1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**An Approach and a
Prototype CASE Tool for
Data Distribution Design**

by

Christian Lamarre, B.Eng.

**A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfilment of the requirements
for the degree of
Master of Engineering**

**Ottawa-Carleton Institute for Electrical Engineering,
Department of Systems and Computer Engineering,
Carleton University,
Ottawa, Ontario, Canada**

March 1, 1990

© Christian Lamarre 1990

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-59044-0

The undersigned recommend to the Faculty of Graduate
Studies and Research acceptance of the thesis

An Approach and a
Prototype CASE Tool for
Data Distribution Design.

Submitted by Christian Lamarre, B.Eng.
in partial fulfilment of the requirements for
the degree of Master of Engineering.

Thesis Co-supervisor

Thesis Co-supervisor

Chairman,
Department of Systems and
Computer Engineering

Ottawa-Carleton Institute for Electrical Engineering,
Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
March 1, 1990

Abstract

This thesis proposes a top down approach for data distribution design which covers the application of a selected data allocation model. The selection of the data allocation model is made by comparing different allocation features covered by existing allocation models. Using an object-oriented approach this thesis also describes the design and implementation of a prototype CASE tool which supports the application of an allocation heuristic that is part of the selected data allocation model. Modifications are made to that heuristic to improve its optimality i.e. bring the solution closer to the global optimum, to ease its implementation and to be able to apply it to practical problems.

The usefulness of the developed approach as well as the potential for adequate performance of the prototype CASE tool was demonstrated by application to a real size problem where hundreds of data fragments are distributed over 17 different sites.

Acknowledgements

First, I would like to express my sincere gratitude to my thesis supervisors, Dr. M. Niktash and Dr. J.S. Riordon for their help and guidance throughout the process of this research work. I would also like to thank Kevin Watson, for his help with ARTTisan.

Special appreciations go to my family and friends for their moral support.

Most of all, I would like to thank my girl-friend Judith for her steady patience and encouragement which helped me go through a difficult period of my life.

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Thesis Objectives	1
1.3	Thesis Overview	2
2	Information System Development Methodologies and Supporting CASE	4
2.1	Information Systems Software Life Cycle	4
2.1.1	Information Systems Planning	5
2.1.2	Business Analysis	5
2.1.3	System Design	6
2.1.4	Construction Design	6
2.1.5	Testing	7
2.1.6	Maintenance	7
2.2	The Three Perspectives	7
2.2.1	Data Oriented Perspective	8
2.2.2	Process Oriented Perspective	8
2.2.3	Behaviour Oriented Perspective	9
2.2.4	Cross-Reference between Perspectives	10
2.3	Information System Development Methodologies	10
2.3.1	Some Existing Methodologies	10
2.3.2	Support of Data Distribution Design with Information System Methodologies	12
2.3.3	Methodology employed as an Example	12
2.4	Overview of CASE Tools	13
2.4.1	Motivation and Objectives of CASE Tools	13
2.4.2	CASE Tools Classifications	14
2.4.3	Key Components of CASE Workbenches	15
2.4.4	Example of a Workbench that supports IEM	19
2.5	Familiarization with IEM and IEW	20
2.5.1	Planning Phase	22
2.5.2	Business Analysis Phase	27
2.5.3	System Design Phase	29
2.6	Summary	30
3	Data Distribution Design	31
3.1	Overview	31
3.2	TOP DOWN versus BOTTOM UP Approaches	31
3.3	Design of Database Fragmentation	32

3.3.1	Horizontal Fragmentation	34
3.3.2	Vertical Fragmentation	36
3.4	The Allocation of Fragments	37
3.4.1	The Different Allocation Features	37
3.5	Data Dictionary Allocation	46
3.6	Analysis of Allocation Models	48
3.6.1	Model 14 Description	52
3.6.2	Model 14 Analysis	53
3.6.3	Model 14A Description	54
3.6.4	Model 14A Analysis	55
3.6.5	Model 15 Description	55
3.6.6	Model 15 Analysis	57
3.6.7	Model 16 Description	60
3.6.8	Model 16 Analysis	67
3.6.9	Subset of Model 16	71
3.7	Selected Model	71
3.8	Summary	72
4	CASE Tool Design Description	73
4.1	Overview	73
4.2	Architecture of CASE Tool for Data Distribution Design	73
4.3	Supported Data Allocation Heuristic	75
4.3.1	Limitations of Supported Heuristic	76
4.3.2	Model 16A Example Description	78
4.3.3	Model 16A Execution Steps Description	79
4.4	Modifications Made to Supported Heuristic	90
4.4.1	First Modification	90
4.4.2	Second Modification	93
4.4.3	Third Modification	97
4.5	CDT General Requirements	98
4.6	An Overview of ARTTisan	98
4.7	CDT Implementation Design	99
4.8	Summary	101
5	A Proposed Approach for Data Distribution Design	102
5.1	Overview	102
5.2	Qualitative Approach	103

5.2.1	Processes versus Subject Databases	104
5.2.2	Processes versus Locations	104
5.2.3	Subject Database versus Process/Location	106
5.2.4	Effect of Queries and Updates Frequencies on Data Distribution	107
5.2.5	Centralization versus Decentralization Issues	109
5.3	Quantitative Approach	114
5.4	Application of the CDT	125
5.4.1	Data Traffic Without Distribution	126
5.4.2	CDT Application Results	129
5.5	Summary	133
6	Conclusion	134
6.1	Contributions of this Research	134
6.2	Future Developments	136
	Bibliography	140
Appendix A	DEM AND IEW APPLIED TO AN AEROSPACE MAINTENANCE INFORMATION SYSTEM PROJECT	147
Appendix B	CDT IMPLEMENTATION DESIGN	243
Appendix C	PROPOSED DATA DISTRIBUTION APPROACH APPLIED TO AEROSPACE MAINTENANCE INFORMATION SYSTEM PROJECTS	276

List of Figures

2-1	CASE Workbench Components	16
2-2	IEW Architecture	20
2-3	Information Engineering Methodology	22
3-1	Top Down Approach for Data Distribution Design	33
3-2	An Example for a Processing-Schedule	42
3-3	Features of Data Allocation Models	49
3-4	Transaction Modification after Addition of Data	59
3-5	Transaction Number 2 Processing-Schedule Graph	63
3-6	Query Processing-Schedule Minimizing Response Time	65
3-7	Query Processing-Schedule Minimizing Transmission Cost	66
4-1	Architecture of CASE Tool for Data Distribution Design	74
4-2	Modified Heuristic 'total-data-allocation'	77
4-3	Global Processing-Schedule Graph	81
4-4	Initial Virtual Sites Assignment	82
4-5	Set P of Virtual Sites to Unite	83
4-6	Cost Increase Incurred by Removal of V1 and V3 from P1	85
4-7	Decrease in Cost Incurred by Union of V1 and V3	86
4-8	Decrease in Cost Incurred by Assigning Union of V1 and V3 to P1	87
4-9	Final Allocation of Virtual Sites	89
4-10	Last Union before Final Allocation	91
4-11	Set P of Virtual Sites to Unite	91
4-12	Union of V4 with V6	92
4-13	Union of V6 with Vu12357	93
4-14	Initial Allocation Graph with Virtual Site V8 to be Updated Only	94
4-15	Final Allocation with Original Heuristic	95
4-16	Final Allocation with Modified Heuristic	96
5-1	Factor Table	104
5-2	Subject Databases versus Location/Process Table	106
5-3	Effect of Updates on Data Distribution	108
5-4	Query Specifications	116
5-5	From Individual to Global Query Processing Schedule	123
5-6	All Data Centralized at NDHQ Ottawa	127
5-7	Data Traffic with Data Centralized at NDHQ	128
5-8	Data Traffic with Data Centralized at NDHQ and One Copy of the Data Generated Locally Stored Locally	129
5-9	Data Allocated to Minimize Transmission Cost	131
5-10	Data Traffic When Transmission Cost is Minimized	132

A-1	BAMEO Organization Chart	148
A-2	BAMEO Functions Decomposition	152
A-3	BAMEO Org. Units versus Funct/Process	172
A-4	Format of Interviewed Data	174
A-5	Input and Output Data Classes of each Processes	176
A-6	Cross-Reference Between Data Classes and Subject Area	178
A-7	Critical Success Factor versus Objectives	180
A-8	E-R Diagram	185
A-9	E-R Diagram with only One-to-Many Relationship	188
A-10	IEW Structured Entry Chart	190
A-11	Structure Entity Chart	192
A-12	Example of Transaction Specification	193
A-13	Process versus Entity Association Matrix	194
A-14	Example of Affinity Analysis	195
A-15	Affinity Analysis	196
A-16	Aircraft Subject Database	201
A-17	Equipment Subject Database	202
A-18	Personnel Subject Database	203
A-19	View 1	204
A-20	View 2	205
A-21	View 3	206
A-22	View 4	207
A-23	DFD of Maintenance Function	208
A-24	Data Store "Maintenance Required"	209
A-25	View 1	211
A-26	View 2	212
A-27	View 3	213
A-28	View 4	214
A-29	Merging View 1 & View 2	216
A-30	Merging Result to View 3	217
A-31	Merging Result to View 4	218
A-32	Final Result	220
A-33	Corresponding Entities	223
A-34	BAMEO Data Model Description	226
A-35	Maintenance Function DFD	235
A-36	Maintenance Function DFD after Correction	236
A-37	Logical Access Map Main Entities	238
A-38	Logical Access Map	239
A-39	An Example of an Action Diagram	240
A-40	IEW Action Diagram	241

B-1	TArc and TNode Classes Inheritance	244
B-2	LNode and LArc Classes Inheritance	245
B-3	GBSpline Class Inheritance	245
B-4	GNode Class Inheritance	246
B-5	SMEModes Class Inheritance	246
B-6	Step-by-Step Through Heuristic 'total-data-allocation'	264
B-7	NonStop Execution of Heuristic 'total-data-allocation'	267
B-8	Breakdown of Method 'Calculate-Total-Union-Cost'	271
B-9	Adjustment of Transmission Edges Values as Result of an Union	273
C-1	Processes versus Subject Database Association Matrix	277
C-2	Factor Table for ADAM Process	278
C-3	ADAM Update Statistics	280
C-4	Subject Database versus Process/Locations Table	281
C-5	Frequency of Updates for Subject Database Personnel	282
C-6	Process P6.2 Transaction Specification	285
C-7	AMMIS (Q2) Transaction Specification	286
C-8	Process P6.2 Processing-Schedule	286
C-9	P6.2 Processing-Schedule Graph Summary	288
C-10	AMMIS Q2 Processing-Schedule Graph	289
C-11	AMMIS Q2 Processing-Schedule Graph Summary	291
C-12	Update Aircraft Fragment Processing-Schedule Graph	292
C-13	Update Transfer Fragment Processing-Schedule Graph	293
C-14	Processing-Schedule Graph Summary for Update of Aircraft Subject Database Fragments	295
C-15	Aircraft Subject Database Initial Processing-Schedule Graph	298
C-16	Enlargement of Figure C-15 for PH1, PH11 and PH12	299
C-17	Enlargement of Figure C-15 for PH2, PH12 and PH16	300
C-18	Enlargement of Figure C-15 for PH10, PH12 and PH16	301
C-19	Enlargement of Figure C-15 for PH9, PH10 and PH16	302
C-20	Enlargement of Figure C-15 for PH15 and PH16	303
C-21	Enlargement of Figure C-15 for PH14 and PH15	304
C-22	Enlargement of Figure C-15 for PH3, PH13 and PH17	305
C-23	Enlargement of Figure C-15 for PH3, PH4, PH13 and PH17	306
C-24	Enlargement of Figure C-15 for PH5, PH6 and PH14	307
C-25	Enlargement of Figure C-15 for PH7 and PH8	308
C-26	Aircraft Subject Database Final Allocation	309
C-27	Aircraft Subject Database Final Allocation	310
C-28	Enlargement of Figure C-27 for PH1, PH11 and PH12	311
C-29	Enlargement of Figure C-27 for PH2, PH12 and PH16	312
C-30	Enlargement of Figure C-27 for PH10, PH12 and PH16	313
C-31	Enlargement of Figure C-27 for PH9, PH10 and PH16	314

C-32	Enlargement of Figure C-27 for B-27 for PH9 and PH15	315
C-33	Enlargement of Figure C-27 for PH14 and PH15	316
C-34	Enlargement of Figure C-27 for PH3, PH4, PH7 and PH13	317
C-35	Enlargement of Figure C-27 for PH5, PH6 and PH14	318
C-36	Enlargement of Figure C-27 for PH6, PH7 and PH8	319

GLOSSARY

ADAM	Automated Data for Aerospace Maintenance : This is the name of a Canadian Air Force project which consists of building an information system which will assist aircraft managers to administer their aircraft resources by automating the current administrative tasks associated with their aircraft maintenance at Base level.
AMMIS	Aircraft Maintenance Management Information System: This is the name of a Canadian Air Force project which consists of building an information system which will assist aircraft managers to administer their aircraft resources by automating the current administrative tasks associated with their aircraft maintenance at the Canadian Forces Headquarters level.
ARTTisan	Advanced Real Time Toolset : This CASE tool has been developed by Carleton University (Canada), in order to aid the designer and programmer alike in developing real-time Unison specific software.
BAMEO	Base Aircraft Maintenance Officer: It is a position in the aircraft maintenance organization of the Canadian Air Force. The BAMEO is responsible to the Base Commander for the organization, planning and implementation of the aircraft maintenance function and for the support of the air operational requirements of the base.
BSP	Business Systems Planning: It was introduced in mid 1970 by IBM. It can be described as a structured approach to assist a business in establishing an information system plan to satisfy its near and long term information needs.
CASE	Computer Aided Software Engineering: It is a computer program directed at automating the various tasks involved in the application of structured methodologies for software development.
CDT	Carleton University Distributed Database Design Toolkit:: It is the prototype CASE tool developed in the scope of this thesis. It automates the data allocation part of the data distribution design step.

