time and decreases the cost of software products.

[1] Berzins and Luqi in their book "Software Engineering with Abstractions" 1990 chapter 1 p 6 about the Software Development Process

Software engineering has also changed the process of software evolution and maintenance. The old approach "to maintain a software system for period of time and then to replace it with an complete new system" when further evolution is not feasible is not an efficient solution. It is too costly to lose the assets that an existent and working system offers, including all the previous work that went into the requirements analysis phase. Also, users have verified the existent system over time and have gained knowledge about the system's operations through years of maintenance and development. To be efficient, the evolution technique for an existent system must respect that the current system is valuable even though it is increasingly difficulty to extend. Therefore, the new approach must respect the assets of the old system and developers should salvage any useful parts from the old system and change only the parts that must be changed.

The most common problem when re-engineering old systems is that they are not implemented according to modern, evolvable methods like the object-oriented design techniques; however, the developers can still use the requirements specification of the old systems in the first phase of the redesign. So the designers must retrieve the requirements specification from the old system and document them with object-oriented techniques. In addition, the most effective approach to redesign is to implement a prototype. In this way, the designer can verify the correctness of the requirements specification and design through reviews with the customer.

Current software engineering techniques are well supported by commercial-off-the-shelf (COTS) products. Today COTS products are increasingly important and highly economical tools for organizations to explore reengineering opportunities and strategies.

The ultimate goal is to reengineer the old system using object-oriented techniques that allows the concurrent evolution of software via component-based development.

B. MOTIVATION

The TRADOC Analysis Center of Monterey, California plans to redesign and re-implement the Campaign Planning Exercises (CAMPEX) discussion war game. The CAMPEX[2] is a software system that was implemented by the Air War College to provide students the opportunity to test their understanding of strategy, leadership, international security, National Security Decision Memorandums (NSDM), General Purpose (GP) forces, unified commands, and joint fundamentals. The current version of the CAMPEX system consists of two modules, the "Deployment" and the "Employment". In the "Deployment" module the student deploys joint forces and in the "Employment" module he employs the forces.

The CAMPEX system lifecycle began in 1986 and the current version (with ID 8.9[3]) was released in 1994. Because CAMPEX is still used today, we can assume that it satisfies most requirements of the Air War College. Moreover, we can conclude that the system requirements and the algorithms and other functions of CAMPEX have been verified in practice, because the CAMPEX was developed, maintained, and used by Air War College personnel. CAMPEX was implemented in the Microsoft Basic Professional Development System Version 7.10, and the object-oriented techniques were not used for

[2] Air War College in the CAMPEX User Manual
[3] Air War College Source Code of CAMPEX last version

its design. Also, a serious problem for further evolution of CAMPEX is the lack of documentation for any phase of its development.

An important constraint that must be satisfied when reengineering a current software system is the available hardware. Today the user has PC's with greater processing power and data storage capacity that often operate on a high capacity network. The reengineering of CAMPEX must account for this computing environment.

The final factor that must be considered when reengineering CAMPEX is the High Level Architecture (HLA) for distributed simulations. The U.S. Department of Defense DoD) mandates use of the HLA in new simulations and the retrofit of legacy simulations by 2001.

C. OBJECTIVES

This thesis describes the reengineering of the CAMPEX "Employment" Module using object-oriented techniques with respect to the current user's needs in combination with the current available hardware. The secondary goal is preparation for High Level Architecture compliance should the user decide to distribute CAMPEX over the network.

So the primary objectives of my work were twofold: 1) research in the area of the Distributed Objects Architectures and 2) analysis and redesign of the CAMPEX "Employment" module with object-oriented techniques and verify the design with a small prototype. The prototype must work in the Microsoft Windows environment and allows user to run some of the CAMPEX procedures through Internet.

D. THESIS ORGANIZATION

Chapter II provides:

background on Distributed Objects Architectures,

high level requirements for the Distributed Components,

a brief description of the partial and "complete" existent solutions in this area,

the advantages and disadvantages of each solution, and

the common characteristics and differences.

Chapter III provides the requirements analysis and functional specification of CAMPEX Employment that results from reverse engineering the current version. The requirements analysis and functional specification are documented using Unified Modeling Language (UML) notation.

Chapter IV describes the new design and architecture of the new CAMPEX using the UML methodology and notation. The existing CAMPEX algorithms and functions are the basis for the design of the object interactions because the Air War College has already verified them in practice.

Chapter V presents the prototype implementation. The prototype is implemented with ACCESS-2000 and Visual Basic. Chapter V contains the basic database design, queries and forms. Moreover, this chapter offers a "User Manual" that allows the user to test and use the prototype easily.

Chapter VI summarizes the key elements of the thesis, provides observations about the difficulties and lessons learned, and provides insights and recommendations for future work to apply the selected Distributed Objects Architecture and to re-implement the entire CAMPEX.

THIS PAGE INTENTIONALLY LEFT BLANK

II. DISTRIBUTED COMPONENTS ARCHITECTURE

A. THE PROBLEM DESCRIPTION

Even after many years the state of the art and science in software development is not as satisfactory as that in hardware development. Different vendors have developed numerous remarkable applications using various methods, computer languages, and platforms. The exploitation of all these old applications in combination with new applications under development on different types and configured machines that run over Internet poses one of the biggest challenges in the software engineering today.

B. REQUIREMENTS AND CONSTRAINTS

This section addresses the high-level requirements and constraints to solve the problem posed in the previous section. First the object-oriented methodology for software implementation comprises a requirement for new applications and a constraint for existing applications. This general requirement[4] is that

- the applications must consist of components,
- the components should be characterized by high cohesion and loose coupling,
- the components should be developed in accordance with object-oriented principles (encapsulation, polymorphism and inheritance), and
- the components should communicate through messages.

[4] Reaz Hoque and Tarun Sharma in their book "WEB Components" 1998 chapter 1 p 7 what are the Distributed Objects

The second requirement is to cross network boundaries. The new architecture must allow the users to utilize applications that consist of components that can be distributed on different machines on WANs or LANs. This requirement demands that the application access the network in standard ways and expand its address space to the entire network effectively.

The third requirement is that the components should be programming language independent. That is, components that are developed in different programming languages be able to interoperate in a distributed system. At the same time the user must be able to choose a unique tool for the creation and the integration of these heterogeneous components.

The fourth requirement is to cross the platform boundaries. The new architecture must allow the old legacy applications as well as the new ones to run on machines of different type or configuration. In other words, machines and their operating system should not block communication among the components of the distributed application.

The fifth requirement is to satisfy the "reduce, reuse, recycle" dictum. This requires that the application be reusable as a whole or in parts (patterns, components or objects). The application must be designed so that it can be deployed in an environment with diminished resources and function in a reduced, but useful, capacity. Finally, it must provide components and design features that can be recycled for the production of new applications.

The sixth and last requirement is that the application must offer a satisfactory level of performance and reliability. This constraint follows directly from the maxim to "reduce". "Reduce" excludes solutions that increase the implementation time and

complexity. This excludes many popular solutions that are used today. Examples include the 4GL languages with fat clients, the development of low level drivers and new network protocols for the crossing of network boundaries, and the use of different tools with each programming language for breaching the programming language barriers.

A final constraint is that the new system should demand no new resources.

C. DISTRIBUTED COMPONENTS SERVICES

Today many vendors have software development solutions that seem to satisfy all these requirements. These solutions are variants of the Distributed Components Based Architecture. All the vendors contend that they have realized the early dream of the software community for distributed computing that properly uses all the capabilities of the current hardware.

Distributed architectures must offer the following services[5]:

- *naming* service to provide a mechanism for locating distributed components in a system,
- *monitor* service to watch the whole system for correctness and alert if something wrong happens to an operator,
- *listening* service to ensure that users of the distributed components have the appropriate privileges,
- *persistence* mechanism to uniformly save, update, and restore an object's state using a persistent data store,

[5] SUN Microsystems in JINI Architecture Specification version 1.0.1 1999 about Distributed Components Services

- *transaction* support mechanism to ensure that a transaction is completed or aborted in its entirety whenever work is performed. (Typically a transaction defines an atomic unit of work in an enterprise system. A distributed transaction is a single unit of work that spans multiple computers.)
- *security* mechanisms to ensure that communication from authorized users to a distributed object is secure,
- *messaging* support to provide an asynchronous programming model, as opposed to the typical request-reply model. (There are many types of applications that require messaging. An example is an application in which the client and server are required to run in different times.)
- *distributed* services to automatically deallocate distributed objects when they are no longer being used by their clients, and
- *resource* management to manage distributed objects in such a way as to maximize scalability and support a large number of clients interacting with a large number of distributed objects in short period of time.

D. EXISTING PARTIAL SOLUTIONS

Today there are many suggestions from different vendors. Some of these suggestions tackle the problem of the distributed computing by trying to satisfy all the requirements and others by trying primarily to satisfy an individual requirement.

The following techniques suggest solutions for various requirements of the distributed computing problem:

The *Remote Procedure Call Mechanism*[6] satisfies the requirement to cross network boundaries. It is probably the most popular technique for crossing network barriers. At first sight it seems well suitable for distributed computing since it allows one application to make functions calls through the network. However, if we examine RPC deeper we find that RPC does not allow the objects to change state. Adding this capability has a very large performance cost.

Sockets mechanisms[7] are another way to cross network barriers. This solution has many advantages, but it usually demands low-level network programming. To be useful for distributed computing architectures this solution should offer network interfaces that hide all the unnecessary low level networking details.

Interpreted Languages[8] can solve the problem of the crossing operating system (OS) boundaries. Since this kind of languages is not compiled to a specific binary format, they can be reused and run in source code format on multiple machines. The drawback of this solution is the lack of speed because interpreted code is typically hundreds of times slower than the compiled code; also, the interpeter may not exists for some platforms.

Binary Compatibility Layers can be used to cross OS barriers. This solution provides a layer of code on top of the OS that runs binary files compiled for a different OS. This solution is feasible if binary compatibility layers exist for all the OS's. One problem with this solution is that it demands a lot of work from programmers. Another

[6] Gopalan Suresh Raj in his book "A Detailed Comparison of CORBA, DCOM and JAVA/JINI

[7] Reaz Hoque and Tarun Sharma in their book "WEB Components" 1998 chapter 1 p.14 what are the Distributed Objects

problem is that most applications use more than one OS layer, so the OS vendor must construct a new binary compatibility layer every time another OS layer is used.

Language Binding can cross language barriers. The biggest weakness of this technique is that users must learn how the solution works in different programming environment. Consequently, it is not standard and can delay the implementation process.

Another way to cross language barriers is a development environment that can compile to multiple target platforms. But this solution has a big drawback too. It demands a binary distribution.

E. EXISTING COMPLETE SOLUTIONS

The previous section discussed the most common partial solutions for distributed computing and emphasized their drawbacks. This section describes standard complete solutions.

The Distributed Component Object Model (DCOM) is Microsoft's solution for distributed computing. DCOM treats the problem of the distributed components as two different subproblems. The first subproblem[9] is the component architecture that describes component packaging and cross language interoperability. The second sub-problem is communication among components over the network and support for remote method invocation.

---

[8] Reaz Hoque and Tarun Sharma in their book "WEB Components" 1998 chapter 1 p 15 what are the Distributed Objects

[9] Software Engineering Institute *DCOM Architecture Overview,* 10 Jan 97, p.2 by Ed Morris and Emil Litvak

COM, an ancestor of the DCOM and now a part of DCOM, allows inter-process communication. COM supports *interoperability* and *reusability* of distributed objects by allowing developers to build systems by assembling reusable components from different vendors. By applying COM to build systems from preexisting components developers hope to reap benefits like *maintainability* and *adaptability*.

COM defines an application-programming interface (API) for creating components to use in custom applications and to allow components to interact. However, the interacting components must adhere to a binary structure specified by Microsoft. As long as they adhere to this binary structure, components written in different languages can *interoperate*.

The DCOM extends the COM by allowing network-based component interaction. While COM processes can run on the same machine in different address spaces, the DCOM extension allows processes to be spread across a network. With DCOM, components operating on a variety of platforms can interact as long as DCOM is available within the environment. COM and DCOM represent "low-level" technology that allows components to interact. Microsoft provides two higher-level application services, OLE and ActiveX, which are built on top of COM and DCOM. OLE provides services such as object "linking" and "embedding" that are used in the creation of compound documents (documents generated from multiple tool sources). ActiveX allows components to be embedded in Web pages.

COM is a binary compatibility specification and associated implementation that allows clients to invoke services provided by COM-compliant components (COM objects). As shown in Figure 1 services implemented by COM objects are exposed

through a set of interfaces that represent the only point of contact between the clients and the object.

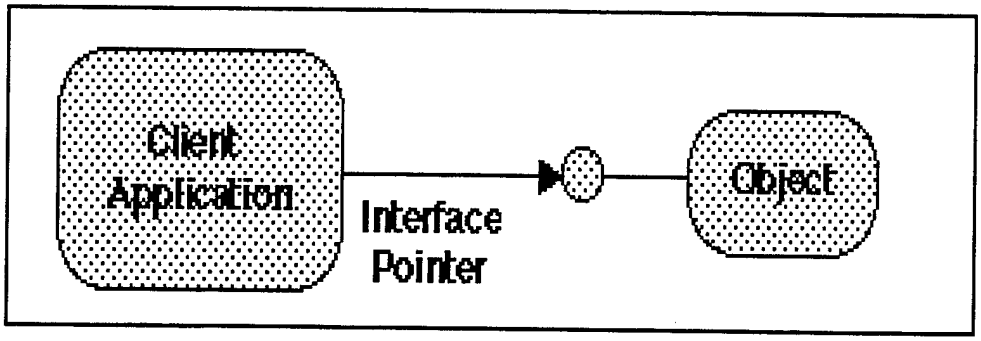


Figure 1. Client Using COM Object Through an Interface Pointer (Source: Software Engineering Institute *DCOM Architecture Overview,* 10 Jan 97, p.2.)

COM defines a binary structure for the interface between the client and the object. This binary structure provides the basis for interoperability among software components written in arbitrary languages. As long as a compiler can reduce language structures down to this binary representation, the implementation language of the client is independent from the run-time binary representation of the object. Thus, COM objects and clients can be coded in any language that supports Microsoft's COM binary structure.

An interface provides a collection of related methods. COM objects and interfaces are specified using Microsoft Interface Definition Language (IDL), an extension of the DCE Interface Definition Language standard. To avoid name collisions, each object and interface must have a unique identifier.

Every COM object runs within of a server. A single server can support multiple COM objects. There are three ways in which a client can access COM objects provided by a server:

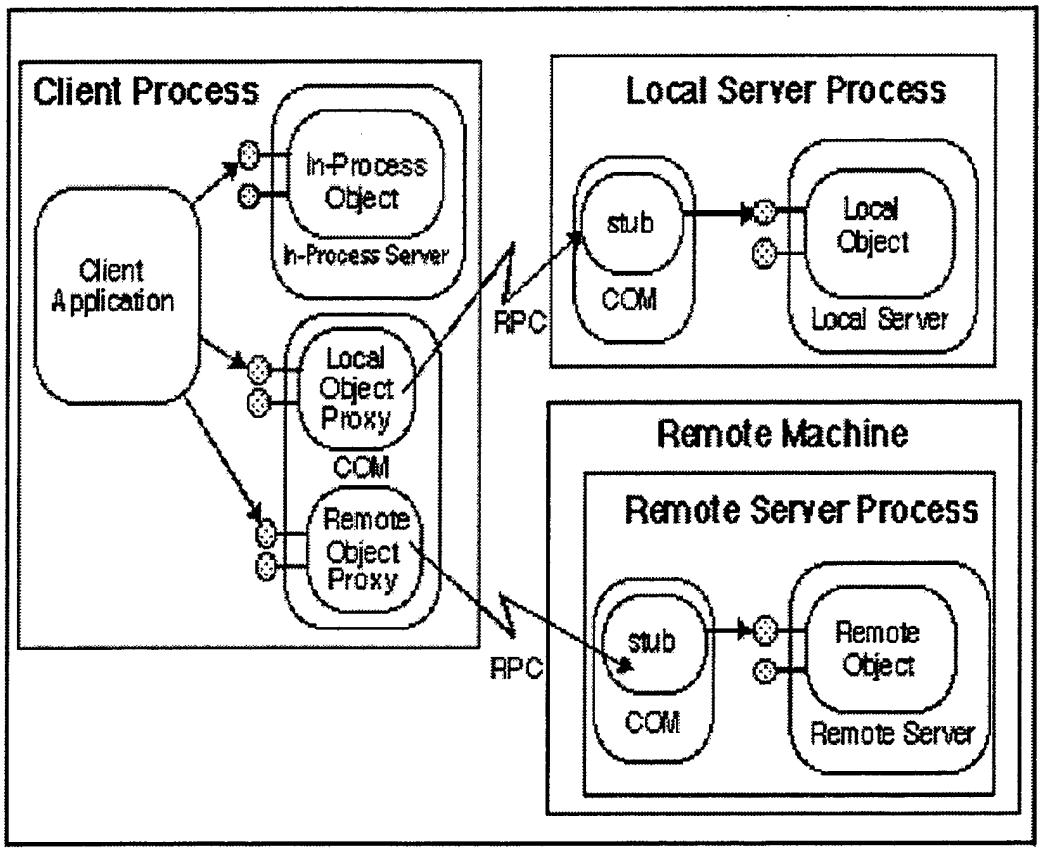


Figure 2. Three Methods for Accessing COM Objects (Source: Software Engineering Institute *DCOM Architecture Overview,* 10 Jan 97, p.4.)

- In-process server: The client can link to a library containing the server directly. The client and server execute in the same process. Communication is accomplished through function calls.
- Local Object Proxy: The client can access a server running in a different process yet on the same machine through an inter-process communication mechanism. This mechanism is actually a lightweight Remote Procedure Call (RPC).

- Remote Object Proxy: The client can access a remote server running on another machine. The network communication between client and server is accomplished through DCE RPC. The mechanism supporting access to remote servers is called DCOM.

If the client and the server are in the same process, the sharing of data between them is simple. However, when the server process is separate from the client process, as in a local server or remote server, COM must format and bundle the data in order to share it. This process of preparing the data is called marshalling. Marshalling is accomplished through a "proxy" object and a "stub" object that handles the cross-process communication details for any particular interface. COM creates the "stub" in the object's server process and has the stub manage to the real interface pointer. Then COM creates the "proxy" in the client's process, and connects it to the stub. Then the proxy supplies the interface pointer to the client.

The client calls the interfaces of the server through the proxy, which marshals the parameters and passes them to the server stub. The stub unmarshals the parameters and makes the actual call inside the server object. When the call is completed, the stub marshals return values and passes them to the proxy, which in turn returns them to the client. The same proxy/stub mechanism is used when the client and server are on different machines. However, the internal implementation of marshalling and unmarshalling differs depending on whether the client and server operate on the same machine (COM) or on different machines (DCOM). Given an IDL file, the Microsoft IDL compiler can create default proxy and stub code that performs all necessary marshalling and unmarshalling.

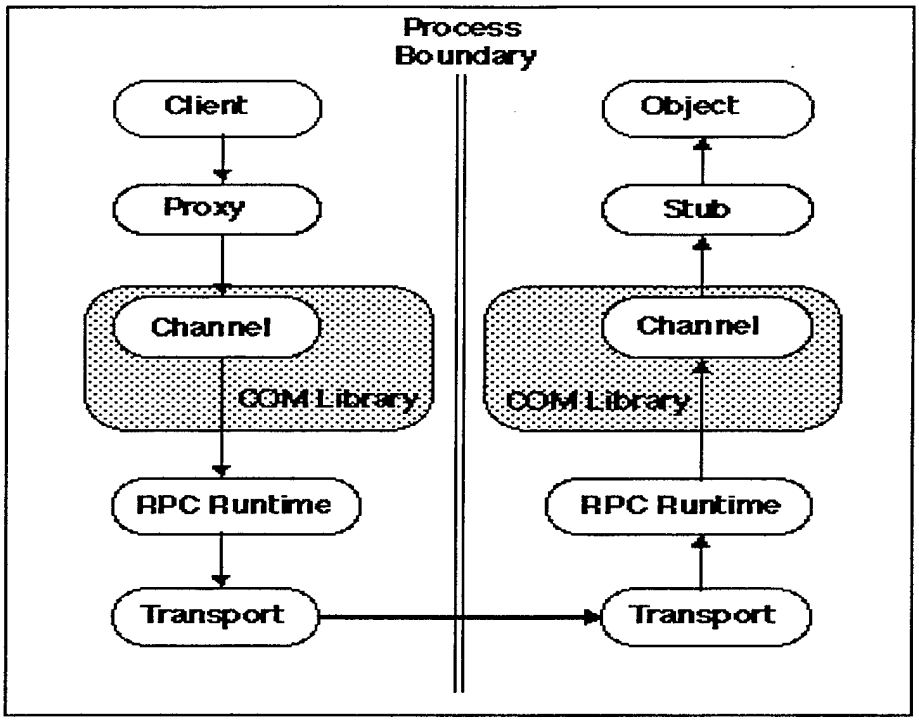


Figure 3. Cross-Process Communication in COM (Source: Software Engineering Institute *DCOM Architecture Overview,* 10 Jan 97, p.5.)

All COM objects are registered with a component database. As shown in Figure 4, when a client wishes to create and uses a COM object:

- it invokes the COM API to instantiate a new COM object,
- COM locates the object implementation and initiates a server process for the object,
- the server process creates the object and returns an interface pointer at the object, and then
- the client can interact with the newly instantiated COM object through the interface pointer.

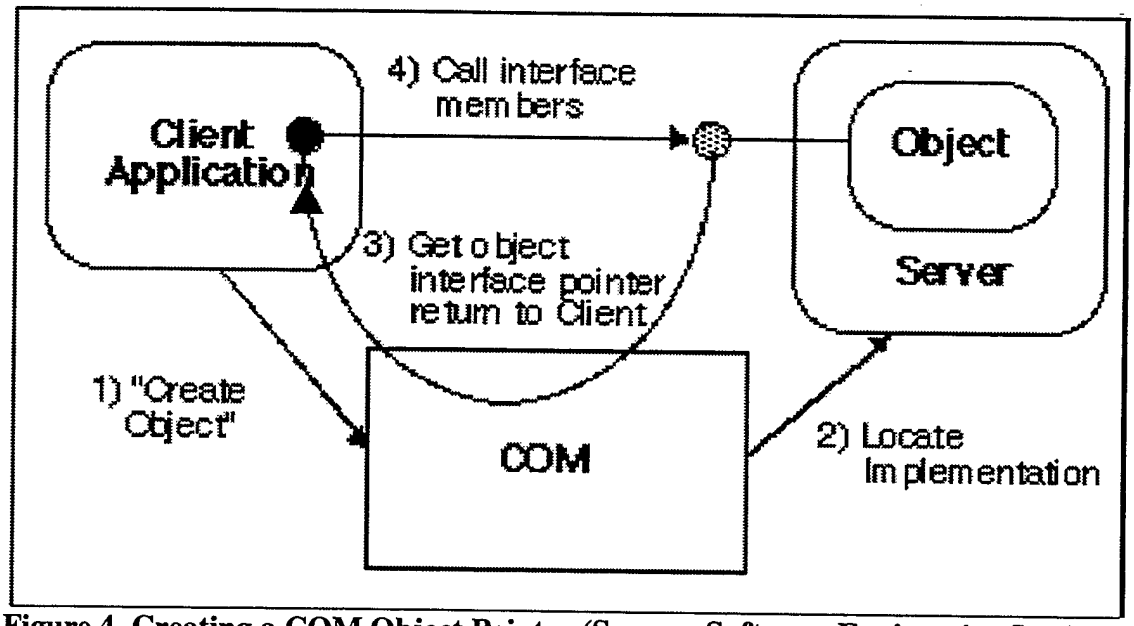


Figure 4. Creating a COM Object Pointer (Source: Software Engineering Institute *DCOM Architecture Overview,* 10 Jan 97, p.4.)

COM includes interfaces and API functions that expose operating system services as well as other necessary mechanisms for a distributed environment such as naming and events.

DCOM advantages are:

- Microsoft supports DCOM so it already has many users.
- DCOM and the other Microsoft tools for implementing components usually have a low price.
- The components that DCOM integrates can be implemented in many programming languages so DCOM succeeds in crossing the language boundaries.
- DCOM offers an interface that crosses network boundaries in a standard and simple way.

- DCOM is a robust application. It is easy for the programmers to learn to use the DCOM (though one must be an expert in areas like distributed systems design, multi-threaded applications, and networking to implement an embedded system with DCOM).

Disadvantages of DCOM are:

- Once an interface is defined, it should not be changed. New methods should not be added and existing methods should not be modified. This interface restriction is not enforced, but it is a rule that component developers should follow.
- DCOM depends on the Windows Operating System. This is its main disadvantage. DCOM does not satisfy the requirement for crossing platform boundaries.
- Older legacy applications are not easy to integrate with a system that uses only Windows.
- Because COM and DCOM are based on a native binary format, components written to these specifications are not really platform independent.
- DCOM is not standard for many vendors. For example Netscape does not support ActiveX.
- DCOM cannot guarantee high performance.
- DCOM does not support security, but there are external packages for security support.
- DCOM does not support applications with real-time requirements and cannot guarantee reliability to applications.

The Common Object Request Broker Architecture[10] (CORBA) is the Object Management Group's (OMG) solution. OMG is an industry group with over six hundred member companies representing computer manufacturers, independent software vendors, and a variety of government and academic organizations. CORBA is a consortium standard, not a "formal" IEEE, ANSI or ISO standard.

The vision behind CORBA is that distributed systems are conceived and implemented as distributed objects. The interfaces to these objects are described in a high-level, architecture-neutral specification language that supports object-oriented design abstraction. CORBA supports distributed systems that feature rapid development, *maintainability* and *adaptability*.

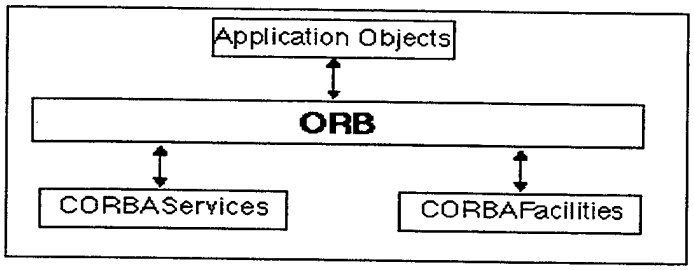


Figure 5. CORBA Architecture Layers (Source: Software Engineering Institute *CORBA Architecture Overview,* 10 Jan 97, p.2.)

Logically the CORBA consists of 4 layers (Figure 5):

- The Object Request Broker (ORB) layer is the core of CORBA and it handles requests for objects.
- The CORBA Common Object Services Layer or CORBAServices supplies the object support service. This layer is fundamental when building non-trivial distributed applications. These services currently include asynchronous event

[10] Kurt Wallnau in his technical paper "Common Object Request Broker Architecture"

management, transactions, persistence, externalization, concurrency, naming, relationships, and lifecycle management.

- CORBA Common Facilities Layer or CORBAFacilities has two sub-layers, the horizontal and the vertical. They may be useful for some distributed applications, but are not as universally applicable as CORBAServices. These facilities include user interface, information management, system management, task management, and a variety of "vertical market" facilities in domains such as manufacturing, distributed simulation, and accounting.

The ApplicationObjects Layer is an extension of the vertical sub-layer of CORBAFacilities layer. Developers must implement this layer for their applications because CORBA offers no standard solution.

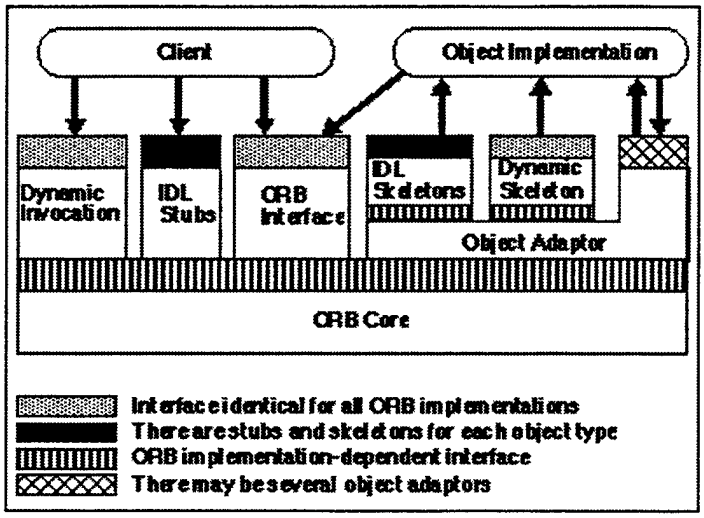


Figure 6. Detailed CORBA Architecture (Source: Software Engineering Institute *CORBA Architecture Overview,* 10 Jan 97, p.3.)

ORB Core is the CORBA runtime infrastructure. The interface to the ORB Core is not defined by CORBA. It is proprietary to a particular vendor. ORB Interface is a

standard interface (defined in IDL) to functions provided by all CORBA-compliant ORBs.

The IDL processor generates IDL stubs for each interface defined in IDL. Stubs hide the low-level networking details of object communication from the client while presenting a high-level, object type-specific application programming interface (API).

Dynamic Invocation Interface (DII) is an alternative way for clients to access objects. While stubs provide an object type-specific API, DII provides a generic mechanism for constructing requests at run time. An interface repository allows some measure of type checking to ensure that a target object can support the request made by the client.

Object Adaptor provides extensibility of CORBA-compliant ORBs to integrate alternative object technologies into the OMA. For example, adaptors may be developed to allow remote access to objects that are stored in an object-oriented database. Each CORBA-compliant ORB must support a specific object adaptor called the Basic Object Adaptor (BOA). The BOA defines a standard API implemented by all ORBs.

IDL Skeletons[11] are the server-side analogue of IDL stubs. IDL skeletons receive requests for services from the object adaptor and call the appropriate operations in the object implementation.

Dynamic Skeleton Interface (DSI) is the server-side analogue of the DII. While IDL skeletons invoke specific operations in the object implementation, DSI defers this

[11] Jason Pritchard in his book "COM and CORBA Side by Side" Jun 1999 p. 50

processing to the object implementation. This is useful for developing bridges and other mechanisms to support inter-ORB interoperation.

Advantages of CORBA are:

- CORBA is a standard architecture for Object Request Brokers. CORBA compliant vendors support portability and interoperability across different programming languages, hardware platforms, operating systems, and ORB implementations.
- When combined with the Object Management Architecture, CORBA can result in distributed systems that can be rapidly developed and can reap the benefits of CORBA like maintainability and adaptability.
- CORBA can cross network, operating systems and programming language boundaries.
- CORBA can support with the current addition of Real-Time Event Service source and type base filtering, event correlations, real-time dispatching and UDP/IP multicast communication. Also with the addition of Scheduling Service, CORBA can support static rate monotonic scheduling and dynamic maximum urgency first scheduling to assign priorities and validate schedulability. These are services that have prevented CORBA from supporting real-time applications and guaranteeing high performance in the past.
- CORBA makes the development of distributed applications easier than with previous technologies.

- CORBA is a mature product with a large and growing number of CORBA implementations available in the marketplace including implementations from most major computer manufacturers and independent software vendors.

The disadvantages of CORBA are:

- CORBA is a complex specification and considerable effort is required to develop expertise in its use.
- CORBA ORB's vary in prices from vendor to vendor and some ORB's are very expensive.
- There is no organization to test in a formal way all aspects of a CORBA implementation, so little information is available.
- Changes to the CORBA specifications while technically justified have resulted in unstable ORB implementations.
- IDL is the "least-common denominator". It does not fully exploit the capabilities of programming languages especially in the definition of abstract data types.
- CORBA specifies only a minimal range of security mechanisms; more ambitious and comprehensive mechanisms have not yet been adopted by the OMG.

JINI is Sun Microsystems' distributed computing solution. JINI is a distributed system that allows a group of computing devices connected by an Intranet or Internet to be used by a group of users as a single computer system. Technically JINI system extends the Java application environment from a single virtual machine to a network of machines. The JINI system is Java centric and assumes that all the co-operating components are

implemented in the Java programming language. It is, however, possible to accept components created in other languages if their compilers can produce Java byte codes.

The high level goals of JINI are:

- Enabling users to share services and resources over the network.
- Providing users easy access to resources anywhere on the network while allowing the user's network location to change.
- Simplifying the task of building, maintaining, and altering a network of devices, software and users

The logical parts of JINI that try to satisfy these goals are:

- A set of components that provides an infrastructure for federating services in a distributed system.
- A programming model that supports and encourages the production of reliable distributed services.
- Services that can be part of a federated JINI system and that offer functionality to any other member of the federation.

JINI services are typically computation, storage, or communication with another service, which is a software filter, a hardware device, or another user. JINI programmers must think in terms of services when they think of a set of servers and clients, users and programs, and programs and files. Users, clients, servers do not exist in JINI; everything is a service. JINI systems provide mechanisms for service construction, lookup, communication and use in a distributed system. Examples of services include devices such as printers, displays, or disks, software applications or utilities, information such as databases and files, and users of the system.

Services in a JINI system communicate to each other using a service protocol which consists of interfaces written in the Java programming language. The set of such protocols is open-ended. The base JINI system defines a small number of such protocols to provide critical service interactions.

The Lookup Service finds and resolves other services. This service is the main mechanism for the interaction between a user and a JINI system. Lookup Service's job is to know the available interfaces of the other services and the methods and functionality that the other services are able to offer. Thus, when a user requests a particular service the lookup service locates the appropriate service to satisfy the user.

Java Remote Method Invocation (RMI) provides the communication among services. In reality it is not a service but an infrastructure that supports the communication among services. RMI provides mechanisms to find, activate, and garbage collect object groups. It provides remote procedures call mechanisms that allow not only the interchange of data among the objects, but also the interchange of whole objects including code for methods.

JINI supports two levels of security. The first level determines if the user of the system who requests the service has the right to use this particular service. The second level checks if a service has the right to request another service; furthermore, the relationships among services are maintained in the access control list responsible for this second level of security.

The access to many services in the JINI system environment is based on leasing. Leasing in JINI means that each object allocates a particular service for particular period of time using predefined rules. JINI offers two kinds of leasing service, exclusive leasing

that means no one else can use the service simultaneously and non-exclusive leasing that allows the reallocation of the same service at the same time from a different user or object.

The JINI Transaction interfaces provide a service protocol to coordinate a two-phase commitment. A series of operations, either within a single service or spanning multiple services, can be wrapped in a transaction. The semantics of a transaction is left up to the service using those interfaces.

The JINI architecture supports distributed events too. An object may allow other objects to register interest in events in the object and receive a notification of the occurrence of the event. This enables distributed event-based programs to be written with a variety of reliability and scalability guarantees.

The components of the JINI system can be segmented into three categories infrastructure, programming model and services.

Advantages of JINI are:

- JINI can cross network and operating systems boundaries.
- JINI has the best methodology for event handling for the communication between objects.
- JINI is a simple technology because it only uses Java's environment.

	Infrastructure	Programming Model	Services
Base Java	Java VM RMI Java Security	Java APIs JavaBeans™ ...	JNDI Enterprise Beans JTS ...
Java + Jini	Discovery/Join Distributed Security Lookup	Leasing Transactions Events	Printing Transaction Manager JavaSpaces™ Service ...

Figure 7. JINI Architecture Segmentation (Source: SUN Microsystems Inc. *JINI Architecture Specification,* Jan 1999, p.12.)

- In JINI it is easy, natural and often automatic for occurrences to join and leave the system.
- JINI systems are currently far more dynamic than other network groups which must be configured by hand in a centralized fashion.
- The model can recognize that the delivery of a distributed notification may be delayed.
- The event and notification interfaces, which are an extension of the event model used by JavaBeans components to the distributed environment, enable event-based communication among JINI services.

The disadvantages of JINI are:

- The JINI architecture gains most of its simplicity by assuming that the Java programming language is the implementation language for components.
- One cannot cross language boundaries with JINI and most legacy applications are not written in Java so they must be rewritten before we can reuse them with JINI.

- JINI is a very new architecture and nobody has ever used it extensively in the real world yet, so it is not mature enough.

F. COMPARISON AMONG THREE ARCHITECTURES

There is no way to provide a complete comparison among these three distributed computing systems. Any comparison depends on the purpose of the comparison and the background of the audience.

Parallels among the main operations of the CORBA, DCOM and JINI include these:

All three support multiple object instantiation, the CORBA and JINI through registration and skeleton instantiation, and DCOM by the server explicitly or dynamically through the COM run-time system.

DCOM uses the Object Remote Procedure Call (ORPC), CORBA the Internet Inter-ORB Protocol (IIOP), and JINI the RMI as their underlying remote protocols.

When a client object needs to activate a server object, DCOM can do it by a method call; on the other hand CORBA offers the same service through naming or trader service, and JINI through a lookup service.

For object handling the client for DCOM uses an interface pointer while CORBA and JINI use the object reference.

The Registry in DCOM maps object names as does the Implementation Repository in CORBA, and the RMIRegistry in JINI.

The type information for methods is held by Type Library in DCOM, by Interface Repository in CORBA, by the object itself in JINI which can be queried by using reflection and introspection.

The responsibility for an object's location and activation falls to Service Control Management (SCM) with DCOM, to Object Request Broker (ORB) for locating and to Object Adapters for activating with CORBA, and to Lookup Service with JINI.

So we can see that the three systems have very similar operations. Other similarities are:

- They have complex ways to define the interface of their components.
- They require that the user know networking to set up remote services.
- They can cross the network boundaries.
- They offer a low level of security.

CORBA is the most complete of the three. First, it is independent of language and operating system. JINI can support only components that are implemented with Java and DCOM works properly only in a Windows OS environment. Second, CORBA with the addition of real-time event service and scheduling can be reliable enough and can offer satisfactory performance. JINI may equal CORBA in this regard, but DCOM does not. Third, CORBA is mature enough. Many applications have already been implemented with it. DCOM is also mature, but JINI is not. CORBA is an open standard. JINI will be offered as part of JAVA and DCOM as part of Windows. Of course, CORBA is not the perfect solution for implementers of a distributed component system because of the high level of effort and training required for success.

III. SOFTWARE REQUIREMENTS SPECIFICATION

A. OVERVIEW

The purpose of this project is to redesign and re-implement the CAMPEX Employment Module with a distributed architecture using the object-oriented methodology. The final product must offer to the Air University of United States Air Force a tool that allows students to improve their skills in air campaign design through practice. Also, this software must allow students to execute exercises from their personal computers and transfer the results of their practice to the war college server.

B. CUSTOMERS

The customers are resident students, nonresident students and instructors at the Air University.

C. GOALS

The ultimate goal is to offer an application to Air War College students that allows them to execute exersices in air campaign planning. This application will also allow instructors at the Air University to check the students' results. This application must be capable of running on the students personal computers as well as on the Air University computers. Thus, the CAMPEX Employment Module must offer:

- Fast and easy downloading.
- Fast and easy installation on computers that runs Microsoft Windows Operating System independent of the configuration of the students' machines.

- Selection of the tactical exercise scenario.
- Cues to the user for correctly sequencing events.
- Information to the user about necessary assumptions used during the exercise.
- Creation and editing of air tasking orders (ATOs).
- Planning of the missions.
- Automatic update of the game state with the execution of a game cycle.
- Creation of reports with the results and estimations.
- The ability to return to a previous state.
- Display of the map of the exercise area.
- Analysis of the student's plan.
- Printing of reports, results, and information lists.

D. USER CHARACTERISTICS

The typical user of the CAMPEX Employment Module requires special education in the air campaign planning. In addition, the user is expected to have a medium or low level of familiarity with the Windows operating system and the Internet. If the user has these characteristics then, he will be able to use the module after a brief training session that will take approximately two to three hours.

E. GENERAL CONSTRAINTS

The CAMPEX Employment Module must be a fully autonomous system capable of functioning for extended periods of time with minimal support. The CAMPEX prototype will be provided in an ACCESS-2000 runtime-software package that will

include all the required bindings. This CAMPEX Employment Module will depend on the Windows Operating system and Microsoft Office commercial product.

F. ASSUMPTIONS AND DEPENDENCIES

The following assumptions and dependencies have been defined in order to simplify the analysis of the CAMPEX Employment Module and the implementation of the CAMPEX Employment Module prototype.

1. The user (student) must know how to operate a personal computer and more specifically how to use a personal computer that runs Microsoft Windows operating system.
2. In this initial phase only the employment module prototype will be implemented.
3. The software application will fully support the basic functionality of the CAMPEX Employment Module and partially support the optional functionality.
4. The user must be connected to the Internet to execute the "Send Exercise" function.

G. SYSTEM FUNCTIONS

Ref#	Function	Use Case	Category
R1	**Student Support Functions**		
R1.1	Creates and displays new Student Card	U1	evident
R1.2	Stores new Student attributes into the objects repository (database)	U1	hidden

Ref#	Function	Use Case	Category
R1.3	Displays existed Students	U1	evident
R1.4	Displays current student's attributes	U1	evident
R1.5	Logs attributes changes to a student's object	U1	evident
R1.6	Changes the object student attributes	U1	hidden
R1.7	Searches student objects instantiations with input key	U1	hidden
R2	**Send Exercise Functions**		
R2.1	Displays existent Exercises objects	U2	evident
R2.2	Queries Exercises objects with input key	U2	hidden
R2.3	Outputs the collected exercise attributes to Air University database	U2	hidden
R2.4	Displays message that informs student for the success of the process	U2	evident
R3	**Starts a CAMPEX module (Employment) Functions**		
R3.1	Retrieves the "Copyright Screen" from database	U3	hidden
R3.2	Displays the "Copyright Screen"	U3	evident
R3.3	Retrieves the "Execute Order" from database	U3	hidden
R3.4	Displays the "Execute Order"	U3	evident
R3.5	Provides inter-process communication mechanisms	U3	evident
R3.6	Retrieves the "DIA Intel Update" from database	U3	hidden
R3.7	Displays the "DIA Intel Update"	U3	evident
R3.8	Retrieves record the "Thai Forces Available" from database	U3	hidden

Ref#	Function	Use Case	Category
R3.9	Displays the "Thai Forces Available" List	U3	evident
R3.10	Retrieves the "Weather Report" from database	U3	hidden
R3.11	Displays the "Weather Report"	U3	evident
R3.12	Retrieves the "Navy Update" from database	U3	hidden
R3.13	Displays the "Navy Update"	U3	evident
R3.14	Retrieves the "Weapon Availability Update" from database	U3	hidden
R3.15	Displays the "Weapon Availability Update"	U3	evident
R3.16	Retrieves record of "Program Notes" from database	U3	hidden
R3.17	Displays "Program Notes"	U3	evident
R3.18	Retrieves the of "Bomb Damages Assessment and Targets Definitions" from database	U3	hidden
R3.19	Displays the "Bomb Damages Assessment and Targets Definitions"	U3	evident
R3.20	Retrieves the "Analysis and Corrections" from database	U3	hidden
R3.21	Displays the "Analysis and Corrections"	U3	evident
R3.22	Retrieves the "Read me file for Employment Module" from database	U3	hidden
R3.23	Displays the "Read me file for Employment Module" text	U3	evident
R4	**ATO Support Functions**		
R4.1	Initiates a new ATO object with the name given by user	U4	evident
R4.2	Loads an existent ATO	U4	evident

Ref#	Function	Use Case	Category
R4.3	Copy an existent ATO to a new one with a name given by user	U5	evident
R4.4	Renames an existent ATO	U4	evident
R4.5	Erases an existent ATO	U4	evident
R4.6	Displays an ATO enter form to enter ATO attributes	U4	evident
R5	**Support Planning Missions Functions**		
R5.1	Enters a target to the priority list	U6	evident
R5.2	Decreases the priority of the existed targets with lower priority than the specified priority by one	U6	evident
R5.3	Increases the priority of the existed targets with higher priority than the specified priority by one	U6	evident
R5.4	Deletes the targets with priority lower than 20	U7	evident
R5.5	Creates and displays a new mission card	U7	evident
R5.6	Stores new mission record to database	U7	hidden
R5.7	Displays a mission object	U7	evident
R5.8	Logs attributes to a mission object	U7	evident
R5.9	Search Missions objects with input key (mission or package)	U7	hidden
R5.10	Deletes a Mission object	U7	evident
R6	**Fly an ATO Support Functions**		
R6.1	Executes the assigned missions and packages	U8	hidden
R6.2	Collects all assigned missions, packages, and actual data objects	U8	hidden
R6.3	Calculates the collected objects	U8	hidden

Ref#	Function	Use Case	Category
R6.4	Creates a new ATO state	U8	evident
R6.5	Save the calculation's results as attributes of the new ATO state	U8	hidden
R7	**Initial Information Support Functions**		
R7.1	Queries data base for Blue Bases objects	U9	hidden
R7.2	Displays total Basing Information	U9	evident
R7.3	Queries database for sorties attributes	U9	hidden
R7.4	Calculates the queries' results	U9	hidden
R7.5	Displays available sorties by Blue Bases	U9	evident
R7.6	Queries database for the necessary for analysis objects	U9	hidden
R7.7	Calculates the queries' results	U9	hidden
R7.8	Displays the missions' analysis	U9	evident
R7.9	Queries the database for Ground Forces objects	U9	hidden
R7.10	Displays the collected by the queries objects	U9	evident
R8	Estimated Results Support Functions		
R8.1	Queries database for Recce by target	U10	hidden
R8.2	Displays Recce by targets for the current program state	U10	evident
R8.2.1	Queries database for missions and actual information objects	U10	hidden
R8.3	Calculates the collected objects	U10	hidden
R8.4	Displays estimated results of planned missions before the missions execution	U10	evident
R8.5	Queries database for missions that have the attribute sorties = empty	U10	hidden
R8.6	Displays missions without sorties	U10	evident

Ref#	Function	Use Case	Category
R8.7	Queries database for necessary objects to calculates the "daily summary"	U10	hidden
R8.8	Calculates the queries (R8.7) results to create daily summary	U10	hidden
R8.9	Displays daily summaries by Aircraft Type	U10	evident
R8.10	Displays daily summaries by Task Type	U10	evident
R8.11	Queries database for necessary objects attributes to create the "logistics requirements"	U10	hidden
R8.12	Calculates the results of the (R8.11) queries to create the report	U10	hidden
R8.13	Displays logistics requirements by Blue Base and Supply category	U10	evident
R9	**Actual Results Support Functions**		
R9.1	Queries database for necessary objects to create "the cumulative summary report"	U11	hidden
R9.2	Calculates the collected objects attributes of query (R9.1)	U11	hidden
R9.3	Displays cumulative summary with the end of past date	U11	evident
R9.4	Queries database for the necessary objects for the report "Recce Targets at the start of the current Date"	U11	hidden
R9.5	Calculates the results of query (R9.5)	U11	hidden
R9.6	Displays Recce for Targets at the start of the current date	U11	evident
R9.7	Queries the database for the necessary objects to create the report "Enemy Planes over Blue Bases during the Past Day"	U11	hidden

Ref#	Function	Use Case	Category
R9.8	Calculates the results of query (R9.7)	U11	hidden
R9.9	Displays the number of enemy planes over the Blue Bases during the past date	U11	evident
R9.10	Queries the database for the necessary objects to create the report of "Overall Indicators of Sorties at the end of the Past Date"	U11	hidden
R9.11	Calculates the results of query (R9.10)	U11	hidden
R9.12	Displays overall indicators of sorties at the end of the past date	U11	evident
R9.13	Queries the database for the necessary objects to create the report of "Overall Indicators of Effort Weight at the End of the Past Date "	U11	hidden
R9.14	Calculates the results of query (R9.13)	U11	hidden
R9.15	Displays overall indicators of effort weight at the end of the past date	U11	hidden
R9.16	Queries the database for the necessary objects to create the report of "Overall Indicators of Blue Attrition at the End of the Past Date"	U11	hidden
R9.17	Calculates the results of query (R9.16)	U11	hidden
R9.18	Displays overall indicators of Blue Attrition at the end of the past date	U11	evident
R9.19	Queries the database for the necessary objects to create the report of "Overall Indicators of Loss Ratio at the End of the Past Date "	U11	hidden
R9.20	Calculates the results of query (R9.19)	U11	hidden

Ref#	Function	Use Case	Category
R9.21	Displays overall indicators of loss ratio at the end of the past date	U11	evident
R9.22	Queries the database for the necessary objects to create the report of "Losses by Mission and by Task Type During the Past Date"	U11	hidden
R9.23	Calculates the results of query (R9.22)	U11	hidden
R9.24	Displays losses by mission and by task type during the past date	U11	evident
R9.25	Queries the database for the necessary objects to create the report of "Losses by Mission and by Aircraft Type During the Past Date "	U11	hidden
R9.26	Calculates the results of query (R9.25)	U11	hidden
R9.27	Displays losses by mission and by aircraft type during the past date	U11	evident
R10	**Support Functions of General Purpose**		
R10.1	Loads the ATO Management screen	U3	optional
R10.2	Loads the program Main Menu screen	U4, U5	optional
R10.3	Displays Operations Area Map	U12	optional
R10.4	Quit the program	None	optional
R10.5	Changes time of displaying messages on the screen 1-5 sec	None	optional
R10.6	Changes color of the screen	None	optional
R10.7	Locked keypad to arrows Yes/No	None	optional
R10.8	Convert box characters Yes/No	None	optional
R10.9	Showing bombing on map Yes/No	None	optional

Ref#	Function	Use Case	Category
R10.10	Highlights steel for LCD/Mono screen Yes/No	None	optional
R10.11	Displays warning messages	ALL	optional
R10.12	Displays reminder messages	ALL	optional
R 10.13	Displays additional explanation messages	ALL	optional
R10.14	Displays interactive messages	ALL	optional

Table 1. System Functions

H. SYSTEM ATTRIBUTES

Attribute	Details and Boundary Constraints
Response time	When entering new values, the system must give the opportunity to enter the next item in 1 sec.
Response time	When user asks to see a report or information list, system must displays the report on the screen in 5 sec
Response time	System must execute an ATO and moves to the next state in 5 sec
Response time	System must start printing the reports in 15 sec
Interface metaphor	Forms-metaphor windows and dialog boxes
Operating system platform (initially)	Microsoft Windows 95 and NT

Table 2. System Attributes

I. USE CASES

1. High Level Use Cases

- U1: Start Employment Module
- U2: Student Info
- U3: Load an ATO
- U4: Manage an ATO
- U5: Describes the 20 Targets with Highest Priority
- U6: Plan an ATO
- U7: Fly an ATO
- U8: Initial Information
- U9: Estimated Results
- U10: Actual Results
- U11: Map
- U12: Send Exercise

USE CASE (U1): START EMPLOYMENT MODULE

Actors:	Student
Purpose:	Start the Employment Module
Overview:	Student selects to use the CAMPEX Employment Module, the program displays the Introduction Reports and "Ground Forces Report". The student can select to see only the "Ground Forces Report" and as alternative to continue with the program.
Type:	Primary and essential
Cross References:	R3, R3.1, R3.2, R3.3, R3.4, R3.5, R3.6, R3.7, R3.8, R3.9, R3.10, R3.11, R3.12, R.3.13, R3.14, R3.15, R3.16, R3.17, R3.19, R3.20, R3.21, R3.22, R7.9, R7.8

USE CASE (U2): STUDENT INFO

Actors: Student

Purpose: Student identifies himself

Overview: Student enters his Personal Information in the CAMPEX. If he has already entered his personal information, he just selects his own name.

Type: Primary and essential

Cross References: R1.1, R1.2, R1.3, R1.4, R1.5, R1.6, R1.7, R10.11, R10.12, R10.13, R10.14, R10.15

USE CASE (U3): LOAD AN ATO

Actors: Student

Purpose: Load an ATO

Overview: Student selects an ATO to load. When completed, student will enter the "Main Menu" and the selected ATO is loaded.

Type: Primary and essential

Cross References: R4.2, R10.1, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed the

- "Start Employment Module" Use Case (U1)
- "Student Info" Use Case (U2)

USE CASE (U4): MANAGE AN ATO

Actors: Student

Purpose: To allow student to manage to the ATO's

Overview:	Student wants to manage to an ATO. When completed, student will return in the "ATO File Management" menu and, he can continue by loading an ATO.
Type:	Primary and essential
Cross References:	R4.1, R4.3, R4.4, R4.5, R4.6, R10.1, R10.11, R10.12, R10.13, R10.14 Use Cases: Student must have completed the • "Start Employment Module " Use Case (U1) • "Student Info" Use Case (U2)

USE CASE (U5): DESCRIBE THE 20 TARGETS WITH HIGHEST PRIORITY

Actors:	Student
Purpose:	Fill the list with 20 targets with highest priority
Overview:	Student has decided which targets of his plan have the highest priority. Student edits the list of 20 targets with highest priority. After the completion of this use case, the student's "20 Highest Priority Target List" can be displayed by the application.
Type:	Primary and essential
Cross References:	R5.1, R5.2, R5.3, R5.4, R10.2, R10.11, R10.12, R10.13, R10.14 Use Cases: Student must have completed: • "Start Employment Module" Use Case (U1) • "Student Info" Use Case (U2)

- "Load an ATO" Use Case (U3)

USE CASE (U6): PLAN AN ATO

Actors:	Student
Purpose:	Enters student's plans
Overview:	Student enters new missions or edits old missions. With completion of this use case the student's plans have been entered.
Type:	Primary and essential
Cross References:	R5.5, R5.6, R5.7, R5.8, R5.9, R5.10, R10.2, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)
- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)

USE CASE (U7): FLY AN ATO

Actors:	Student
Purpose:	To execute the planned missions
Overview:	Student is running fly missions. System calculates the result of the planned missions. Saves the results in a new ATO. Loads the new ATO.
Type:	Primary and essential
Cross References:	R6.1, R6.2, R6.3, R6.4, R6.5, R10.2, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use case (U1)
- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)
- "Plan an ATO" Use Case (optional) (U6)

USE CASE (U8): INITIAL INFORMATION

Actors:	Student
Purpose:	Inform student for the initial data of an ATO
Overview:	Student asks for initial information. With completion of this use case, the student has seen or printed the information that he has asked for.
Type:	Primary and essential
Cross References:	R7.1, R7.2, R7.3, R7.4, R7.5, R7.6, R7.7, R7.8, R7.9, R7.10, R10.2, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)
- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)

USE CASE (U9): ESTIMATED RESULTS

Actors:	Student
Purpose:	Display the estimated results of the student plan before executing the plan

Overview: Student asks for the estimated results. With completion of this use case estimated results are displayed on the screen.

Type: Primary and essential

Cross References: R8.1, R8.2, R8.3, R8.4, R8.5, R8.6, R8.7, R8.8, R8.9, R8.10, R8.11, R8.12, R8.13, R10.2, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)
- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)
- "Plan an ATO" Use Case (optional) (U6)

USE CASE (U10): ACTUAL RESULTS

Actors: Student

Purpose: Inform student of the Results Reports of an ATO

Overview: Student asks for information. With completion of this use case, the student has seen or printed the information that he has asked for.

Type: Primary and essential

Cross References: R9, R9.1, R9.2, R9.3, R9.4, R9.5, R9.6, R9.7, R9.10, R9.11, R9.12, R9.13, R9.14, R9.15, R9.16, R9.17, R9.18, R9.19, R9.20, R9.21, R9.22, R9.23, R9.24, R9.25, R9.26, R9.27, R10.2, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)

- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)
- "Fly an ATO" Use Case (U7)

USE CASE (U11): MAP

Actors:	Student
Purpose:	See the map of the exercise area
Overview:	Student selects to see the map via the Main Menu. When completed, the map is displayed on the screen.
Type:	Primary and essential
Cross References:	R10.2, R10.3

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)
- "Load an ATO" Use Case (U3)

USE CASE (U12): SEND EXERCISE

Actors:	Student
Purpose:	To send the results of an exercise to "Air University"
Overview:	Student must be connected to the "internet" first. Then, selects the Option "Send Exercise Results to the Air University" and selects an exercise from the displayed. With the completion of this use case, the selected exercise results are sent to the Air University server.
Type:	Primary and essential

Cross References: R2.1, R2.2, R2.3, R2.4, R10.15

Use Cases: User must have completed the

- "Student Info" Use Case (U2)
- User must have connected to the Internet

2. CAMPEX Employment Module Use Case Diagram

User starts the CAMPEX by identifying himself, and by this way the results of the exercise execution will have an owner.

Then the user selects to start the CAMPEX Employment module.

The last use case that is described by the CAMPEX Employment Module use case diagram is "Send Exercise." With this use case, the user can send the results of an exercise to the Air War College server. The sent results have the user ID and the Air War College Instructors can identify the owner.

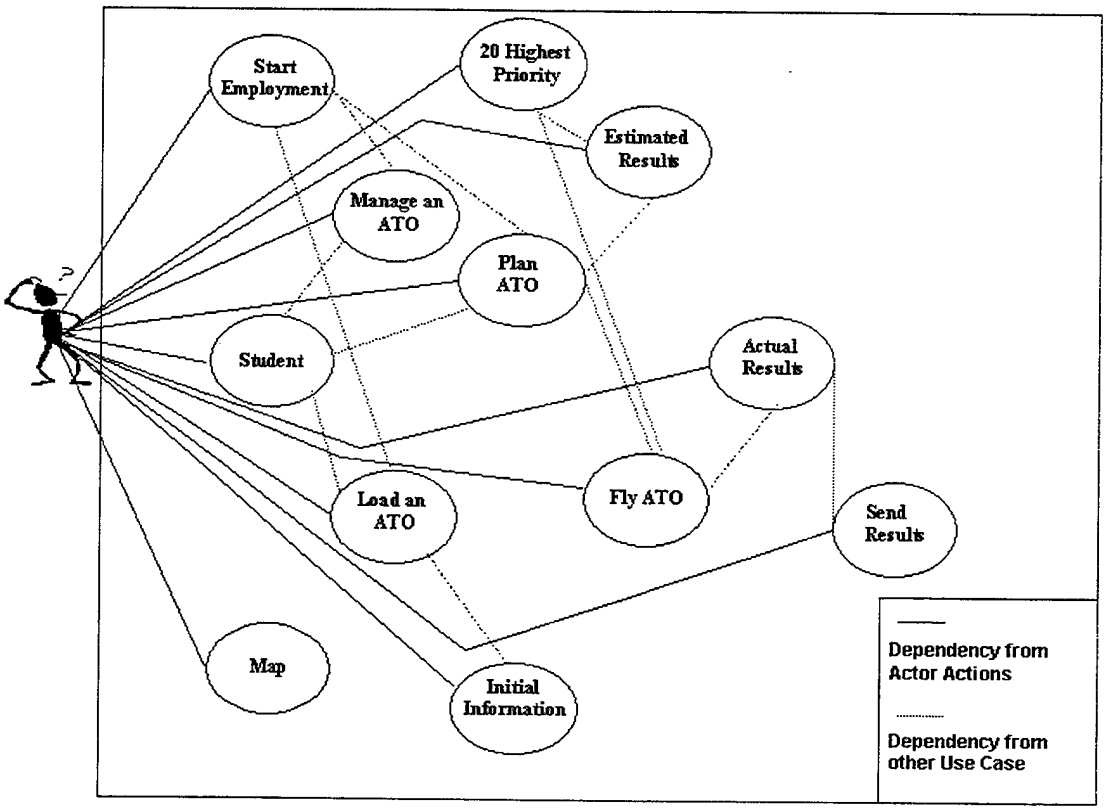


Figure 8. Use Cases Diagram

J. RANKING USE CASES

Rank	Use Case	Justification
High	Load an ATO (U3)	Important because it initiates the Employment module.
	Fly ATO (U7)	Most important and highest risk process.
	Plan an ATO (U6)	Important because it allows the student enter his plan.
Medium	Describes the 20 targets with highest priority (U5)	Important because it allows the student to enter his plan

Rank	Use Case	Justification
Medium	Student Info (U2)	Affects the identification of the exercise results.
	Start CAMPEX Employment Module (U1)	Affects the initial phase of the CAMPEX
	Send Exercise (U12)	Affects the process of ranking the exercises.
	Manage an ATO (U4)	Affects the initial phase of the module.
	Initial Information (U8)	Informs student about the current data.
	Estimated Results (U9)	Informs student about the changes that will happen to the data.
	Actual Results (U10)	Informs student about the new state data.
Low	Map (U12)	Minimum effect on the processes

Table 3. Ranking Use Cases

K. CONCEPTUAL MODEL

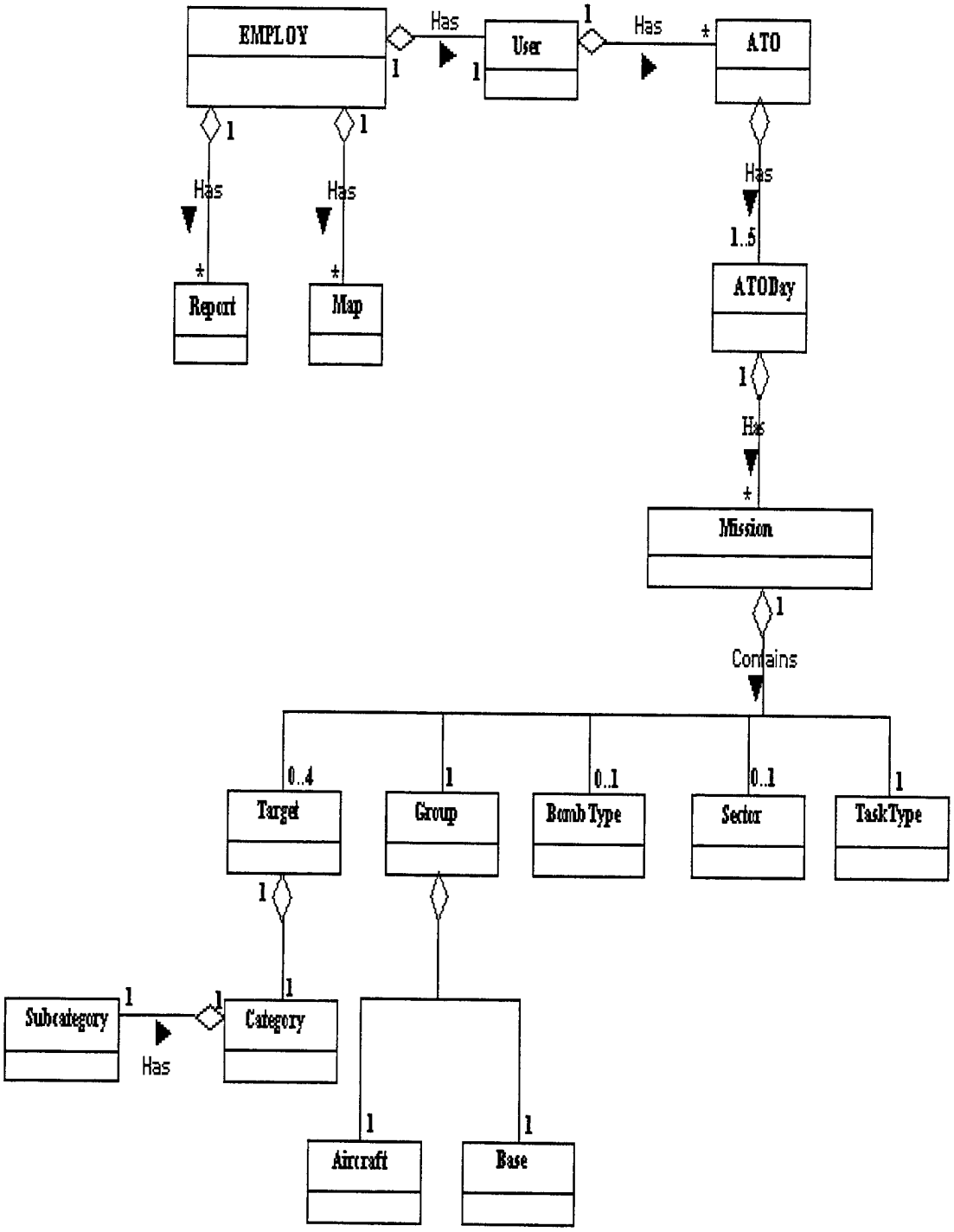


Figure 9. Conceptual Model.

IV. SOFTWARE DESIGN SPECIFICATION (SDS)

A. INTRODUCTION

1. Purpose

This chapter specifies the design for the Campaign Planning Exercises (CAMPEX) Employment Module and presents the interaction (collaboration) and object diagrams that describe the overall CAMPEX Employment Module software.

2. Scope

This SDS provides extensive information concerning the designed and proposed functionality of the CAMPEX Employment Module. This chapter describes the subsystems that comprise the CAMPEX Employment Module. The sequence diagrams describe the individual CAMPEX object interactions via messages/methods. The object diagrams illustrate the specifications for software classes and the interfaces for the CAMPEX Employment Module.

3. Definitions, Acronyms, and Abbreviations

All definitions, acronyms, and abbreviations are included in Appendix C: "Abbreviations, Acronyms, and Definitions". Abbreviations, acronyms, and definitions from the previous chapter have been carried forward for consistency.

B. THE SYSTEM ARCHITECTURE

1. The Detailed Architecture Diagram

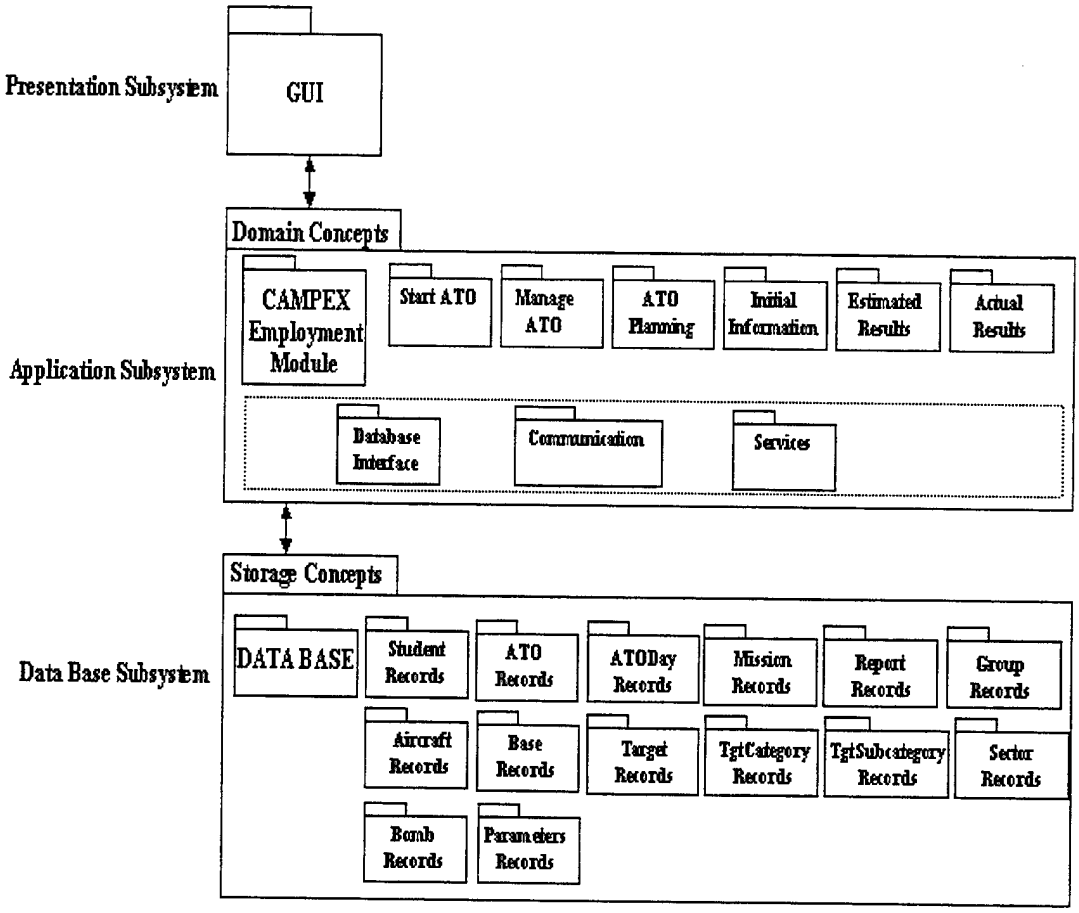


Figure 10. Detailed Architecture Diagram.

The CAMPEX Employment Module has three subsystems. The Presentation Subsystem/Layer contains the Graphical User Interfaces (GUIs). The Application Subsystem/Layer contains the CAMPEX Employment Module high level object-oriented services, the services for communication with external devices and interface to the local

and remote database (Server of the Air War College). The Storage Subsystem contains the actual database.

a. Presentation Subsystem

- *Object Class*: Graphical User Interface.
- *Interface to Other Subsystems*: Application Subsystem.
- *Human Interface*: The GUI provides the only interface to the user of the CAMPEX employment Module. Chapter V User Manual provides a pictorial representation of the CAMPEX Employment Module human interfaces.
- *Overall Control Structure*: The GUI is consisting of a number of Microsoft ACCESS 2000 Forms and Sub-forms.
- *Resource Allocation*: The GUI first allocates resources that support the system screen and then allocates part of the system processing power and data storage space.
- *Data Stores and Management*: The GUI subsystem does not store or manage data; it has no direct access to data, but it does offer a communication channel between the user and the Application Subsystem.
- *Global Resources and Management*: The management of the global resources associated with the three CAMPEX Employment Module Subsystems is conducted through a time-sharing approach. Resources

are not shared equally among all the subsystems. The GUI uses most of the resources that support the screen operation, for example.

- *Boundary Conditions*: None.
- *Constraints*: The GUI of CAMPEX Employment Module cannot be made autonomous from the other subsystems of the CAMPEX Employment Module. The entire CAMPEX Employment Module prototyping will be provided in a Microsoft ACCESS 2000 runtime software package that includes all required bindings.
- Trade-Off Priorities: None.
- Design Decision /Rationale:
  - The GUI provides the only external interface to CAMPEX Employment Module.
  - ACCESS 2000 will be used in the GUI Subsystem.
  - All Forms and Sub-Forms are considered essential.

b. Application Subsystem

- Object Class: See Figure 11.
- *Interface to Other Subsystems*: Presentation Subsystem.
- Human Interface: None
- *Overall Control Structure*: The CAMPEX Employment Module prototype uses sequential method to control its tasks, with one active object to monitor and control all tasks. The CAMPEX Employment

Module is procedurally driven; it uses fixed procedural loops to control the system.

- *Resource Allocation*: Allocation of resources is focused towards the timed processes required to monitor the GUI. A process that also sends objects to the Air War College allocates communication resources.
- *Data Stores Management*: The Data Base Subsystem is responsible for the management and storage of the data. The Application Subsystem has direct access to the data storage resources.
- *Global Resources and Management*: The objects within the Application Subsystem share the resources equally.
- *Boundary Conditions*: Startup, shutdown, termination, and failure should be performed/investigated by the system user. Details are provided in the user's manual in Chapter V.
- *Constraints*: Employment Module prototype can be used as fully autonomous system when packaged as a Microsoft ACCESS 2000 run-time including all the required bindings. Otherwise it can only work through Microsoft ACCESS 2000. Furthermore, the user must establish an Internet connection first to communicate and send information to the Air War College Server.
- Trade-Off Priorities: None
- Design Decision/Rationale:
    - Efforts were taken to minimize the use cases but the number was constrained by the required functionality.

- The CAMPEX Employment Module fully supports all physically challenged persons.
- Visual Basic and SQL will be used through Microsoft ACCESS 2000 in the Application Subsystem.
- All functions are considered essential.
- For the communication with the Air War College Server a DSN address will be used.

c. *Storage Subsystem*

- Object Class: Data Base
- *Interface to Other Subsystems*: Application Subsystem.
- Human Interface: None
- *Overall Control Structure*: The Data Base consists of a number of tables and various types of queries.
- *Resource Allocation*: The Data Base uses about 10 MB of hardware storage in the system at any time and allocates additional space for queries.
- *Data Stores and Management*: Microsoft ACCESS 2000 handles the storage for the Data Base.
- *Global Resources and Management*: As described above in Resource Allocation.
- *Boundary Conditions*: None.

- *Constraints*: Data Base is constrained by the ACCESS 2000 general constraints.
- Trade-Off Priorities: None.
- Design Decision/Rationale:
  - ACCESS 2000 will be used in the Data Base Subsystem.
  - DSN address describes the address of the Air War College Server.
  - All functions are considered essential.
  - The user has no direct access to the Data Base.

C. OBJECT/CLASS DIAGRAMS

1. Object Diagram

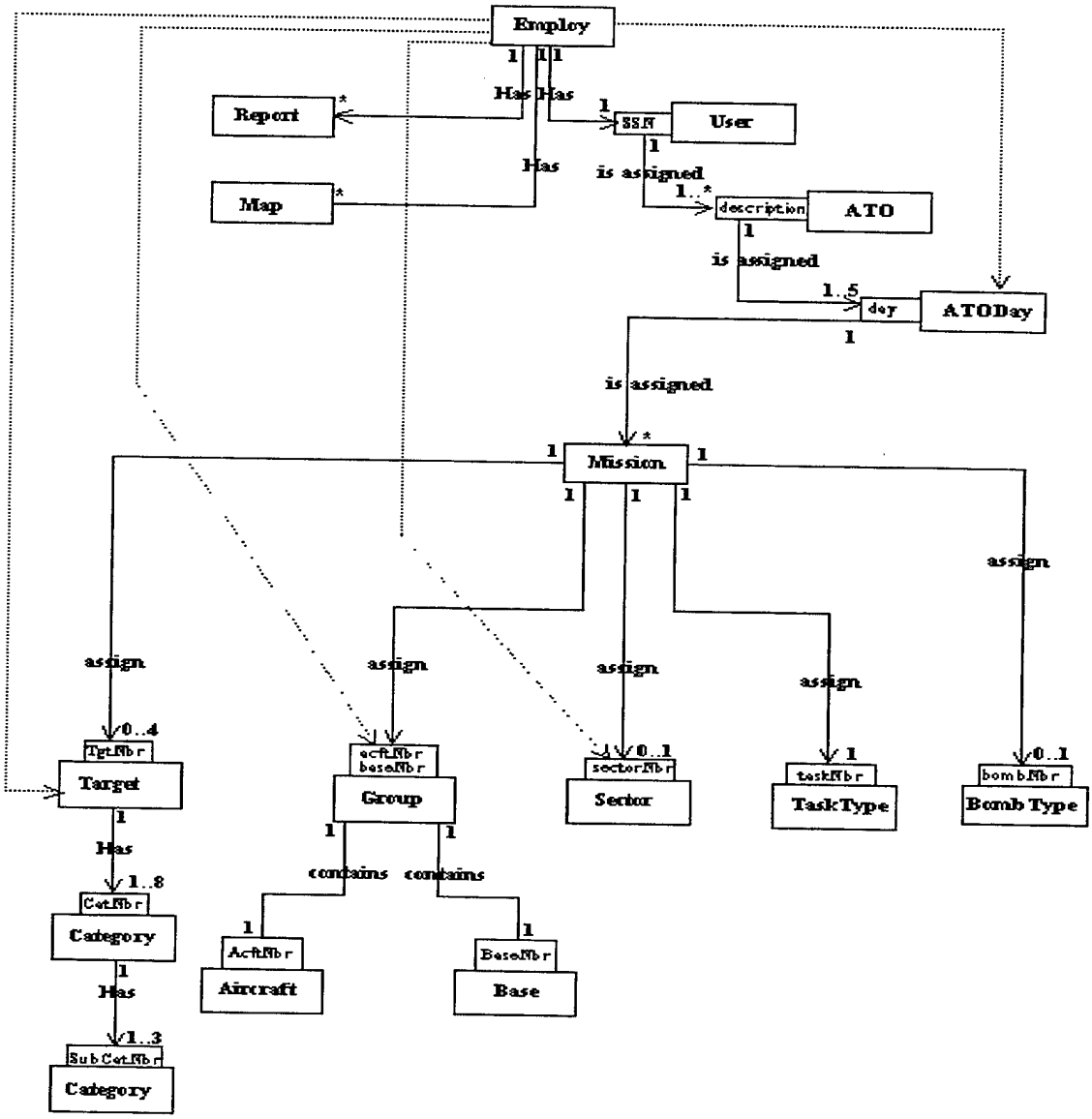


Figure 11. CAMPEX Employment Module Object.