COBOL	Common Business Oriented Language : It is a procedure-oriented computer language mainly used for business data processing.
CPU	Central Processing Unit : The unit of a computing system that includes the circuits controlling the interpretation of instructions and their execution.
CSF	Critical Success Factor : CSF is a method of eliciting requirements by asking users to define the factors that are critical to success in performing their functions or making decisions.
DBMS	Database Management Systems : The DBMS available today are pieces of software which provide the capability of defining a database in terms of record types, data items and relationships. DBMS offer integrity, privacy and data independence.
DFD	Data Flow Diagram : A graphic representation of a system, showing data sources, data sinks, storage, processes performed on data as nodes, and logical flow of data as links between the nodes.
E-R	Entity Relationship Model : This model provides the analyst with three main semantic concept : entities, relationships and attributes to describe an organization. Entities are distinct objects within a user organization. Relationships are meaningful interactions between these objects and attributes describe these entities and relationships.
FS	Fragment Set : This is a set of fragments. A fragment is a fraction of an entity. There are two possible type of fragments, either horizontal or vertical fragments.
IEM	Information Engineering Methodology : It is an information system development methodology which cover the first four life cycle phases. It stresses the development of integrated systems, combines data and process oriented features, at each phase.
IEW	Information Engineering Workbench : A CASE tool which supports IEM.
IRDS	Information Resource Dictionary System : A software system which

conforms to the US Federal Information Processing Standard for data dictionary systems.

JSD	Jackson System Development : An information system development methodology which is process oriented and models the business application as a network of communicating processes, add new processes during system design to provide the desired outputs and transform the whole during construction design into a working system.
NP-Complete	Non-deterministic Polynomial-Time Complete : This class of problems is likely to include only intractable problems. A problem is intractable if all algorithms to solve that problem are of at least exponential time complexity.
NDHQ	National Defence Headquarters : This is the Canadian Forces Headquarters located in Ottawa, Canada.
JSP	Jackson System Programming : It is a language used in the JSD methodology which is used to describe process behaviour.
OS	Operation Set : A set of relational algebra and relational operations such as projection, semi-join etc...
PhS	Physical Site : It represents a real site in the computer network.
PSL/PSA	Problem Statement Language / Problem Statement Analyser : An information system development methodology which is process oriented and supports the system design phase only.
SDL ¹	System for Distributed Database : It was the first prototype (1976-78) of a distributed database management system.
VS	Virtual Site : It represents a set of fragments and associated relational operations which will eventually be allocated to a PhS.

Chapter 1

Introduction.

1.1 Motivation.

Due to the demand for information system availability, performance, reliability and autonomy on one side, and enabled by advances and the proliferation of database technology as well as the extensive development of computer networks, distributed databases are becoming widespread in information systems. A distributed database is, in brief, an integrated database which is built on a computer network rather than on a single computer. The data which constitutes the database are stored at the different sites of the computer network, and the application programs running at different sites access and share data.

The designers of distributed information systems are facing an important problem: how to distribute the data on different computers/sites in order to obtain the intended availability, performance, flexibility and reliability at a minimum cost. Data distribution design is thus becoming an integrated part of information system design which requires its own theory, design methodologies and support tools.

1.2 Thesis Objectives.

This thesis proposes an approach for the design of distributed information systems and develops a prototype CASE tool to support the proposed approach. This thesis addresses the following areas:

- Application of an existing information system design methodology with its supporting CASE tool on a specific problem (ADAM);

- Analysis of different models for data distribution and especially for the allocation of data. Selection of a data allocation model;
- Design and implementation of a prototype CASE tool in order to support the application of the selected data allocation model;
- Proposal of an approach that can possibly be added to information system design methodologies in order to enhance them with data distribution capabilities. The proposed approach shall as a minimum cover the application of the selected data allocation model; and
- Demonstration of the usefulness of the proposed approach and the performance of the prototype CASE tool with a case study e.g. ADAM.

1.3 Thesis Overview.

With chapter two this thesis begins by the description of the different software life cycle phases and the different perspectives to be covered by information system development methodologies. It also gives the objectives and describes the key components of CASE tools. In order to show how a methodology can cover the different life cycle phases and perspectives, a selected methodology is applied to an Aerospace Maintenance Information System project. A selected CASE tool is also applied to the same project to acquire knowledge about the project as well as the application of a CASE tool to the development of a software system for this project.

Chapter three describes the different steps of data fragmentation, data allocation and data dictionary allocation to cover in order to follow a top-down approach for data distribution design. It then analyses different data allocation models by comparing their different allocation features to finally select a data allocation model for further implementation in a prototype CASE tool.

Chapter four starts by presenting the functions to be provided by a CASE tool for data distribution design. It then gives a detailed description of the heuristic (a subset of the selected model) supported by the prototype CASE tool. Finally using an object-oriented approach, the design and implementation of the prototype CASE tool is given.

Chapter five describes the proposed approach for data distribution design. In order to show how the approach can be applied to a practical problem, it is applied to the same project described in chapter two i.e. the Aerospace Maintenance Information System project. The prototype CASE tool is also applied to the same project to show how it performs when applied to real size problems and how it can support the proposed approach. Finally chapter six conclude by describing the contributions of this research and the future developments to be done to the prototype CASE tool as well as extensions to the proposed approach for data distribution design.

Chapter 2

Information System Development Methodologies and Supporting CASE Tools.

2.1 Information Systems Software Life Cycle.

Software engineering is composed of steps that encompass the methods, tools and procedures required to build software. These steps are often referred to as software engineering paradigms. The following three paradigms have been widely recognized and discussed[1]:

1. Classic life cycle: The classical software life cycle paradigm sometimes called the 'waterfall model' demands a systematic, sequential approach to software development that begins at the system level for requirement analysis and progresses through analysis, design, coding, testing and maintenance.
2. Prototyping: It is a process that enables the developer to create a model of the software to be build. Prototyping begins with requirements gathering. A quick design then occurs. The quick design leads to the construction of a prototype. The prototype is evaluated by the customer/user and is used to refine requirements for the software to be developed. These differents steps are repeated as many times as required.
3. Fourth generation techniques: This paradigm focuses on the ability to specify software to a machine at a level that is close to natural language. The tool then automatically generates source code based on the developer's specification. However, a data structure with relevant information must exist and be readily accessible by the fourth generation language.

Based on the classic life cycle paradigm, the following phases correspond to the information systems software life cycle:

1. information systems planning (requirement analysis);
2. business analysis (analysis);
3. system design (design);
4. construction design (coding);
5. Testing; and
6. Maintenance.

2.1.1 Information Systems Planning.

This phase identifies the business objectives of the enterprise and determines the broad nature of its information requirements. This stage may include a feasibility study to determine the possible alternatives to proceed further [2].

This phase may focus on the existing information systems, their scope and whether they are due for phase out or are adaptable for future use. It can also focus on the need for new information systems and how they should be bounded.

2.1.2 Business Analysis.

The area of an enterprise covered by a business analysis stage is more constrained than for information system planning. Business analysis should cover a subarea of the area covered in the preceding information systems planning stage [2].

Business analysis includes an analysis of various aspects of the enterprise, depending on the methodology being used. It would usually include the business activities performed in the area of interest. It is important to note here that a business activity is not synonymous with a computerized process.

Another technique used in business analysis is to study the properties of the existing system irrespective of whether it is computerized or manual. Another possible technique used in business analysis is data analysis . The analysis of information flow in an organization and the associated analysis of material flow are other techniques to be considered.

2.1.3 System Design.

A system design phase typically includes the preparation of components of different kinds. If the construction tool such as a DBMS is known in advance, then the data oriented components can be prepared in such a way as to take the structuring capability into account. With the increasing emphasis on relational theory and the growing acceptance of the associated relational DBMS products, it is advisable to refer to data oriented components resulting from the design stage in the relational terms, namely in terms of tables, columns, rows and constraints [2].

As for the data design oriented components, the process oriented components can similarly be prepared in such a way that they are dependent or independent of the specific tool used to construct the information system. A process in this context is an item on a user menu which the user can invoke from a terminal.

The construction tool independent components of data design and process design can be referred to as a data model and a process model respectively. The two should be cross-referenced to identify the data to be used by each process and also each process which uses a given part of the data.

2.1.4 Construction Design.

This phase involves designating how the system, as designed in the previous phase is

to be constructed, therefore it involves a knowledge of the tools to be used to construct the system. If during the previous phase the construction tool was already taken into consideration then this phase can be considered as part of the previous phase [2].

Trade-offs will have to be made during this phase. The set of processes identified as part of system design will not necessarily equate on a one-to-one basis with the programming involved and it may be possible to construct the system cost-effectively, such that several processes as perceived in user terms can be handled by the same program.

2.1.5 Testing.

Once implemented, system testing begins. The testing process focuses on the logical internals of the software assuring that software statements have been tested. It also focuses on the functional externals, that is, conducting tests to assure that defined input will produce actual results that agree with required results.

2.1.6 Maintenance.

Software will undoubtedly undergo changes after it is delivered to the customer. Changes will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment or because the customer requires functional or performance enhancements. Configuration control is done to ensure control over these different software versions.

2.2 The Three Perspectives

The entire process of planning, analysis, system design and construction design is treated from the standpoint of the development and refinement of a series of models derived from information system development methodologies [3]. Some of these methodologies will emphasize one or two perspectives more than the others. Therefore, the perspective

concept has been introduced to create a link between these different information system development methodologies.

These perspectives are as follows:

1. data oriented;
2. process oriented; and
3. behaviour oriented.

2.2.1 Data Oriented Perspective.

The data oriented perspective places emphasis on a complete and thorough analysis of the data and its relationship. The data oriented perspective is seen to concentrate on the different following aspects:

1. business data;
2. logical dB design, construction and performance independent;
3. logical dB design, considering the construction tool to be used; and
4. stored representation of data, including indexes and access path.

The first aspect corresponds to the information system life cycle phase of business analysis, the second to system design and the third and the fourth to construction design. The main components involved within this perspective are entities and relationships.

2.2.2 Process Oriented Perspective.

It is the earliest perspective to be recognized and covers the different phases of the information system life cycle.

In the business analysis phase the aspect of business activity has to be considered. A business activity is a task to be accomplished in an enterprise and its recognition is quite

independent of the presence or absence of a computerized information system. Each activity can be broken down into lower-level activities until a basic activity is arrived at.

In the construction design phase, the aspect of computerizable processes and compilable units of programming have to be considered. A computerizable process is an executable piece of software built by the designer, using whatever tools are appropriate. A typical computerizable process may consist of one or more compilable units of programming. A compilable unit of programming is several lines of source code, as part of the construction stage.

The components involved within this perspective are business activities, the kinds of information and material used in the business activities called information/material sets, the flow of information /material sets and the organization units responsible for the business activities.

2.2.3 Behaviour Oriented Perspective.

This perspective focuses on the dynamic nature of the data and the need to analyse and understand events in the real world which impact data recorded in the information system. There is definitely a difference between business activities and business events. An event is an instantaneous, atomic state transition in the computation history of a system. Atomic means an event happens completely or not at all [4]. 'Sales order processing' is an example of a business activity and 'receipt of a sale order' is an example of a business event. Business events analysis should be done at the same time as business data and activity analysis as part of the business analysis phase.

Events happening in the real world can trigger a requirement for appropriate action in the computerized system. An application area can be "event intensive" which means that

there are several business events that can occur and there are constraints on the sequence associated with business events. Business events have their own event precedence which means that a business event of one kind cannot happen until some other business event has already happened. There is a preference to identify the precedence of events in a running system during the construction design phase at the same time as the design of computerizable processes.

2.2.4 Cross-References between Perspectives.

The main advantages of using different methodologies which emphasize on different perspective or using only one methodology which cover more than one perspective is to be able to cross-refer the components of these different perspectives.

To ensure consistency, it is possible to cross-refer the business activity with the entity types, the business events with the entity types and the business activities with the business events. For example, a verification could be done to ensure that each business event can be related to at least one entity type and at least one business activity.

2.3 Information System Development Methodologies.

2.3.1 Some Existing Methodologies.

Numerous methodologies have been developed to cover some or all of the first four phases of the information system software life cycle [2] [5]. Different methodologies will emphasize with a different degree the three perspectives during the first four life cycle phases. Among the most popular ones, we have the following [2], [5], [6]:

1. Problem Statement Language / Analyser (PSL/PSA, 1969): This data oriented methodology supports the system design phase only. It uses an entity-relationship

(E-R) model to describes various aspects of the information system, such as I/O flow, system structure, data structure, and system dynamics [2],[5],[7].