2. Classes-Objects Attributes and Operations

a. Class Employ

1/ Attributes

Attribute	Type	Description
Hi_5Damage	Integer	Parameter for calculation
Lo_5Damage	Integer	Parameter for calculation
Hi_7Damage	Integer	Parameter for calculation
Lo_7Damage	Integer	Parameter for calculation
NotFireAt	Integer	Parameter for calculation
DogFightBasic	Integer	Parameter for calculation
KillRatioBad	Integer	Parameter for calculation
KillRatioGood	Integer	Parameter for calculation
MinWorstRatio	Integer	Parameter for calculation
MinBstRatio	Integer	Parameter for calculation
PercentRedDCA	Integer	Parameter for calculation
PercentRedOCA	Integer	Parameter for calculation
PercentRedFlying	Integer	Parameter for calculation
MaxPercentRedAcftLost	Integer	Parameter for calculation
RedAbort	Integer	Parameter for calculation
RedFEBALoss	Integer	Parameter for calculation
RedFtrLoss	Integer	Parameter for calculation
RedTermLoss	Integer	Parameter for calculation
RedNotFind	Integer	Parameter for calculation
DoAndShowArmyMsns	Integer	Parameter for calculation
ShowEnemyBmbrsOverBlueBases	Integer	Parameter for calculation
TightenessFactor	Integer	Parameter for calculation
AIMfudgeFactor	Integer	Parameter for calculation

Attribute	Type	Description
PercentDailyDegradeGnd	Integer	Parameter for calculation
AcftGndEquivFactor	Integer	Parameter for calculation
GndDiffTomoveFEBA	Integer	Parameter for calculation

Table 4. Class Employ–Attributes

2/ Operations

Operation	Input	Output
initStudentServicesGUI	None	Student (all existed)
initSelectATOGUI	None	ATODay (all existed)
InitCopyATOGUI	None	ATODay (all existed)
init20TgtHighestPriority	None	Targets (all existed, displays the first 20 with higher priority)
InitPlanEditMsns	None	m: Mission (all existed)
InitDeletePackage	None	p: Integer
SelectStudent	SSN	None
AddStudent	SSN, rank, firstName, lastName, country, address, e-Mail, section	None
getIntroReport	None	Report (all reports existed)
loadATODay	SSN, description, day	None
AddATO	SSN: Text, description: Text	None
selectATOToCopy	SSN: Text, description1: Text, description2: Text	None

Operation	Input	Output
eraseATO	SSN: Text, description: Text	None
modifyTgt	SSN: Text description: Text day: Integer tgtNbr: Integer priority: Integer	None
enterMsn	st: Student, a: ATO ad: ATODay taskType: Integer acftType: Integer base: Integer sector: Integer tgtCategNbr: Integer tgtSubNbr: Integer bmbTypeNbr: Integer nbrOfSorties: Integer package: Integer	None
deleteMsn	SSN: Text description: Text day: Integer msnNbr: Integer	None
deletePackage	SSN: Text description: Text day: Integer package: Integer	None
flyATO	None	None
estimatedResults	None	r: Report

Operation	Input	Output
flightsWithoutSorties	None	r: Report
dailySummary	None	r: Report
logisticsRequirements	None	r: Report
blueBasingSum	None	r: Report
recceTgt	None	r: Report
analysis	None	r: Report
sortiesAvailable	None	r: Report
cumulativeSum	None	r: Report
enemyOverBlueBase	None	r: Report
groundWarSummary	None	r: Report
measuresOfMerit	None	r: Report
yesterdayLossesByAcft	None	r: Report
yesterdayLossesByTask	None	r: Report
selectMap	mapNbr: Integer	map: Map
selectATOToSend	SSN: Text, description1: Text, description2: Text	None
updateStudent	st: Student selection: Boolean	st: Student
addStudent	st: Student	st: Student
updateReport	r: Report g: GroundUnit:=Null	None
updateATODay	ad: ATODay selection: Boolean	None
deleteATOday	ad3: ATODay	None
addGroup	ad: ATODay g: Group	None
addSector	ad: ATODay s: Sector	None

Operation	Input	Output
addTgt	ad: ATODay t: Target	None
addMsn	ad: ATODay m: Mission	None
checkMsn	m: Mission	exist: Boolean
clacMsnLosses	m: Mission	l: Integer Array
updateMsn	m: Mission ad: ATODay l: Integer Array	None
calculateEstimatedResults	m: Mission	er: Integer Array
updateReport	r: Report m: Mission er: Integer Array	None
calcLogReq	m: Mission	lr: Integer Array
calculateAnalysis	m: Mission	ana: Integer Array
calculateSortiesAvailable	m: Mission g: Group	as: Integer
calculateCumSumP1	m: Mission	cs1: Integer Array
calculateCumSumP2	g: Group	cs2: Integer Array
calculateCumSumP3	t: Target	cs3: Integer Array
calculateEnemyOverBlueBases	b: Base g: Group	eobb: Integer
calcSector	m: Mission	sl: Integer
calclateOverIndicators	m: Mission m: Mission	oi: Integer Array
calculateLosses	m: Mission m: Mission	los: Integer
addStudent	st: Student	None

Operation	Input	Output
updateReport	r: Report gu: Null	None

Table 5. Class Employ–Operations

b. Class Student

1/ Attributes

Attribute	Type	Description
SSN	Text	Unique number for each student
Rank	Text	Rank of Student
First Name	Text	First name of the Student
Last Name	Text	Last name of the Student
Country	Text	Country of the Student
Address	Text	Address of the Student
Section	Integer	Section of the Student
E-Mail	Text	E-mail
Selection	Boolean	True for the current Student

Table 6. Class Student–Attributes

2/ Operations

Operation	Input	Output
getStudentAll	None	St: Student
getStudentSel	selection : Boolean	St: Student
getStudent	SSN: Text	St: Student
checkStudent	SSN: Text	Exist: Boolean

Table 7. Class Student–Operations

c. *Class ATO*

1/ Attributes

Attribute	Type	Description
Description	Text	The ATO name
Selection	Boolean	True for the current ATO

Table 8. Class ATO–Attributes

2/ Operations

Operation	Input	Output
getATO	st: Student	a: ATO (all existed)
addATO	st: Student description: Text	None
getATO	st: Student description: Text	a: ATO
getATO	st: Student selection: Boolean	a: ATO
deleteATO	st: Student description: Text	None

Table 9. Class ATO–Operations

d. *Class ATODay*

1/ Attributes

Attribute	Type	Description
Day	Integer	The ATODay name
Selection	Boolean	True for the current ATODay

Table 10. Class ATODay–Attributes

2/ Operations

Operation	Input	Output
getATODay	a: ATO	ad: ATODAY
getATODaySel	selection: Boolean	ad: ATODAY
getATODayDay	a: ATO day: Integer	ad: ATODAY
addATODay	a: ATO day: Integer	None
deleteATOdAY	a: ATO day: Integer	None
getATODaySelandATO	a: ATO selection: Boolean	ad: ATODAY

Table 11. Class ATODay–Operations

e. Class Mission

1/ Attributes

Attribute	Type	Description
MsnNbr	Integer	The number of Mission
Package	Integer	The package number of the Mission
NbrOfSorties	Integer	The number of sorties assigned in the Mission
Priority	Integer	The priority for execution, describe by the Task Type that assigned to the Mission and by the user selection in 20 highest priority list
SortieRate	Integer	Parameter for calculation

Attribute	Type	Description
blueBombLoadQuantity	Integer	Parameter for calculation

Table 12. Class Mission–Attributes

2/ Operations

Operation	Input	Output
updateMsn	taskType: Integer acftType: Integer base: Integer sector: Integer tgtNbr: Integer tgtCaNbr: Integer tgtSubNbr: Integer bmbTypeNbr: Integer nbrOfSorties: Integer package: Integer	None
deleteMsnAll	ad: ATODay	None
getMsnAll	ad: ATODay	m: Mission (all existed)
getMsn	ad: ATODay msnNbr: Integer	m: Mission
addMsn	taskType:Integer acftType: Integer base: Integer sector: Integer tgtNbr: Integer tgtCaNbr: Integer tgtSubNbr: Integer bmbTypeNbr: Integer	None
	nbrOfSorties: Integer package Integer	

Operation	Input	Output
deleteMsn	ad:ATODay msnNbr:Integer	None
getMsnPkg	ad:ATODay	package:Integer
deleteMsnPkg	ad:ATODay package:Integer	None
deleteMsnBad	ad:ATODay	None
updateSector	ad:ATODay m.sector:Integer l:Integer	None
updateGroup	ad:ATODay m.group:Integer l:Integer	None
updateTarget	ad:ATODay m.tgt:Integer l:Integer	None

Table 13. Class Mission–Operations

f. Class Target

1/ Attributes

Attribute	Type	Description
TgtNbr	Integer	The number of Target
Name	Text	The description of Target
Zone	Integer	The zone of Target
Nationality	Boolean	True for the Chinese Targets, meaning that they cannot be assigned to a Mission
%Operational	Integer	The percentage operational of the Target

Attribute	Type	Description
ShelterStatus	Integer	Parameter for calculation
RiskInZone	Integer	Parameter for calculation
RedBaseAcftTypeShetlerSatus	Integer	Parameter for calculation

Table 14. Class Target–Attributes

2/ Operations

Operation	Input	Output
deleteTgt	ad:ATODay	None
getTgtAll	ad:ATODay	t:Target (all existed)
updateTgt	ad:ATODay tgtNbr:Integer priority:Integer	None

Table 15. Class Target–Operations

g. Class Category (Target)

Attributes

Attribute	Type	Description
CategoryNbr	Integer	Unique number for every category
Description	Text	The description of Category
RiskInZone	Integer	Parameter for calculation

Table 16. Class Category–Attributes

h. Class Target Subcategory

Attributes

Attribute	Type	Description
SubcategoryNbr	Integer	Unique number for every subcategory
Description	Text	The description of subcategory

Table 17. Class Subcategory–Attributes

i. Class Group

1/ Attributes

Attribute	Type	Description
GroupNbr	Integer	Unique number for every group
AcftAvailable	Integer	Number of aircraft available in group
Losses	Integer	Losses of group from the start of the game

Table 18. Class Group–Attributes

2/ Operations

Operation	Input	Output
deleteGroup	ad:ATODay	None
getGroup	ad:ATODay	g:Group
getGroupInit	None	g:Group

Table 19. Class Group-Operations

j. Class Aircraft

1/ Attributes

Attribute	Type	Description
AcftTypeNbr	Integer	Unique number for every aircraft

Attribute	Type	Description
AcftName	Text	Description of aircraft
InCommissionRate	Integer	Parameter for calculation
AbortPercent	Integer	Parameter for calculation
FuelPerSortie	Integer	Parameter for calculation
AirAirAbility	Integer	Parameter for calculation
AirToGroundAbility	Integer	Parameter for calculation
Stealth	Boolean	True for aircrafts without losses

Table 20. Class Aircraft–Attributes

2/ Operations

Operation	Input	Output
getAcftType	None	acft:Aircraft

Table 21. Class Aircraft–Operations

k. Class Sector

1/ Attributes

Attribute	Type	Description
SectorNbr	Integer	Unique number for every sector
SectorName	Text	Description of sector
BlueFrontLine	Integer	Parameter for calculation
RedFrontLine	Integer	Parameter for calculation
BlueReinfRate	Integer	Parameter for calculation
RedReinfRate	Integer	Parameter for calculation
CurFEBApsn	Integer	Parameter for calculation
GroundResults	Integer	Parameter for calculation
BluePosition	Integer	Parameter for calculation

Attribute	Type	Description
RedPosition	Integer	Parameter for calculation

Table 22. Class Sector–Attributes

2/ Operations

Operation	Input	Output
deleteSector	ad: ATODay	None
getSectorInit	None	s: Sector
getSector	ad: ATODay	s: Sector

Table 23. Class Sector–Operations

m. Class Task Type

1/ Attributes

Attribute	Type	Description
TaskTypeNbr	Integer	Unique number for every task
TaskTypeName	Text	Description of task type (abbreviation of ATO type mission)
FEBAlossBasic	Integer	Parameter for calculations
FEBAlossIfDSA	Integer	Parameter for calculations
FEBAlossIfDSUP	Integer	Parameter for calculations
FEBAlossIfDSAandDSUP	Integer	Parameter for calculations
DofFightProbBasic	Integer	Parameter for calculations
DogFightProbIfDSE	Integer	Parameter for calculations
DogFightProbIfC3	Integer	Parameter for calculations
DogfightProbIfDSEandC3	Integer	Parameter for calculations

Attribute	Type	Description
TermLossBasic	Integer	Parameter for calculations
TermLossIfC3	Integer	Parameter for calculations
TermLossIfDSA	Integer	Parameter for calculations
TermLossIfDSUP	Integer	Parameter for calculations
TermLossIfC3andDSA	Integer	Parameter for calculations
TermLossIfC3andDSUP	Integer	Parameter for calculations
TermLossIfDSAand DSUP	Integer	Parameter for calculations
TermLossIfC3andDSAandDSUP	Integer	Parameter for calculations
ProbNotFind	Integer	Parameter for calculations
Priority	Integer	Parameter for calculations

Table 24. Class Task Type–Attributes.

2/ Operations

Operation	Input	Output
getTaskType	None	tsk:TaskType

Table 25. Class Task Type–Operations

n. Class Bomb Type

Attributes

Attribute	Type	Description
BombTypeNbr	Integer	Unique number for every task
bombTypeName	Text	Description of bomb type

Table 26. Bomb Type–Attributes

o. Class Map

1/ Attributes

Attribute	Type	Description
MapNbr	Integer	Unique number for every map
MapDescription	Text	Description of map
MapImage	GIF	Image of map

Table 27. Class Map–Attributes

2/ Operations

Operation	Input	Output
getMap	None	map:Map
getMap	mapNbr:Integer	map:Map

Table 28. Class Map–Operations

p. Class Report

1/ Attributes

Attribute	Type	Description
ReportTitle	Text	Unique number for every report
ReportBody	Text	Text of report

Table 29. Class Report–Attributes

2/ Operations

Operation	Input	Output
getIntroReport	None	r: Report
createReport	title: Text	r: Report

Operation	Input	Output
createReport	r: ATODay title: Text	r: Report

Table 30. Class Report–Operations

THIS PAGE INTENTIONALLY LEFT BLANK

V. PROTOTYPE

A. PURPOSE

The purpose of this chapter is to describe the way that the prototype is implemented. It also provides a "User Manual" of the prototype for easy using.

B. PROTOTYPE IMPLEMENTATION

1. General

The prototype was implemented with Microsoft Access 2000 for the following two reasons.

First, most of the use cases can be implemented as interactions between the GUI subsystem and the Database subsystem without involving the Application subsystem.

Second the number of functions of the application is big and the available time is not enough to implement the Design Specification with a classical object-oriented language.

The use of ACCESS 2000 cannot fully prove the object-oriented Design Specification of Chapter IV. Yet it proves the correct design of Database, can be used for the verification of the Requirements Specification of Chapter III, and partially prove the correctness of the Objects' attributes and interactions.

2. Database Design

The Prototype consists of Tables, Queries, Forms, Macros and Reports. The Database Entity-Relationship Diagram constructed by the ER-Win 3.5 tool of "Logic Works" is displayed in Figure 12.

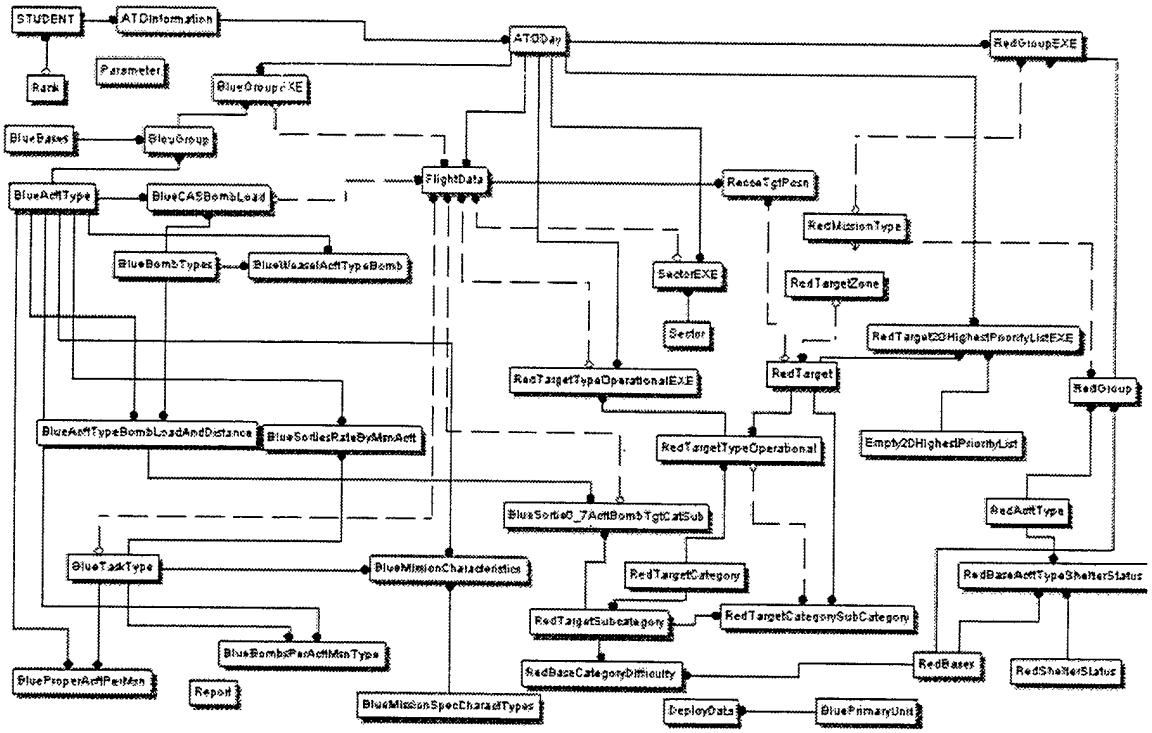


Figure 12. Entity-Relation Diagram (Logic Works ER-Win 2.0 Database Design Tool.)

Entities – Relations Attributes

	Relation Name	Attribute Name	Key	Type
1.	**STUDENT**			
		StudentSSN	Primary Key (PK)	Text

	Relation Name	Attribute Name	Key	Type
		StudentFirstName	None	Text
		StudentLastName		Text
		RankAbbr	Foreign Key (FK)	Text
		StudentClassNbr		Integer
		StudentSelection		Yes/No
		StudentCountry		Text
2.	**ATOINFORMATION**			
		ATODescription	Primary	Text
		StudentSSN	Primary	Text
3.	**ATODAY**			
		ATOExecutionTime	Primary	Integer
		ATODescription	Primary	Text
		StudentSSN	Primary	Text
		Selection		Yes/No
4.	**RANK**			
		RankAbbr	Primary	Text
		RankDescrption		Text
5.	**BLUEACFTTYPE**			
		BlueAcftTypeNbr	PK	Integer
		BlueAcftTypeName		Text
		BlueAcftTypeInComPercent		Integer
		BlueAcftTypeAbortPercent		Integer
		BlueAcftTypeFuelPerSortie		Integer
		BlueAcftTypeAirAirAbilityPercent		Integer
		BlueAcftAirToGndCapab		Integer
		BlueAcftAirToGndCapab		Integer
		AcftFuel		Text

	Relation Name	Attribute Name	Key	Type
		Stealth		Yes/No
6.	**BLUEBASE**			
		BlueBaseNbr	PK	Integer
		BlueBaseAbrreviation		Text
		BlueBaseFullName		Text
		BlueBasePercentOCA		Double
7.	**BLUEGROUP**			
		BlueBaseNbr	PK-FK	Integer
		BlueAcftTypeNbr	PK-FK	Integer
		BlueAcftAmmount		Integer
8.	**BLUEGROUPEXE**			
		ATOExecutionTime	PK-FK	Integer
		ATODescription	PK-FK	Text
		StudentSSN	PK-FK	Text
		BlueBaseNbr	PK-FK	Integer
		BlueAcftTypeNbr	PK-FK	Integer
		BlueAcftAmmount		Integer
9.	**BLUETASKTYPE**			
		BlueMissionTypeNbr	PK	Integer
		BlueMissionName		Text
		FEBAlossBasic		Double
		FEBAlossIfDSA		Double
		FEBAlossIfDSUP		Double
		FEBAlossIfDSAandDSUP		Double
		DofFightProbBasic		Double
		DogFightProbIfDSE		Double
		DogFightProbIfC3		Double
		DogfightProbIfDSEandC3		Double
		TermLossBasic		Double

	Relation Name	Attribute Name	Key	Type
		TermLossIfC3		Double
		TermLossIfDSA		Double
		TermLossIfDSUP		Double
		TermLossIfC3andDSA		Double
		TermLossIfC3andDSUP		Double
		TermLossIfDSAand DSUP		Double
		TermLossIfC3andDSAandDSUP		Double
		ProbNotFind		Double
		Priority		Integer
10.	**BLUEBOMBTYPE**			
		BlueBombTypesNbr	PK	Integer
		BlueBombTypeName		Text
11.	**BLUEPROPERACFTPERMISSION**			
		BlueAcftTypeNbr	PK-FK	Integer
		BlueMissionTypeNbr	PK-FK	Integer
		BlueProperAcftTypePerMissionNbr		Integer
12.	**BLUEBOMBSPERACFTMSNTYPE**			
		BlueAcftTypeNbr	PK-FK	Integer
		BlueMissionTypeNbr	PK-FK	Integer
		BlueBombNbrPerAcftMsnBomb		Integer
13.	**BLUESORTIESRATEBYMSNACFT**			
		BlueAcftTypeNbr	PK-FK	Integer
		BlueMissionTypeNbr	PK-FK	Integer
		BlueMsnAcftTypeRate		Integer
14.	**BLUEACFTTYPEBOMBLOADANDDISTANCE**			
		BlueSpCharacteristicNbr	PK	Integer
		BlueBombTypesNbr	PK -FK	Integer
		BlueBombLoadAmmount		Integer
		BlueRangeWithLoad		Integer

	Relation Name	Attribute Name	Key	Type
15.	**BLUECASBOMBLOAD**			
		BlueAcftTypeNbr	PK-FK	Integer
		BlueBombTypesNbr	PK -FK	Integer
		BlueBombLoadQuantity		Integer
16.	**BLUEWEASELACFTYPEBOMB**			
		BlueAcftTypeNbr	PK-FK	Integer
		BlueBombTypesNbr	PK -FK	Integer
17.	**BLUEMISSIONSPECCHARACTTYPES**			
		BlueSpCharacteristicNbr	PK	Integer
		BlueSpCharacteristicDescriptionr		Integer
18.	**BLUEMISSIONCHARACTERISTICS**			
		BlueSpCharacteristicNbr	PK-FK	Integer
		BlueAcftTypeNbr	PK-FK	Integer
		BlueMissionTypeNbr	PK-FK	Integer
19.	**SECTOR**			
		SectorNbr	PK	Integer
		SectorName		Text
		SectorBlueCASPercntNeed		Double
		SectorBlueSAPercentNeed		Double
		SectorBlueReccePercentNeed		Integer
		SectorBlueFrontLine		Integer
		SectorBlueReinforce		Integer
		SectorRedFrontLine		Integer
		SectorRedReinforce		Integer
20.	**SECTOREXE**			
		SectorNbr	PK -FK	Integer
		ATOExecutionTime	PK-FK	Integer
		ATODescription	PK-FK	Text
		StudentSSN	PK-FK	Text

	Relation Name	Attribute Name	Key	Type
		CurrentFEBAPosition		Integer
		GndResults		Integer
		BluePosition		Integer
		RedPosition		Integer
		TotalFEBA		Integer
21.	**REDTARGET**			
		RedTargetNbr	PK	Integer
		RedTargetDescription		Text
		ZoneNbr	FK	Integer
		RedTargetLatidute		Double
		RedTargetLongidute		Double
		RedTargetChinese		Yes/No
22.	**REDTARGETZONE**			
		ZoneNbr	PK	Integer
		RiskInZone		Integer
23.	**REDMISSIONTYPE**			
		RedMissionTypeNbr	PK	Integer
		RedMissionDescription		Text
24.	**REDTARGETCATEGORY**			
		TargetCategoryNbr	PK	Integer
		TargetCategoryDesc		Text
25.	**REDTARGETSUBCATEGORY**			
		TargetSubCategNumber	PK	Integer
		TargetCategoryNbr	PK- FK	Integer
		TargetSubCategDescr		Text
		TargetSubCategRebRate		Double
		TargetSubCategRebTimes		Integer
26.	**REDTARGETCATEGORYSUBCATEGORY**			

	Relation Name	Attribute Name	Key	Type
		TargetSubCategNumber	PK- FK	Integer
		TargetCategoryNbr	PK- FK	Integer
		RedTargetNbr	PK -FK	Integer
27.	**REDTARGETOPERATIONAL**			
		TargetCategoryNbr	PK- FK	Integer
		RedTargetNbr	PK -FK	Integer
		ReccePercentOperational		Integer
		MaxPercentOperational		Integer
28.	**REDTARGETOPERATIONALEXE**			
		ATOExecutionTime	PK-FK	Integer
		ATODescription	PK-FK	Text
		StudentSSN	PK-FK	Text
		TargetCategoryNbr	PK- FK	Integer
		RedTargetNbr	PK -FK	Integer
		ReccePercentOperational		Integer
		MaxPercentOperational		Integer
29.	**REDBASE**			
		RedBaseNbr	PK	Integer
		RedBaseDescription		Text
30.	**REDACFTTYPE**			
		RedAcftTypes	PK	Integer
		RedAcftDescription		Text
31	**REDGROUP**			
		RedAcftTypes	PK -FK	Integer
		RedBaseNbr	PK -FK	Integer
		RedMissionTypeNbr	FK	Integer
		RedGroupNbr		Integer
		RedAcftNumber		Integer

	Relation Name	Attribute Name	Key	Type
		RedMissionTypeSortyRate		Integer
32	**REDGROUPEXE**			
		ATOExecutionTime	PK-FK	Integer
		ATODescription	PK-FK	Text
		StudentSSN	PK-FK	Text
		RedAcftTypes	PK -FK	Integer
		RedBaseNbr	PK -FK	Integer
		RedAcftNumber		Integer
		RedLosses		Integer
33.	**REDSHELTERSTATUS**			
		ShetlerStatusNbr	PK	Integer
		ShetlerDescription		Text
34.	**REDBASEACFTTYPESHELTERSTATUS**			
		RedAcftTypes	PK -FK	Integer
		RedBaseNbr	PK -FK	Integer
		ShetlerStatusNbr	FK	Integer
35.	**EMPTY20HIGHESTPRIORITYLIST**			
		Priority	PK	Integer
		RedTargetNbr	FK	Integer
36.	**REDTARGET20HIGHESTPRIOTIYLISTEXE**			
		Priority	PK-FK	Integer
		ATOExecutionTime	PK-FK	Integer
		ATODescription	PK-FK	Text
		StudentSSN	PK-FK	Text
		RedTargetNbr	FK	Integer
37.	**REDBASECATEGORYDIFIICULTY**			
		RedBaseNbr	PK -FK	Integer
		TargetCategoryNbr	PK- FK	Integer
		TargetSubCategNumber	PK- FK	Integer

	Relation Name	Attribute Name	Key	Type
38.	**RECCETGTPSN**			
		ATOExecutionTime	PK-FK	Integer
		ATODescription	PK-FK	Text
		StudentSSN	PK-FK	Text
		RedTargetNbr	PK-FK	Integer
		FlightNbr	PK-FK	Integer
39.	**FLIGHTDATA**			
		ATOExecutionTime	PK-FK	Integer
		ATODescription	PK-FK	Text
		StudentSSN	PK-FK	Text
		FlightNbr	PK	Integer
		BlueBaseNbr	FK	Integer
		BlueAcftTypeNbr	FK	Integer
		BlueMissionTypeNbr	FK	Integer
		SectorNbr	FK	Integer
		TargetSubCategNumber	FK	Integer
		TargetCategoryNbr	FK	Integer
		RedTargetNbr	FK	Integer
		BlueBombTypesNbr	FK	Integer
		FlightBlueSortiesAssigned		Integer
		MssnPkgNbr		Integer
40.	**BLUESORTIE0_7ACFTBOMBTGTATSUB**			
		TargetCategoryNbr	PK-FK	Integer
		TargetSubCategNumber	PK-FK	Integer
		BlueAcftTypeNbr	PK-FK	Integer
		BlueBombTypesNbr	PK-FK	Integer
		BlueSortie0_7		Integer
41.	**BLUEPRIMARYUNIT**			
		PrimeUTC	PK	Text

	Relation Name	Attribute Name	Key	Type
		PrimeUnitName		Text
		PrimeUnitAmmunition		Integer
42.	**DEPLOYDATA**			
		PrimeUTC	PK-FK	Text
		ArmyDivs		Integer
		Nbr2		Integer
		Nbr3		Integer
		Nbr4		Integer
43.	**REPORT**			
		Reportid	PK	Integer
		ReportLabel		Text
		ReportText		OLE Object
44.	**PARAMETERS**			
		Hi_5Damage		Integer
		Lo_5Damage		Integer
		Hi_7Damage		Integer
		Lo_7Damage		Integer
		NotFireAt		Integer
		DogFightBasicLoss		Integer
		KillRatioBad		Integer
		KillRatioEven		Integer
		KillRatioGood		Integer
		MinWorstRatio		Integer
		MinBestRatio		Integer
		PercentRedDCA		Integer
		PercentRedOCA		Integer
		PercentRedFlying		Integer
		MaxPercentRedAcftLost		Integer

Relation Name	Attribute Name	Key	Type
	RedAbort		Integer
	RedFEBALoss		Integer
	RedFtrLoss		Integer
	RedTernLoss		Integer
	RedNotFind		Integer
	DoAndShowArmyMissions		Integer
	ShowEnemyBpmbersOverBlueBases		Integer
	TightnessFactor		Integer
	AIMfudgeFactor		Integer
	PercentDailyDegradeGnd		Integer
	AcftGndEquivFactor		Integer
	GndDiffToMoveFEBA		Integer
	TopLat		Integer
	BottomLat		Integer
	LeftLong		Integer
	RightLong		Integer

Table 31. Entities – Relations Attributes.

The Database Entity-Relation Diagram of Figure 12 and the Relations' description of the table above can also be used as Database design for the implementation of the final Application. The prototype's database contains some extra relations that are used for the calculations only and should not be included in the final application database. This kind of relation has the prefix "TMP" for easy recognition from the main relations of the database.

The prototype was a combination of ACCESS 2000 Forms, Visual Basic code, queries, and macros to provide user friendly GUI. The GUI design and probably the implementation can be used for the final application.

The queries are written by the special ACCESS 2000 SQL. In the case when a query is very complicated to be executed, nested queries that construct temporary tables are used. The use of nested queries reduces the prototype's performance because of its extensive use of the storage devices, which are the slowest components of a computer. On the other hand, it offers the functionality that the prototype needs.

Macros are used mainly for two reasons to create a sequence of events (queries) and to offer runtime information to the GUI.

Reports are used only for printing necessities and they are the printable versions of GUIs (Forms.)

C. USER MANUAL

1. Installing CAMPEX Employment Module Prototype

- Recommended System: IBM PC-compatible 486 computer running Microsoft WINDOWS (NT 3.5 or 98) or higher, with minimum 16 MB RAM and minimum 256-color monitor.
- Copy the file CAMPEX2000.mdb from its source to desktop (a shortcut is created.) The size of the prototype is expected to be 10 MB when it needs ACCESS 2000 to run (current version) and above 25 MB if it can run as an independent application.

2. Running the CAMPEX Employment Module Prototype

- Double click on the application icon and the program starts to execute.
- In every screen student has the choices *"Return"* and *"Exit."* With *"Return,"* the system returns to previous screen, and with *"Exit,"* the system quits the application.

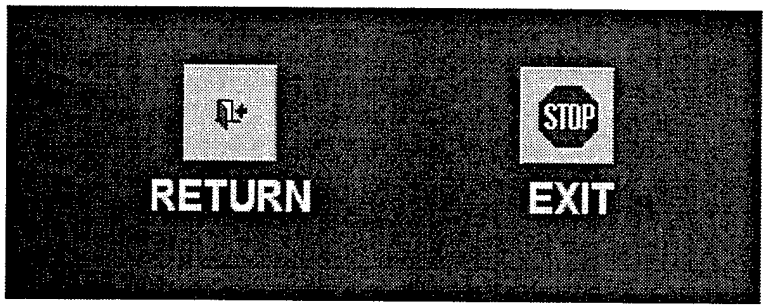


Figure 13. Return and Exit

3. CAMPEX Employment Module Initial Screen

The Initial screen of CAMPEX Employment Module appears with one menu bar on the top of the screen named "Student", with three choices:

- *"New Student"* must be selected if the student executes the program for first time.
- *"Select Student"*, must be selected if student has already input his personal information in the program.
- "Exit" to quit the application.

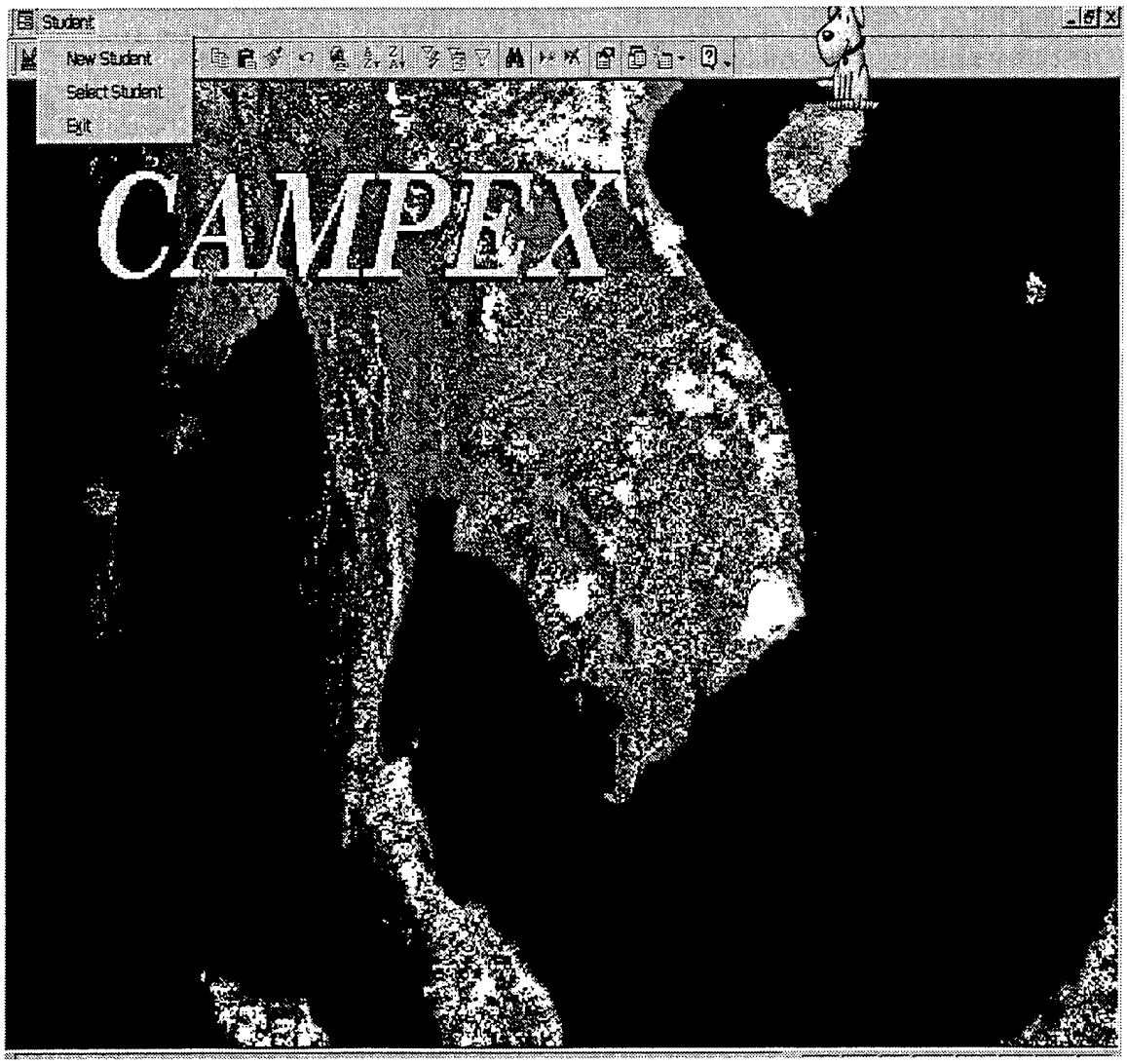


Figure 14. Initial Screen

4. "New Student"

- An empty "Student" record is displayed.
- Enter the necessary student information.
- Fill all data fields.
- Select return.

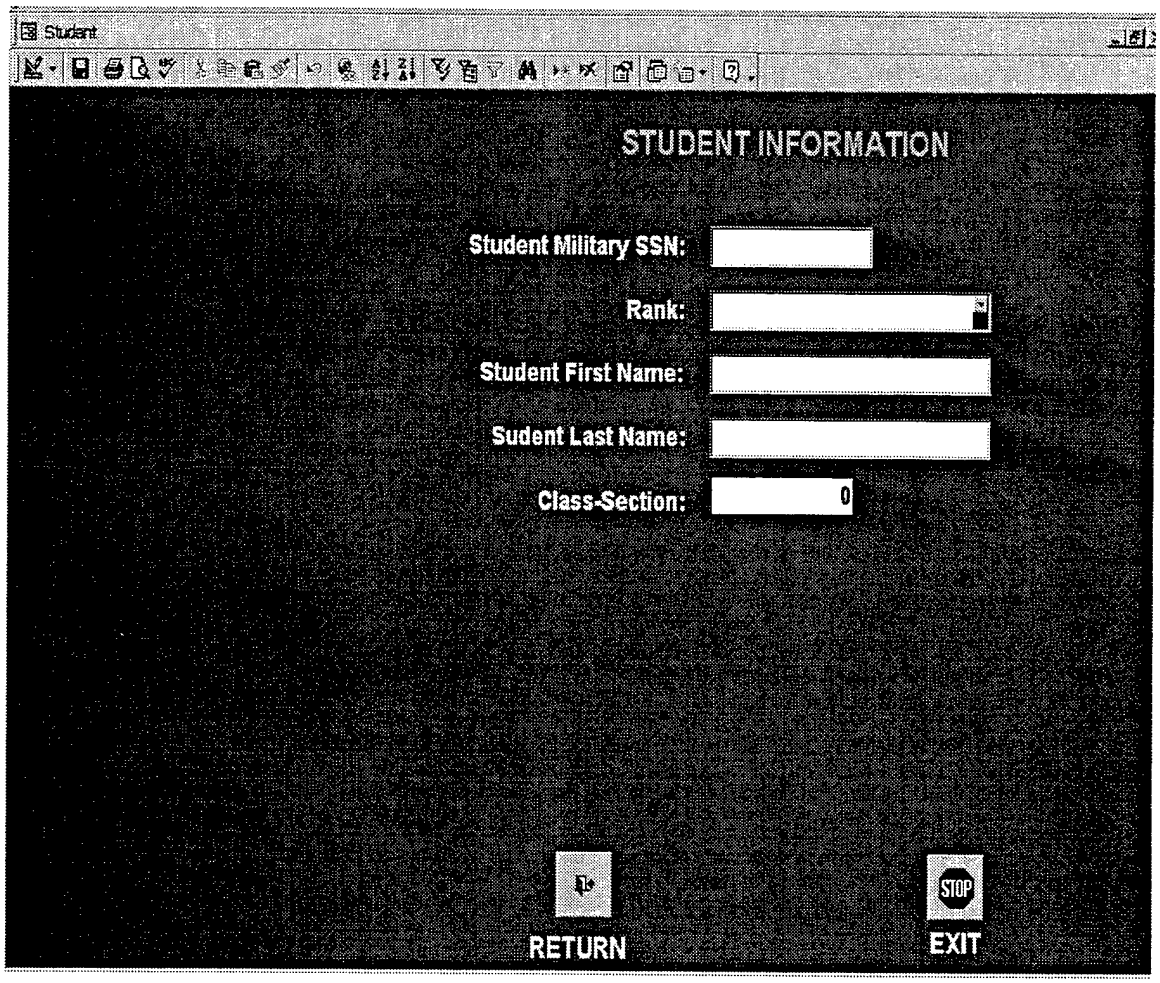


Figure 15. New Student

5. **"Select Student "**

- System returns a list of student names.
- Select a name from the list.
- Select "Continue."

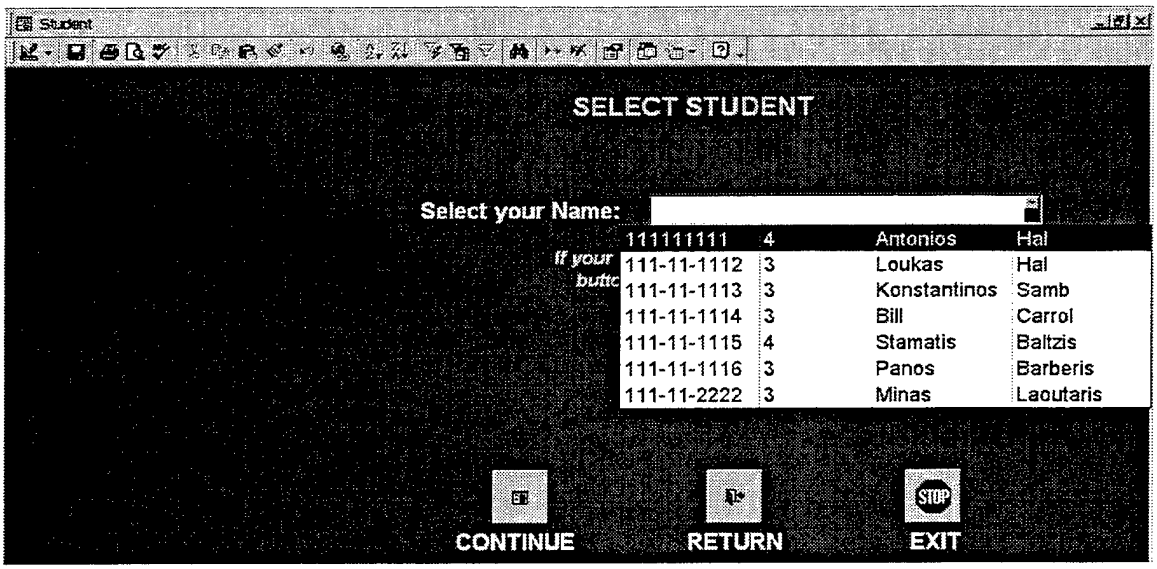


Figure 16. Select Student

6. "Start Employment Module"

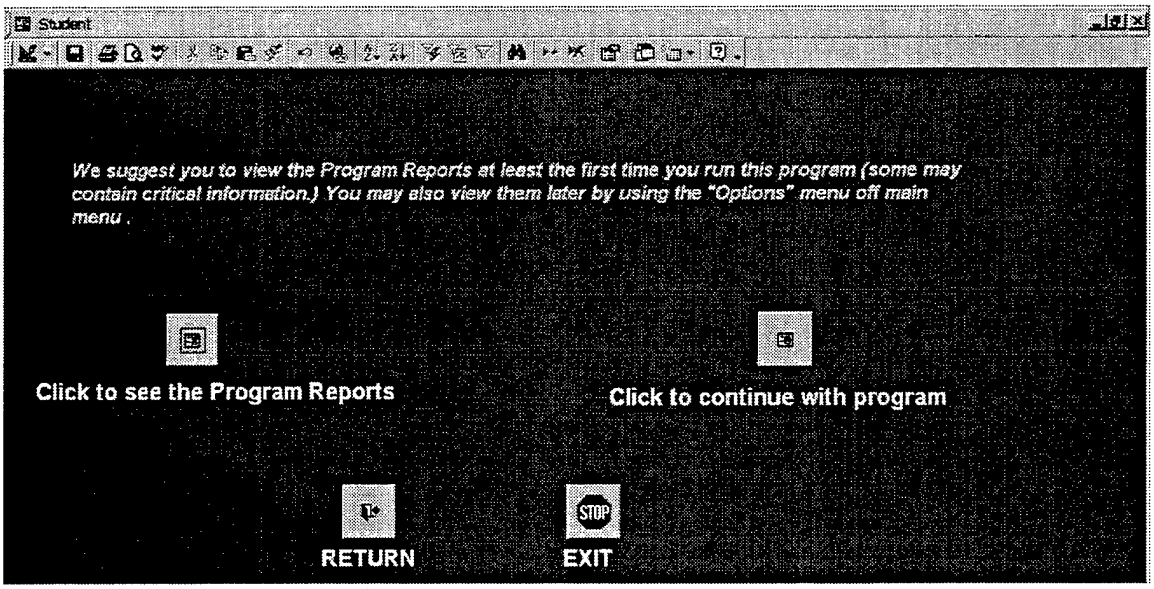


Figure 17. Choose to see Reports or Not

- "Click to See the program Reports" must be selected if the student wants to see the "Introductory Reports."

- "Click to Continue with Program" must be selected if the user wants to continue with the program without seeing the "Introductory Reports."

7. "Click to See the Program Reports"

A screen with the first Introductory Report will be displayed. Student can see the report by using the right screen scrollbar, by clicking inside the report and by using the keyboard up and down arrows. Notice that the student has seen the whole report only if he is able to see the note "Last Line," written with red letters.

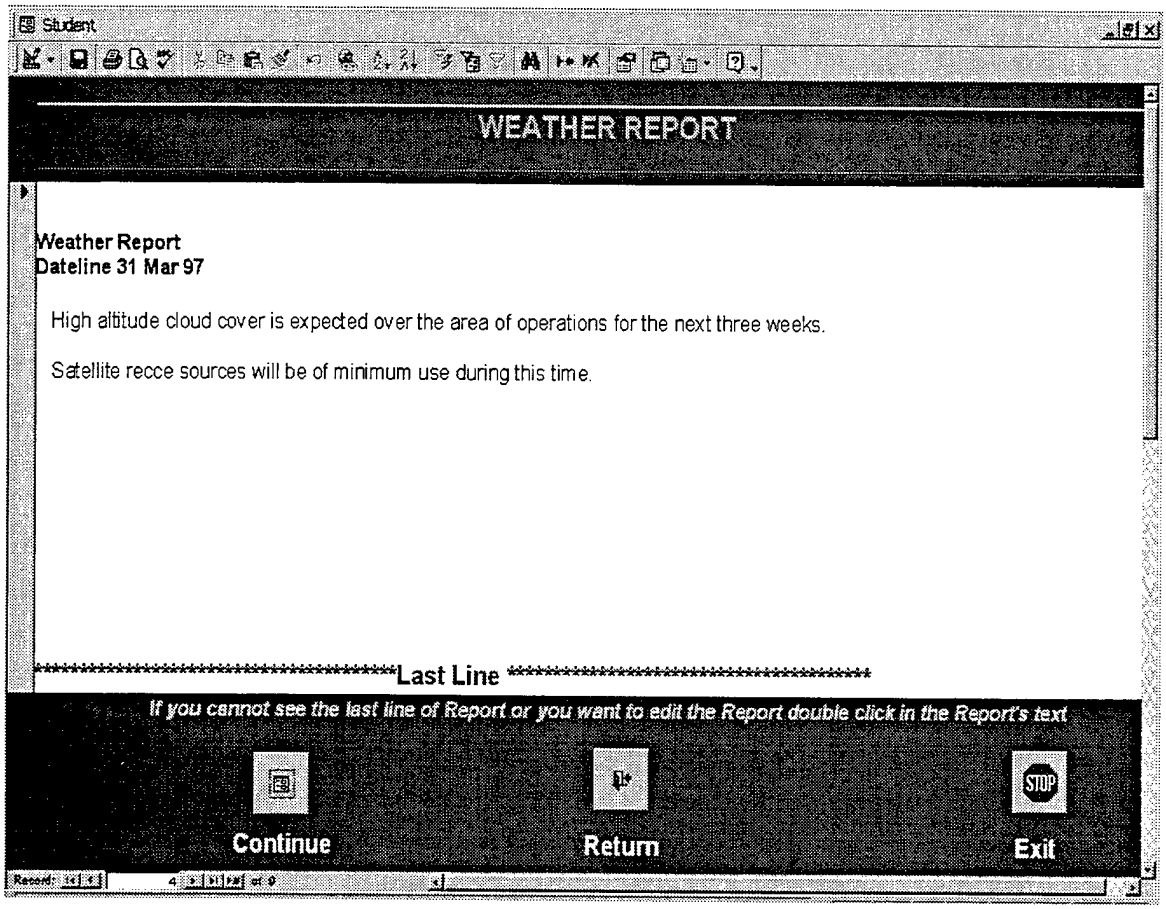


Figure 18. Introductory Report

Select (▶) to continue with the next "Introductory Report."

8. "Click to Continue with Program"

A screen with the "Ground Combat Units as of Day 1" will be displayed. This report is always empty, because the information it contains is a product of "CAMPEX Deployment Module" that has not yet been implemented.

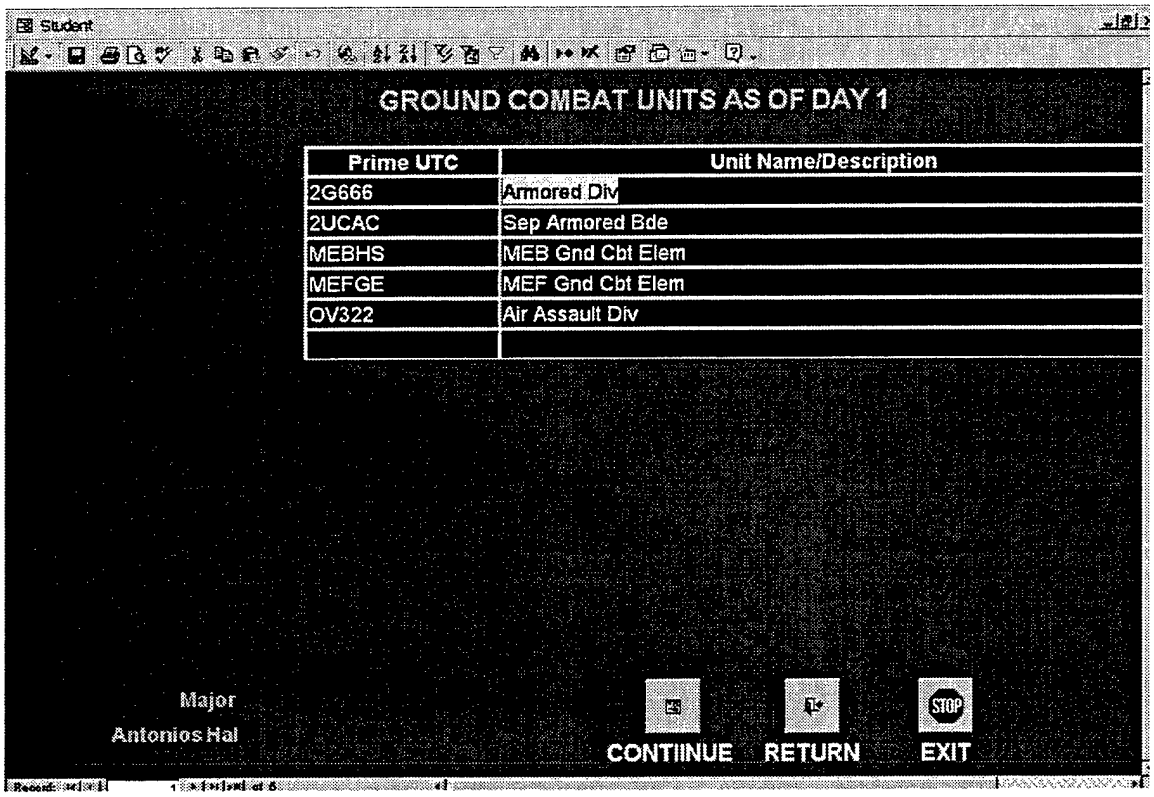


Figure 19. Ground Combat Units

Select "Continue."

9. "ATO Selection"

- "Select an ATO" must be selected if the student wants to execute an existing ATO.
- "If the ATO is not on the list Click the Button to Enter" must be selected if the user wants to enter an ATO that is not existed in the list.

10. "Select an ATO"

- Select an ATO description.
- Select an ATO day.

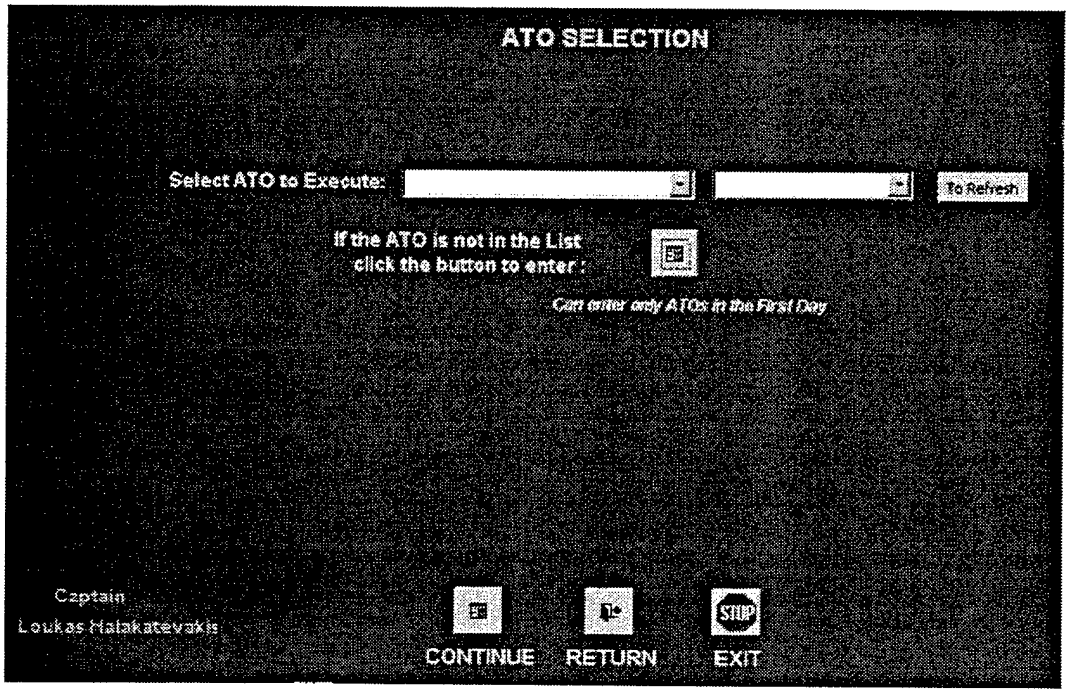


Figure 20. ATO Selection

If the student selects an ATO day, all the days following the selected day and their information will be deleted.

11. "New ATO "

An empty ATO record is displayed, if the student selects "If the ATO is not on the list click the Button to Enter".

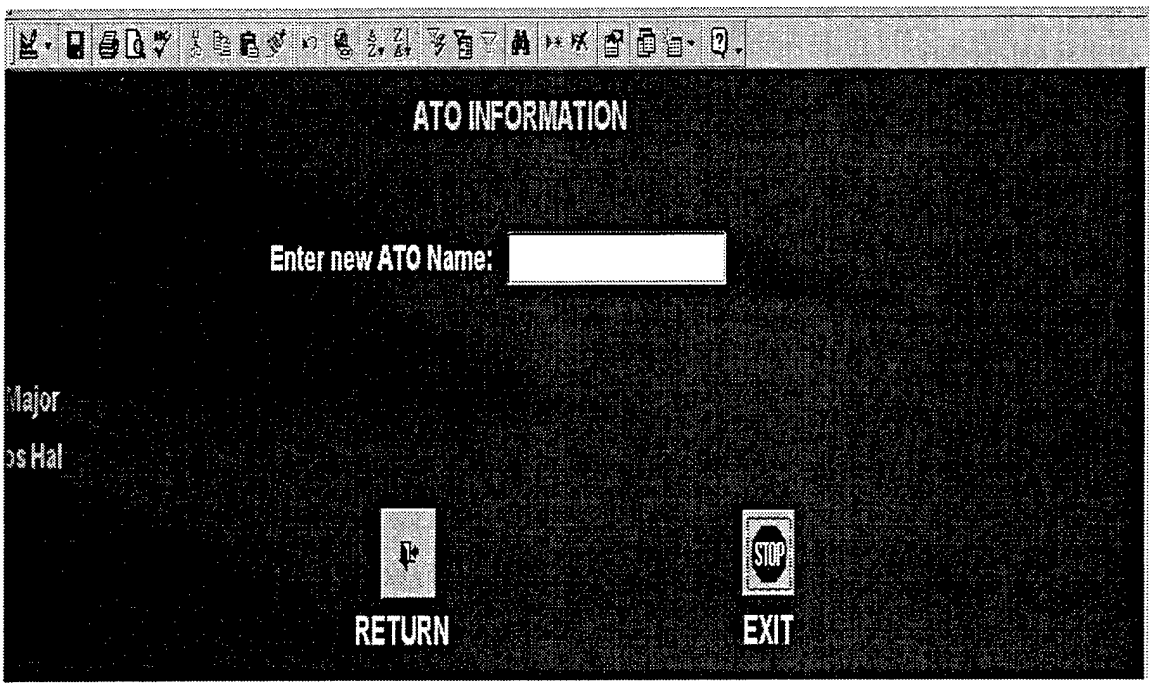


Figure 21. Enter a New ATO

- Enter the necessary information.
- Select “Return”.

Program returns to the previous screen and user must select an ATO to continue.

12. "Main Menu Screen"

On the top of the screen a new "Menu Bar" will be displayed with three choices:

- "Options."
- "ATO Planning."
- "Intel and Results."

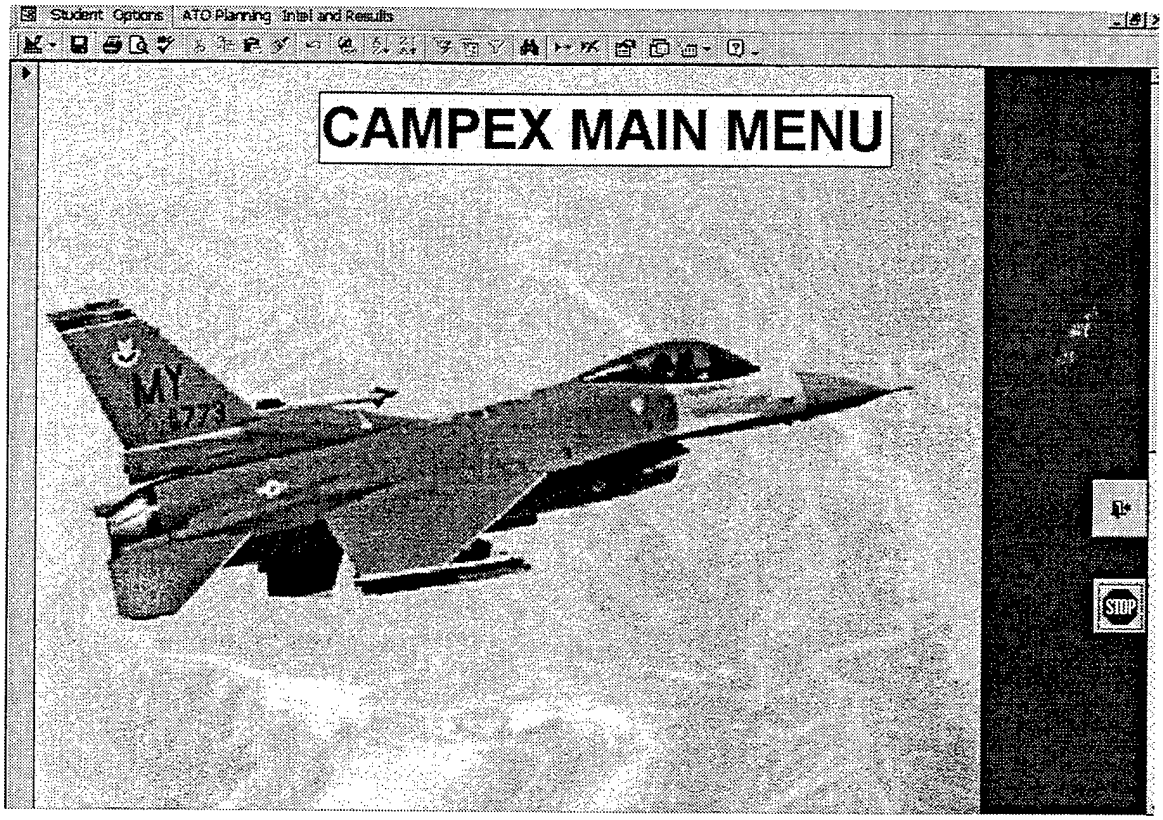


Figure 22. Main Menu

13. **"Options"**

- "Open Introductory Report" must be selected to see an Introductory Report.
- "Area Map" must be selected to see the Area Map.
- "Change ATO" must be selected to return to the state of selecting an ATO.
- "Blue Basing Summary" must be selected to see the blue "Basing Summary" report.
- "Sorties Available" must be selected to see the blue "Sorties Available" report.
- "Analysis" must be selected to see the blue "Analysis" report.

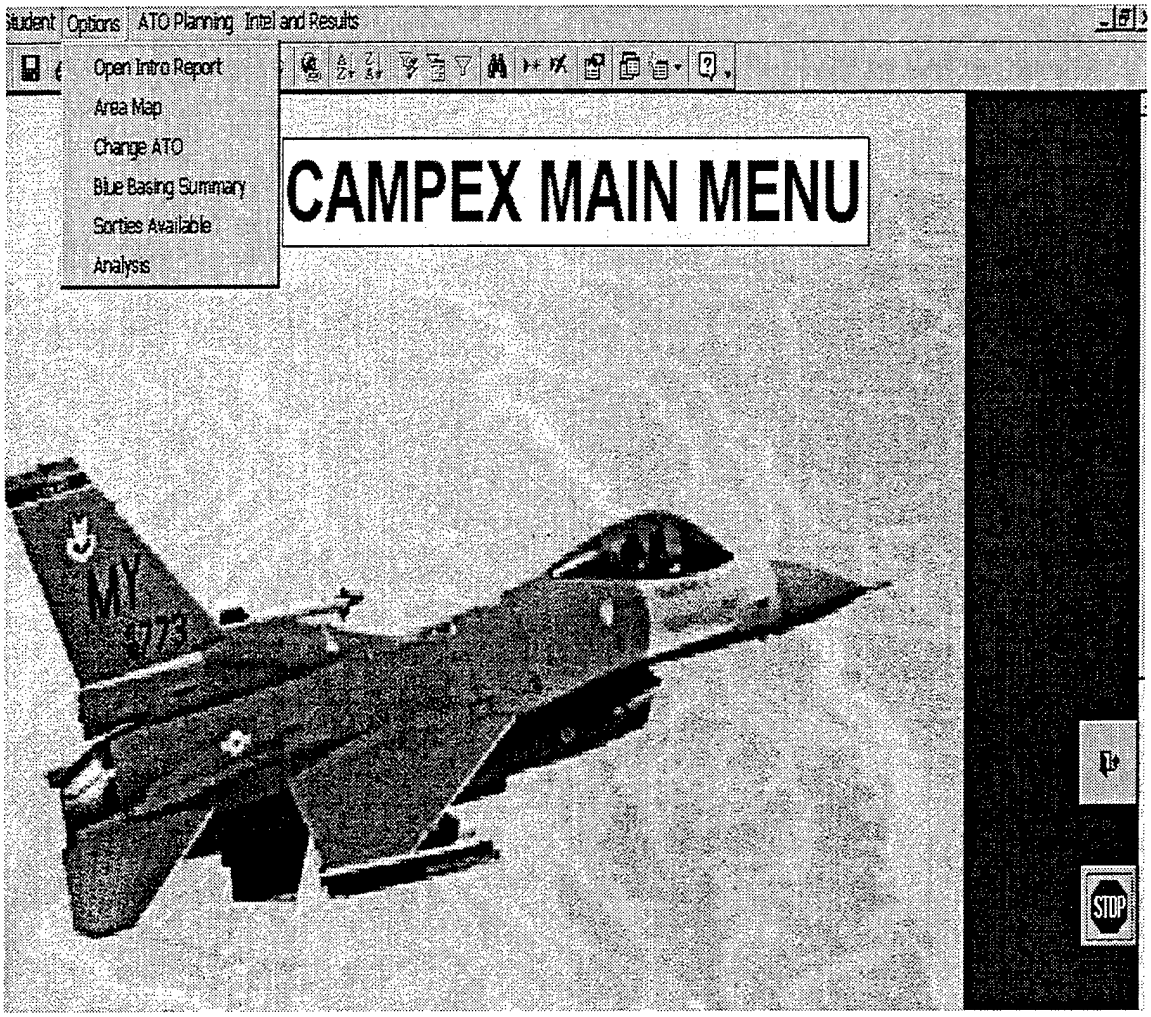


Figure 23. Options Menu

14. "Open Intro Report"

System returns a list with all Introductory Reports' Titles.

- "Select a title from the list." The selected Report will be displayed, pressing the "Return" button will return to the "Report Management Screen."

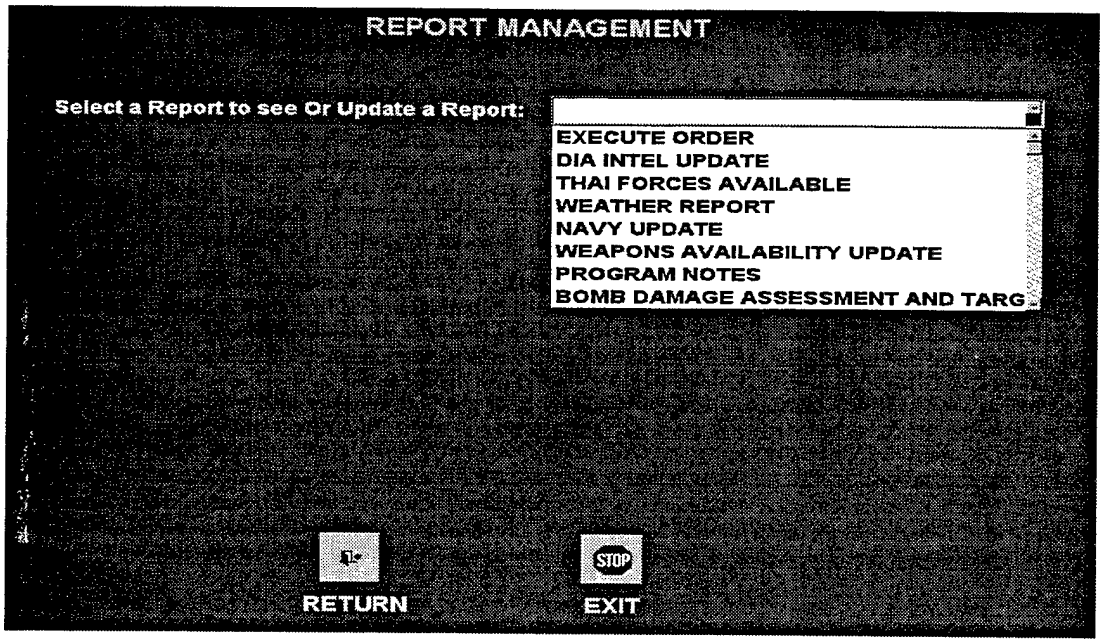


Figure 24. Report Management

15. "Area Map "

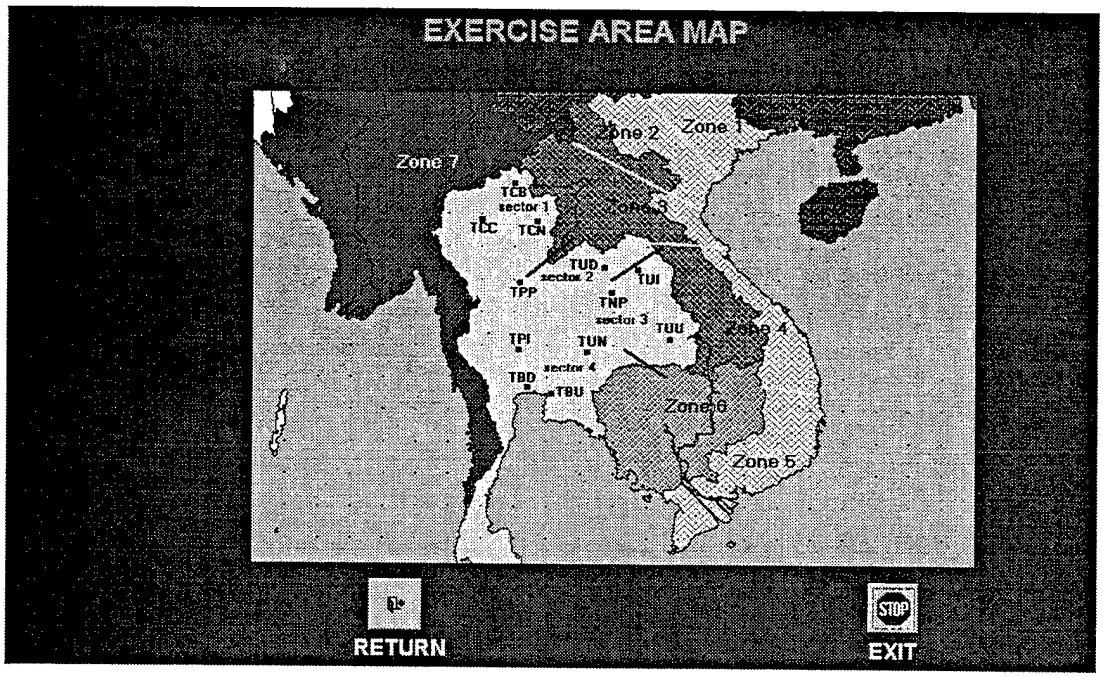


Figure 25. Area Map

16. "Change ATO"

"ATO Selection" screen will be displayed. User must follow the same known sequence as in the section 4.b.11.

17. "Blue Basing Summary "

BLUE BASING SUMMARY FOR ATO

MAP	Base Name	Acft Type	OP	Lost	OCA	DCA	SA	AI
TBD	Bangkok	F-117	18	0	1	—	1	1
TCC	Chiang Mai	AC130	6	0	—	—	—	—
TUN	Korat	FA-18	12	0	2	2.5	2	2
TUN	Korat	WW	48	0	—	—	—	—
TUN	Korat	F-16	24	0	2	3	2	2
TUN	Korat	EA-6	4	0	—	—	—	—
TNP	Nam Phong	OA-10	24	2	—	—	—	1
TCN	Nan	F-16	24	0	2	3	2	2
TPP	Phitsanulok	OA-10	24	0	—	—	—	1
TPI	Takhli	F-15E	24	0	2	2.5	2	2
TUU	Ubon	F-15	24	0	—	3	—	—
TUU	Ubon	F-16	24	0	2	3	2	2

Major
Antonios Hal

PRINT RETURN EXIT

Figure 26. Blue Basing Summary

18. "Sorties Available "

SORTIES AVAILABLE AT AIRBASES Odyseas 1

Aircraft Type	Sorties Available	Base Number	Base Name
F-16	20	6	Korat
F-16	20	8	Nan
F111	22	13	U-Taphao
EF111	22	13	U-Taphao
F-117	16	3	Bangkok
B-52	11	1	Anderson
A-6	20	14	Navy
EA-6	6	14	Navy
EA-6	6	6	Korat

Major
Antonios Hal

PRINT RETURN EXIT

Figure 27. Sorties Available

19. "Analysis "

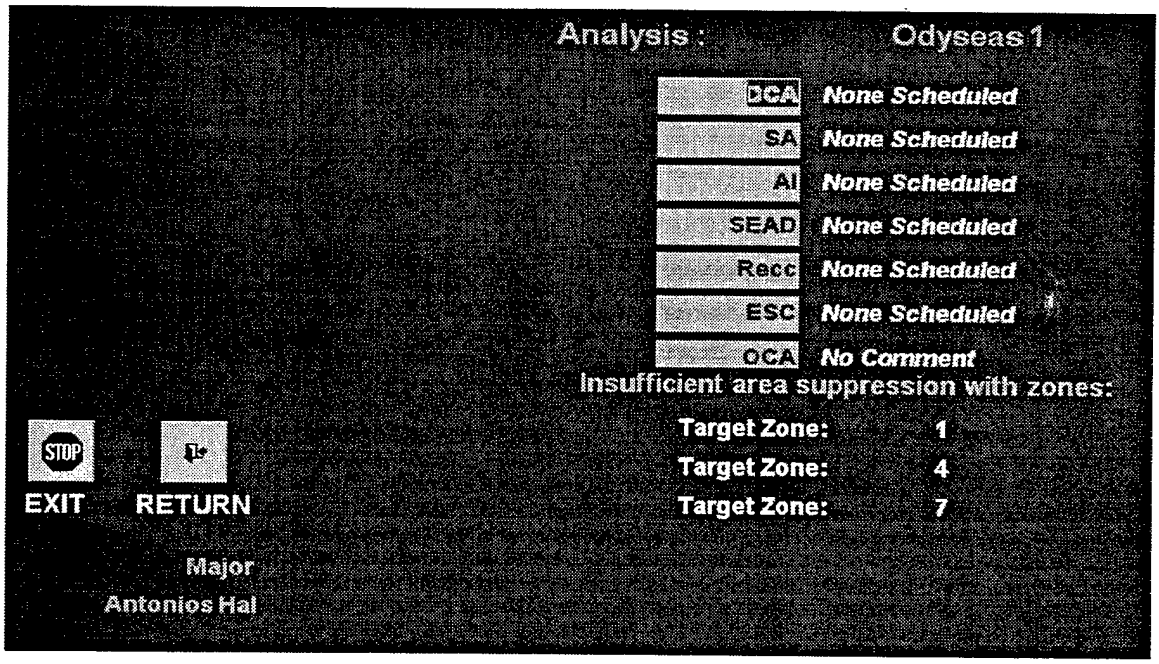


Figure 28. Analysis

20. "ATO Planning"

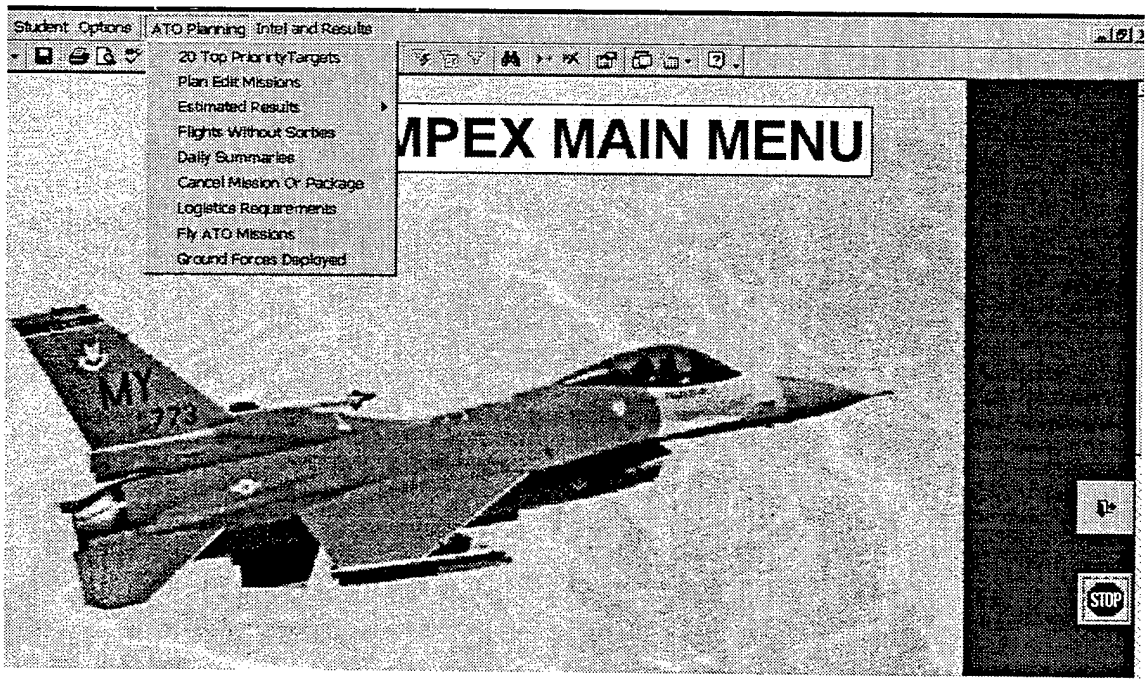


Figure 29. Menu "ATO Planning"

- "20 Top Priority Targets" must be selected to describe the 20 targets with highest priority.
- "Plan Edit Missions" must be selected to plan missions for an ATO.
- "Estimated Results" must be selected to see the "Estimated Results" report of his planning.
- "Flights Without Sorties" must be selected to see the "Flights Without Sorties " report.
- "Daily Summaries" must be selected to see the "Daily Summaries " report.
- "Cancel Mission or Package" must be selected to delete a planned mission or package of missions.
- "Logistic Requirements" must be selected to see the "Logistic Requirements " report.
- "Fly ATO Missions" must be selected to execute the planned ATO. System changes state to the next ATO day.
- "Ground Forces Deployed" must be selected to see the "Ground Forces Deployed " report.