2. Jackson System Development (JSD, 1975): This process oriented methodology models the business application as a network of communicating processes, adds new processes during system design to provide the desired outputs and transforms the whole during construction design into a working system. This methodology includes the earlier Jackson System Programming (JSP) language which is used to describe process behaviour [1], [2], [8], [9], [10].
3. Information Engineering Methodology (IEM,1982): A methodology which covers the four life cycle phases. It stresses the development of integrated systems, combines data and process oriented features, at each phases. The approach package and integrates a large number of techniques inspired from different methodologies such as IBM Business System Planning (BSP) [2], [5], [11], [12], [13], [14], [15], [16], [17].
4. Yourdon (1979, 1984): Prior to 1984 system design was addressed with data flow diagram and construction design through functional decomposition. Now only the planning phase is not covered. The business analysis defines the environment in terms of events and behaviour through data flows in response to events, but incorporates entity-relationship modelling. System design focuses on data flow and state transitions for process [2], [18], [19], [20], [21].

2.3.2 Support of Data Distribution Design with Information System Methodologies.

Since most of the present information systems development methodologies only support partially or do not support at all the design of distributed information systems, there exists a need to enhance the present existing methodologies with the capability of data

distribution design to address the data oriented perspective. It is felt that some of the data distribution design issues could be covered during the planning and analysis phase of the information system life cycle; however even if missed during these phases, they could be done during the system design phase. Actually the whole data distribution design could be done during the system design phase.

A popular methodology has been used as an example in order to illustrate how methodologies could be enhanced with data distribution design capabilities. Chapter 5 describes an approach to use in order to do the data distribution design during the system design phase. Besides the example methodology, as explained in section 5.1 under some constraints the proposed approach is general enough to enhance other information system development methodologies with distribution capabilities if these methodologies support objects of various types as well as relations (e.g. Object Oriented Databases).

2.3.3 Methodology employed as an Example.

In this thesis, the design phase of the information engineering methodology (IEM) can be selected to be enhanced to support data distribution design. IEM is a popular methodology and is also valuable for the following reasons:

1. it covers the four phases of the software life cycle;
2. it considers the data and process perspectives and cross-refer their different components;
3. it is a flexible methodology which can be subject to modifications;
4. it produces a relational data model of the organization;
5. it is well documented and provides a set of guidelines, a framework to study an organization in a systematic manner; and
6. it has the required formalism and is presently supported by different CASE tools.

2.4 Overview of CASE Tools.

2.4.1 Motivation and Objectives of CASE Tools.

Despite the merits of the different existing structured methodologies, their adoption and acceptance has been limited. Ed Yourdon estimates that while 90 percent of software professionals are familiar with structural methodologies and 50 percent of their organizations have experience with them, only 10 percent are actively using them[22]. This small percentage of usage is largely attributed to their labour intensive nature.

CASE technology is directed at automating the various tasks involved in the application of structured methodologies. It is oriented towards making the application of structured techniques practical. It automates the labour-intensive tasks so that the developer can focus on the creative aspects of software development. Software now accounts for about 80 percent of the development expense for new systems, with hardware development relegated to about 20 percent of the system development budget. Software maintenance cost are even higher[23].

With the tremendous effort expended on maintaining software, software backlog is a major issue, within data processing and management information systems organizations. The productivity of the software developer has not increased at a rate sufficient to meet the increasing software demand considering the limited supply of software developers[24].

CASE tools help the software development process by automating and integrating the following[25]:

1. software development phases and maintenance;
2. generation of software documentation;
3. generation of code; and

4. error checking to get errors out early [26].

CASE tools seek to formalize software development and standardize software documentation. In addition they promote reusability in order to benefit from the same cost reductions and productivity improvements associated with the modularization of hardware components. The objective is not only to reuse source code modules, but also to reuse prototype models, data models and design specifications.

2.4.2 CASE Tools Classifications.

A considerable number of CASE products or tools are available on the market which differ in the extent to which they automate and integrate the various software engineering tasks. Based on these criteria, CASE tools can generally be classified as follows [25],[27]:

1. Toolkits: Toolkits are the simplest type of CASE tools. They are sets of integrated tools that automate one type of system life cycle function such as program design or one type of job class such as system analysis. Toolkits which address the design, analysis and planning phases are referred to as front-end or Upper CASE tools. Those which automate the subsequent construction, testing and maintenance phases are referred to as back-end or Lower CASE tools[28].
2. Workbenches: These provide integrated tools for automating the complete software life cycle. Some of them can support one or more methodology.
3. Methodology Companions: These provide computerized assistance for a particular software development methodology and provide automated help to guide the developer through the development steps. Some methodology companions support only a subset of all the software life cycle phases and they are classified as a methodology companion toolkit. Some others support all the different phases

of the software life cycle and they are classified as a methodology companion workbench.

From the above classification, a workbench supporting different methodologies is the more complete and therefore the more desirable type of CASE tool. Assuming that all phase of the software life cycle were well covered, the Workbenches type of CASE tool would be able to meet the CASE tools objectives previously described.

2.4.3 Key Components of CASE Workbenches.

As far as support provided to the classic software life cycle is concerned figure 2-1 illustrates the key components of a CASE Workbench environment [6]. A limited number of tools currently support fully/partially some/all of these components. Commercially available tools differ in the scope and range of the capabilities provided by the various components. These various components support to a certain degree the different phases of the software life cycle.

Figure 2-1: CASE Workbench Components

The following is a description of these different components and which life cycle phase they support:

1. **Front-End Diagramming Tools:** This component allows the user to create and manipulate diagrams interactively through a powerful graphics interface. Supported diagramming techniques are associated with the software development methodologies discussed at paragraph 2.3. The Data Flow Diagram (DFD), the Hierarchical Tree Decomposition diagram and the Entity-Relationship (E-R) diagram are among the most popular diagrams [6],[25],[26],[29]. This component

mainly supports the planning, analysis, design and construction design of the software life cycle.

2. **Design Checker & Analyzer:** There are four basic types of analysis/checking provided by current CASE tools [25],[26],[29]:
 - a. Syntax and type checking verify that a diagram conforms to a set of rules usually dictated by the associated methodology,
 - b. Consistency and completeness checking ensures that a diagram contains all the required information and is consistent with all other system information in the central repository,
 - c. Level balancing is concerned with errors in the same family of diagrams. For example, in a DFD it ensures that data flow shown in and out of a process at one level are represented in the diagram at the next lower level, and
 - d. Functional decomposition which is concerned with checking for the quality of hierarchical tree decomposition diagrams. Some methodologies provide guidelines for decomposing functions into more detailed subfunctions. For example, no function should be decomposed into itself or decomposed into more than seven subfunctions [29].

This component mainly supports the planning, analysis and design phase of the software life cycle.

3. **Code Generator:** A number of tools provide partial code generation capabilities by automatically generating the data declarations, by generating subroutines or procedure frameworks and call statements for structure chart models. The user must then manually code the rest.

So far, emphasis has been placed on generating COBOL code automatically. This

may be attributed to the fact that a significant portion of COBOL program consists of data declarations which are easier to generate than procedural code. The generation of the code for scientific and engineering systems is considered to be a more difficult task due to the complexity of the target environments [30] and the need for tight performance requirements [31]. This component supports the construction design phase of the software life cycle.

4. Configuration Control System: This component provides capabilities to correct, change and enhance existing systems. Some CASE tools support a testing environment, a maintenance environment that automatically updates original design documents to reflect changes made in automatically generated object code and a reverse engineering environment that translate a system back into its specification for analysis [30]. This component supports the testing and maintenance phase of the software life cycle.
5. Information Repository: The central information repository is the real heart of the Workbench. It stores and analyzes all the information needed to create, modify and evolve a software system. It contains a description of the object and its properties, its relationships to other objects. It can also be called a data dictionary which contains a wealth of information about the data (Also called Metadata). It is the basis for integration, software reusability, code generation, system specification and integrity [6],[25],[26],[27],[28],[29],[32]. It supports all the different phases of the software life cycle.

There is presently a standard Information Resource Dictionary System (IRDS) being established in the U.S. in order to have effective management of information resources [33]. There is an emerging ISO Standard IRDS which apply directly to the

Information Repository of any CASE tool. In order to increase user efficiency and improve metadata transportability from CASE tool to CASE tool, large organizations with multiple CASE products from different vendors should ensure that these CASE products conform with IRDS.

2.4.4 Example of a Workbench that supports IEM.

Since the proposed approach for data distribution design will be added as extra steps to labour intensive information system methodologies, there is also a need to develop or modify a CASE tool in order to automate these extra steps as much as possible. In order to do so, a first step is to get familiar with an existing CASE tool. For the following reasons Information Engineering Workbench (IEW) is chosen for the example given in section 2.5:

1. It is accessible;
2. IEW supported diagramming techniques and different types of analysis/checks are directly associated with the Information Engineering Methodology;
3. Even if it calls itself a Workbench, it is not a complete Workbench as defined in the previous sub-section, since it does not provide components for code generation and for configuration control. However, it has a central information repository and with its present modular architecture (figure 2-2), if enhanced with the missing component it has the potential to become a Workbench (A Code Generator component is predicted for the next year). It is therefore a good tool to be familiar with; and
4. Because of IEW modular architecture, only some module would have to be modified to support data distribution design.

Figure 2-2 : IEW Architecture

2.5 Familiarization with IEM and IEW.

Before proposals for enhancements to IEM to incorporate data distribution design capabilities as well as proposals for a CASE tool to support these data distribution design capabilities be made, the following exercises were conducted for familiarization purposes. The exercise consists of applying IEM and IEW to an Aerospace Maintenance Information System project in order to have a good understanding of each of them.

With James Martin books as main references [34], [35], [36], [37], [38], & [39] each steps of the Information Engineering Methodology (Steps 1 to 21 of figure 2-3) will be described to cover the first three phases of the software development life cycle. Figure 2-3 is adapted from [34]. In appendix A, each step has been applied to an Aerospace Maintenance Information System project to show concrete results and eventually come up with a relational data model for centralized databases. This relational data model will be used in chapter 5 for data distribution design. At every possible step IEW is applied to show how it can support the application of the Information Engineering Methodology.

Figure 2-3 : Information Engineering Methodology

2.5.1 Planning Phase:

It is unthinkable to build a battleship without an overall plan for the entire ship. Once the overall plan exists, separate teams can go to work on the components [34]. The planning phase of this methodology focus on building the overall "TOP DOWN" plan of the entire organization. The analysis phase focus on working in detail on these separate

components, it is the “BOTTOM UP” approach. The following paragraphs will describe the different steps of the planning phase illustrated at figure 2–3. Some of these steps override each other or are done in parallel in order to add flexibility to the methodology:

1. Step (1)-> Enterprise Model: This step consists of decomposing in a hierarchical manner the functional area of an organization into functions and then decomposing the functions into processes. The functional area corresponds to the major area of activity of an organization such as engineering, maintenance etc... The processes correspond to the basic activities under a function. An organization chart showing hierarchical decomposition of organization positions also has to be produced.
2. Step (2)-> Business System Plan Study: This step consist of the following different parts:
 - a. The first two parts are steps 1&3 of this methodology;
 - b. The third part consist of cross-referencing between each other the different elements of functions, process, activities and positions derived at step 1 and also cross-referencing these different elements with the existing Data Processing System elements. This cross-referencing is done to derived information for the next parts of the study;
 - c. The fourth part will consist of having interviews with high level executives of the organization to identify the information system needs and their corresponding processes. This information will be required during the next step for the design of implementable systems;
 - d. This step is to reduce and organize the interviewed data; and
 - e. This part correspond to step 7.

3. **Step (3)-> Subject Database Study:** Subject Database represent classes of data/organizational subjects within an organization. They are derived by the following steps:
 - a. Identify the basic organizational subjects that the enterprise works with;
 - b. Look at the list of processes and write down what classes of data are used as input and output to each processes; and
 - c. Cross-refer the data classes to the subject databases to ensure that all the possible data classes are covered by these subject databases.
4. **Step (4)-> Critical Success Factor (CSF) Analysis:** At this step CSF of the organization have to be identified. CSF is a method of eliciting requirements by asking users to define the factors that are critical to the successful performance of their functions or making decisions [40],[41]. The controlling individual or group should identify what factors are the most critical for success and what pressures can brought to optimize these factors. Some critical success factors have revealed the need to build new information systems since in some case information system could be critical for the success of an organization. The effect that any kind of data has on the CSF should be identified and a cost benefit analysis should be done to decide if it is worthwhile to incorporate this data into an information system.
5. **Step (5)-> Management by Objectives:** Functional analysis sets out to examine the mission and objectives in an organization. The overall objectives can be broken down into sub-objectives for each level and each department of the organization. Then the three following questions can be asked:
 - a. how can these objectives be measured?

- b. what data are needed for their measurement?
- c. how should decisions be made at each level so as to best achieve these objectives?

Questions “a” and “c” relate to management issues. However question “b” can lead to the identification of new data to measure the accomplishment of the objectives.