21. "20 Top Priority Targets "

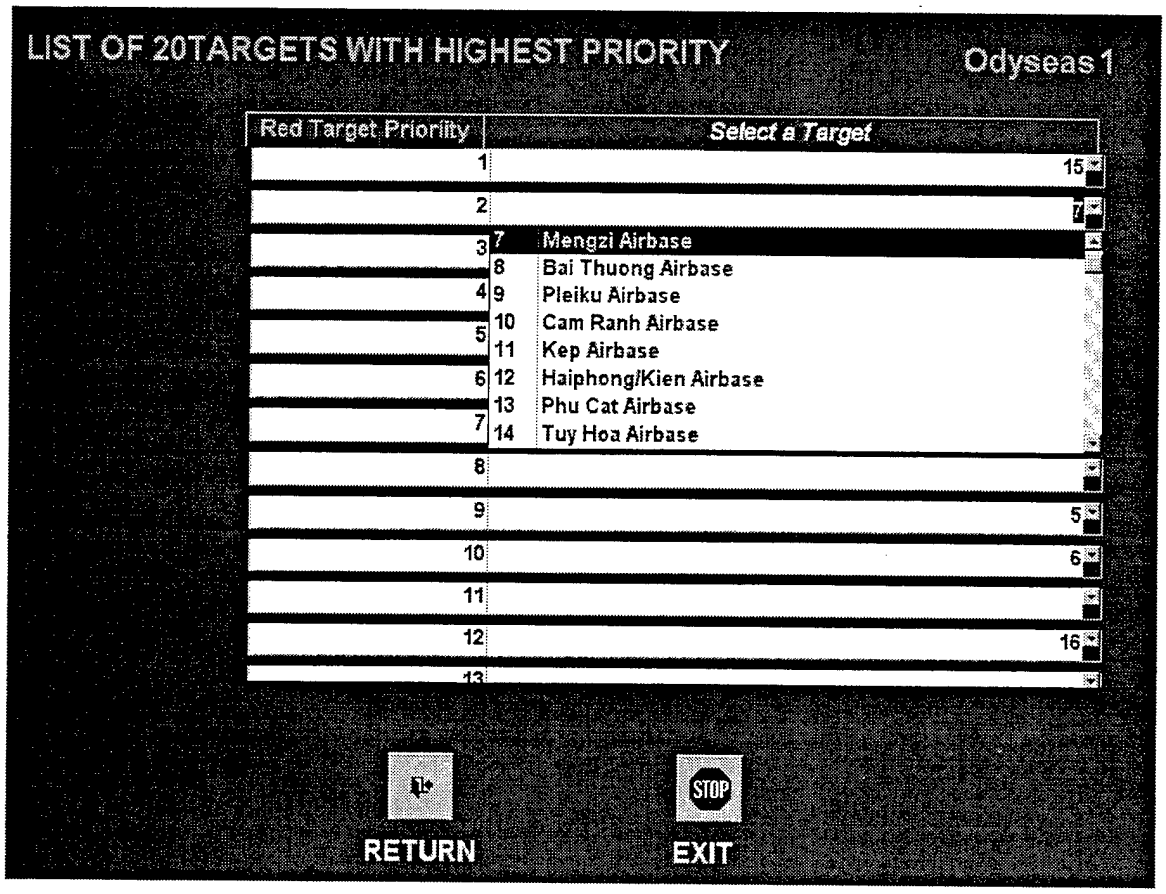


Figure 30. List of 20 Targets with Highest Priority

- Select the priority number, and then select a Target from the list. The user cannot select the same target more than once in a Priority List.

22. "Plan Edit Missions "

- "Aircraft Type."
- "Blue Base."
- "Red Base or Target Number."

- "Target Type."
- "Zone of Target."
- "Task Type."
- "Packages."
- "List All."

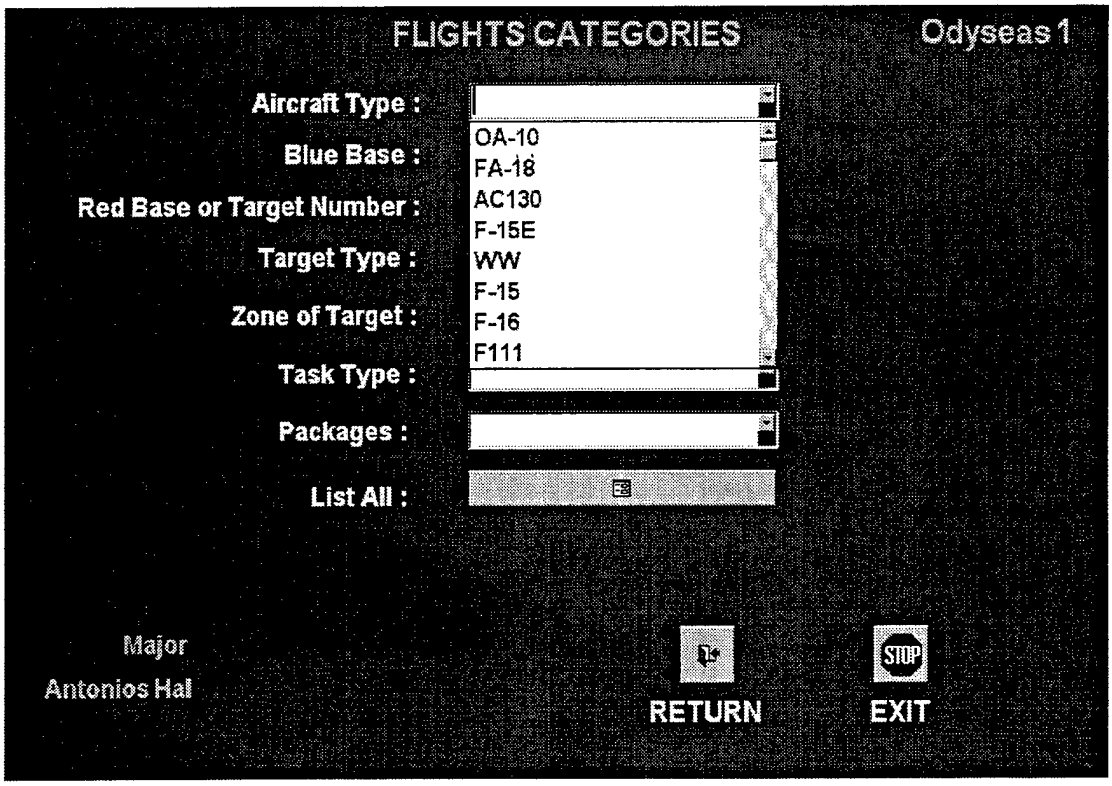


Figure 31. Flight Categories

Only the Flights in the planned missions will be displayed. Selecting the "List All" option will results in the display shown in Figure 32. The user can see the planned missions up to that moment now by the selected choice.

Select "To Enter a new Flight or to edit an existing click the button below" to modify flights.

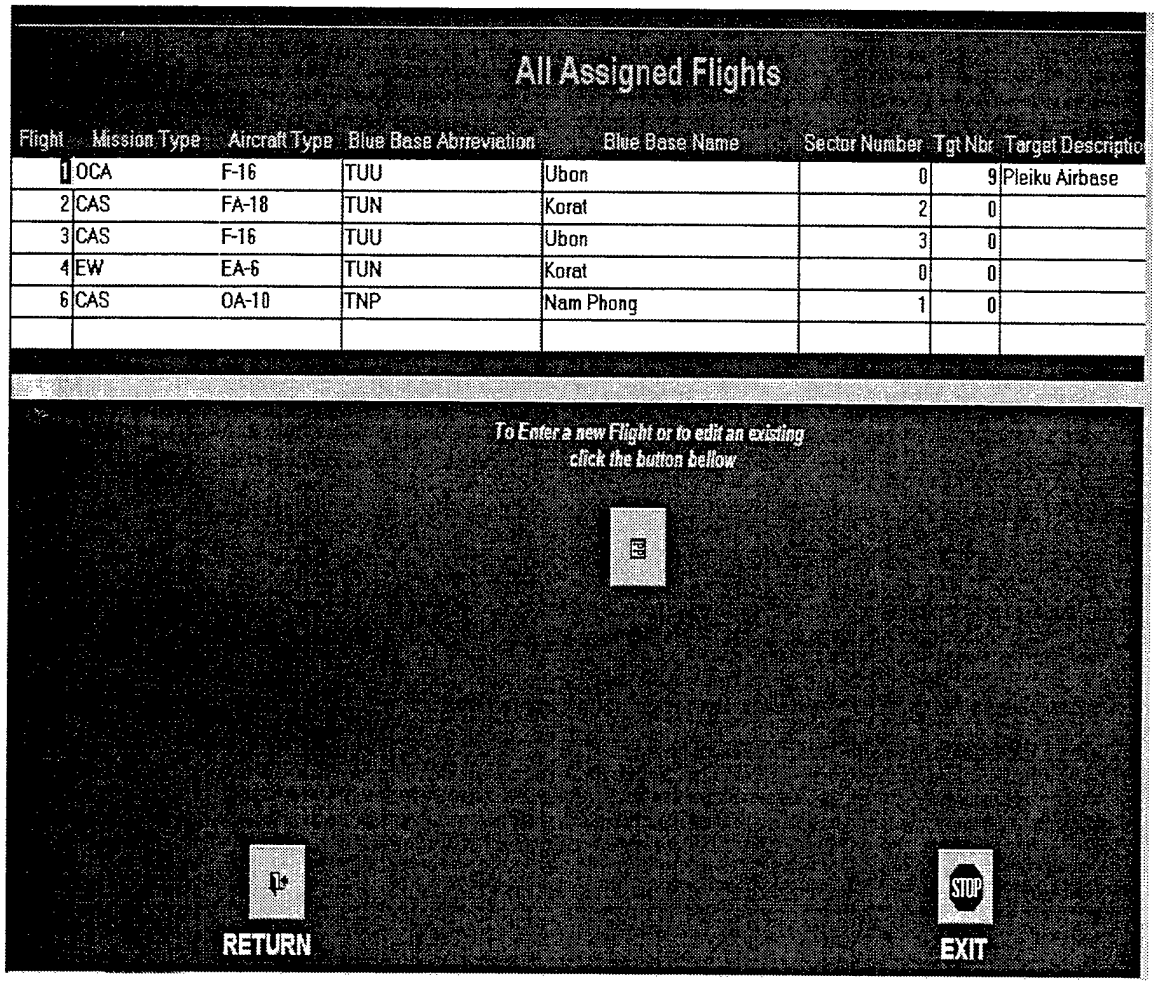


Figure 32. Assigned Flights By Category

23. "Enter or Edit Flight "

- "To Enter a new Flight Enter the Flight Number and click on the Button below" must be selected to enter a new Mission.
- "The numbers in the list already exist so you cannot enter them but you can edit them" must be selected to edit a mission.
- Select an existed flight number or enter a new number will result in the display shown in Figure 34.

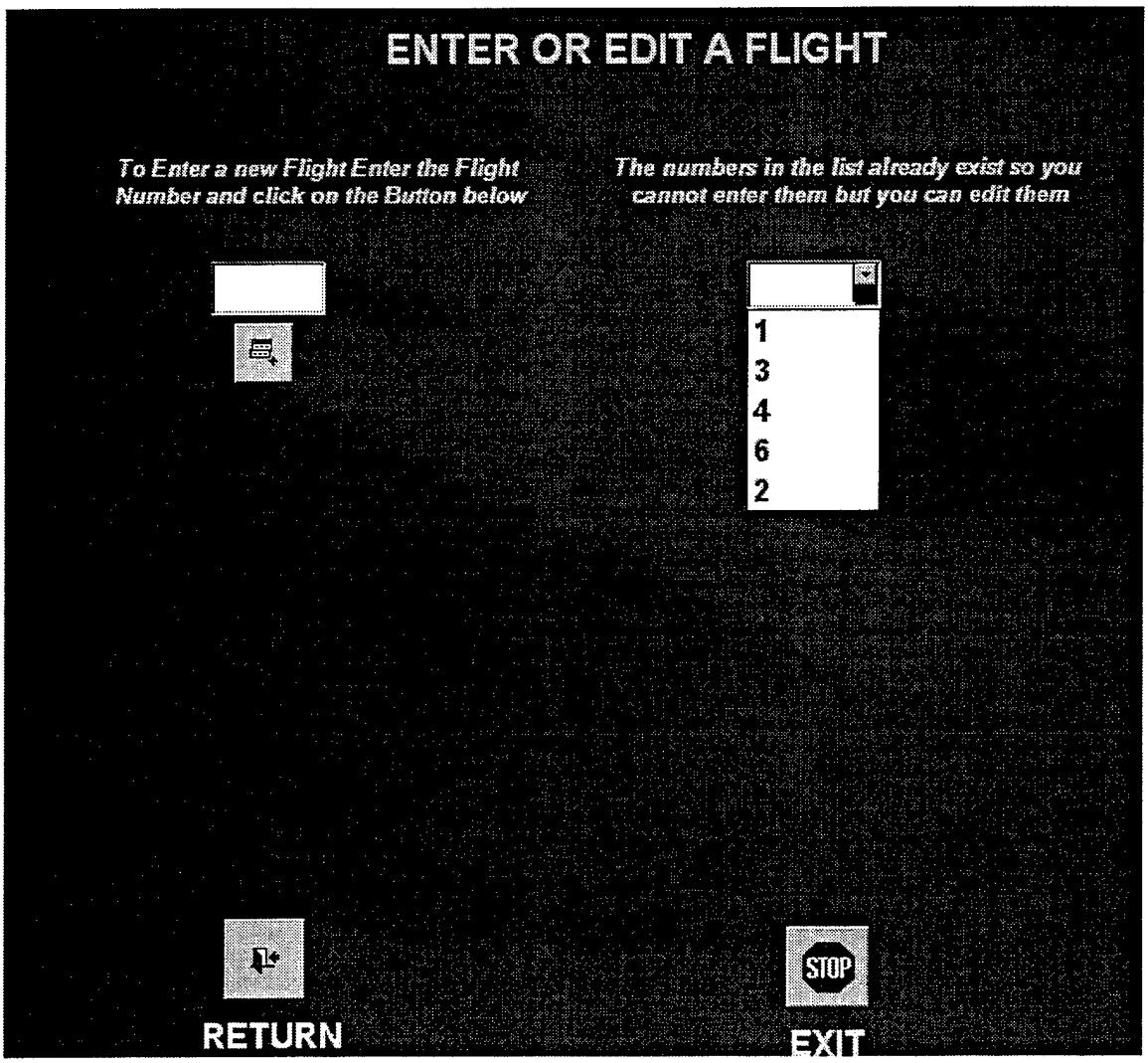


Figure 33. Enter or Edit a Flight

24. "Flight Data "

- Enter the necessary flight information.
- Fill all data fields.
- Select "Return."

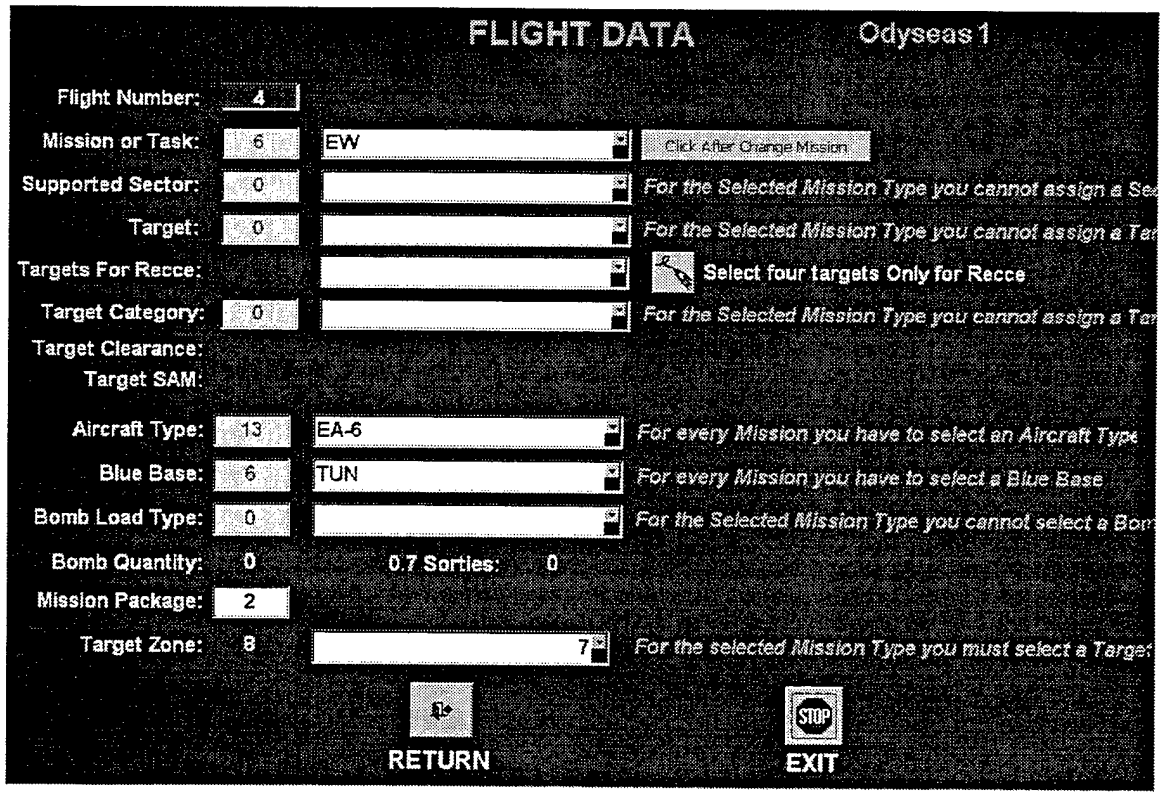


Figure 34. Input or Update Flight Data

25. **"Estimated Results Choices"**

- "Aircraft Type."
- "Blue Base."
- "Red Base or Target Number."
- "Target Type."
- "Zone of Target."
- "Task Type."
- "Packages."
- "All Flights."

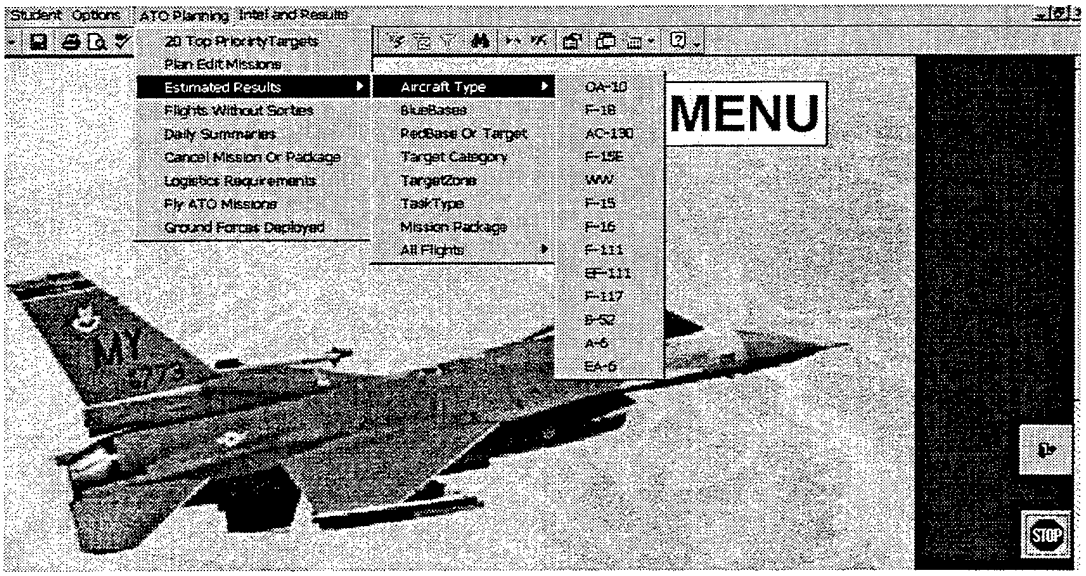


Figure 35. Estimated Results Choices

Each of these choices leads to another menu of choices that describes a filter for the way that the user can see the "Estimated Results" report.

26. "Estimated Results"

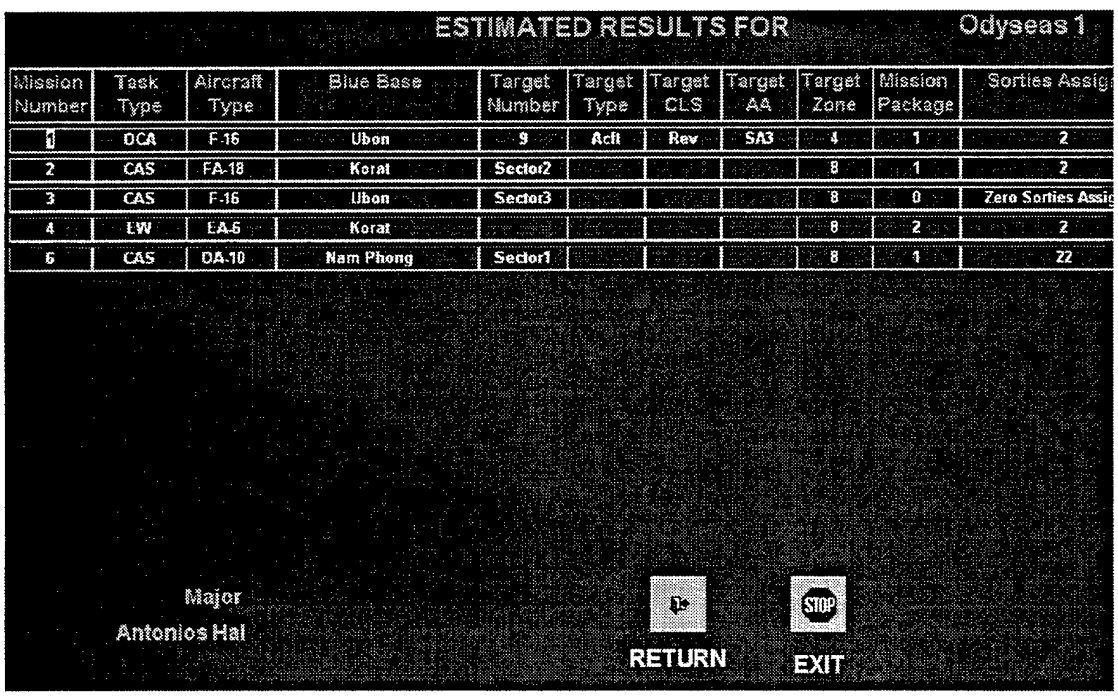
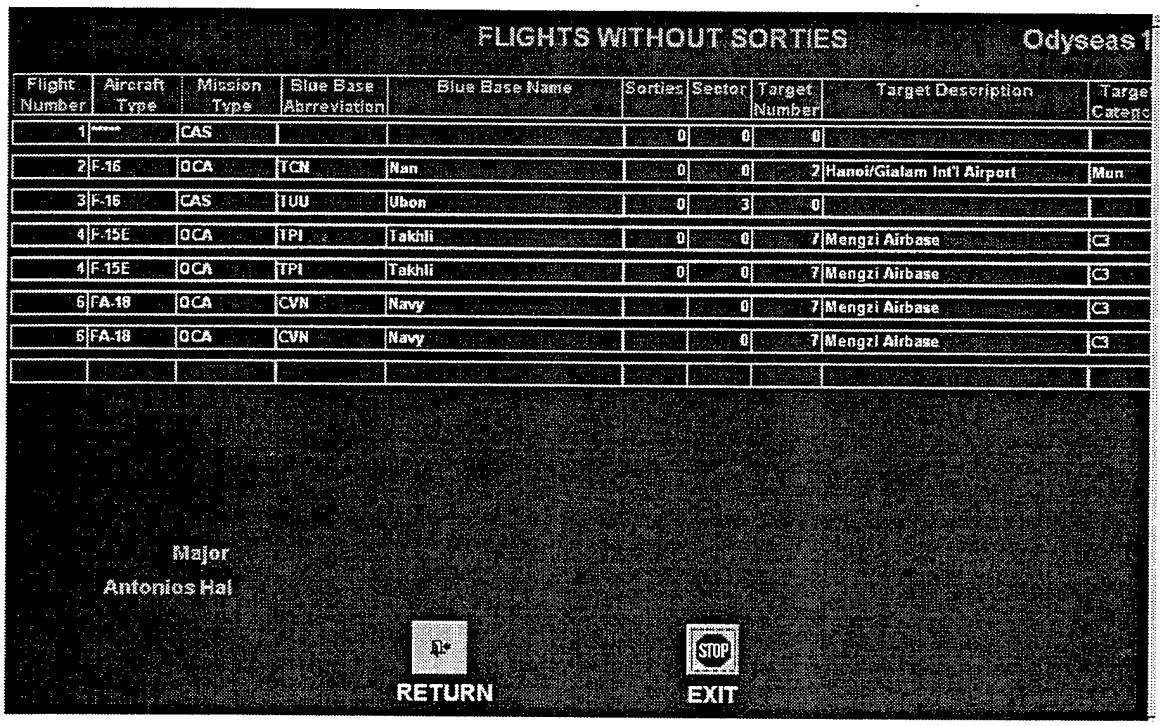


Figure 36. Estimated Results

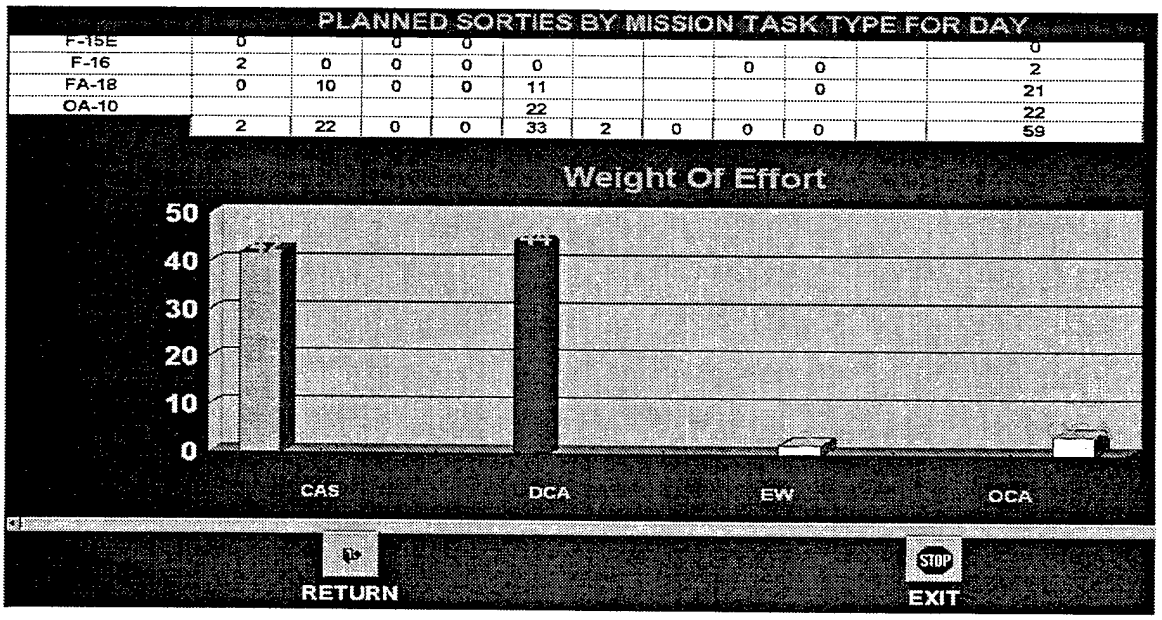
27. "Flights without Sorties"



Flight Number	Aircraft Type	Mission Type	Blue Base Abrreviation	Blue Base Name	Sorties	Sector	Target Number	Target Description	Targe Catego
1		CAS			0	0	0		
2	F-16	OCA	TCN	Nan	0	0	2	Hanoi/Gialam Int'l Airport	Mun
3	F-16	CAS	TUU	Ubon	0	3	0		
4	F-15E	OCA	TPI	Takhli	0	0	7	Mengzi Airbase	C3
4	F-15E	OCA	TPI	Takhli	0	0	7	Mengzi Airbase	C3
6	FA-18	OCA	CVN	Navy		0	7	Mengzi Airbase	C3
6	FA-18	OCA	CVN	Navy		0	7	Mengzi Airbase	C3

Figure 37. Flights without Sorties

28. "Daily Summaries "



F-15E	0		0	0						0
F-16	2	0	0	0	0		0	0		2
FA-18	0	10	0	0	11			0		21
OA-10					22					22
	2	22	0	0	33	2	0	0	0	59

Figure 38. Daily Summary

29. "Cancel Mission or Package"

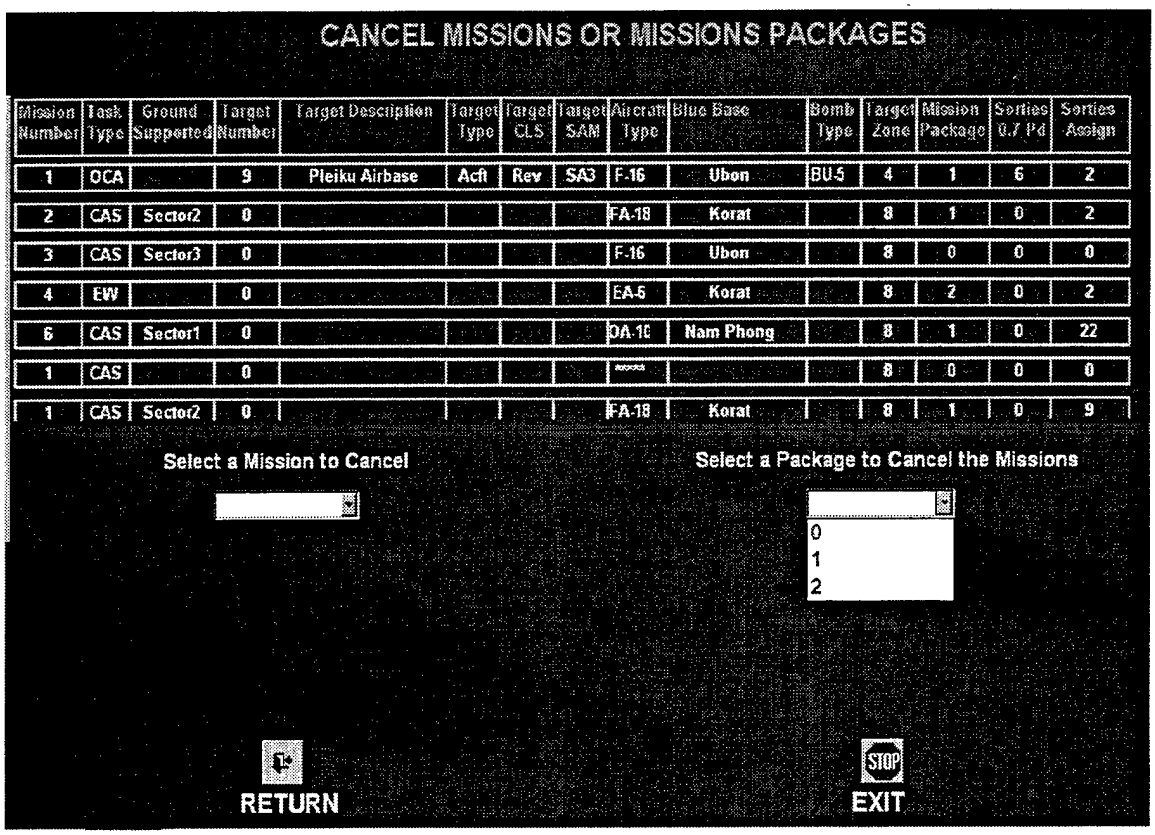


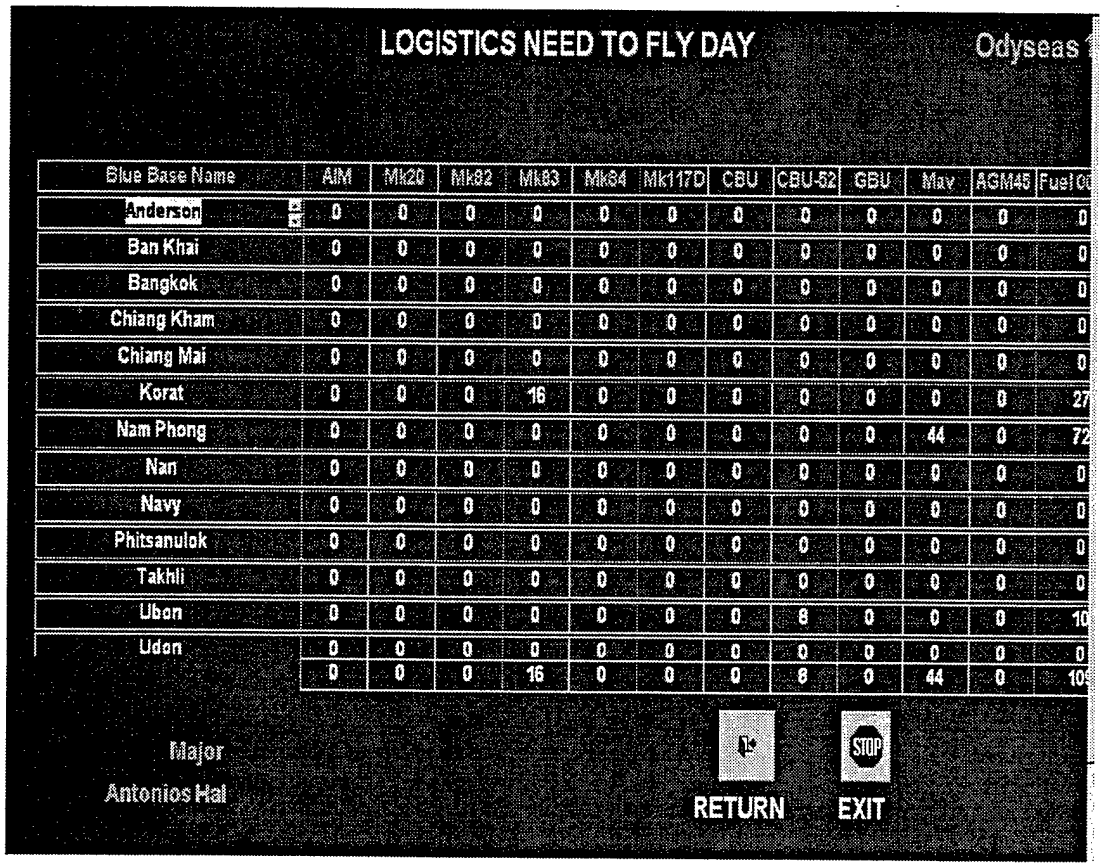
Figure 39. Cancel Mission or Missions' Package

- "Select a Mission to Cancel."
- "Select a Package to Cancel the Missions."

System returns a List of the planned missions or packages (depending on the user's choice.)

Select a Mission or Package Number. The system deletes the selected mission or the missions of the selected package.

30. "Logistic Requirements "



Blue Base Name	AIM	Mk20	Mk82	Mk83	Mk84	Mk117D	CBU	CBU-52	GBU	Mav	AGM45	Fuel
Anderson	0	0	0	0	0	0	0	0	0	0	0	0
Ban Khai	0	0	0	0	0	0	0	0	0	0	0	0
Bangkok	0	0	0	0	0	0	0	0	0	0	0	0
Chiang Kham	0	0	0	0	0	0	0	0	0	0	0	0
Chiang Mai	0	0	0	0	0	0	0	0	0	0	0	0
Korat	0	0	0	16	0	0	0	0	0	0	0	27
Nam Phong	0	0	0	0	0	0	0	0	0	44	0	72
Nan	0	0	0	0	0	0	0	0	0	0	0	0
Navy	0	0	0	0	0	0	0	0	0	0	0	0
Phitsanulok	0	0	0	0	0	0	0	0	0	0	0	0
Takhli	0	0	0	0	0	0	0	0	0	0	0	0
Ubon	0	0	0	0	0	0	0	8	0	0	0	10
Udon	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	16	0	0	0	8	0	44	0	105

Figure 40. Logistics Requirements

31. "Fly ATO Missions"

System executes mission and changes state to the next day.

32. "Ground Forces Deployed"

As shown Figure 18

33. "Intel and Results"

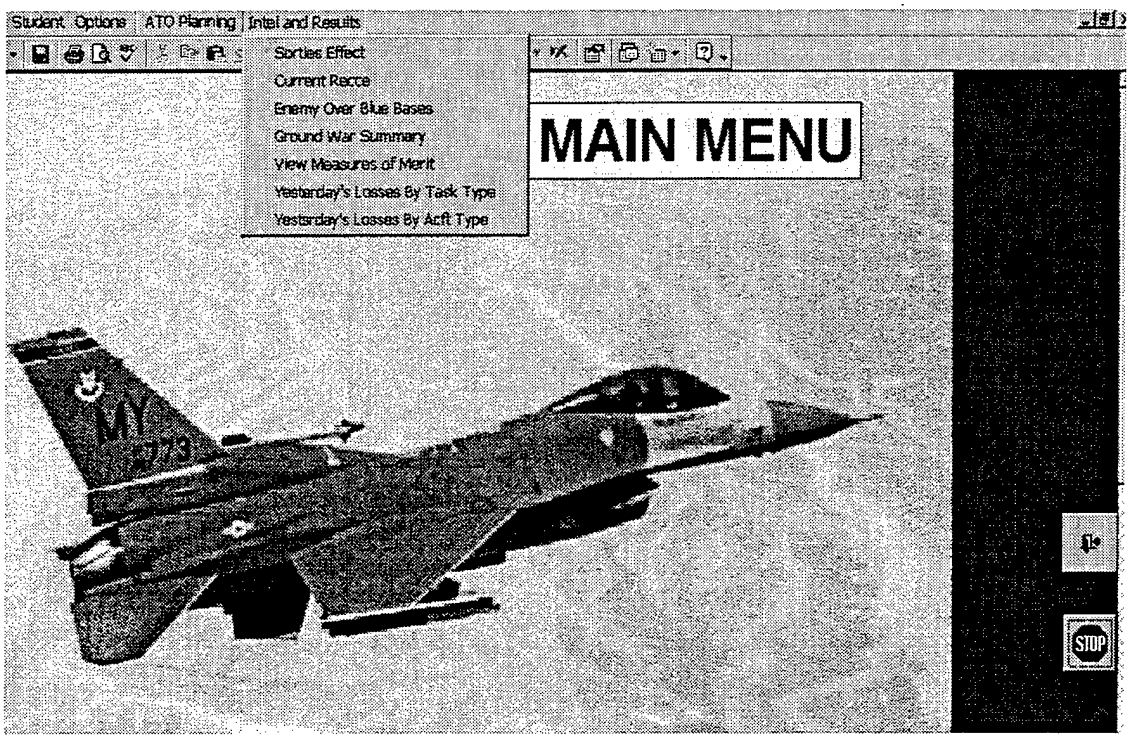


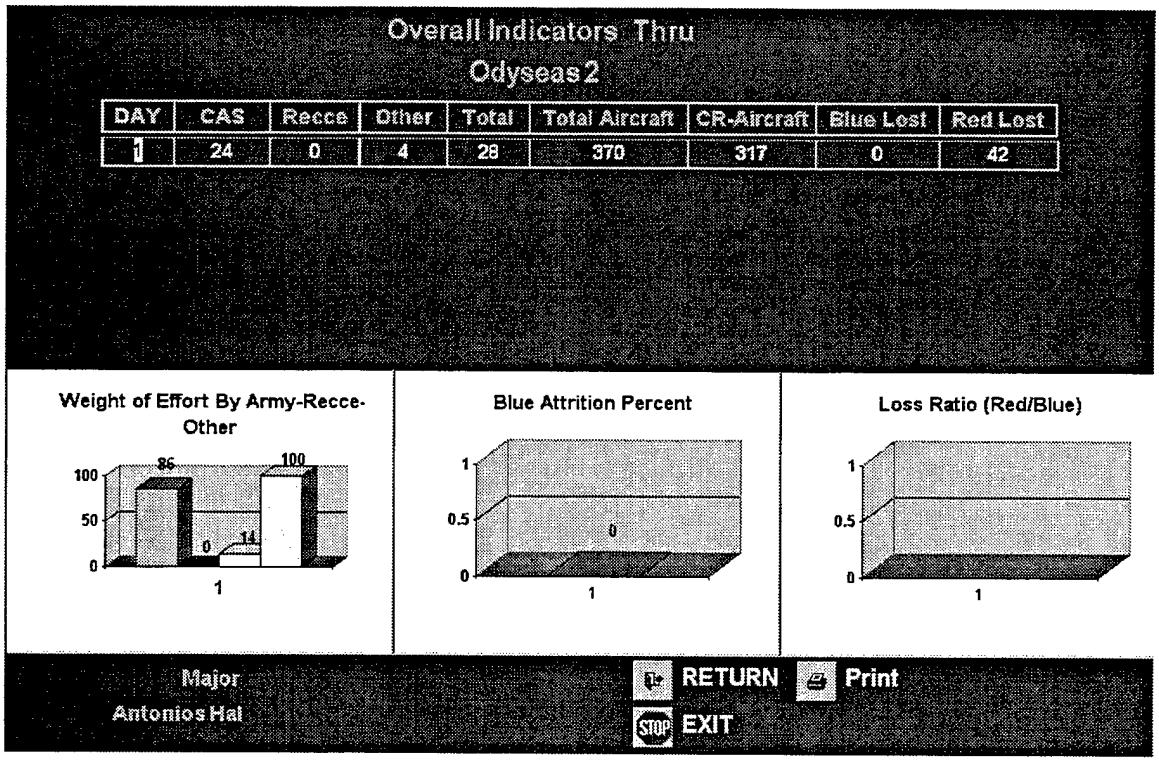
Figure 41. Menu Intel Results

They are active only if the user has executed a planning (Fly an ATO):

- "Sorties Effect" must be selected to see the "Sorties Effect" report of his planning.
- "Current Recce" must be selected to see the "Current Recce" report of his planning.
- "Enemy Over Blue Bases" must be selected to see the "Enemy Over Blue Bases" report of his planning.
- "Ground War Summary" must be selected to see the "Ground War Summary" report.
- "View Measures of Merit" must be selected to see the "View Measures of Merit" report.

- "Yesterday Losses by Task Type" must be selected to see the "Yesterday Losses by Task Type" report.
- "Yesterday Losses by Acft Type" must be selected to see the "Yesterday Losses by Aircraft Type" report.

Depending on the user's choice, the selected report will be displayed. In addition, user can choose to print the report.



DAY	CAS	Recce	Other	Total	Total Aircraft	CR-Aircraft	Blue Lost	Red Lost
1	24	0	4	28	370	317	0	42

Figure 42. Sample Intel Report "Measures of Merit"

VI. CONCLUSIONS

This thesis examined three distributed component architectures: Microsoft's DCOM, Sun Microsystems's JINI, and the OMG's CORBA, and concluded that they have similar capabilities, but each has distinct strengths and weaknesses. CORBA seems to be the best solution for using legacy applications as components because of its ability to cross languages boundaries. In practice, however, most old applications cannot be used as components because they are not implemented using object-oriented technology. Thus, it is necessary to redesigned and re-implement the legacy applications using object-oriented technology before they can be used as distributed components. In this case, JINI in combination with the Java programming language comprises the most complete solution. JINI implementations are fully compatible to each other and Java supports extensive object-oriented design and implementation.

The analysis and redesign of CAMPEX proved the concept that old applications are not useless. The old version of CAMPEX was the main source for the Requirements Analysis and Design Specification. The Requirements Analysis was a product of reverse engineering the old CAMPEX functionality and much of the detailed design are resulted from reverse engineering the CAMPEX source code. The process of reverse engineering was not cost free. It required more time and added difficulties and risks to the whole process.

Using the Unified Modeling Language (UML) for the analysis and the design of a software product adds risk to the development process when the designer is not experienced in using the language. The primary advantage of the UML Methodology is that the processes overlap each other, thus the designer gain a more complete knowledge

for the whole problem. On the other hand the overlap in the UML process consumes additional time. Moreover the design results are often ambiguous and different people can design the same product differently and different people can read UML products differently. For UML to give a clear view of a problem to everyone involved in the development process, it must be used in combination with customer and team reviews.

CAMPEX uses a "Two Tier Architecture" through the choice of ACCESS 2000 for the prototype implementation. A "Three Tier" object-oriented design could provide added benefits only under special circumstances. The designer must keep in his mind that one can validate fully the Requirements Specification, use the same GUI design for the final application and verify the correctness of objects, but one can only partially validate the design of object interactions.

Keeping the prototype's Database and GUI and implementing the computations with a classic object-oriented programming language will complete the CAMPEX Employment Module implementation. CAMPEX will be the basis for implementation of other Air War College planning modules. Parts of CAMPEX will be components of other modules, and CAMPEX itself will be a component in a distributed environment where students exercise air campaign plans.

APPENDIX A – USE CASES

USE CASE (U1): START EMPLOYMENT MODULE

Actors: Student

Purpose: Select to start the CAMPEX Employment Module

Overview: Student Selects to use the CAMPEX Employment Module, the program displays the Introduction Reports and "Ground Forces Report." Alternative the student can select to see only the "Ground Forces Report" and continue with the program.

Type: Primary and essential

Cross References: R3.1, R3.2, R3.3, R3.4, R3.4, R3.5, R3.6, R3.6, R3.7, R3.8, R3.9, R3.10, R3.11, R3.12, R3.13, R3.14, R3.15, R3.16, R3.17, R3.18, R3.19, R3.20, R3.21, R3.22, R10.1, R10.11, R10.12, R10.13, R10.14

Use Cases: -

Section: Main

Typical Course Events

	Actor Action	System Response
1.	This use case begins when student selects to load the CAMPEX Employment Module.	
2		Displays the initial screen of

3. Selects "Continue"

4. Asks student if he wants or does not want to see the Introduction Reports.

5. The student chooses an option to continue:

   To see Introduction Reports, go to "Intro Screen" section

   To continue without seeing the "Introduction Reports," go to "Continue" section

6. Displays Ground Combat Units as of Day 1 screen

7. Selects to Continue

8. Displays the ATO Management screen

Alternative Courses: -

Section: Intro Screens

Typical Course Events

Actor Action	System Response
1. Selects to see "Intro Screens"	

2. Gets all the Introduction Reports

i

3. Displays "Read me file for CAMPEX Employment Module"

4. Selects to Continue

5. Displays the "Execute Order"

6. Selects to Continue

7. Displays "DIA Intel Update"

8. Selects to Continue

9. Displays "Thai Forces Available"

10. Selects to Continue

11. Displays "Weather Report "

12. Selects to Continue

13. Displays "Navy Update"

14. Selects to Continue

15. Displays "Weapon Availability Update"

16. Selects to Continue

17. Displays "Program Notes"

18. Selects to Continue

19. Displays "Bomb Damages Assessment and Target Definitions"

20. Selects to Continue

21. Displays "Analysis and Corrections"

Section: Continue

Typical Course Events: No additional events

USE CASE (U2): STUDENT INFO

Actors:	Student
Purpose:	Student identifies himself
Overview:	Student inserts his personal information in the CAMPEX. If he has already entered his personal information just selects his own name. Alternative he can change his information.
Type:	Primary and essential
Cross References:	R1.1, R1.2, R1.3, R1.4, R1.5, R1.6, R1.7, R10.11, R10.12, R10.13, R10.14 Use Cases: "Start Employment Module" Use Case (U1)

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case begins when student selects to identify himself to execute an exercise	
	2. Presents available "Student" options

3 Chooses:

Select an "Existing User Name," go to "Select Existing User" section.

Enter a "New User," go to "New User" section.

"Change Student Information," go to "Change User Information" section.

4. Presents CAMPEX Main Menu screen

Section: Select Existing User

Typical Course Events

Actor Action	System Response
1. This use case begins when student selects "Existing User"	
	2. Presents available students' names
3 Chooses his name	
	4. Creates the selected student instantiation

Section: New User

Typical Course Events

Actor Action	System Response
1. This use case begins when student selects "New User"	
	2. Presents user input screen
3 The student enters requested personal information	
4. Select to "Return"	
	5. Stores student entered information
	6. System returns student to previous GUI from where student can follow the process of section "Existing Student"

Section: Change User Information

Typical Course Events

Actor Action	System Response
1. This use case begins when student selects "Update User Information"	
	2. Presents the selected user information

3 The student inputs his new information.

4. Select to "Return"

5. Stores student entered information

6. System returns student to previous GUI from where student can follow the process of "Existing Student" section.

USE CASE (U3): LOAD AN ATO

Actors:	Student
Purpose:	Load an ATO
Overview:	Student selects an ATO. With completion student is entered to "Main Menu" and the selected ATO is loaded.
Type:	primary and essential
Cross References:	R4.1, R.4.2 Use Cases: Student must have completed the • "Start Employ" Use Case (U1) • Student must already entered the ATO that wants to load

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case starts when the student selects to "Load an ATO"	
2. Selects an existing ATO from the list.	
	3. Creates and updates the necessary objects of the selected ATO
3. Select to continue	
	4. Displays the "Main Menu" screen of ATO Employment Module

USE CASE (U4): MANAGE AN ATO

Actors:	Student
Purpose:	Mange an ATO
Overview:	Student selects to manage to an ATO. With completion selected or new ATO information should be in CAMPEX Employment Module.
Type:	primary and essential
Cross References:	R4.1, R4.3, R4.4, R4.5, R4.6
	Use Cases: Student must have completed the

- "Start Employment Module " Use Case (U1)
- "Student Info" Use Case (U2)

Section: Main

Typical Course Events

Actor Action	System Response
1. Chooses one of the "ATO Management" options. Student selects "Start a new ATO from scratch," go to "Start new ATO from scratch" section. "Erase an existing ATO," go to "Erase an ATO" section. "Copy an ATO to a new file, load new," go to "Copy an ATO" section.	1.
2. Continue with program	
	3. Displays the ATO Selection screen

Section: Start a new ATO from scratch

Typical Course Events

Actor Action	System Response
1. Selects Start a new ATO from scratch	
	2. Asks to enter the new ATO information
3. Input the new ATO information	
	4. Stores the new ATO information

Section: Copy an ATO

Typical Course Events

Actor Action	System Response
1. Selects Copy an ATO	
2 Enters the new ATO name	
	3. Creates necessary ATO objects
4. Selects an existent ATO	
	5. Updates the entered ATO to the selected ATO information.

Section: Erase an ATO

Typical Course Events

Actor Action	System Response
1. Selects Erase an existent ATO	

2. Selects an existent ATO

3. Deletes selected ATO File information

4. Displays the "ATO Selection" screen

USE CASE (U5): DESCRIBE THE 20 TARGETS WITH HIGHEST PRIORITY

Actors:	Student
Purpose:	Fill the list with 20 targets with highest priority
Overview:	Student has decided which targets of his plan have the highest priority. Student edits the list of 20 targets with highest priority. After the completion of this use case the student's "20 Highest Priority Target List" can be displayed by the application.
Type:	Primary and essential
Cross References:	R5.1, R5.2, R5.3, R5.4, R10.2, R10.11, R10.12, R10.13, R10.14 Use Cases: Student must have completed: • "Start Employment Module" Use Case (U1) • "Student Info" Use Case (U2) • "Load an ATO" Use Case (U3)

Section Main

Typical Course Events

Actor Action	**System Response**

1. This use case begins when student has already decided about the 20 targets with the highest priority
2. Student selects "Top 20 Targets"

3. Displays the current list of top 20 targets

4. Student chooses a Priority List place

5. Displays all targets

6. Student select one of the displayed targets

7. Updates targets priority

If the target is already in the list then the program doesn't accept the selection and displays message to the student to select another Target.

USE CASE (U6): PLAN AN ATO

Actors:	Student
Purpose:	Enters student's plans
Overview:	Student enters new missions, or edits old missions, or erases missions. With completion of this Use Case the student's plans have been entered.
Type:	Primary and essential
Cross References:	R5.5, R5.6, R5.7, R5.8, R5.9, R5.10, R10.2, R10.11, R10.12,

R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)
- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case begins when student is ready to enter his plan in the application	
2. Student selects: "Plan an ATO," go to "Plan Edit Mission" section "Cancel a Mission or a Package of Missions," go to "Cancel a Mission or a Mission Package" section	
	3. Return to Main Menu screen
	4. Displays list of missions or packages of the selected by student type

Section: Plan or Edit a Mission

Typical Course Events

Actor Action	System Response
1. Student select to Plan or Edit a Mission	
	2. Displays the available options that user can see the lists of existent Missions.
3. Chooses one option	
	4. Displays list of missions of the student's selected type
5. Selects to Enter or Edit Mission	
	6. Asks from the student to enter a new mission number or to select a mission from the list to edit
7. Student enters or selects a mission	
	8. Displays an input form for the entered or selected mission
9. Enters the mission's information	
10. Exits from the form	
	11. Enters new mission information to

the system

Section: Cancel a Mission or a Missions' Package

Typical Course Events

Actor Action	System Response
1. Student selects to Cancel a new Mission or a Mission's Package	
2. Selects an existent mission or package to cancel	
	3. Deletes selected mission or package of missions

USE CASE (U7): FLY AN ATO

Actors:	Student
Purpose:	To execute the planned missions
Overview:	Student is running fly missions. System calculates the result of the planned missions. Saves the results in a new ATO. Loads the new ATO.
Type:	Primary and essential
Cross References:	R6.1, R6.2, R6.3, R6.4, R6.5, R10.2, R10.11, R10.12, R10.13, R10.14
	Use Cases: Student must have completed:

- "Start Employment Module" Use case (U1)
- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)
- "Plan an ATO" Use Cases (optional) Use Case (6)

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case begins when student has finished his plan editing	
2. Student selects "Fly ATO"	
	3. Calculates the results of the planned missions
	4. Creates a new ATO
	5. Saves the results of the calculation to the new ATO
	6. Loads the new ATO

USE CASE (U8): INITIAL INFORMATION

Actors: Student

Purpose: Inform student for the initial data of an ATO

Overview: Student asks for initial information. With completion of this Use

Case the student has seen or printed the information that he has asked for.

Type: Primary and essential

Cross References: R7.1, R7.2, R7.3, R7.4, R7.5, R7.6, R7.7, R7.8, R7.9, R7.10, R10.2, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)
- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case begins when student has loaded an ATO or in any moment of the program	
2. Chooses to see actual data by: "Flights without Sorties" "Daily Summaries" "Logistic Requirements" "Blue Basing Summary as of Start of Current Day" "Sorties Available to Task at	

Airbases"

"Analysis"

"Recce for Targets at Start of Current Date"

3. Calculates data to create the report

4. Displays the selected report

5. Chooses one option:

print report, go to "Print Report" section

exit

6. Displays the Main Menu screen

7. Displays Main Menu screen

Section: Print Report

Typical Course Events

Actor Action	System Response
1. Selects "Print".	
	2. Prints the displayed report

USE CASE (U9): ESTIMATED RESULTS

Actors: Student

Purpose: Display the estimated results by the student plan before these plans execute

Overview: Student asks for the estimated results. With completion of this Use

Case estimated results are displayed on the screen.

Type: Primary and essential

Cross References: R8.1, R8.2, R8.3, R8.4, R8.5, R8.6, R8.7, R8.8, R8.9, R8.10, R8.11, R8.12, R8.13, R10.2, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)
- "Student Info" Use Case (U2)
- "Load an ATO" Use Case (U3)
- "Plan an ATO" Use Cases (Optional) (U6)

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case begins when student has finished (editing) his plan editing	
2. Student selects "Estimated Results"	
	3. Displays the available reports that user can see the estimated results
4. Chooses one option	
	5. Displays report of estimated results
6. Chooses available options Selects "Return" Selects "Print", go to the "Print"	

section

10. Displays Main Menu screen

Section: Print

Typical Course Events

Actor Action	System Response
1. Selects "Print"	
	2. Prints the displayed report

USE CASE (U10): ACTUAL RESULTS

Actors:	Student
Purpose:	Inform Student for the Results Reports of an ATO
Overview:	Student asks for Information. With completion of this Use Case, the student has seen or printed the information that he has asked for.
Type:	primary and essential
Cross References:	R9, R9.1, R9.2, R9.3, R9.4, R9.5, R9.6, R9.7, R9.10, R9.11, R9.12, R9.13, R9.14, R9.15, R9.16, R9.17, R9.18, R9.19, R9.20, R9.21, R9.22, R9.23, R9.24, R9.25, R9.26, R9.27, R10.2, R10.11, R10.12, R10.13, R10.14

Use Cases: Student must have completed:

- "Start Employment Module" Use Case (U1)
- "Student Info" Use Case (U2)

- "Load an ATO" Use Case (U3)
- "Fly an ATO Missions" Use Case (U7)

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case begins when student has flown an ATO and at any other moment of the program after that	
2. Chooses to see actual data by: "Cumulative Summary During the Previous Date" "Current Recce" "Enemy Over Blue Bases" "Ground War Summary" "Measures of Merit" "Yesterday Looses by Task Type" "Yesterday Looses by Aircraft Type" "Final Target Status"	
	3. Calculates data to create the report
	4. Displays the selected report
5. Chooses one option:	

print report, go to "Print Report"

section

exit

6. Displays the Main Menu screen

10. Displays Main Menu screen

Section: Print the Report

Typical Course Events

Actor Action	System Response
1. Selects "Print".	
	2. Prints the displayed report

USE CASE (U11): MAP

Actors:	Student
Purpose:	See the map of the exercise area
Overview:	Student selects to see the map by Main Menu. With completion map is displayed on the screen
Type:	Primary and essential
Cross References:	R10.2, R10.3 Use Cases: Student must have completed: • "Start Employment Module" Use Case (U1) • "Load an ATO" Use Case (U3)

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case begins when student selects from "Main Menu" screen to see the "Exercise Map."	
	2. Displays the Map on the screen

USE CASE (U12): SEND EXERCISE

Actors: Student

Purpose: To send the results of an exercise to "Air University"

Overview: Student must be connected to the "internet" first. Then, he selects Option of CAMPEX "Send Exercise Results to the Air University" and selects an exercise from the displayed. With the completion of this use case the selected exercise results are been sent to the Air University server.

Type: Primary and essential

Cross References: R2.1, R2.2, R2.3, R2.4, R10.15

Use Cases: User must have completed the

- "Student Info" Use Case (U2)
- User must have connected to the Internet

Section: Main

Typical Course Events

Actor Action	System Response
1. This use case begins when student	

decides to send the result of his exercise to the server of Air War College

2. Connect to Internet

3. Selects "Send the Result to the Air War College"

4. Collects the necessary information

5. Sends the necessary Information to Air College Server

APPENDIX B – SYSTEM SEQUENCE DIAGRAMS

SELECT STUDENT

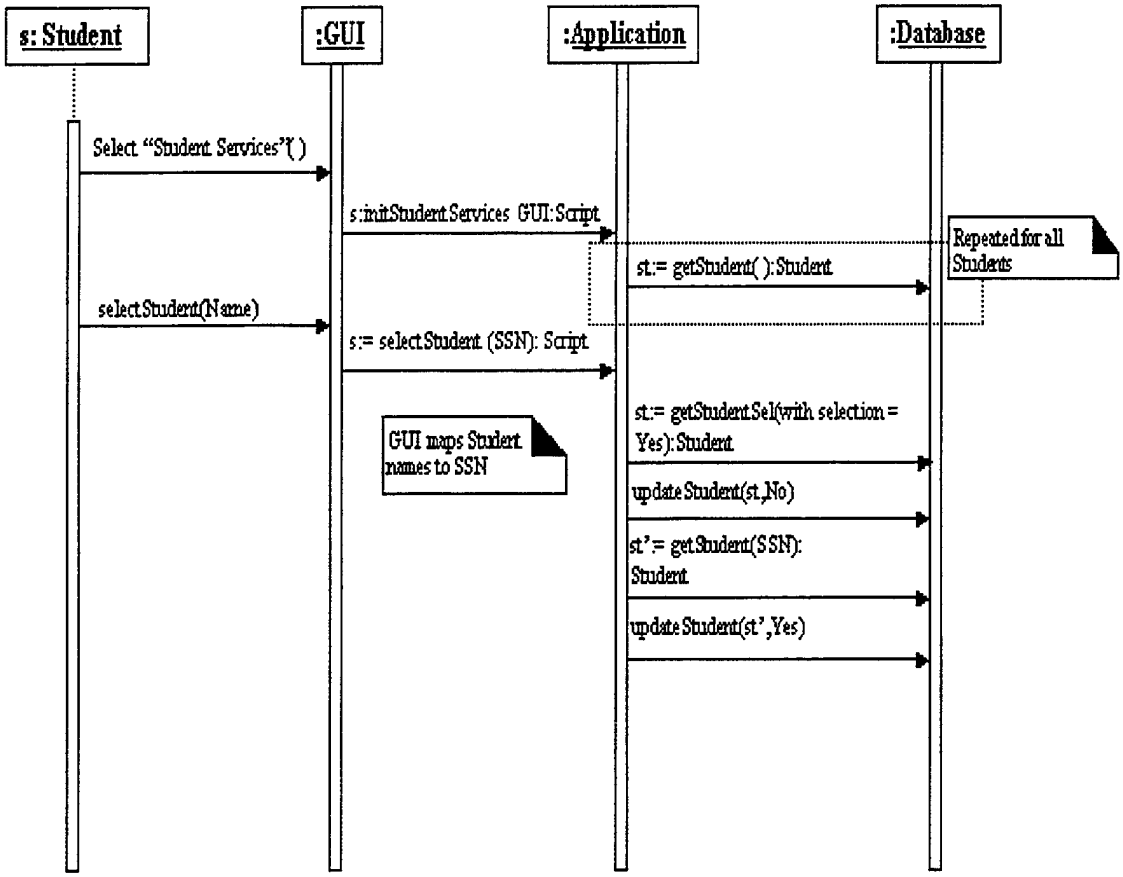


Figure 43. Sequence Diagram "Select a Student"

ADD STUDENT

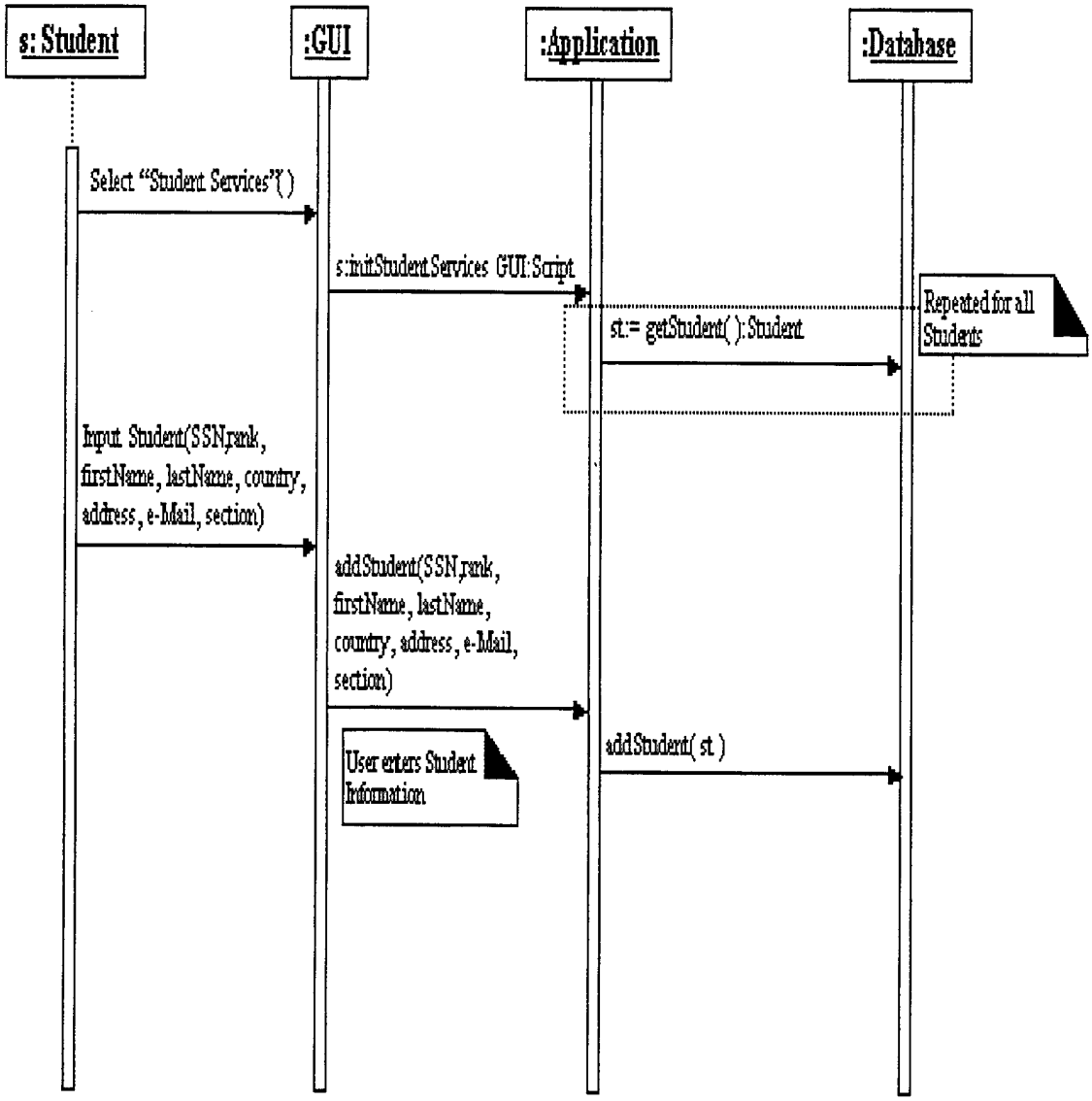


Figure 44. Sequence Diagram "Add Student"

START EMPLOY

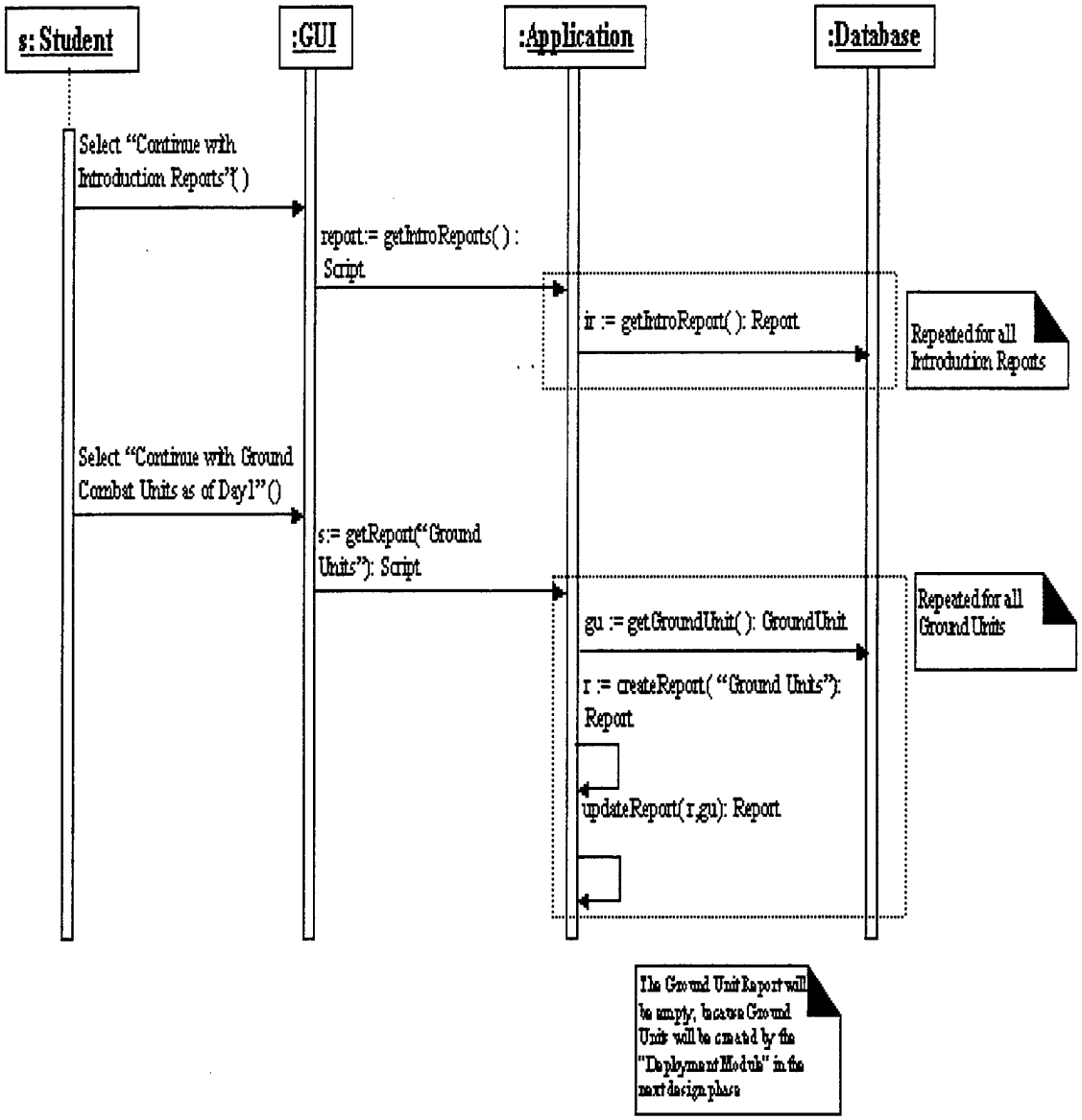


Figure 45. Sequence Diagram "Start an ATO"

SELECT ATO

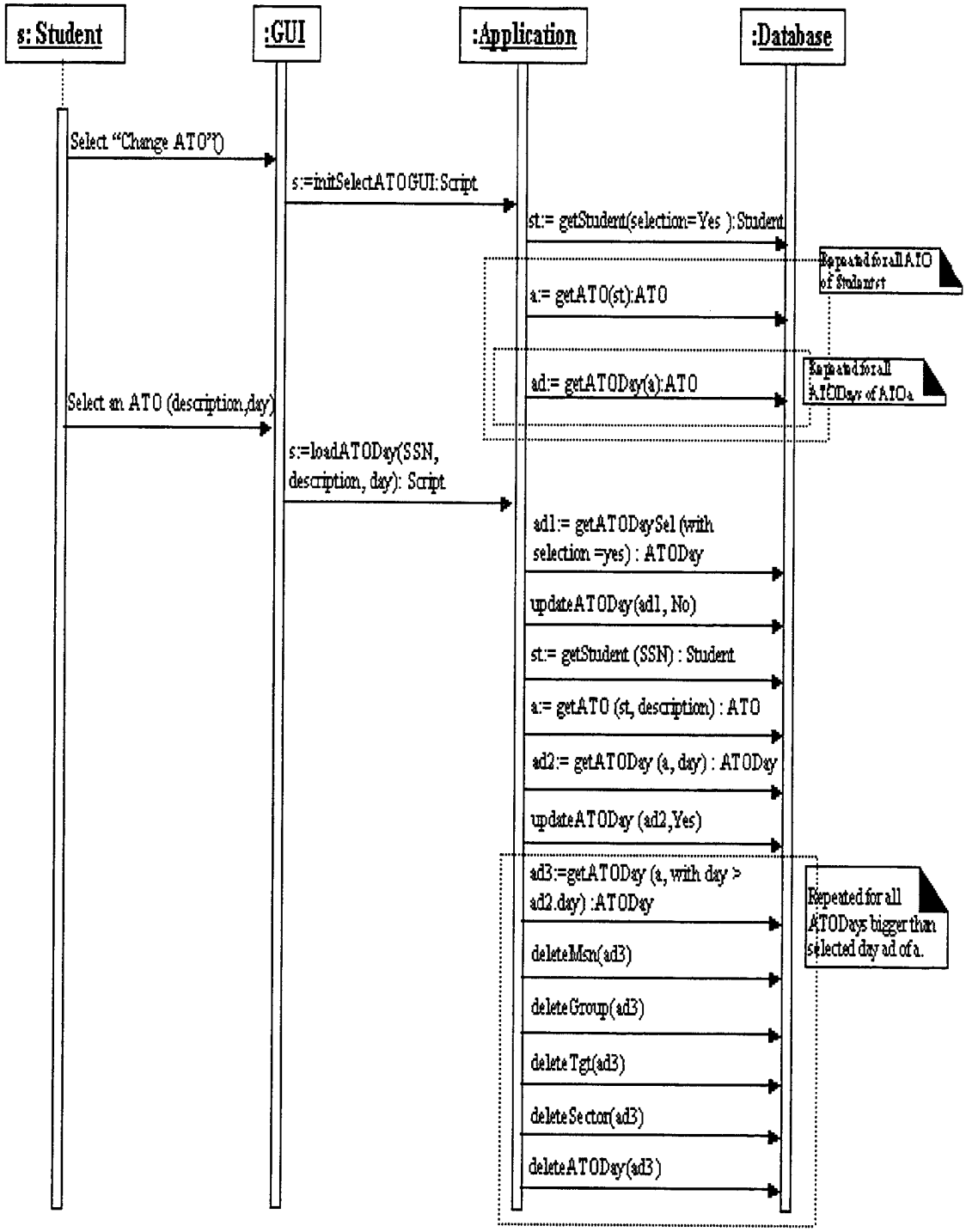


Figure 46. Sequence Diagram "Select an ATO"