6. Step (6)-> Entity Analysis: Using the information gathered during steps 1 to 5, the next step is to derive the entities and the relationship required by the enterprise.
7. Step (7)-> Business System Chart: This step shows how to group subject databases into implementable systems and subsystems. Usually subject databases are ordered by the occurrence of their corresponding process into the life cycle. Then for implementation reason, the subject databases are grouped into implementable systems[34]. At this step existing data processing systems have to be considered.
8. Step (8)-> Structured Entity Chart: First the depth of each entity of the E-R diagram is determined. A depth one entity is an entity with single arrow(s) only pointing to it. A depth two entity is an entity pointing to a depth one entity. A depth N entity is an entity pointing to a depth N-1. The E-R diagram is then ordered as follow:
 - a. each entity of depth one is put at the top;
 - b. entity of depth two are put below corresponding depth one entity; and
 - c. entity of depth N are put below corresponding depth N-1 entity.

9. Step (9)-> Entity Cluster Analysis: Three possible approaches can be taken to cluster entities together:

- a. Subject Databases: Entities can be grouped by the subject databases determined at step 3. This approach is straightforward but imprecise;
- b. Affinity Analysis: The affinity between two entities is defined as the probability that these two entities will be used by the same process. If the two entities are never used by the same process their affinity is zero. If the two entities are always used together by the same processes their affinity is 1. Therefore, the entities having a high affinity are grouped together. This approach is more precise than a. above but requires more information; and
- c. Frequency of Association Paths Between Entities: To calculate the frequency of associations between entities the frequency of the transaction is required as well as the entities involved in that transaction. The different transaction to consider can be defined following the 20/80 rule. [42].

For this approach pairs of entities united by an association (relationship) having a high frequency are grouped together. This approach is the most precise one but is also the one which require the more information.

10. Step (10)-> Partitioned Entity Chart: The relationships still uniting the different groups formed in step 9 are removed to create independent group of entities.

This step is the last one of the Planning phase. It is to note that the IEW Planning Workbench supports most of the different steps of the planning phase. Step 8 which designs the structure of entity charts, and step 9 which considers the frequency of associations between paths, are not supported. (See Appendix "A" for more details).

2.5.2 Business Analysis Phase:

Step 11 of the figure 2-3 is a transition from the Planning phase (TOP DOWN) to the Analysis phase (BOTTOM UP). Using the partitioned entity charts of the previous phase, the analyst has a way to decompose the overall data of the enterprise in a partitioned group of entities of reasonable size. The attributes of the user View can be associated to the appropriate entity of the entity chart. The entities can be compared to the main elements of the user Views. If a View main element does not correspond to any of the entity, appropriate modifications would have to be made to the user View or the entity chart [35],[36],[37].

The following paragraphs will describe the different steps of the business analysis phase illustrated at figure 2-3. Some of these steps override each other or are done in parallel in order to add flexibility to the methodology:

1. Step (12)-> User Views: During this step Views are requested from user. The attributes of the user Views are associated to the appropriate entity of the entity chart.
2. Step (13)-> Users Data Flow Diagrams (DFDs): From this step, using the inputs from users data flow diagrams are built for each functions and processes identified in the TOP DOWN Planning phase.
3. Step (14)-> Compare DFDs and User Views: This is the time to compare Data Oriented Perspective with the Process Oriented one. The contents of the DFDs could be compared with the contents of the E-R diagram to ensure that nothing is missing on both sides and one perspective is in accordance with the other.
4. Step (15)-> Canonical Synthesis: Following different steps, each view are normalized individually to get the third normal form. Then all the different views

are merged together and following different sub-steps the result is also normalized to 3NF. At that step, the end result has to be normalized up to the 5NF. Then to ensure data stability a five step stability analysis is made on the end result. The data model needs to be as stable as possible. "It is a bridge between the past and the future: the past because it must take into consideration conversion and coexistence of old systems; the future because the data administrator persuades the end users and analyst to think about future uses of the data, and structures the data in the fashion which is as stable as possible" [37]. The process of normalization aspire to a design which is devoid of redundancy. The reason for eliminating redundancy is to remove the difficulties associated with updating duplicated information. On the other hand, a normalized design involves separate tables which may have to be joined to satisfy given queries, and this in itself is an overhead. Thus there could be cases (For specific frequent queries) where the designer might reintroduce some redundancy (Denormalize selectively starting from 5NF) for performance reasons. For more details on this step see Appendix A.

5. Step (16) & (16A)-> Structured Data Model: The final result of step 15 is a stable structured data model. This data model can be compared to the structured entity chart of step 8, to ensure that they correspond to each other, if they don't appropriate adjustments can be done.
6. Step (17)-> Overall DFDs: Individual DFD are merged together to give an overall DFD.
7. Step (18)-> Compare Overall DFDs with Overall Data Model: As for step 14, the overall E-R diagram can be compared with the overall DFD to ensure that they correspond to each other, if they don't appropriate adjustments can be made.

For example, some data store considered in the user DFD could have to be transformed as external agents.

It is to note that the IEW Analysis Workbench offers E-R and DFD diagrams required to support this phase. However nothing is offered to support the step 12 user Views as well as the step 15 canonical synthesis (See Appendix "A" for more details).

2.5.3 System Design Phase:

The following paragraphs describe the different steps of the design phase illustrated in figure 2-3 [34],[39]:

1. Step (19)-> Logical Access Map: For each procedures or sub-procedures (Transactions) determined at step 1 of the Planning phase the sequence and the content of the accessed entities is identified following five different steps [37].
2. Step (20)-> Action Diagram: With the help of different diagrams, the sequences in which the transaction actions must occur and the condition for their occurrence is represented in a procedural format [37],[38] .
3. Step (21)-> Distribution Analysis: As prone by the Information Engineering methodology, this step should be divided in two parts: the qualitative and the quantitative approaches.

The first part is the qualitative approach and provides management with a framework to decide at the subject database level which subject databases should be distributed and if yes how they should be distributed at the different sites.

The second part is the quantitative approach and it is done at the entity level. Neither reference [34] nor [39] treat data distribution at a satisfactory level. They briefly indicate different issues such as transmission cost, response time, availability,

fragmentation etc... to be considered when it is time to allocate data, but none of them provide choices for algorithms to decide how the data should be distributed. Besides, none of these references indicate how these different issues impose constraints on the application of these different algorithms during data allocation.

It is important to note that IEW Design Workbench offers the Action diagram required to support step 20. It does not support Logical Access Maps. It even offers extra feature to support some of the implementation issues. Naturally IEW offers nothing to support step 21 of Distribution Analysis since the methodology itself does not suggest any well defined approach for data distribution (See Appendix "A" for more details). As such, requirements exists for enhancement of IEM with an approach for data distribution design.

2.6 Summary.

In this chapter, the different issues of software life cycle phases as well as the different perspectives to be covered by information system development methodologies have been discussed. The different objectives and key components of CASE tools were also described. Then a selected methodology has been applied to a practical problem to illustrate these different life cycle phases and perspectives. A selected CASE tool was also applied to that same project to show how it can support an information system methodology.

In the next chapter, a number of models for data distribution design are examined. Various distribution models are reviewed. A comparative study of these models are presented. Finally one model is selected and employed for development of a prototype CASE tool.

Chapter 3

Data Distribution Design.

3.1 Overview.

In spite of limited experience in the design of distributed systems, the problem of data distribution design has been extensively studied as an advanced research area, mostly from a technical viewpoint, and many contributions can be found in the literature.

The next section presents the two possible approaches for data distribution design, the top-down and the bottom-up approach. A breakdown of the top-down approach is given. Section 3.3 discuss different approaches developed for horizontal and vertical data fragmentation. Section 3.4 describes the different features to be considered in order to allocate data. Section 3.5 describes existing approaches for data dictionary allocation with their respective advantages and disadvantages. Section 3.6 analyses different data allocation models by comparing their data allocation features to the different data allocation features described at section 3.4 . The result of this analysis is section 3.7 where an allocation model is selected for further implementation in a prototype CASE tool.

3.2 TOP DOWN versus BOTTOM UP Approaches.

There exist two alternative approaches for the design of data distribution, the top-down and the bottom-up. The bottom-up approach can be used when existing databases are to be integrated. This approach is based on the integration of existing schemata into a single, global schema. Integration means the merging of common data definition and the resolution of possible conflicts among different representations of the same data.

Figure 3–1 illustrates the top-down approach. This approach consists of designing the global enterprise data model and from there, proceed by first designing the fragmentation of the database, then by allocating the fragments to the sites and finally by allocating the data dictionary. This approach is completed by performing, at each site, the “physical design” of the data which are allocated to it. As for centralized database this step consist of mapping the conceptual schema to the storage access and determining appropriate access methods[43].

Both approaches can be incorporated into existing information system development methodologies. For example the bottom-up approach can be incorporated into the IEM life cycle at the business analysis phase, the top-down approach can be incorporated into the life cycle at the system design phase of any information system development methodology which produce a relational data model of the organization such as IEM. This thesis will limit itself to the first three steps of the top-down approach i.e. the fragmentation, the data allocation and the data dictionary allocation. The following paragraphs will describe these three different steps but will put more emphasize on data allocation.

3.3 Design of Database Fragmentation.

The purpose of fragmentation design is to determine fragments which are “units of allocation” , that are appropriate starting-points for the subsequent data allocation phase. Designing fragments consists in grouping tuples (in the case of horizontal fragmentation) or attributes (in the case of vertical fragmentation) which have the same properties from the viewpoint of their allocation. Therefore fragmentation is tightly related to allocation and the criteria used to determine a fragment should be the same as the one used to determine an allocation. For example, if the allocation criteria is to minimize transmission

Figure 3-1: Top Down Approach for Data Distribution Design

cost then the criteria to fragment should also be one that minimizes the same transmission cost. In the following two sub-sections various approaches for horizontal and vertical fragmentation will be presented.

3.3.1 Horizontal Fragmentation.

Horizontal fragments consist of subsets of the instances (tuples) of an object having all the attributes of that same object. Determining the horizontal fragmentation of a database is not an easy task and many different data properties as well as data allocation criteria can be considered. A well known data property is the logical relationship between the data which result in a fragment predicate for horizontal fragments [43], [44], [45], [46], [47].

With this logical approach, each fragment is associated with a minterm predicate which indicates the distinguishing properties possessed by the instances or tuples of that fragment[44]. A simple predicate is a predicate of the type "Attribute θ value" (Where θ is '=', 'not =', '<', '>', '<=' and '>=') . A minterm predicate "y" for a set P composed of simple predicates is the conjunction of all predicates appearing in "P". Therefore a fragment is defined as the set of all tuples for which a minterm predicate holds[43], [45].

For example, suppose there is an entity named 'EMPLOYEE'. Among different attributes, this entity has the attribute type 'DEPT-NUMBER' and 'JOB'. Assume now that there are only two departments, 1 and 2. Two simple predicates for this example could be 'DEPT-NUMBER = 1' and 'JOB=Clerk'. The minterm predicates for these two predicates would be:

1. 'DEPT-NUMBER = 1' and 'JOB = Clerk';
2. 'DEPT-NUMBER = 1' and 'JOB not = Clerk';
3. 'DEPT-NUMBER not = 1' and 'JOB = Clerk'; and
4. 'DEPT-NUMBER not = 1' and 'JOB not= Clerk'.

The first fragment would consist of all the tuples for which the minterm predicates 1 holds, the second fragment would consist of all the tuples for which the minterm predicates 2

holds and so on.

Another data property which can be considered is the “statistical” property of the data such that each fragment can be formed based on the number of references made to its instances by applications. This property can be referred to as the affinity of tuples. This affinity has the same definition as the affinity defined in section 2.5.1 of the IEM Planning Phase. Tuples having a high affinity are grouped together [43], [44], [45], [48]. The resulting fragments are used [43], [44], [48] by algorithms that allocate data based on the criteria of locality. Locality is defined as follow: “The data objects obtained from the fragmentation are allocated to the site where they are required the most often” [48]. With this definition, locality tries to minimize total transmission cost by allocating a fragment to a site where it is most used. Therefore the “statistical property of the data” can be used to determine horizontal fragments if these fragments are used in an algorithm that try to minimize transmission cost.

Apers proposed to create different horizontal fragments as required for each transaction. Only the tuples required for a transaction are part of its associated horizontal fragment. This approach is totally different from the previous one. These fragments are used in an allocation algorithm that minimizes total transmission cost [46], [47].

When the objective of the allocation approach is to minimize the response time of the system, one alternative solution is to allow as much parallelism as possible. Apers proposed to perform horizontal fragmentation randomly into sets containing an equal number of tuples. This will increase parallel processing which may result to an enhanced response time for query processing [46], [47].

In addition to the approaches mentioned above, other approaches for horizontal

fragmentation have been investigated by Maier and Ullman [49], Paradaens and DeBru [50] and Chang and Cheng [51].

3.3.2 Vertical Fragmentation.

Vertical fragmentation consists of selecting a subset of attributes of an object and grouping into a set all tuples (instances) of that same object which only have that subset of attributes [43], [44], [45], [46], [47]. Dismantling the groups of attributes goes against the analysis phase of IEM where attributes were grouped by entities for logical reasons (See section 2.5.2). Besides that, for each extra vertical fragment created, an extra copy of the key has to be added to the fragment to allow joins. Vertical fragmentation is not as frequently applied as horizontal fragmentation for logical and complexity reasons. Therefore there are not as many useful ways to create vertical fragments as there are to create horizontal ones. Two general approaches exist for vertical fragmentation [43], [52]:

1. The split approach in which global relations are progressively split into fragments [52]; and
2. The grouping approach in which attributes are progressively aggregated to constitute fragments [46], [47], [53].

Except for Apers' approach for vertical fragmentation [46], [47], the affinity between pairs of attributes (Affinity as previously defined) is used to decide if an existing group of attribute has to be split or if different attribute(s) should be grouped together [52], [53]. The split approach is actually the preferred one. The rationale behind is that if the analysis and planning phase were well done, the "optimal" solution is much closer to the groups composed of all attributes (entities) than the groups that are single-attribute partitions.

3.4 The Allocation Of Fragments.

The selection of an optimal file/data allocation for a distributed computing environment is a problem that researchers have been extensively studied. Over the years, the development of allocation algorithm progressively considered more and more features. For example, earlier work in this area was done in the context of non-redundant file allocation problem. Later for obvious cost optimization, the possibility of allocating redundant fragments(data) was also considered. Some researchers even went beyond by considering allocation of data as well as its associated resources such as computers, application programs etc.. [54], [55], [56]. This thesis will limit itself to the allocation of data.

3.4.1 The Different Allocation Features.

Numerous algorithms considering many different allocation features presently exist. In order to have a framework to evaluate different selected algorithms, these features have been logically grouped and defined as follow:

1. Basic: Under this group, the basic features of the data allocation have been included. Any selected data allocation algorithm has to consider as a minimum the following basic features [42], [43], [44], [46], [47]:
 - a. Multiple Data: More than one kind of data. For example, with entity A, B and C, there is multiple data. The opposite of multiple data would be single data where only one data kind is considered. For example, only entity A;
 - b. Redundant Data: More than one copy of the same data. For example, multiple copies of entity A;
 - c. Vertical Fragments: As defined in the previous sub-section; and
 - d. Horizontal Fragments: As defined in the previous sub-section.

2. **Solution Characteristics:** This feature group includes basic characteristics differentiating one solution function from another. The basic characteristics have been identified as follow:
 - a. **Algorithmic Solution:** This feature means that since no heuristic has been developed, the only way to solve the algorithm is to compute all the possibilities or use a branch and bound approach which will give the global optimal solution of the objective function. The objective function to be resolved, as well as its associated constraint, can be either linear or non linear. However, this solution is an optimal one as far as the specific allocation features covered by the objective function are concerned. It is not an optimal solution to the overall allocation problem since, as will be shown in section 3.6, no model exists with a function considering all the different allocation features described in this sub-section. Some allocation features such as storage cost and reliability are also in conflict with each other. For example, an increase to the number of copies of a file would add to the reliability of the overall system; however this would also increase the storage cost (It would not minimize the storage cost). Besides, the quality of the input data may not be sufficient to guarantee that the computed solution is in fact the best one. There is also the fact that only the quantifiable features covered in this section can be part of a function. As will be shown in chapter 5, other qualitative features can be considered for data distribution. For a specific application, usually only a subset of the overall quantitative and qualitative features is significant enough to be considered. As shown in [57], the process of finding optimal allocation is a NP-complete problem which means that the compu-

tation time for all known optimal algorithms for this class increase at least exponentially with the problem size.

B. W. Wah [58] has demonstrated that using a linear integer programming approach for the allocation of multiple copies of a single file, the time complexity of the allocation algorithm increases as fast as $K^n - 1$ where K represents the number of decisions and n represents the number of sites. This linear integer programming approach considers all the different possibilities to assign data (represented by a '1') or not to assign data (represented by a '0') to all the different number of sites therefore there are two possible decisions i.e. $K=2$. For example, if there are three sites there are seven possible combinations of 0's and 1's to consider (eg. 001 to 111). Since the time complexity for multiple copies of a single file increases as fast as $K^n - 1$, the time complexity for multiple copies of multiple files will at least increase as fast or even faster. This demonstrates that the time complexity of allocating multiple copies of multiple files is a NP-complete problem. As far as computation time is concerned this time complexity can be considered as a disadvantage since for large size problems the user would have to wait hours if not days to get results. However, as for the previous feature of linear programming a fast computation time is not necessarily that important if data allocation is not done too often in the lifetime of a distributed database;

- b. Heuristic Solution: This feature means that heuristics have been developed to resolve the solution space of the objective function. Of course, these heuristics give only a local optimum. However, some of these heuristics are tractable i.e. they determine how far the local optimum is from the global one. Since heuristics reduce the search space, this feature is desirable as far

as the selection of the algorithm is concerned. To have a tractable heuristic is even more attractive.

3. Cost: This feature groups includes all the different possible costs that can be considered by the allocation approach. The different costs have been identified as follow:
 - a. Communication Cost: This cost is defined as the monetary cost which depends on the volume of all data transmitted due to the query/update processing [59]. If the application environment requires allocation of data in a long haul network, it is vital to consider this cost [48];
 - b. Storage Cost: This cost considers the fact that at different sites, different media can be used to store data. This would result in different storage costs. For example, at site 'A' it can cost five times more to store a unit of data than at site 'B'. There exists economies of scales in that the cost per bit is lower with large storage units than small ones;
 - c. Database Software and Application Program Cost: Redundant copies over multiple sites means redundant database software and application programs. Multiple copies of database software and application programs means extra cost incurred by issues such as configuration control, multiple copies of software on disks etc... Therefore this cost is proportional to the number of copies of data; and
 - d. I/O and CPU Cost: These costs are different at each sites and they represent monetary cost of CPU processing due to authorization, concurrency, recovery etc... [42]. If the application environment involves the allocation of data in a local area network, this cost is important enough and should be considered.

4. **Execution Time:** This feature group involves time which is usually considered as a cost [59]; it can also be considered as a performance [60]. The different execution time features have been identified as follow:
 - a. **Queuing Delays:** The delay that data encounters in a queue before being processed; and
 - b. **Query Response Time/Throughput:** Total time elapsed from query submission to delivery of the response is the query response time. This time can be broken down into access, processing and transfer time. The throughput of the system is the rate at which query and update are being processed by the system (Number of queries and updates per second).
5. **Query Optimization:** This feature group includes the different query optimization issues that are considered for allocation. Before explaining the different query optimization issues, the concept of a processing schedule has to be explained. A query optimizer, also called query processing algorithm will determine a processing schedule to process a query or an update. A processing schedule contains a number of tasks, each task is a basic operation to be performed by the sites involved, which together deliver the final result to the user [46]. Figure 3-2 depicts a processing schedule:

Figure 3-2: An Example for a Processing-Schedule

In this figure, let:

- $S_i, i = 1, 2 \dots n$ be the possible sites. Here we have site S_1 to S_3 ;
- $IR_x, x=1, 2 \dots n$ be the n different fragments. Here we have three different fragments i.e. IR_1, IR_2 & IR_3 ;
- $TP_i(IR_x)$ be the queuing time for fragment IR_x before being processed at site S_i . Here we have $TP_1(IR_1)$, $TP_2(IR_2)$, and $TP_2(IR_1, IR_2)$;
- $TW_i(IR_x)$ be the waiting time for fragment IR_x at site S_i before being transmitted. Here we have $TW_1(IR_1)$ and $TW_2(IR_3)$;
- $T(IR_x)$ be the time to transmit a fragment IR_x from one site to another. For example, with $T(IR_1)$ we have fragment IR_1 transmitted from site S_1 to S_2 ; and

- f. $P(IR_x)$ be different relational operations made on fragment IR_x . For example, with $P(IR_1)$ & $P(IR_2)$ we have selection operations made on fragment IR_1 & IR_2 . With $P(IR_3)$ we have a semi-join and a projection operation made on fragment IR_3 .

In figure 3-2 the query originates from site S_3 and is sent to site S_1 and S_2 for processing. It is sent to these two sites since they have the fragments IR_1 and IR_2 required by the query. At each site (S_1 & S_2) the query waits for time $TP_1(IR_1)$ and $TP_2(IR_2)$ (representing a processing delay in the node) before being processed. It is then processed in parallel at site S_1 & S_2 by the two selection operations of $P(IR_1)$ and $P(IR_2)$. When the operation $P(IR_1)$ is completed it waits for time $TW_1(IR_1)$ (representing a transmission delay in the node) before being transmitted to site S_2 . It takes time $T(IR_1)$ to transmit fragment IR_1 from site S_1 to site S_2 . Once received at site S_2 , fragment IR_1 and IR_2 wait for time $TP_2(IR_1, IR_2)$ before being processed. They are then processed (For a semi-join and a selection operation e.i. $P(IR_3)$). The result IR_3 waits an amount of time $TW_2(IR_3)$ before being transmitted to site S_3 . It then takes time $T(IR_3)$ to transmit the result IR_3 to site 3.

The different query optimization issues have been identified as follows:

- a. Multi-Query Optimization: Optimization of the overall query processing cost of the system by applying a query strategy which consider the existence of the queries/updates being executed in parallel. This query processing strategy has to consider the competition in parallel between queries/updates for data items and resources [59]. Its opposite is individual query optimization where the processing strategy does not consider the competition in parallel between queries/updates for data items and resources,

- b. **Static Query Optimization:** In this case, the complete individual/multi query processing schedule (From a query optimizer with an individual or a multi-query optimization strategy) is formed prior to the allocation phase and will not be modified during the allocation of the fragments and/or operators. A static query processing schedule (See figure 3-2) assumes a priori knowledge of the execution cost of the different sequences of query's relational operators (Semi-joins, Selection etc...) [59]; and
 - c. **Dynamic Query Optimization:** In this case, the complete individual/multi query processing schedule (From a query optimizer with an individual or a multi-query optimization strategy) is not formed prior to the allocation phase but it evolves with the allocation of the fragments and operators. For example, suppose we allocate initially according to figure 3-2. Then during the allocation it has been decided to allocate fragments IR1 & IR2 to the same site say, site S2. If immediately after that decision the query optimizer is executed, the figure 3-2 processing schedule would have to be modified accordingly.
- 6. **Network:** This feature group includes the different computer network issues presently considered for allocation. The different issues have been identified as follows:
 - a. **Channel Capacity:** The maximum data transmission rate that a link can support. This feature is a real constraint imposed on the model which can not be exceeded;
 - b. **General Topologies:** Models are based on existing specialized topology such as star, ring etc... or can be applied to general network topologies; and

- c. **Nonfixed Routing:** Models are based on either fixed or dynamic routing with different routing strategies [61]. A dynamic routing strategy will change the route borrowed by the data/request to go from one node to another. In the objective function, this will be reflected by adding a routing assignment variable which will reflect the percentage of data/request routed from one node to another. Obviously for each data/request the total of that variable has to be equal to 100 percent. A model with this feature has the potential to improve fault tolerance since it considers the possibility that the data can borrow different route to go from one node to another.
7. **Operational Features:** This feature group includes all the other issues which do not belong to any of the previous groups yet considered in the different allocation models. These extra features have been identified as follows:
- a. **Locality:** As previously defined in the section of horizontal fragmentation, locality is defined as follow: "The data objects obtained from the fragmentation are allocated to the site where they are required the most often." Allocating data to the sites where they are the most often required minimizes remote data accesses which reduces the total transmission cost of the system. The feature of locality has therefore an indirect effect of reducing the transmission cost of the system. As indicated before, this feature should be taken into account in a long haul network environment;
 - b. **File/Node Availability:** This feature is defined as the probability that a file can be accessed from where it is located. A model with this feature has the potential to improve fault tolerance since a system with high availability is less dependant on telecommunication failures (Most processing can be done

locally);

- c. **Storage Capacity:** This feature represents the maximum amount of data that a node can store. This feature is a constraint imposed on the model which can not be exceeded;
- d. **File/Program Dependence:** Any given transaction will execute a program file before accessing a data file. This feature considers the dependence between files and programs. Data files and program files must not be assumed to be independently accessed [60], [62]; and
- e. **Reliability:** For any reasons such as disaster, sabotage etc..., in a network with geographically distributed sites, there is always the possibility of having the data that resides at one or many sites destructed. The data reliability in a distributed database is defined as the ratio of intact data instance to the overall data instance at any point in time in the distributed database system. For example, if in a distributed database system out of 100,000 data instances, 5 of them are corrupted, then the data reliability for that system is 99.995 %.

3.5 Data Dictionary Allocation.

The data dictionary is the repository of a variety of information concerning objects of interest. In the case of a distributed system in particular, the dictionary entry for a given object must specify the site(s) at which that object is stored, in addition to other relevant information.