NEW ATO

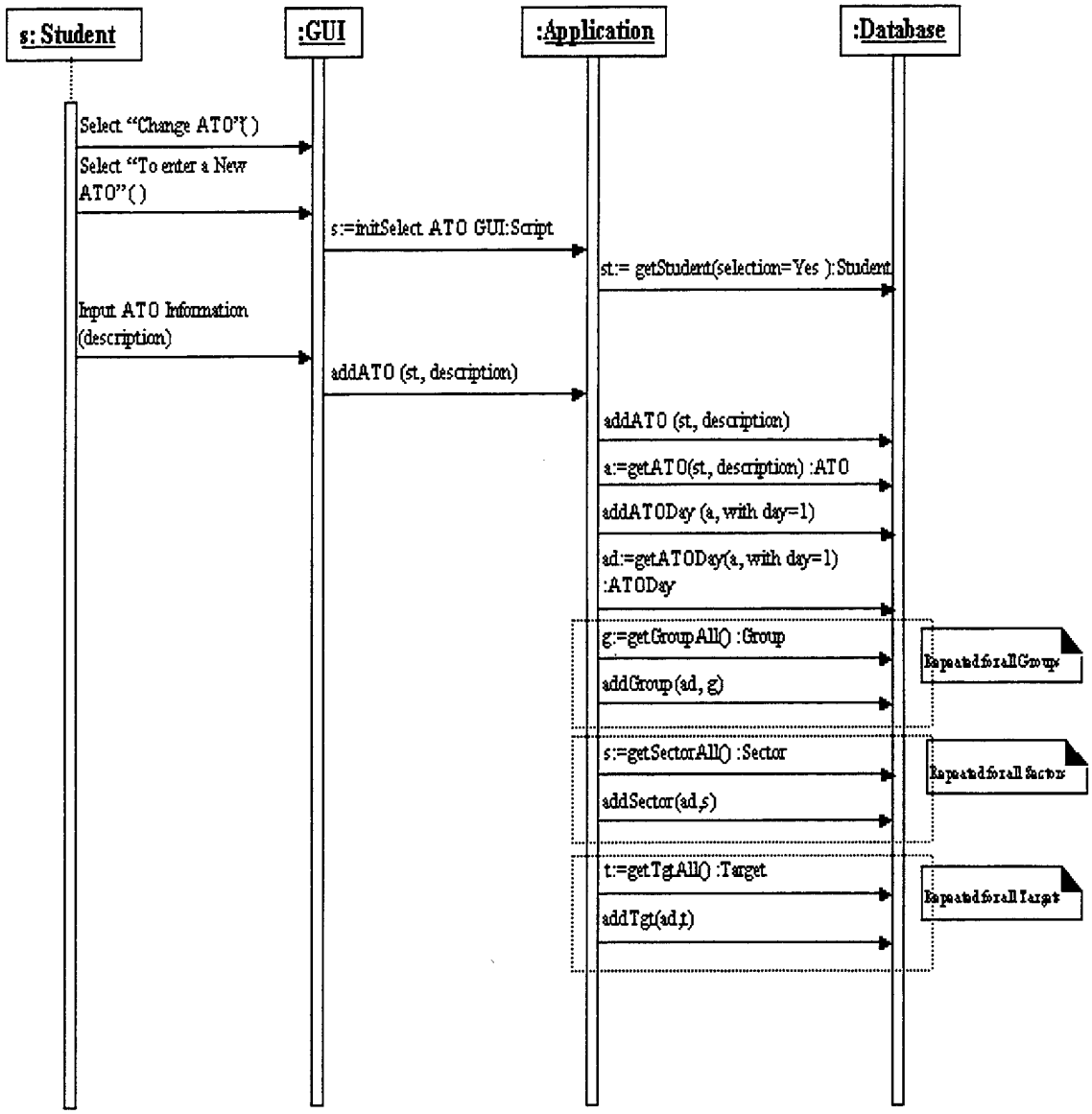


Figure 47. Sequence Diagram "New ATO"

COPY AN EXISTING ATO

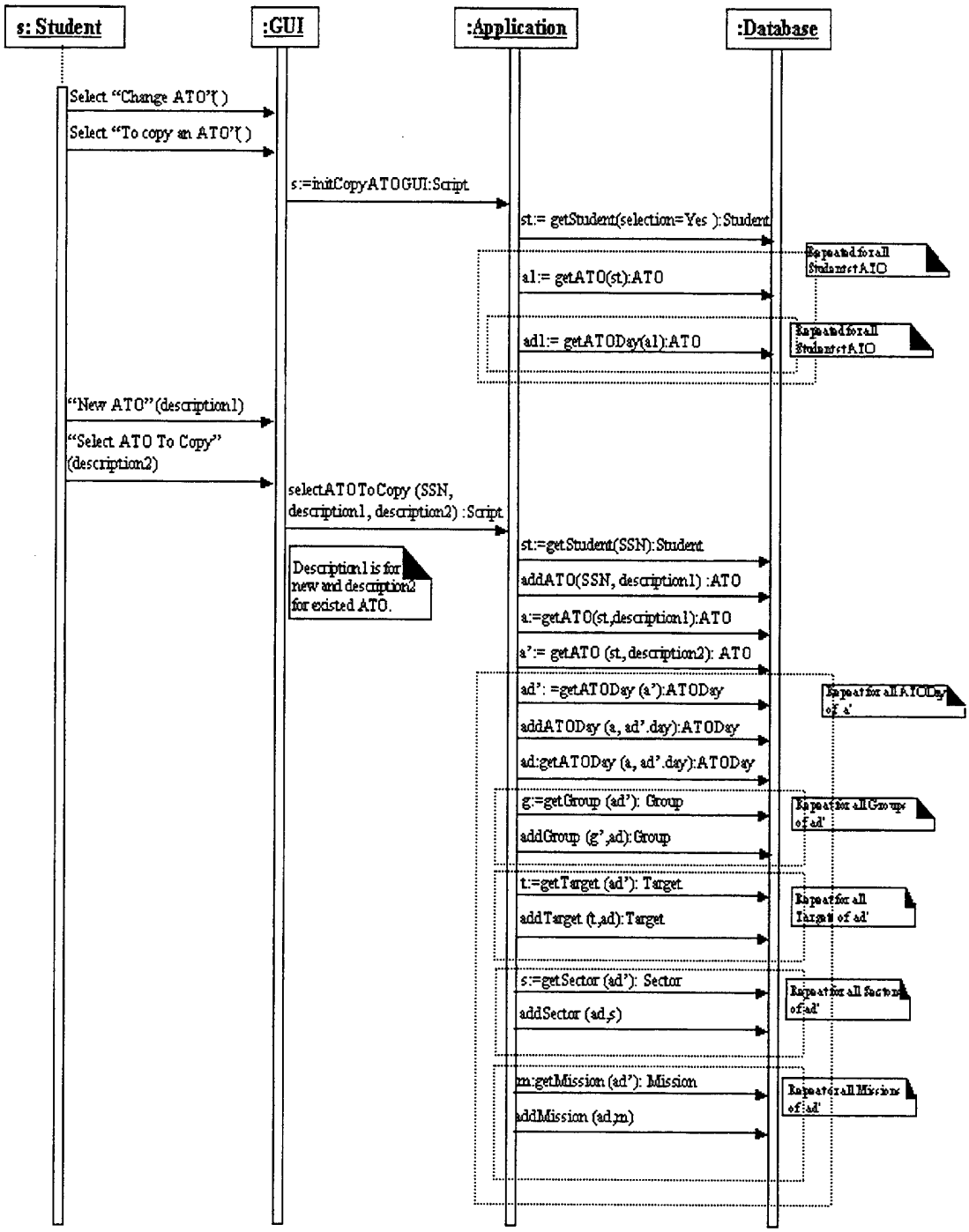


Figure 48. Sequence Diagram "Copy an Existing ATO"

ERASE ATO

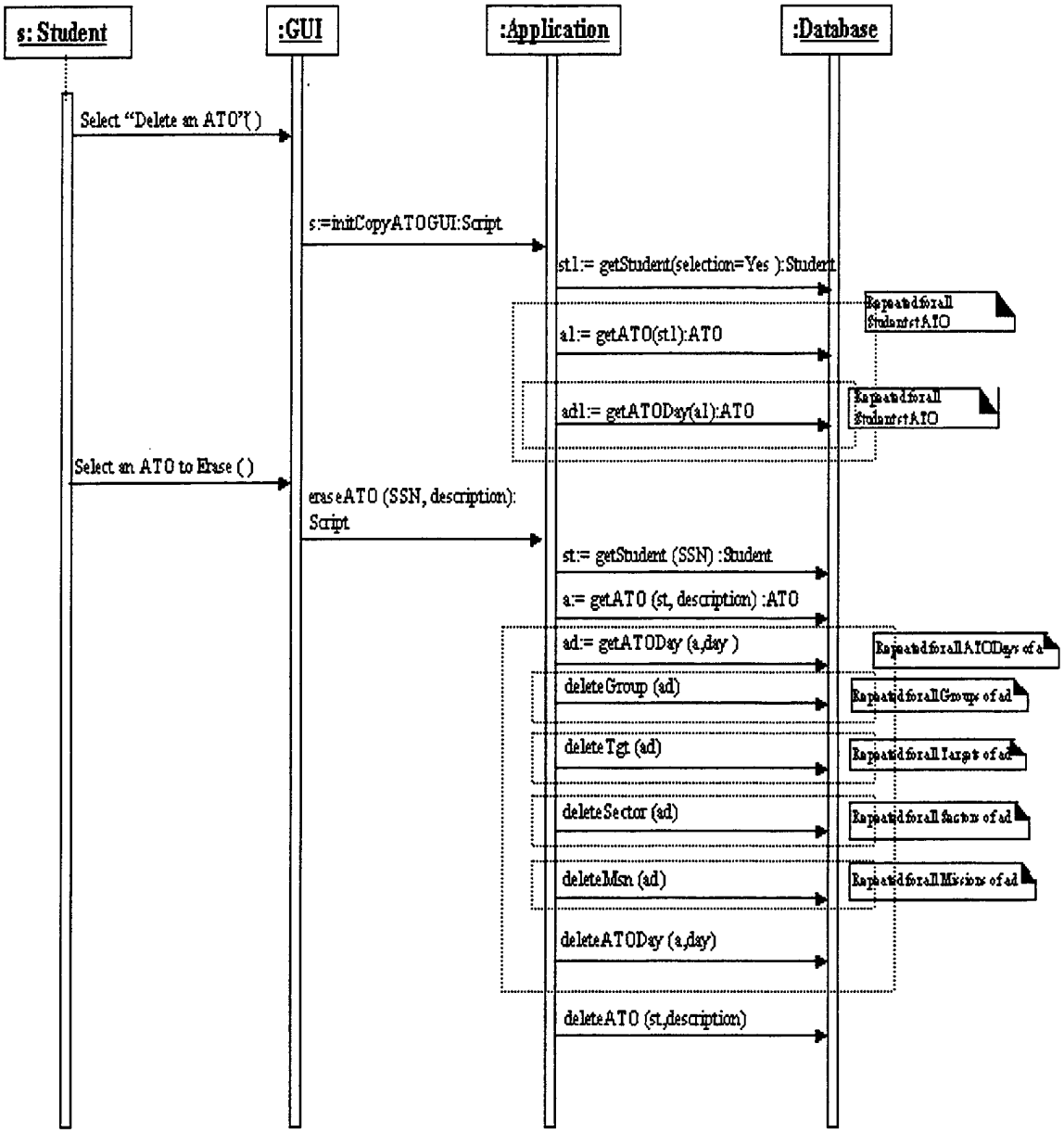


Figure 49. Sequence Diagram of "Erase an ATO"

20 TARGETS WITH HIGHEST PRIORITY

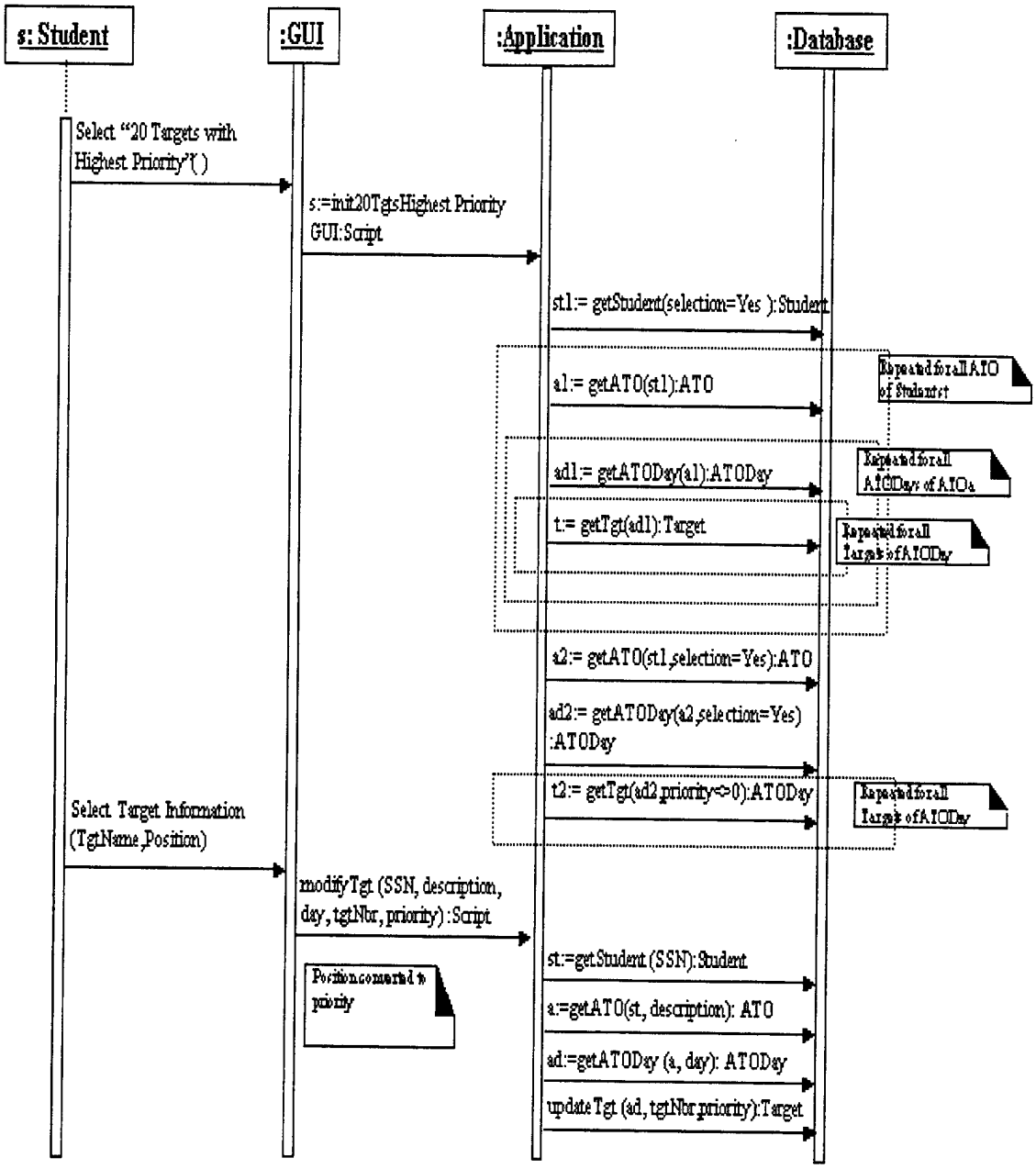


Figure 50. Sequence Diagram "20 Highest Priority Targets"

PLAN ATO ENTER A NEW MISSION

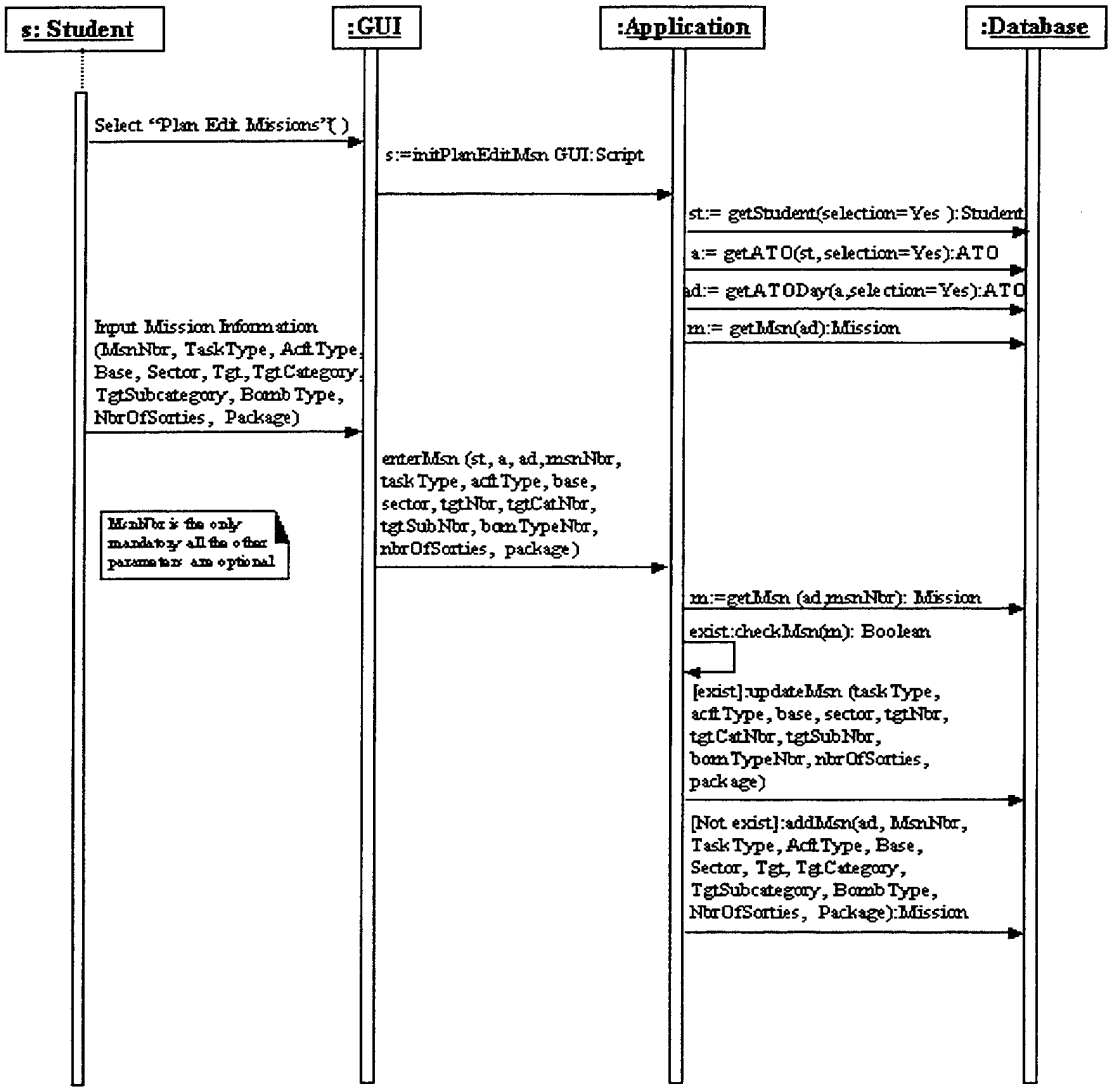


Figure 51. Sequence Diagram "Plan or Edit Mission"

CANCEL MISSION

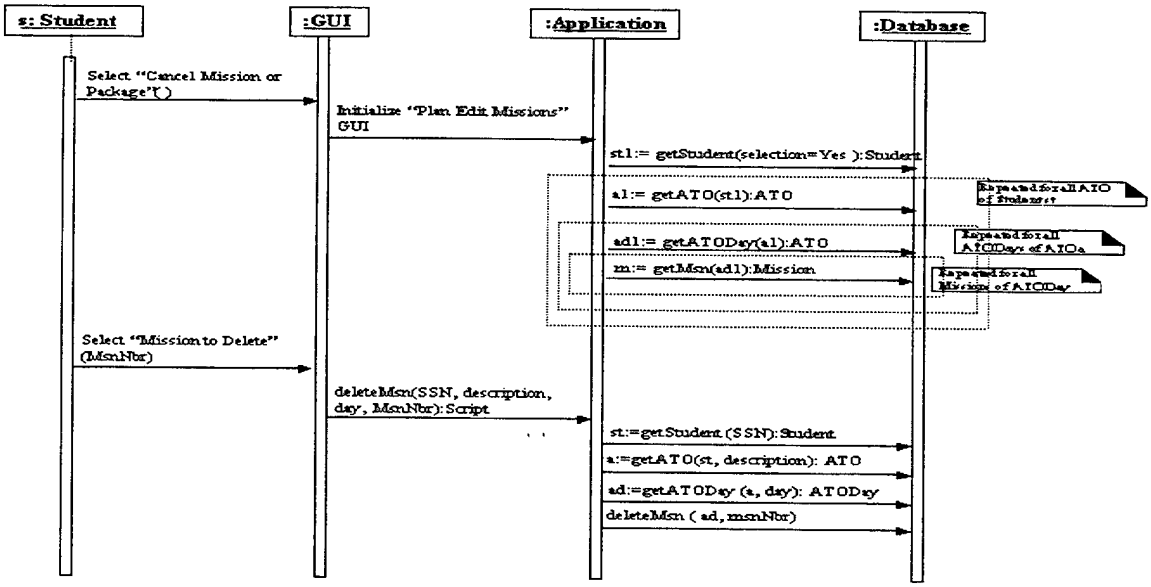


Figure 52. Sequence Diagram "Cancel Mission"

CANCEL MISSIONS OF A PACKAGE

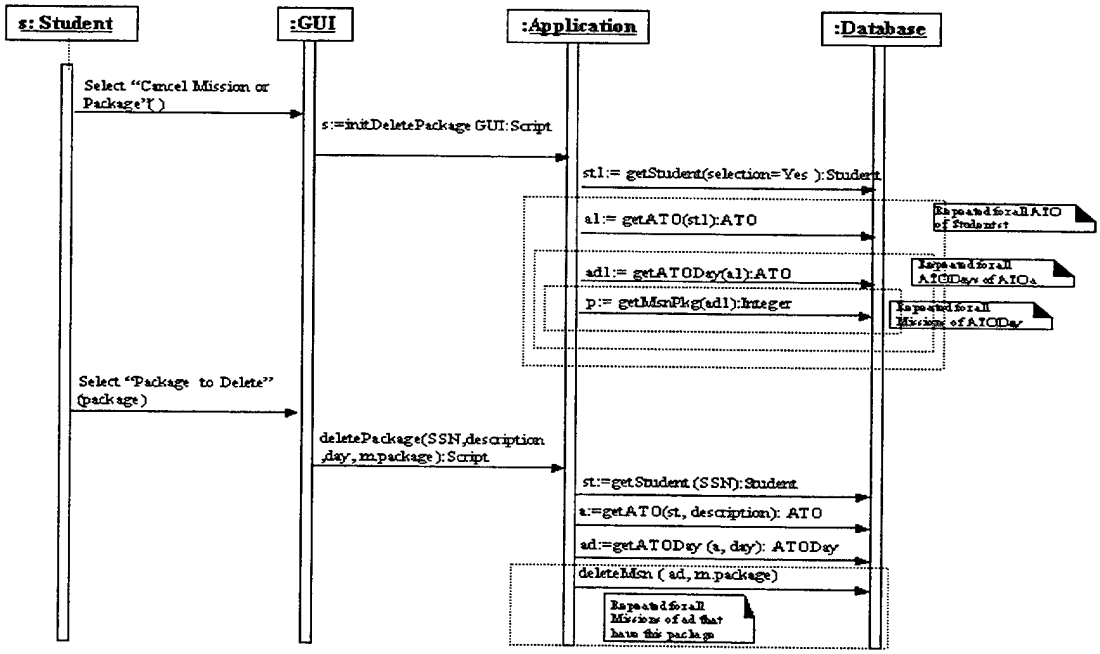


Figure 53. Sequence Diagram "Cancel Package of Missions"

FLY ATO

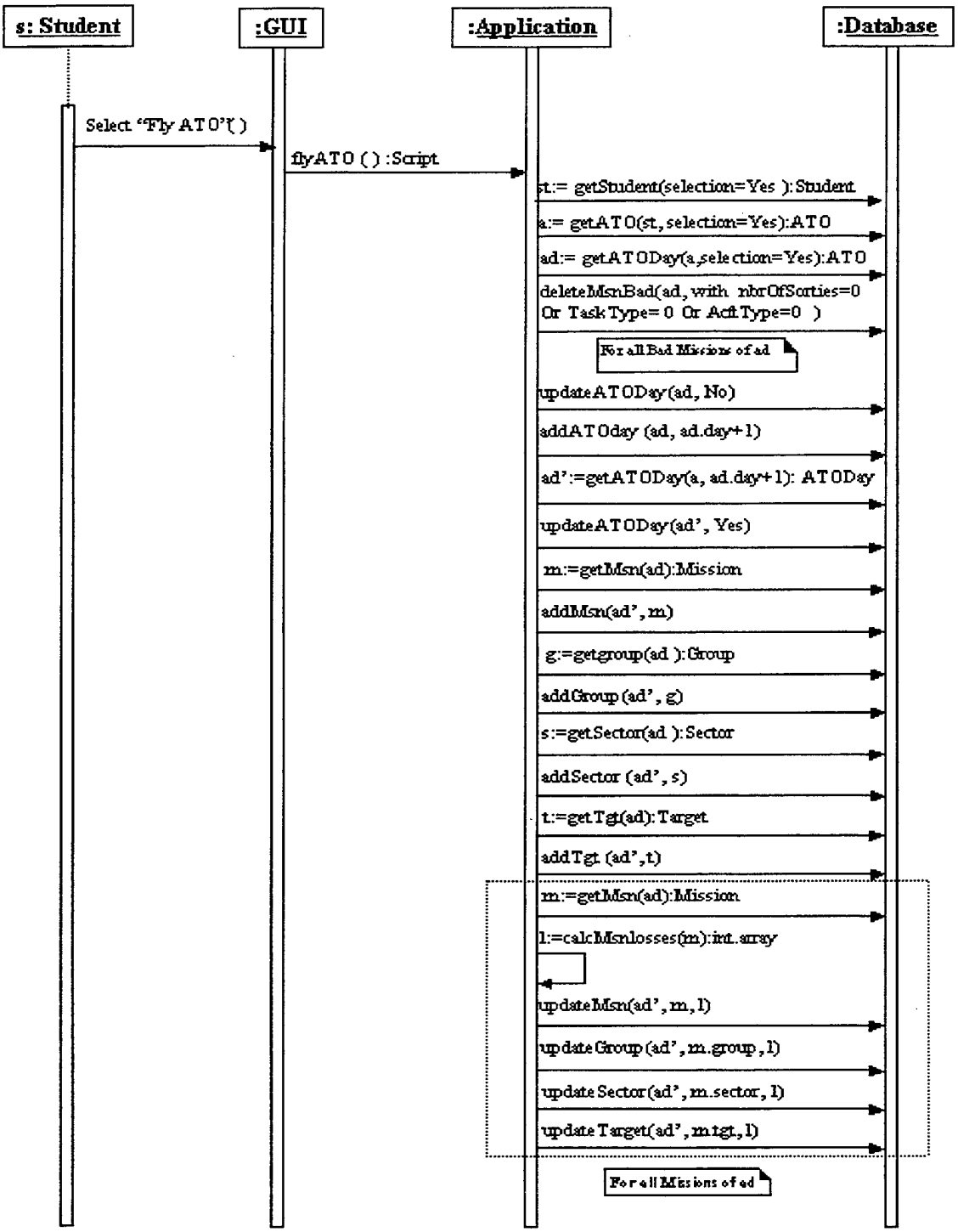


Figure 54. Sequence Diagram "Fly an ATO"

ESTIMATED RESULTS

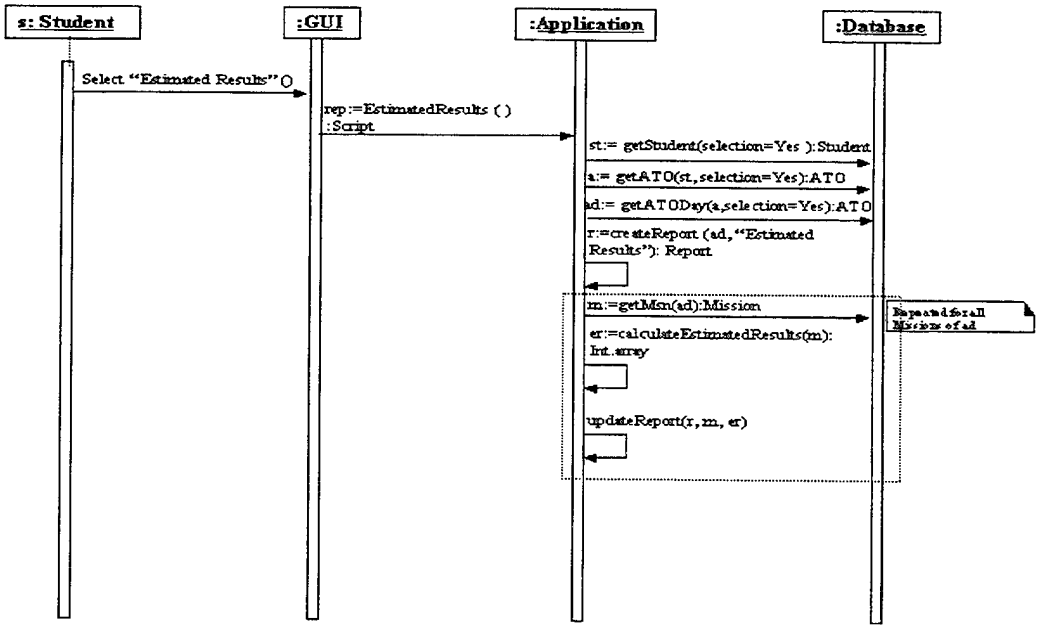


Figure 55. Sequence Diagram "Estimated Results"

MISSIONS WITHOUT SORTIES

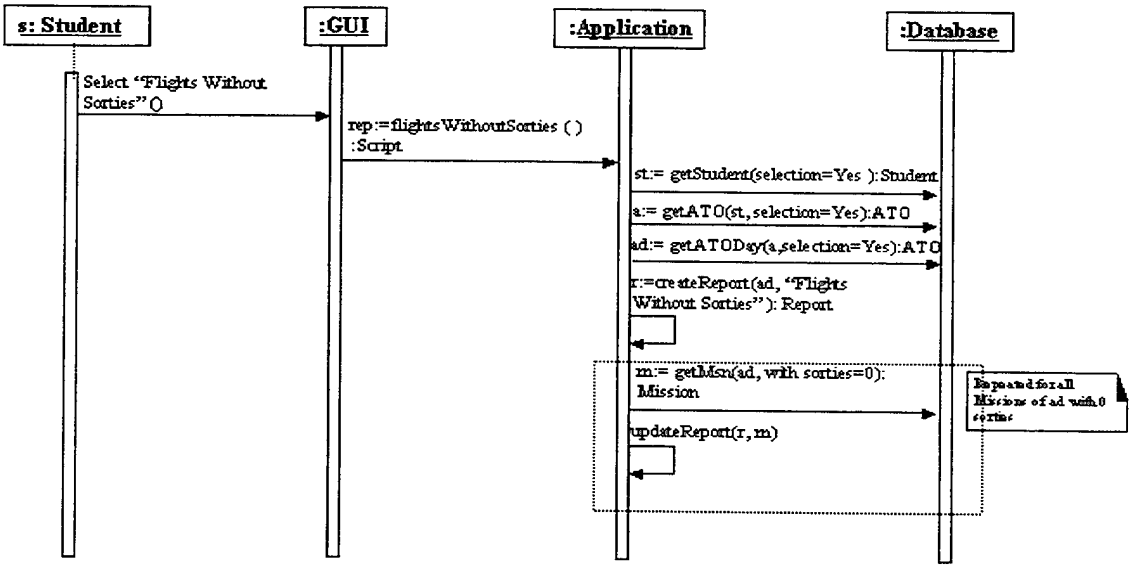


Figure 56. Sequence Diagram "Flights Without Sorties"

INITIAL INFORMATION - DAILY SUMMARY

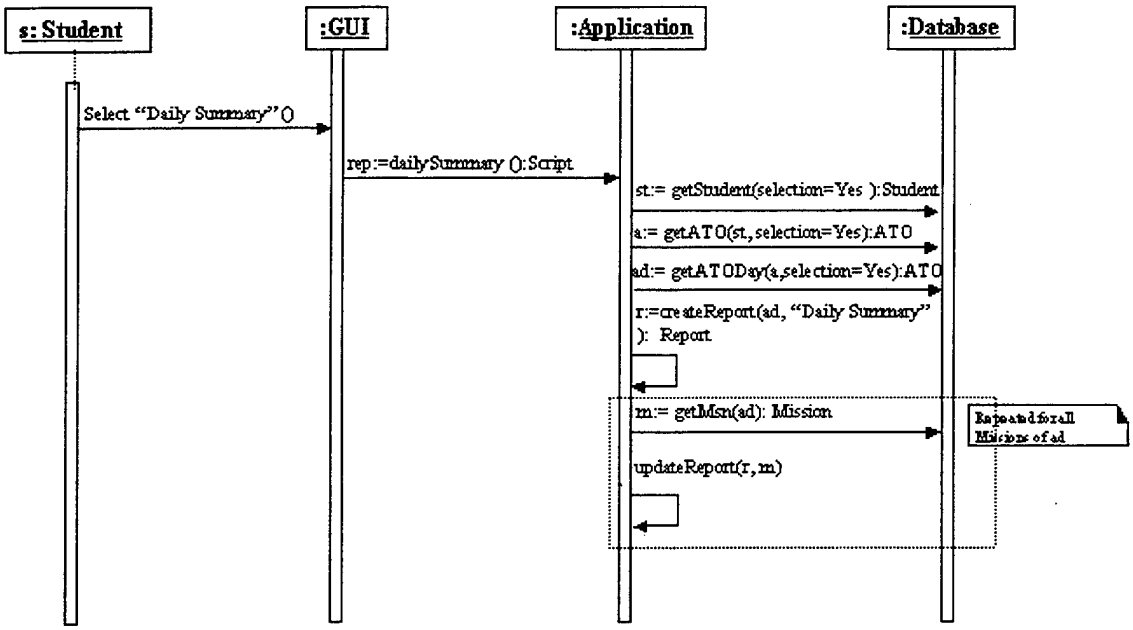


Figure 57. Sequence Diagram "Daily Summary"

INITIAL INFORMATION – LOGISTICS REQUIREMENTS

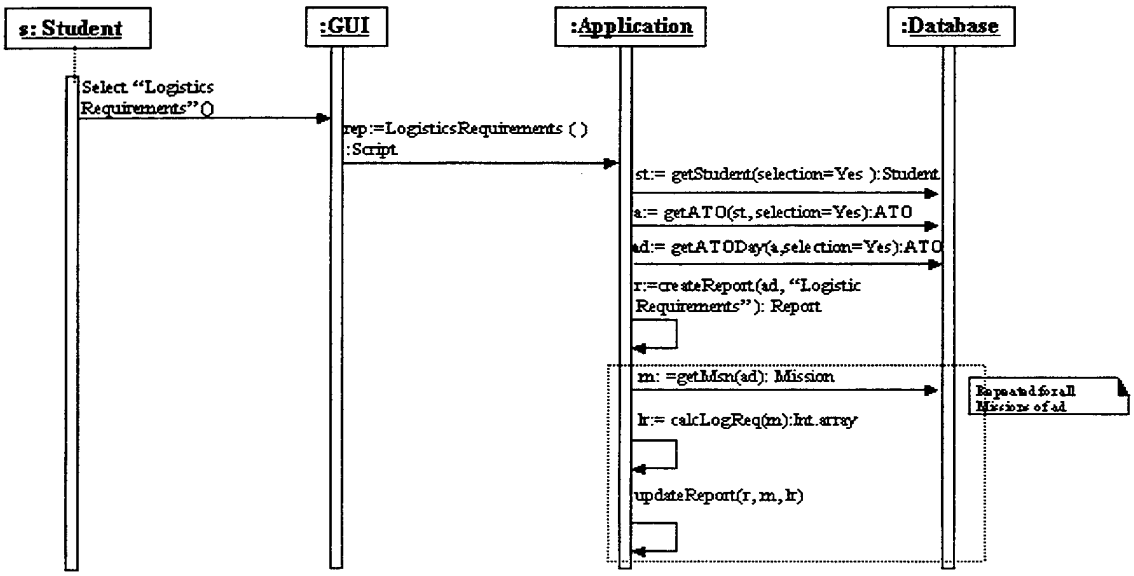


Figure 58. Sequence Diagram "Logistic Requirements"

INITIAL INFORMATION – BLUE BASING SUMMARY

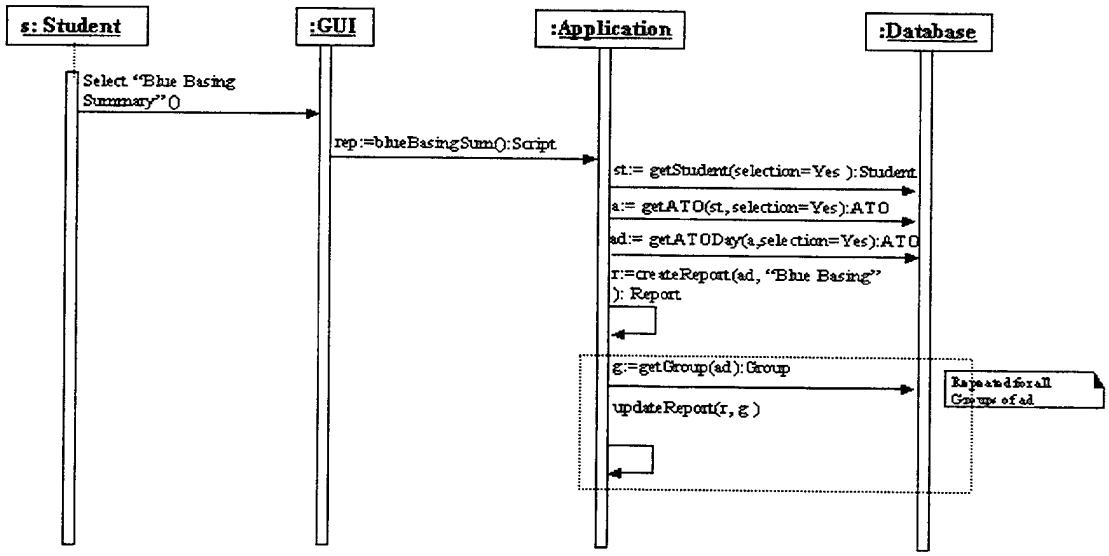


Figure 59. Sequence Diagram "Blue Basing Summary"

INITIAL INFORMATION – "RECCE FOR TARGETS"