There exist different alternatives to allocate different portions of the data dictionary to various sites. Their principal advantages and disadvantages are noted below[63]:

1. **Centralized:** The total data dictionary is stored exactly once, at a single central site. The disadvantage of this approach is that there is a severe loss of local autonomy and the entire system is critically dependent on the availability of the site with the data dictionary.
2. **Fully Replicated:** The total data dictionary is stored in its entirety at every site. It has the disadvantage that every catalog update must be propagated to every site, and the addition of a new site to the network is a highly nontrivial operation. However, the advantage is that every site has local autonomy.
3. **Partitioned:** Each site maintains its own data dictionary for objects stored at that site. The data dictionary is the union of these disjoint local data dictionaries. This approach has the advantage that purely local operations do not involve any remote access. However, requests for nonlocal operations require a broadcast operation be made to all sites in order to locate the required data dictionary entries.
4. **Combination of Centralized and Partitioned:** Each site maintains its own local data dictionary, as in the partitioned approach in addition; a single central site maintains a copy of the union of all of those local data dictionaries. This approach is an improvement, in the sense that request for nonlocal operations involve not a broadcast but an access only to the central catalog site. That central site is still important for the overall system but not critical since if that site fails, a broadcast to all the different sites can still be made and transactions be processed.
5. **Other Combinations:** One such combination is that employed in the SDD-1 [63]. In SDD-1, the entire data dictionary is treated as a single relation/entity. Like any other relation it is fragmented and allocated. There is a higher-level data dictionary which contains information about the location of the different fragments of the data dictionary. This higher-level dictionary is fully replicated.

Any change to the physical distribution of the data dictionary will require the higher-level data dictionary to be updated at every sites.

The first three approaches are relatively simple but depending of the application environment they can have some serious disadvantages associated with them. The last two approaches are relatively more complex to implement but depending of the application environment they can have fewer drawbacks than the first three.

3.6 Analysis Of Allocation Models.

Features of twenty selected data allocation models have been studied and the results are summarized in figure 3–3. The identification of the features considered by models 1 to 12 has been treated elsewhere[60] and are therefore reviewed briefly in the following paragraphs. Models 14 to 16 will be described in more detail in the following subsections.

Covered Features	Models Number																			
	1	2	3	3A	4	5	5A	6	7	8	9	10	11	12	13	14	14a	15	16	16a
BASIC																				
a. Multiple Data					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
b. Redundant Data	X	X	X	X																
c. Vertical Fragments								X	X		X	X				X	X	X	X	X
d. Horizontal Fragments								X	X	X	X	X				X	X	X	X	X
SOLUTION Characteristic																				
a. Algorithmic Solution	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
b. Heuristics	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
COSTS																				
a. Communication Cost	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
b. Storage Cost	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
c. DB S/W Apl.Prg. Cost					X															
d. I/O & CPU Cost				X														X	X	X
EXECUTION TIME																				
a. Queuing Delays					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
b. Query Resp. Time/Thrg					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
QUERY Optimization																				
a. Multi-Query Optim.																		X	X	X
b. Static Query Optim.																			X	X
c. Dynamic Query Optim																			X	X
NETWORK																				
a. Channel Capacity						X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
b. General Topologies	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
c. Nonfixed Routing			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
OPERATIONAL																				
a. Locality																X	X	X	X	X
b. File/Node Availability					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
c. Storage Capacity					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
d. File/Prog. Dependance			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
e. Reliability							X	X	X	X	X	X	X	X	X	X	X	X	X	X

✕ : Model covers feature.

✕ : Model has potential to cover feature.

Figure 3-3: Features of Data Allocation Models

Model 1 was developed by Case in 1972 to minimize the cost for distributing a single

file redundantly across a network [60], [64]. Model 2 was developed by Chandy in 1976 to also minimize the cost for a single file. However the contribution of this model is that it offers a heuristic to resolve the objective function [60], [65]. Model 3 was developed by Morgan in 1977 to minimize cost considering that files and programs must not be assumed to be independently accessed [60], [66]. Model 3A is an extension to model 3 in the sense that it considers everything that 3 does plus it considers the cost of database and programs software, computer equipment and CPU cost [67], [68]. These four models are restricted to the allocation of a single file. They do not meet all the basic features and are therefore not applicable to applications with multiple files.

Model 4 was developed by Chu in 1973 to minimize the cost for distributing multiple files across a network. It considers queuing delays and response time as constraints [60], [68]. Model 5 was developed by Mahmoud and Riordon in 1976 to minimize the cost for distributing multiple files. It provides an objective function which find file and channel capacity assignment that minimizes the sum of communication and storage cost. The response time as well as the file/node availability are considered as constraint. Since both constraints are non-linear, it is quite complex to derive an algorithmic solution, the authors developed a heuristic technique to find local optimum [60], [69]. Model 5A was developed by Irani and Khabbaz in 1979, it is an extension to model 5, it considers everything that model 5 does and, in addition, it considers reliability as a constraint [68], [70]. These three models also do not meet all the basic features and will not be selected as candidates for the prototype CASE tool.

Each of the previous five models determines a file assignment that minimize a cost objective function. Models 6 to 12 inclusive attempt to optimize an execution time objective function. Model 6 was developed by Ramamourthy in 1970, its main

contribution is the realization that under certain network assumption, known memory-hierarchy results are applicable to the file allocation problem; however these files have to be split. This model seeks to minimize the average access time subject to a storage cost constraint. It does not considers the processing and transfer time [60], [71].

Model 7 was developed by Arora in 1973, it is an extension of model 6, i.e. in addition to minimizing the access time, it minimize the processing and transfer time[60], [72]. Model 8 was developed by Hughes in 1973, it maximizes the throughput by further consideration of queuing delays. The model was developed and geared for the closed star network topology [60], [73]. Model 9 was developed by Chen in 1973; whereas model 8 maximizes throughput in a closed network, this model minimizes response time in an open network. This model is geared to an open star network topology [60], [74]. Model 10 was developed by Kleinrock in 1976; it minimizes response time in a general topology network. As in model 9, files in model 10 have to be split [60], [75].

Model 11 was developed by Jones in 1978; it maximizes the throughput assuming that files cannot be fragmented and it is restricted to a star network topology [60], [76]. Model 12 was developed by Dowdy in 1977. As an extension of model 11, it maximizes throughput for general topologies [60], [77]. As for the first seven models, model 6 to 12 do not meet all the basic features and therefore have not been selected for the prototype CASE tool. Based on the previous 12 models, it can be concluded that models developed prior to 1980 can be classified into two groups: the first group emphasize on costs features whereas the second group emphasizes execution time features.

Model 13 was published by S. Navathe and co-searchers in 1984 [52]. Instead of considering multiple data (See section 3.4 under Basic), the objective function of this

model considers only single redundant data and can be applied to vertical fragments only. The objective is to find an optimal allocation which will result in a minimum storage and communication cost. A heuristic is offered to find a solution. It starts first with a non-replicated solution and then add an extra fragment copies one at a time as long as the total cost decreases. This model can be used for general network topology. It does not cover all the basic features and will not therefore be considered for the final selection.

The following models cover all the basic features required for data allocation and will therefore be considered as candidates for the final selection. In order to have a good understanding of each model, they will be described and analyzed in more detail in the next sub-sections.

3.6.1 Model 14 Description.

This model was published by S. Ceri et al in 1987 [44]. It attempts to distribute each fragment to the site where it is mostly used, it therefore uses the feature of locality to allocate data. It provides the following function to quantify the difference between the benefit of the number of retrieval accesses that become local due to the addition of one copy versus the cost of the number of additional remote update accesses required to maintain that copy:

$$D_{kj} = \sum_{i: \text{reads } k} F_{ij} P_{ijk} - \sum_{i: \text{writes } k} \sum_{h \text{ not } = j} F_{ih} P_{ihk} \quad (3.1)$$

Where:

- a. F_{ij} = frequency of transaction "i" from site "j"; and

- b. P_{ijk} = Polarization of fragment "k" for transaction "i" from site "j". As per [44] the polarization is the probability that a given fragment is accessed by a given transaction running at a given site.

In equation 3.1 the first term reflects the increase in benefits from reads, whereas the second term represents the costs incurred by writes.

3.6.2 Model 14 Analysis.

The following paragraphs will identify the different features supported by this model:

- a. Basics: The polarization " P_{ijk} " can apply to vertical or horizontal fragments. The summation term " Σ " can consider non-replicated as well as replicated fragments. Therefore all the basic features are covered;
- b. Solution Characteristics: Each term of that function is linear and therefore fast to compute. A greedy heuristic is offered for the redundant allocation by selecting additional sites for each fragments with respect to the initial non-redundant allocation. At each iteration, the most beneficial site for storing a copy of the fragment is added to the set of allocation. The method suggests the addition of extra fragment when the benefit D_{kj} is large and positive. The method however does not define what is a large positive number e.g. 10, 100 or 1000 ? As suggested at [78] this value could be normalized to range between -1 and 1. A value close to 1 indicates that a duplicate copy should be allocated. A value close to zero or negative suggests that no further duplication should occur. This rule exists because the overhead for a remote update operation is considerably larger than the overhead for a remote read operation. Actually the designer can set a minimum value for D_{jk} beyond which further replication is not allowed;

- c. Cost: No cost is considered;
- d. Execution Time: No execution time feature is considered;
- e. Query Optimization: No query optimization feature is considered;
- f. Network: There is no network topology constraint, therefore it supports general topologies; and
- g. Operational Features: As previously explained, this formula uses the feature of locality. Locality indirectly reduces the transmission cost.

3.6.3 Model 14A Description.

This model was published by S. Ceri and G. Pelagatti in 1984 [43]. As for model 14, this model maps fragments to the sites where they are mostly used. It provides the following function:

$$B_{ij} = \sum_{k: k \text{ reads } i} F_{kj} R_{ki} - C \times \sum_{k: k \text{ writes } j, j' \text{ not} = j} \sum F_{kj'} U_{ki} + B(D_i) \quad (3.2)$$

$$\text{and : } B(D_i) = (1 - 2^{1-D_i}) F_i$$

In equation 3.2:

- a. D_i : denotes the degree of redundancy of fragment 'i'. For example, if there is only one copy of fragment 'i' then $D_i = 1$ and $B(1) = 0$. If there is three copies of fragment 'i' then $D_i = 3$ and $B(3) = 3F_i/4$, and so on.
- b. F_i : denotes the benefit of having the fragment 'i' fully replicated at each site, this number has to be specified by the designer.

- c. C : is a constant which measures the ratio between the cost of an update and a retrieval access,
- d. F_{kj} : is the frequency of transaction 'k' at site 'j' (number of times/month),
- e. i : is the fragment number,
- f. j : is the site number,
- g. k : is the transaction index,
- h. R_{ki} : is the number of retrieval references of transaction 'k' to fragment 'i'.
- i. U_{ki} : is the number of update references of transaction 'k' to fragment 'i'.

3.6.4 Model 14A Analysis.

This function is similar to model 14 function. To multiply the frequency of a transaction (F_{ij}) by the probability that this fragment is accessed by this transaction (P_{ijk}) as model 14 does, is analogous to multiplying the frequency of a transaction (F_{kj}) by the number of references of a transaction to that fragment (R_{ki} or U_{ki}), as model 14A does. Actually this formula covers similar features addressed in model 14 and in addition because of its extra variable " $B(D_i)$ ", it reflects the reliability benefit of having a fragment replicated at different sites[42].

3.6.5 Model 15 Description.

This model was published by S. Ceri et al in 1983 [42]. The objective function of this model takes into account more parameters than those found in model 14 or 14A. The following is the objective function used to calculate the optimal non-replicated data allocation.

$$\min z = \sum_{i,p} C_{ip} X_{ip} + \sum_{i,j} D_{ij} Y_{ij} - \sum_{h,p} A_{hp} W_{hp} - \sum_{h,j} B_h V_{hj} \quad (3.3)$$

In equation 3.3:

- a. C_{ip} : is the cost of transaction processing related to the decision of partitioning the object "i" according to the partition "p". This cost takes into account both the communication and processing cost;
- b. D_{ij} : is the cost of transaction processing related to the decision of allocating the whole object "i" on node "j". This cost takes into account both the communication and processing costs;
- c. A_{hp} : is the communication cost which can be saved because of the use of the same partitioning criteria "p" on the owner and the member of a link "h". A link represent a binary relationship among two objects called the "owner" and the "member" of the link. For example, there is a 1:n relationship among one instance of the owner and the "n" instances of the member object. It is to note that a constraint of this model is that it applies only to 1:n relationship; and
- d. B_h is the communication cost which are saved because both the owner and the member of the link "h" are stored at the same site "j" as a whole. Note that this parameter does not depend on a particular site "j", as uniform communication costs are assumed.

The first processing-schedule graph of figure 3-6 would have a lower response time than the one of figure 3-7. This is because in figure 3-6 there are three join operations O1, O2 and O3 done in parallel at VS2, VS3 and VS4 whereas in figure 3-7 there are three join operations done sequentially at VS1. However, the second processing-schedule graph of figure 3-7 has a lower transmission cost (48,400 bytes versus 55,400 bytes) than the first one of figure 3-6.

These two figures (figure 3-6 and 3-7) illustrate how a different query optimization strategy can create different processing-schedules used later on for data allocation. It is essential that the objective of the data allocation strategy comply with the objective of the query processing strategy. For example, the objective of the allocation strategy used on the figure 3.7 should be to minimize the transmission cost, since the objective of the query optimization strategy used to create the figure 3-7 query processing-schedule was also to minimize the transmission cost.

3.6.8 Model 16 Analysis.

1. More Advanced Notions from Model 16:

To understand some of the features covered by this model, the following two notions have to be understood:

- a. **Assignment:** A physical site may have assigned to it a set of virtual sites, this set will be called the assigned set. A virtual site can be assigned to at most one physical site which means that it is added to the assigned set of that physical site; and
- b. **Union:** The union of two nucleus-sites is a nucleus-site whose set of fragments is the union of the sets of fragments of the two nucleus-sites, whose set of

entities the frequency of the different transactions is required as well as the entities involved in the transactions. Once the frequency of association between entities is known, then entities united by an association (relationship) having a high frequency are grouped together under the same sub-schema. Then using the objective function on each sub-schema a non-replicated optimal solution is determined. For the replicated part a greedy heuristic is used. At each step an extra fragment is allocated to a site in order to decrease the total cost the most. The algorithm terminates when no additional fragment can decrease the total cost any further;

- c. Cost :As previously mentioned C_{ip} and D_{ij} consider both the transmission and processing (CPU and I/O) costs;
- d. Execution Time: No execution time feature is considered;
- e. Query Optimization : For the replicated part when a new data is added, every time there is an evaluation which considers all the different transactions and the best allocation is chosen for a total minimum cost. This total minimum cost is the sum of the transmission and processing cost of each transaction considered individually. Figure 3-4 shows what happen when new data is added

Figure 3-4: Transaction Modification after Addition of Data

In figure 3-4 (a) there are three transactions T1, T2 and T3. Transaction T1 starts at site 1 to process data 1, it then goes through site 2 to process data 3 and ends at site 3 to process data 4. Transaction T2 starts at site 1 to process data 1, it then goes to site 3 to process data 2 and ends at site 2 to process data 3. Transaction T3 could be described the same way. In figure 3-4(b) an extra copy of data 2 has been added at site 1. Transaction T1 and T3 stay the same, however transaction T2 is adjusted to reflect the addition of an extra data. Transaction T2 stays now at site 1 to process data 1 and 2 and goes to site 2 to process data 3. This is multi-query optimization since the overall query processing cost is optimized, however

the parallel competition for resources and data items is not considered, therefore we have only partial multi-query optimization;

- f. Network : Since there is nothing about network topology constraint, it can adapt to any topology. There is a fixed transmission cost variable "TC" , therefore the model apply to fixed routing only; and
- g. Operational Features: As previously mentioned, there is a storage limitation constraint. For the replicated part, there is a function "f(dri)" which reflects the benefit of having an object replicated.

3.6.7 Model 16 Description.

This model was published in P.M.G. Apers PhD dissertation in 1982 [46], [47]. It is based on the fact that query optimization and data allocation are dual problems and cannot be done separately. "The data allocation problem cannot be studied without detailed knowledge about query processing" [46]. The rationale is that query processing is the problem of determining a view of the database and deciding at which sites to execute certain operations, such that the required result can be presented at the site where the query was issued. The dual of this is the data allocation problem, where the access patterns as well as the data to be allocated are given, and their allocation have to be determined. Aper's processing-schedule graph model illustrate transactions in a totally different way than all the previous models.

1. Some Notions from Model 16:

Before explaining this model, some notions peculiar to that model have to be explained. A nucleus-site is a pair(FS, OS) where FS is a set of fragments (one or many) and OS is a set of relational operations (one or many). There exists two types of

nucleus-site, namely the physical site (PhS) and the virtual site (VS). A PhS represents a real site in the computer network whereas a VS represents a set of fragments and associated relational operations which will eventually be allocated to a PhS. Putting a fragment F in the set of fragments of PhS correspond to allocate F to the site in the computer network which correspond to PhS. If we have $VS = (\{F1, F2\}, \{\})$, it means that fragments $F1$ and $F2$ are to be allocated to the same site in the computer network but that which site is not specified.

Now that the notion of PhS and VS have been explained, the processing-schedule graph can be explained in more detail. The processing-schedule graph takes its information from the transaction processing-schedule determined by the query optimizer (refer to figure 3-2 for an example of a processing-schedule). For each site in figure 3-2 which is not a site of origin or a site of result (i.e. $S1$ and $S2$), a VS is created. The site of origin and the site of result correspond to a PhS. For a query, in the processing-schedule graph, a physical site (PhS) corresponds to a destination site whereas in an update, the PhS of a process-schedule graph corresponds to the site of origin. For example in figure 3-2 we have $VS1 = (\{IR1\}, \{O1\})$ where $O1 = P(IR1)$ is a selection operation, $VS2 = (\{IR2\}, \{O2, O\})$ where $O2 = P(IR2)$ and $O3 = P(IR3)$ are respectively a selection and a semi-join operation. Since it is a query, the physical site $S3$ correspond to a destination.

2. Examples of Processing Schedule Graph:

The following example will be considered to illustrate the processing schedule graph. Consider two entities:

- a. WINE with attributes YEAR, NAME, PRODUCER, AREA and COUNTRY where the keys of that entity are the attributes YEAR and NAME; and

- b. **WEATHER** with attributes **YEAR**, **AREA**, **COUNTRY**, **SUN** and **RAIN**, where **SUN** stands for hours of sun and **RAIN** stands for millimeter of rain in a particular area. The keys of that entity are the attributes **YEAR** and **AREA**.

Assume that the transaction below is transaction number T2 and the results are required at site 2.

"Give the name of the wine, its country, its producer and the weather characteristics such as number of hours of sun and number of millimeter of rain, if that wine is from the Napa Valley and its year is 1970."

That query could be translated as follow: (WEATHER(YEAR = 1970) (WINE {AREA = Napa Valley})) [YEAR, AREA, NAME, PRODUCER, COUNTRY, SUN, RAIN]. Assume that:

- a. WEATHER (YEAR = 1970). The parenthesis '()' means a selection operation (represented by O1) done on the entity WEATHER (represented by R1);
- b. WINE {AREA = Napa Valley} (represented by R2) represents a horizontal fragment of the entity WINE;
- c. (WEATHER(YEAR =1970) * (WINE {AREA = Napa Valley})) is a natural join operation (represented by O2) done between the result of a) above and the horizontal fragment WINE{AREA = Napa Valley} of b) above; and
- d. [YEAR, AREA, NAME, PRODUCER, COUNTRY, SUN, RAIN] is a projection operation (represented by O3) done on the result of step c.

The query optimizer decides which operations are required and their sequence. Figure 3-5 depicts a transaction processing schedule for the transaction T2. The choice

of operations and their sequences has been decided arbitrarily and not using a query optimizer.

Figure 3-5: Transaction Number 2 Processing-Schedule Graph

In this figure, the following is assumed:

- a. 'i' is the ith transaction; for example we have the second transaction;
- b. 'f' is the frequency at which this transaction is executed; for example here we have 50 times/month;
- c. 'd' is the amount of data transmitted between sites (Number of bytes). For example, from VS1 to VS2 there is 1000 bytes per transaction occurrence;
- d. 'PhS2' is the physical site 2 and correspond to a real physical site; and
- e. 'VS1' is virtual site 1, a site to be allocated. At VS1 there is R1 which is the entity WEATHER. The operation O1 correspond to a selection operation.

The following paragraphs will illustrate how a processing-schedule graph can vary with two different query optimization strategies. Let suppose we have the following query required at site 1

"Give the name and the year of wines and the hours of sun of areas where the grapes were picked and where more than 1,700 mm of rain fell. "

That query could be translated as follow: ((WEATHER {RAIN > 1,700}) (WINE)) [YEAR, NAME, SUN, AREA]. That query requires all the instances of the entity 'WINE'. Assume that the entity 'WINE' is fragmented as follow:

- a. F1 = WINE {COUNTRY = France};
- b. F2 = WINE {COUNTRY = Italy}; and
- c. F3 = WINE {COUNTRY = USA}.

This query may have many processing schedules for executing it. Two of them will be discussed below:

- a. The first query optimization strategy minimizes the response time. It would give a processing-schedule graph given in figure 3.6. On this graph, the number on the edges is not (i, f, d) any more but rather f times d which represents the number of bytes transmitted between sites per month:

VS1 = ({F4}, { })

VS2 = ({F1}, {O1, O4})

VS3 = ({F2}, {O2, O5})

VS4 = ({F3}, {O3, O6})

Figure 3-6: Query Processing-Schedule Minimizing Response Time.

In this figure fragment F4 is the horizontal fragment WEATHER{RAIN > 1,700}. Operations O1, O2 and O3 are join operations and O4, O5 and O6 are projection operations. The total number of bytes transmitted is $3 \times 18,000 + 400 + 800 + 200 = 55,400$ bytes;

- b. With a second query optimization strategy which minimizes the transmission cost, it would give a processing-schedule graph as given in figure 3-7.

VS1 = ({F4}, {O1, O2, O3, O4})
 VS2 = ({F1}, { })

VS3 = ({F2}, { })
 VS4 = ({F4}, { })

Figure 3-7: Query Processing-Schedule Minimizing Transmission Cost

In this figure, operations O1, O2 and O3 are join operations and O4 is a projection operation. The total number of bytes transmitted is $12,000 + 15,000 + 20,000 + 1,400 = 48,400$ bytes.

The first processing-schedule graph of figure 3-6 would have a lower response time than the one of figure 3-7. This is because in figure 3-6 there are three join operations O1, O2 and O3 done in parallel at VS2, VS3 and VS4 whereas in figure 3-7 there are three join operations done sequentially at VS1. However, the second processing-schedule graph of figure 3-7 has a lower transmission cost (48,400 bytes versus 55,400 bytes) than the first one of figure 3-6.

These two figures (figure 3-6 and 3-7) illustrate how a different query optimization strategy can create different processing-schedules used later on for data allocation. It is essential that the objective of the data allocation strategy comply with the objective of the query processing strategy. For example, the objective of the allocation strategy used on the figure 3.7 should be to minimize the transmission cost, since the objective of the query optimization strategy used to create the figure 3-7 query processing-schedule was also to minimize the transmission cost.

3.6.8 Model 16 Analysis.

1. More Advanced Notions from Model 16:

To understand some of the features covered by this model, the following two notions have to be understood:

- a. **Assignment:** A physical site may have assigned to it a set of virtual sites, this set will be called the assigned set. A virtual site can be assigned to at most one physical site which means that it is added to the assigned set of that physical site; and
- b. **Union:** The union of two nucleus-sites is a nucleus-site whose set of fragments is the union of the sets of fragments of the two nucleus-sites, whose set of

operations is the union of the sets of operations and whose assigned set is the union of the assigned set. The result of an union of two virtual sites is again a virtual site and the union of a virtual site and physical site is that physical site.

The difference between an assignment and an union is that applying the union to an allocation means changing it permanently. The assignment, however allows for changing the allocation back and forth. These two operations make it possible to search through the space of possible allocation by branch-and-bound techniques and heuristic algorithm.

2. Analysis:

The following paragraph will identify the different features covered by this model:

- a. Basic : This model allow the virtual sites to have any kind of fragments, horizontal as well as vertical. With this model, every transaction has to be illustrated by a processing schedule graph before allocation starts. This automatically imply multiple redundant data. Therefore all the basic features of data allocation are covered;
- b. Solution Characteristics : Two different approaches are given, one to minimize the transmission cost and the other one to minimize the response time. A method to derive an optimal as well as a heuristic solution is given for both approaches. Selected test demonstrated that on the average the heuristic solution for the total transmission cost minimization approach is 3% higher than the optimal one. For the response time minimization, selected test demonstrated that if queries reference a number of fragments which is less than the number of sites in the computer network, the heuristic algorithm computes near optimal solution:

- c. **Cost :** As far as the present heuristic solution minimizing the total transmission cost is concerned only the transmission cost is considered. However, since the processing-schedule graph represent operation as well as data at sites, by modifying the present heuristic it is possible to consider I/O and CPU cost as well as storage cost;
- d. **Execution Time:** A queuing model is introduced which considers queuing delays to compute the minimum operation processing time and data transmission time. The average response transmission time is minimized for arbitrary processing schedules;
- e. **Query Optimization:** In the approach which minimizes the transmission cost, a heuristic is given which starts with a global processing schedule graph composed of individual processing schedule graph and from there applies the different steps of allocation. During the overall allocation process none of the processing schedule graphs are recomputed using a query optimizer, therefore we have static query optimization.

This model treats dynamic query optimization as follows. Processing-schedule graph explained in figure 3-5 is replaced by a new graph called the LINK-graph. In this graph each edge has a label which represents the cost saving when the two associated nodes of this edge are united (For a pair of Virtual Sites) or the cost saving when one node is assigned to another (For a Virtual Site assigned to a Physical Site). To compute the label on the edge, the query processing algorithm is applied twice, once when the two nodes are united/assigned (For a pair of Virtual Sites or Virtual/Physical Site respectively) and once when they are not. The difference between these two costs goes on the labels of these edges. This computation is done once at the beginning.

After completion of the above initial step, the LINK-graph is created where the labels of each arc represents the cost saving when the arc's two associated nodes are united. The next steps are as follows:

- i. Select the arc with the highest cost saving;
- ii. Unite its associated nodes;
- iii. Using the query optimizer, recompute the cost saving on the arcs that are connected to the new node that represents the union of the nodes associated with the step i selected arc; and
- iv. Repeat step i to iii until no arc exists with a positive label value. Negative values on the arcs means that no cost saving is made possible by another union of nodes.