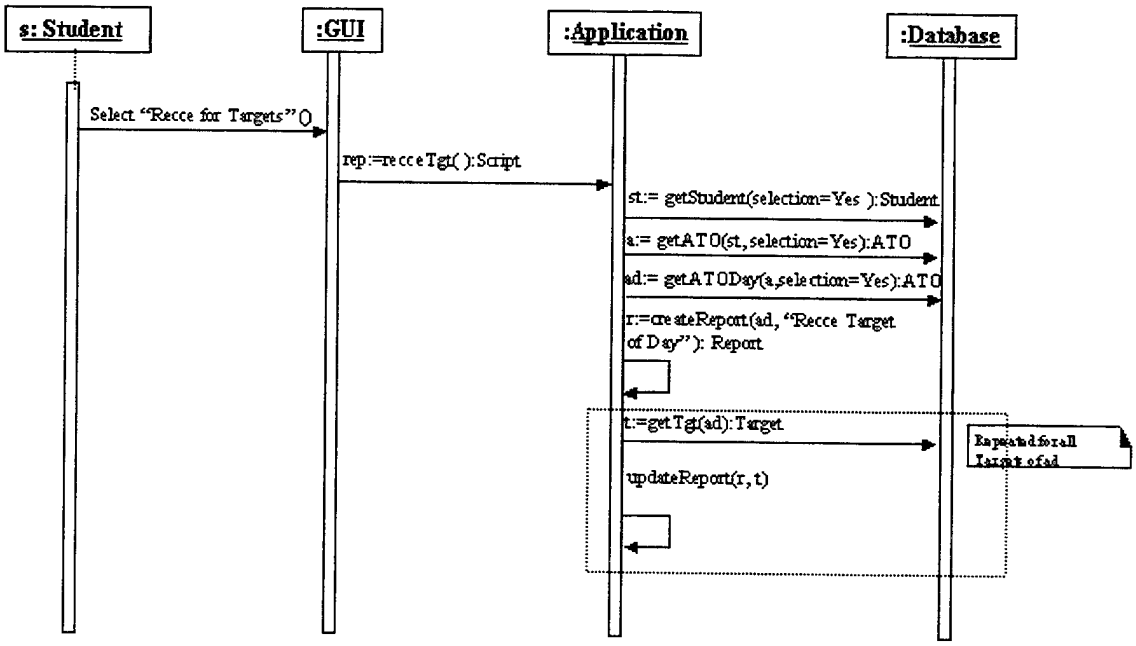


Figure 60. Sequence Diagram "Recce for Targets"

INITIAL INFORMATION – "ANALYSIS"

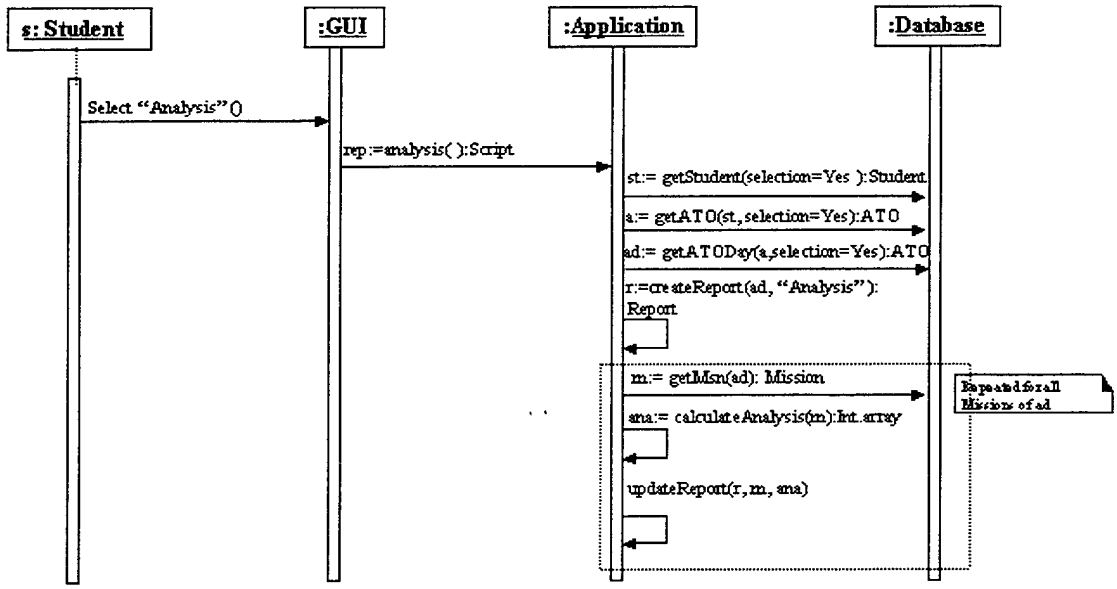


Figure 61. Sequence Diagram "Analysis"

INITIAL INFORMATION – "SORTIES AVAILABLE"

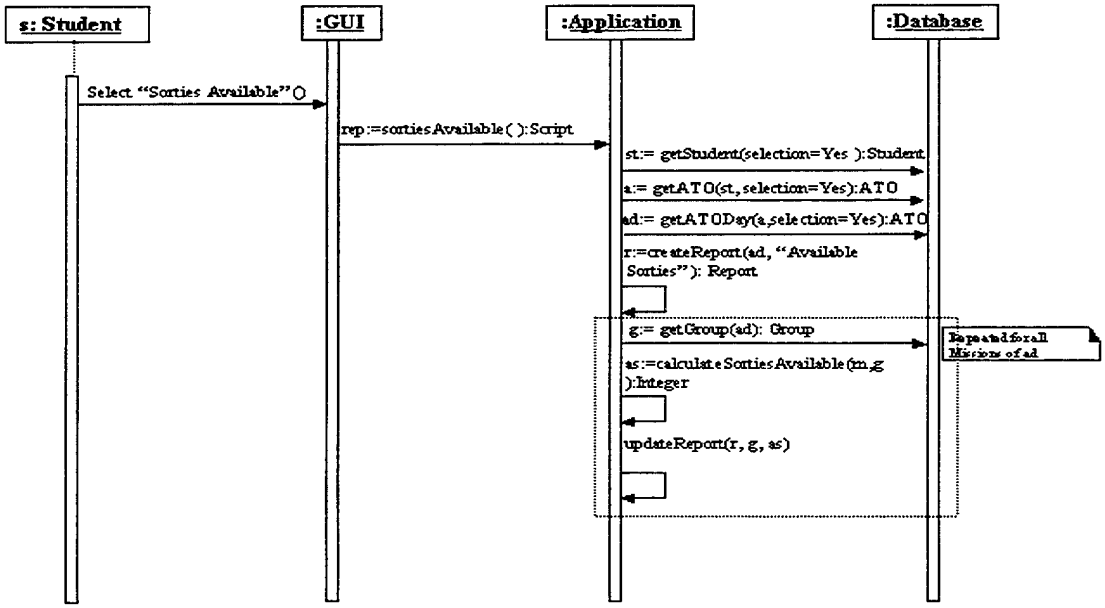


Figure 62. Sequence Diagram "Sorties Available"

ACTUAL RESULTS – "CUMULATIVE SUMMARY"

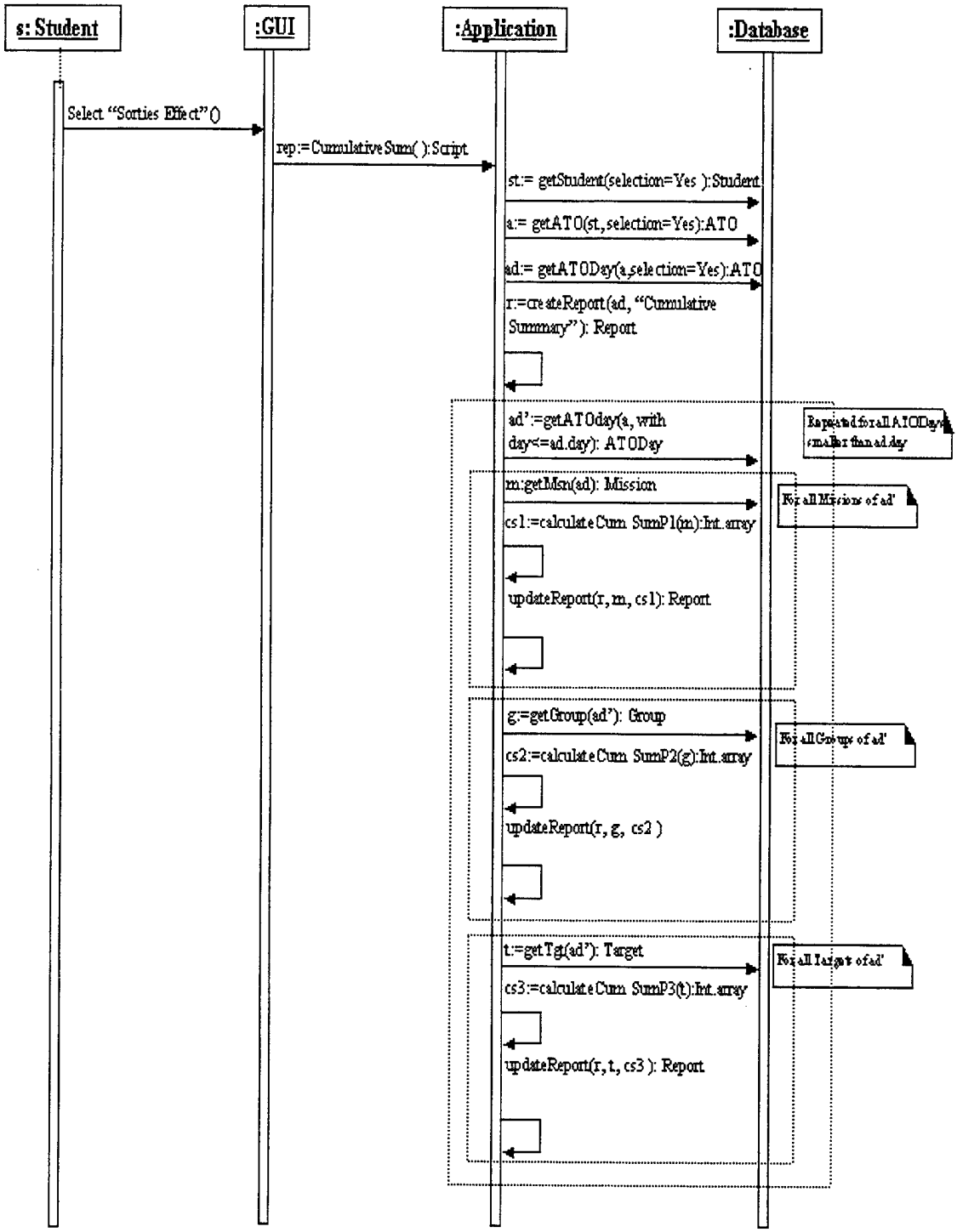


Figure 63. Sequence Diagram "Cumulative Summary"

ACTUAL RESULTS – "ENEMY OVER BLUE BASES"

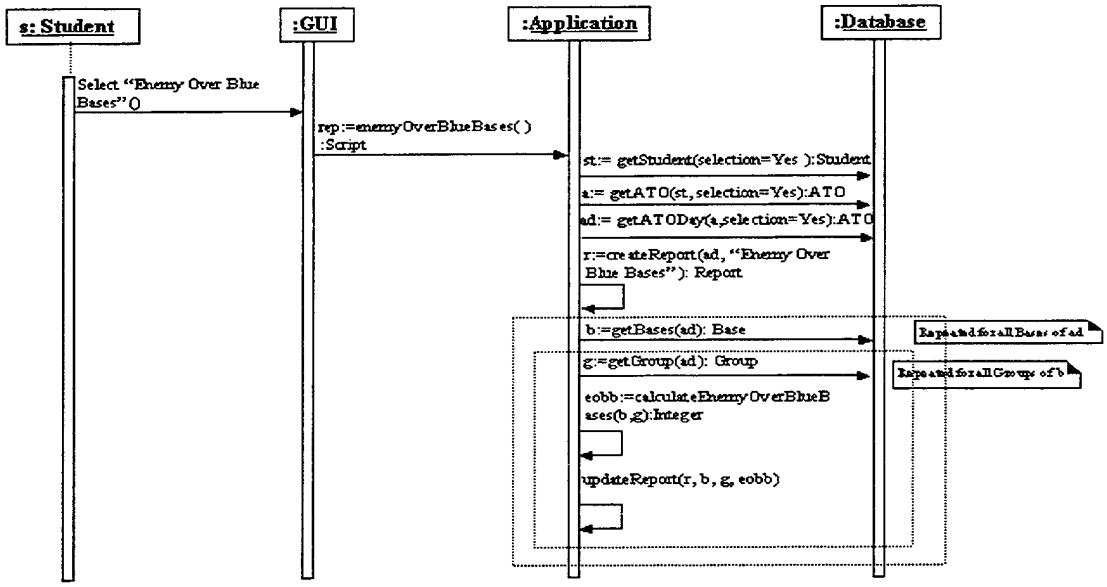


Figure 64. Sequence Diagram "Enemy over Blue Bases"

ACTUAL RESULTS – "GROUND WAR SUMMARY"

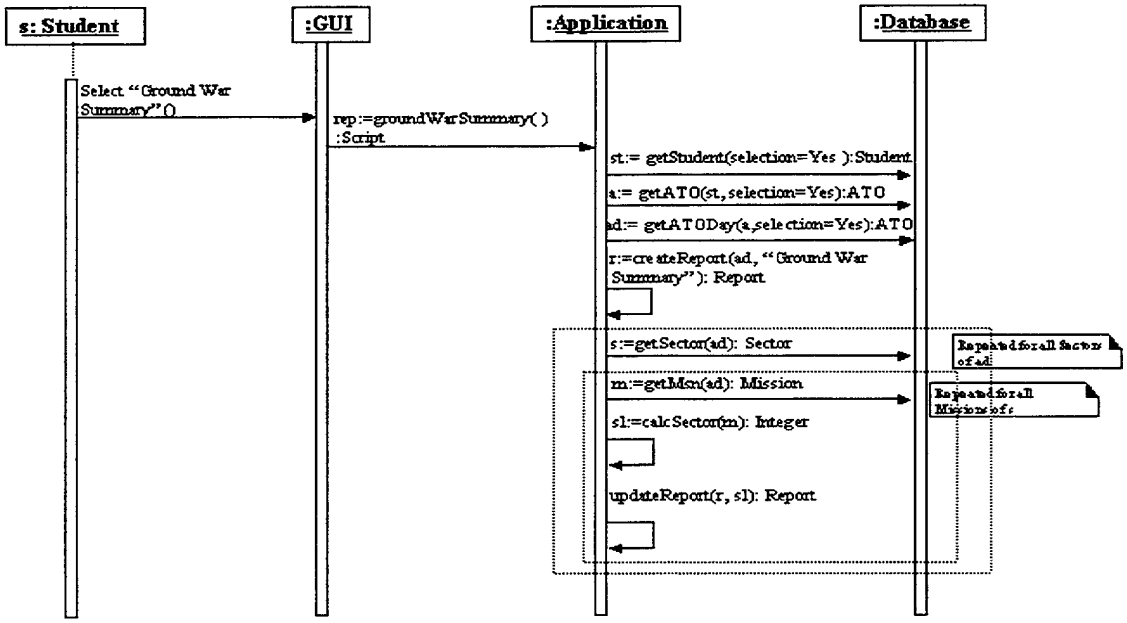


Figure 65. Sequence Diagram "Ground War Summary"

ACTUAL RESULTS – "MEASURES OF MERIT"

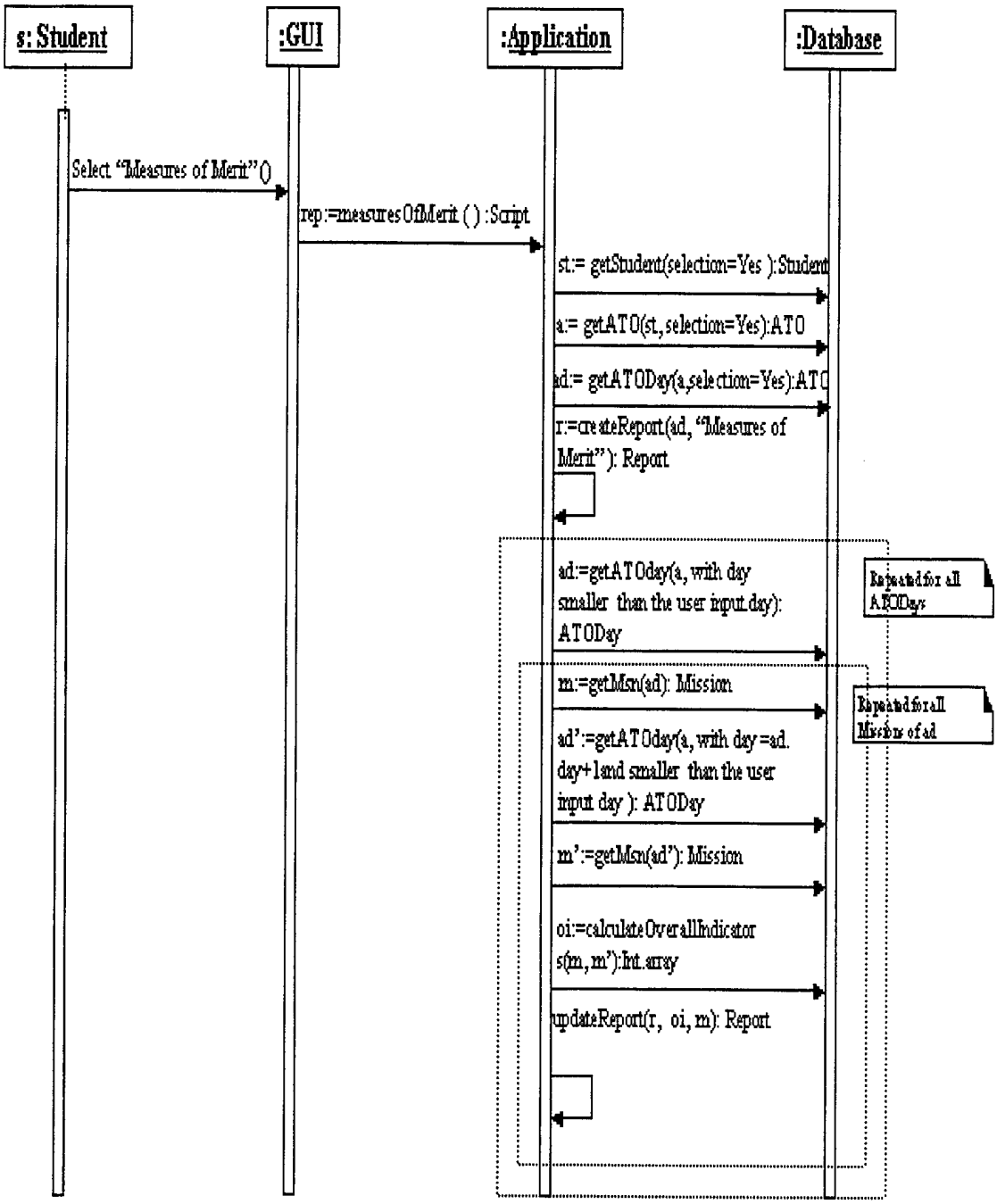


Figure 66. Sequence Diagram "Measures of Merit"

ACTUAL RESULTS – "YESTERDAY LOSSES BY AIRCRAFT TYPE"

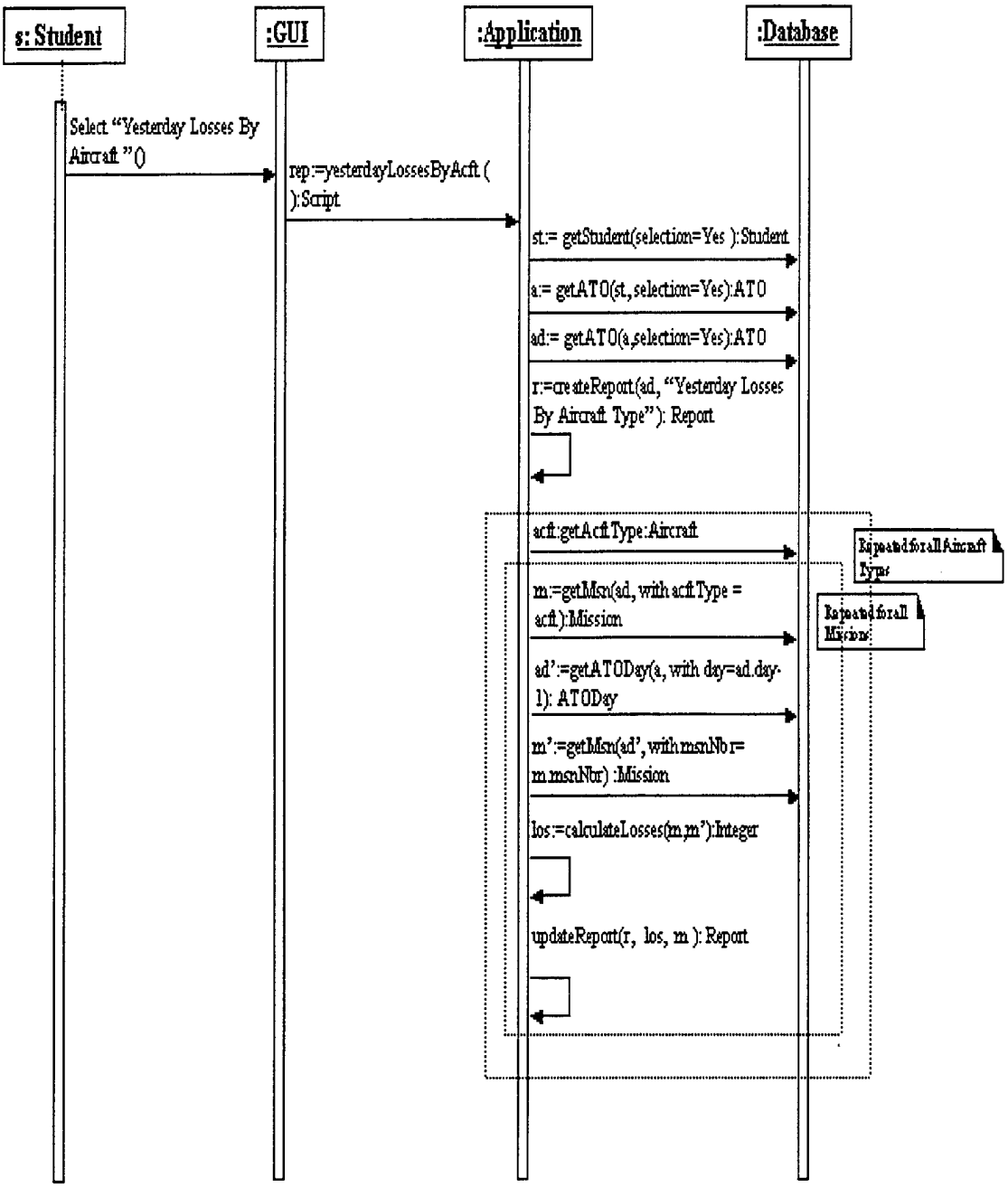


Figure 67. Sequence Diagram "Yesterday Losses By Aircraft Type"

ACTUAL RESULTS – "YESTERDAY LOSSES BY TASK TYPE"

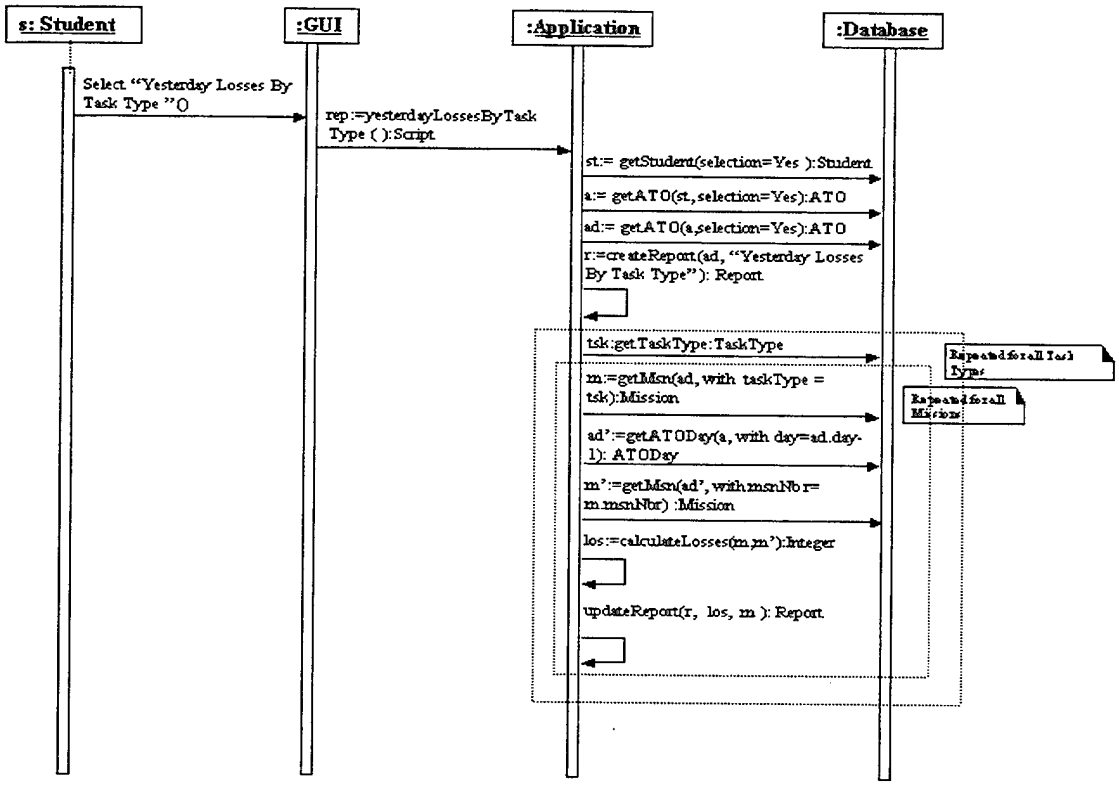


Figure 68. Sequence Diagram "Yesterday Losses By Task Type"

MAP

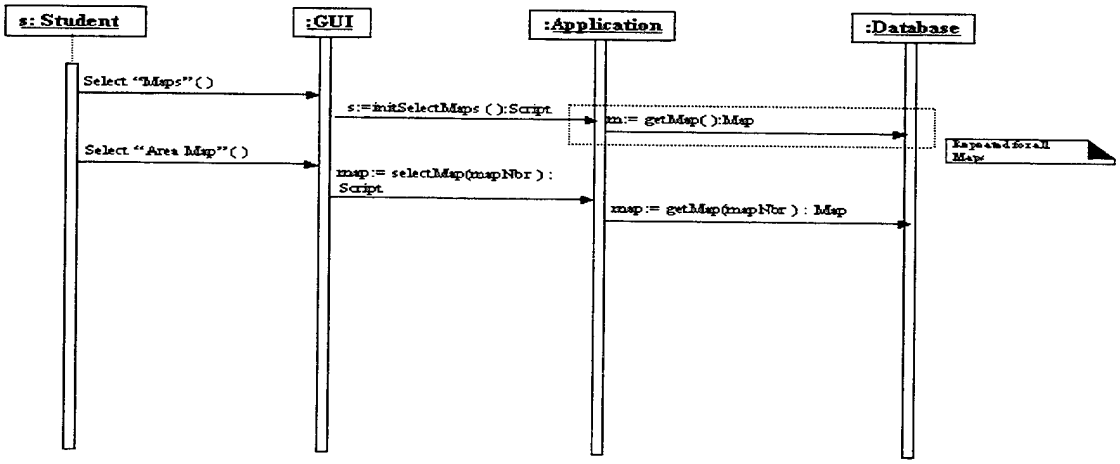


Figure 69. Sequence Diagram of "Map"

SEND ATO

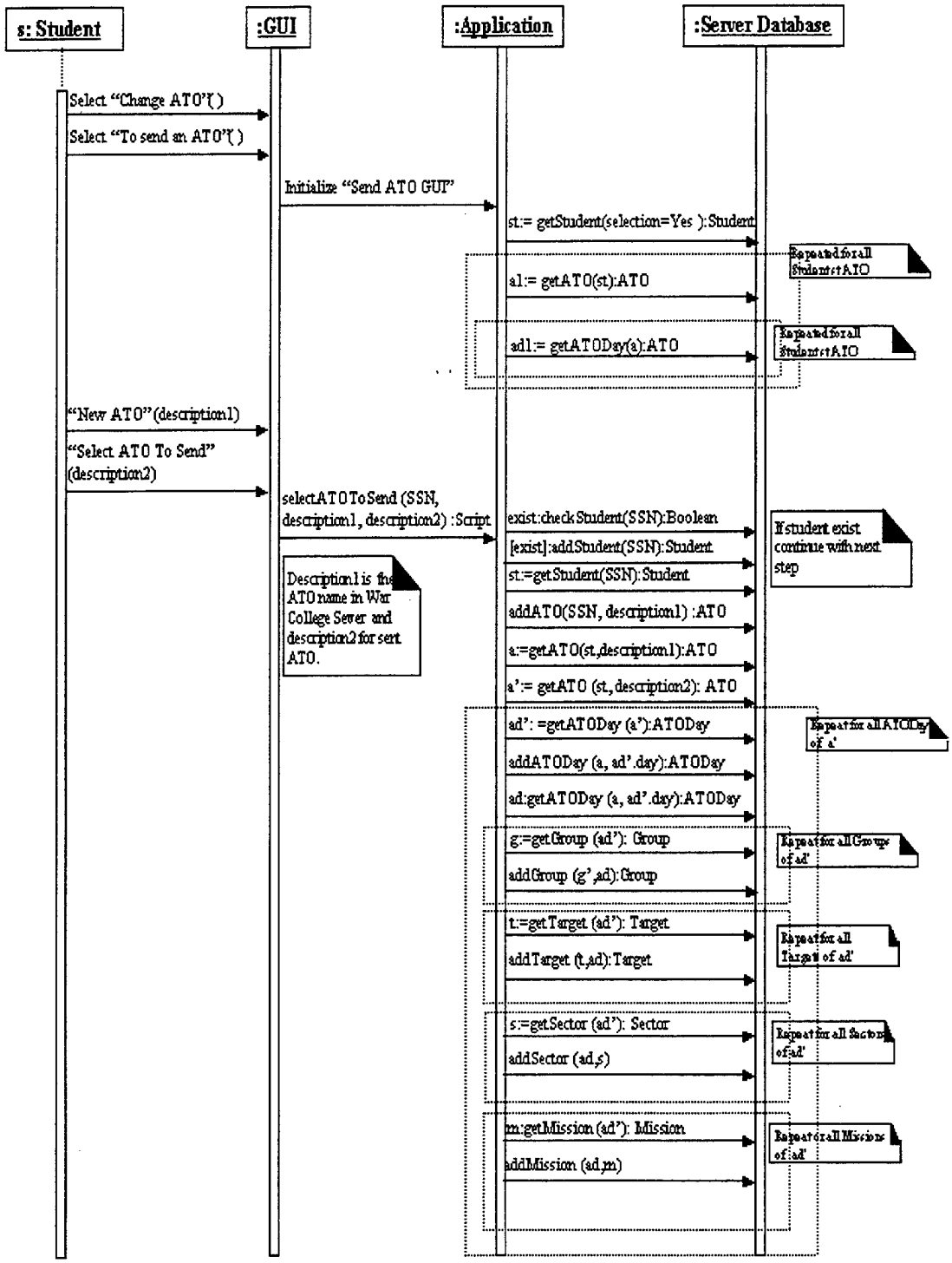


Figure 70. Sequence Diagram "Send ATO"

APPENDIX C ABBREVIATIONS ACRONYMS DEFINITIONS

Abbreviation	Word	Explanation
	Active X	Transition from OLE of Microsoft
	Adaptability	The ease with which software can be modified to meet new requirement [DODSTR 92, p.47]
	Cohesion	The manner and degree to which the tasks performed by a single component are related to one another
	Coupling	1The degree of data or control connectivity between different assets of a software system. Because coupled assets cannot be separated from each other easily and used alone, the scope of reuse is narrowed. 2. The manner and degree of interdependence between components.
DCE	Distributed Computing Environment	
DCOM	Distributed Component Object Model	
4GL	Four Generation Languages	
	Interoperability	Measure of the ability to share information between different systems.
NSDM	National Security	
OS	Operating System	
OLE	Object Linking and Embedding	

Abbreviation	Word	Explanation
	Reengineering	The process of examining and altering an existing software component in order to reformat or configure it. Reengineering is comprised of the subprocesses of reverse engineering, retargeting, restructuring, redocumentation, and forward engineering.
	Stub	COM or CORBA Mechanism that allow seamless access to remote objects
AI	Air Interdiction	Air operations conducted to destroy, neutralize, or delay the enemy's military potential before it can be brought to bear effectively against friendly forces at such distance from friendly forces that detailed integration of each air mission with the fire and movement of friendly forces is not required. (Air War College Glossary.)
ATO	Air Task Order	
AWACS	Airborne Warning and Control System	
BAI	Battlefield Air Interdiction	BAI was a subset of Air Interdiction. Missions, which are tasked against enemy forces, and/or resources that are in a position to directly influence and affect ongoing land operations, but which are not yet directly engaged in combat. These missions are included in Interdiction in the new AFM 1-1 and BAI no longer is an Air Force mission. (Air War College Glossary.)
BDA	Battlefield Damage	

Abbreviation	Word	Explanation
	Assessment	
CAS	Close Air Support	
C3	Command, Control, and Communication	
DCA	Defense Communications Agency	
DSA	Defense Suppression Area	The area EF-111 or Wild Weasel aircraft employing stand-off jamming or suppression would be responsible for negating enemy capability. They would not penetrate enemy territory. (Air War College Glossary.)
EW	Early Warning	
FEBA	Forward Edge of the Battle Area	The foremost limits of a series of areas in which ground combat units are deployed, excluding the areas in which the covering or screening forces are operating, designated to coordinate fire support, the positioning of forces, or the maneuver of units. (Air War College Glossary.)
OCA	Offensive Counterair	
Recce	Reconnaissance	
SAM	Surface to Air Missile	
SEAD	Suppression of Enemy Air Defense	
	Sortie	In air operations, an operational flight by one

Abbreviation	Word	Explanation
		aircraft. (Air War College Glossary.)
UTC	Unit Type Code	A five-character alphanumeric code that uniquely identifies each type unit of the Armed Forces. (Air War College Glossary.)

BIBLIOGRAPHY

Books

Berzins and Luqi. *Software Engineering with Abstractions.* Naval Postgraduate School, USA, 1990.

Elmasri and Navathe, *Fundamentals of Database Systems*, USA, 1994.

Reaz Hoque and Tarun Sharma, *WEB Components*, NY, USA, 1998.

Jason Pritchard, *COM and CORBA Side by Side*, USA, 1999.

Frederick P. Brooks, *The Mythical Man-Month*, University of North Carolina, USA, September 1995.

Dirk Slama, Jason Garbis, Perry Russel, *Enterprise CORBA,* USA, 1999.

Robert Orfali and Dan Harkey, *Client/Server Programming with JAVA and CORBA,* USA, 1998.

Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language User Guide,* Rational Software Corporation, USA, February 1999.

Graig Larman, *Applying UML and Patterns*, USA, 1997.

Hans-Eric Ericsson and Magnus Penker, *UML Toolkit*, USA, 1998.

Desmond F. D' Souza and Alan Cameron Wills, *Object, Components and Frameworks with UML*, USA, October 1998.

Ivar Jacobson, Martin Griss, Patrik Jonsson, *Software Reuse Architecture, Process and Organization for Business Success*, USA, 1997.

Wayne C. Lim, *Managing Software Reuse*, A Comprehensive Guide to Strategically Reengineering the Organization for Reusable Components, USA, 1998.

Danny Ayers, Hans Bergsten, Michael Bogovich, Jason Diamond, Matthew Ferris, Marc Fleury, Ari Halberstadt, Paul Houle, Piroz Mohseni, Andrew Patzer, Ron Phillips, Sing Li, Krishna Vedati, Mark Wilcox, Stefan Zeiger, *Professional JAVA Server Programming*, USA, 1999.

Michael Blaha and William Premerlani, *Object-Oriented Modeling and Design for Database Applications*, USA, 1998.

W. Keith Edwards, *Core JINI*, USA, September 1999.

Technical Papers

Sun Microsystems. "JINI Architecture Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JINI Distributed Event Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JAVASPACES Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JINI Technology Glossary", Version 1.0.1 November 1999.

Sun Microsystems. "JINI Transaction Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JINI LookupService Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JINI Distributed Leasing Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JINI Discovery and Join Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JINI Discovery Utilities Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JINI Entry Utilities Specification", Version 1.0.1 November 1999.

Sun Microsystems. "JINI Lookup Attribute Schema Specification", Version 1.0.1 November 1999.

Gopalan Suresh Raj. "A Detailed Comparison of CORBA, DCOM and Java/RMI" 21 September 1998.

Kurt Wallnau. Software Engineering Institute. "Common Object Request Broker Overview Architecture". 10 Jan1997.
Ed Morris,Emil Litvak, Software Engineering Institute "Component Object Model (COM), DCOM, and Related Capabilities" 23 Jun 1997.

Internet Sites

"CORBA." *Object Management Group.* Available [Online]: http://cgi.omg.org/corba/beginners.html.

"CORBA." *Douglas C. Schmidt.* Available [Online]: http://www.cs.wustl.edu/~schmidt/.

"Interoperability Showcase." *Distributed System Technology Center-CORBANet.* Available [Online]: http://corbanet.dstc.edu.au/html/arch.html.

" COM-DCOM and Related Cababilities." Software Engineering Institute. Available [Online]: http://www.sei.cmu.edu/str/descriptions/com.html.

"JINI Specifications." *Sun Microsystems Corporation.* Available [Online]: http://www.sun.com/jini/specs/index.html.

"Component Object Specification." *Microsoft Resources.* Available [Online]: http://www.microsoft.com/Com/resources/comdocs.asp.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Dan Boger 1
   Chairman, Computer Science Code CS
   Naval Postgraduate School
   833 Dyer Road
   Monterey, CA 93943-5118

4. Dr. Man-Tak Shing 1
   Computer Science Department, Code CS/Sh
   Naval Postgraduate School
   833 Dyer Road
   Monterey, CA 93943-5118

5. Major Leroy A. Jackson 2
   TRAC-MTRY
   Naval Postgraduate School
   166 Bouldry Road
   Monterey, CA 93943-0692

6. Major Antonios Chalakatevakis 5
   Moshopoulou 7-9
   Erithros, 11526
   Athens, Greece