Every time step 'iii' is repeated, a query optimizer has to be used. This is why it can be said that the different steps of allocation require dynamic query optimization. The various query optimization strategies (Individual versus multi query optimization) of section 3.4.1 paragraph 5 can be used. For the second approach which minimizes the average response time, only static query optimization is considered.

Since the processing-schedule graph depends of the strategy of the query optimizer utilized, if the query optimizer does multi-query optimization as defined in section 3.4.1 paragraph 5, the result would be reflected in the processing-schedule graph. That model can therefore support multi-query optimization;

- f. Network : There is nothing about topology constraints; therefore it can be applied to general topologies. For a given link the transmission cost is assumed to be proportional to the amount of data transmitted per unit of time. Therefore there

is a fixed transmission cost for every edge. The labels on the edges of the processing-schedule graph represent the amount of data per unit of time to be transmitted between two sites. By considering this information and modifying the heuristic accordingly, channel capacity could be considered as a constraint in the allocation process; and

- g. **Operational Features:** The processing-schedule graph contains the information to know how much data is stored at each site, therefore if the heuristic is modified accordingly, the storage capacity could also be considered.

3.6.9 Subset of Model 16.

Model 16A is a subset of model 16 [46], [47] and will be explained in chapter 4. In order to understand the chapter 4 explanation of that model, the reader could have to refer back to this chapter, section 3.6.7 where the different notions of model 16 are explained.

3.7 Selected Model.

Between the four finalist models, model 14, 14A, 15 & 16, model 16 is the preferred one for the following reasons:

1. It is the model which cover the most features. Model 16 covers 12 features, 11 features for model 15, 9 for model 14A and 8 for model 14;
2. Model 14 and 14A rely heavily on the assumption of locality;
3. If heuristics employed in model 16 are appropriately modified, it has the potential of covering four extra features which are storage cost, I/O and CPU cost, channel capacity as well as storage capacity;
4. It is the only model which cover the two key group of features which are costs and execution time. Unlike the other models, it considers query optimization; and

5. It is a graphical model which lets the designer follow the progression of the chosen allocation algorithm to help compare the result of different approaches (Response time minimization versus transmission cost minimization) different heuristics or different inputs. It is therefore a very good choice for adoption in the prototype CASE tool.

3.8 Summary.

In this chapter the different steps of distribution design have been discussed. Then an analysis of the different data allocation models has been made by comparing their different allocation features to finally select model 16. In the following chapter, a subset of model 16 i.e. model 16A will be employed for the development of a CASE tool. That model is first described in detail, extended appropriately and then used for the design and implementation of a prototype CASE tool.

Chapter 4

CASE Tool Design Description.

4.1 Overview.

This thesis will limit itself to the design and implementation of a prototype CASE tool previously classified as a Toolkit. The prototype Case tool is classified as a Toolkit since it automates only the data allocation part of the data distribution design. For now on, this prototype will be called CDT which stands for a Carleton University Distributed Database Design Toolkit.

The next section presents the functionalities that should be provided by a CASE tool in order to automate data distribution design. Section 4.3 describes in detail the heuristic supported by the CDT, i.e. the model 16A heuristic. Section 4.4 describes the modifications made to this heuristic to improve its optimality, i.e. to bring the solution closer to the global optimum and also to be able to apply it to practical problems. Section 4.5 describes the CDT general requirements. Section 4.6 gives an overview of ARTTisan, a CASE tool used to support the CDT implementation. Finally, section 4.7 describes the CDT design itself. The CDT implementation consists of about 3,000 lines of Objective-C source code.

4.2 Architecture of CASE Tool for Data Distribution Design.

This section will describe the different functions that a CASE tool should support in order to cover the different steps of the top-down approach for data distribution. As in section 3.2 this thesis will limit itself to the first three steps of the top-down approach

i.e. the fragmentation, the data allocation and the data dictionary allocation. Figure 4-1 proposes an architecture for a CASE tool in order to support the three first steps of data distribution. As for figure 2-2, figure 4-1 shows the different function modules (i.e. Fragment Designer, Fragment Allocator etc...) with their respective components (i.e. Graphic Interface Tool, Diagramming Tool etc...)

Figure 4—1: Architecture of CASE Tool for Data Distribution Design.

Since the data distribution is part of the design phase, the data distribution design tool when added to a workbench would have to be integrated into the design tool module of figure 2-2. The central repository could be the same one as the one of figure 2-1

with modifications made to manage extra data which are peculiar to the data distribution design such as fragment description, fragment allocations description etc...

In figure 4-1 an extra module to cover the query optimization has been added to the three functions/steps of the top down approach. This has been done since some allocation models such as model 16 at chapter three require query optimization. Besides, query optimization is a complex issue and should be treated individually (There exists so many different query optimization strategies). If a model does not require query optimization, then figure 4-1 architecture allows to go from the fragment allocator to the data dictionary allocator. The proposed modular architecture allows the use of different models/approaches when going from one data distribution step to another [79].

To design and implement a CASE tool such as figure 4-1 which cover the different data distribution design steps, which consists of all the different applicable CASE tool component (From graphic interface tool to central repository) and which can also support many data distribution design methodologies is too much for a one year thesis. It was therefore decided to limit this thesis to the design and implementation of a CDT which would be a fragment allocator as per figure 4-1, supported by graphics interface tool, diagramming tool and limited design checker and analyzer tool. CDT also supports a subset of model 16 selected at chapter 3 i.e. model 16A which will be described in detail within the next section.

4.3 Supported Data Allocation Heuristic.

The CASE tool allocates data using the heuristic called "total-data-allocation" of model 16 reviewed in chapter 3. This greedy heuristic covers the allocation features of model 16A described in the next sub-section. The pseudo-code of figure 4-2 illustrates the different

execution steps of the modified heuristic 'total-data-allocation'. In the 4.3.2 sub-section an example will be introduced and will be used at sub-section 4.3.3 to demonstrate the different execution steps of the modified heuristic 'total-data-allocation'.

4.3.1 Limitations of Supported Heuristic.

As can be seen at figure 3–3, the heuristic of model 16A covers all the basic features, i.e. multiple and redundant data as well as vertical and horizontal fragments. It allocates data in order to minimize the communication cost. On the query optimization side, it covers the features of static and multi-query optimization. It can also be applied to any general network topologies.

As previously mentioned, selected tests demonstrated that on the average the final cost associated with using the heuristic solution for this model has a margin of 3 % higher than the optimal one [46]. With the first heuristic modification described in section 4.4, this margin is probably smaller. Because of the second modification described in section 4.4, this heuristic can be applied to any practical problem where the data is either used by updates or queries, by queries only or by updates only.

Line #

```

1  proc    total-data-allocation
2  begin
3      Build_Set_P, Final_Set_P, Previous_Cost, Cost;
4      boolean    goon := true;
5      for 1 to n; /* Where n represents the number of different Virtual Sites. */
6      assign VSi to PhSk with LINKik is maximum;
7      end_for;
8      while goon
9      Build_Set_P := set of pairs of virtual sites that are not yet united;
10     goon := false;
11     while    Build_Set_P not = { } and not goon;
12     do
13         take (VSi, VSj) from Build_Set_P such that LINKij is maximum;
14         add (VSi, VSj) to Final_Set_P;
15     end_do;
16     Previous_Cost := 0;
17     first (VSi, VSj) := first element (VSi, VSj) from Final_Set_P;
18     next (VSi, VSj) := first (VSi, VSj);

    /* Calculate All Possible Union Cost */

19     while    next (VSi, VSj) still exists;
20     Cost :=  $\max_k LINK_{ik} + \max_k LINK_{jk} - (LINK_{ij} + \max_k LINK_{uk})$ ;
21     If    Cost < Previous_Cost;
22     then
23         Remembered (VSi, VSj) := next (VSi, VSj);
24         Previous_Cost := Cost;
25     end_if;
26     next (VSi, VSj) := next element (VSi, VSj) from Final_Set_P;
27     end_while;

    /* Unite Pair of Virtual Sites */

28     take VSi and VSj from remembered (VSi, VSj);
29     VSu := union of VSi and VSj;
30     remove VSi and VSj from processing-schedule graph;
31     add VSu to processing-schedule graph and recompute its LINK-values;
32     goon := true;
33     end_while;
34 end_while;
35 Do final union of virtual sites;
36 end.

```

Figure 4—2: Modified Heuristic 'total-data-allocation'.

4.3.2 Model 16A Example Description.

To illustrate how the heuristic works it will be applied to an example. This example consists of the two entities "WINE" and "WEATHER" of chapter 3, model 16 and the following two queries Q1 and Q2 as well as the following two updates U1 and U2 described below:

1. Q1: The following query is required at physical site 1 (P1):

"From all the previous years and all the different possible areas, give the name of the wine, its area, its producer, its country and its weather characteristics such as number of hours of sun and number of millimeter of rain"

That query could be translated as follow: (WEATHER(YEAR=YEAR and AREA=AREA)WINE)[YEAR, AREA, NAME, PRODUCER, COUNTRY, SUN, RAIN]. Where:

- a. WEATHER(YEAR =YEAR and AREA=AREA) the parenthesis '()' means a selection operation 'O1' done on the entity WEATHER identified by 'R1';
- b. WINE represents all the instances of the entity WINE;
- c. (WEATHER(YEAR=YEAR and AREA=AREA)WINE) is a join operation 'O2' done between the result of 'a' and 'b' above; and
- d. [YEAR, AREA, NAME, PRODUCER, COUNTRY, SUN, RAIN] is a projection operation 'O3' done on the results of step 'c'.

2. Q2: The following query is required at physical site 2 (P2):

"From all the previous years and from AREA=Napa Valley, give the name of the wine, its year, its area, its producer, its country and its weather characteristics such as number of hours of sun and number of millimeter of rain"

That query could be translated as follow: (WEATHER (YEAR = YEAR) (WINE { AREA = NapaValley })) [YEAR, AREA, NAME, PRODUCER, COUNTRY, SUN, RAIN]. Where:

- a. WEATHER(YEAR=YEAR) the parenthesis '()' means a selection operation 'O4' done on the entity WEATHER identified by 'R1';
- b. WINE {AREA = Napa Valley} identified by 'R2' represents an horizontal fragment of the entity WINE and WINE{AREA not= Napa Valley} is identified by 'R3';
- c. (WEATHER(YEAR=YEAR) (WINE{AREA = Napa Valley})) is a join operation 'O5' done between the result of 'a' and 'b' above; and
- d. [YEAR, AREA, NAME, PRODUCER, COUNTRY, SUN, RAIN] is a projection operation 'O6' done on the result of step 'c'.

3. U1: This update is generated from physical site 2 (P2):

“Add new wines to WINE from Napa Valley.”

This update would update all the instances of the entity WINE having the attribute AREA=Napa Valley.

4. U2: This update is generated from physical site 3 (P3):

“Add information about the weather.”

This update would update all the instances of the entity WEATHER.

4.3.3 Model 16A Execution Steps Description.

The 4.3.1 example will now be used to demonstrate the different execution steps of the modified heuristic 'total-data-allocation' as illustrated at figure 4–2. The different

notions (such as virtual site assignment, union etc...) introduced at chapter 3, section 3.6.7 and 3.6.8 that apply to model 16 also apply to model 16A heuristic. These notions will be used during the demonstration of the different execution steps of the modified heuristic.

Before executing the different steps of the heuristic 'total-data-allocation' a Global Processing-Schedule Graph has to be created. A Global Processing-Schedule Graph is a graph where for each query and update, a processing-schedule graph is created (as per figure 3-5). Then the processing-schedule graph of each query and update are merged together into one graph called the global processing-schedule graph.

Figure 4-3 represents a global processing-schedule graph for the 4.3.1 sub-section transactions. For query Q1, the virtual site V3 contains entity WEATHER identified by R1 and a selection operation O1. Virtual site V1 contains horizontal fragment WINE {AREA not = Napa Valley} identified by R3, a join operation O2 done between R1 and R3 and a projection operation O3 done on the results. Virtual site V2 contains horizontal fragment WINE{AREA = Napa Valley} identified by R2, a join operation O4 done between R1 and R2 and a projection operation O5 done on the results. For query Q2 the virtual site V5 contains entity WEATHER identified by R1 and a selection operation O6. Virtual site V4 contains horizontal fragment WINE{AREA = Napa Valley} identified by R2, a join operation O7 done between R1 and R2, a projection operation O8 done on the results.

For the update U1, the virtual site V6 contains the horizontal fragment WINE{AREA = Napa Valley} identified by R2. For the update U2, the virtual site V7 contains the entity WEATHER. As in figure 3-6 the number on the transmission edge represents 'f' times 'd' where 'f' is the frequency of the transaction and 'd' is the amount of data per transaction. This value of 'f' times 'd' represents the transmission cost.

Figure 4—3: Global Processing-Schedule Graph.

The first arc which connects the transmission edges between V2-V6 and V4-V6 and the second arc which connect the transmission edges between V3-V7 and V5-V7 are called forking arcs. The forking arcs are there to note to the heuristic that these transmission edges belong to the same update.

The following are the major execution steps of the modified heuristic 'total-data-allocation' :

1. Initial Virtual Sites Assignment: The forking arcs are first deleted and then the virtual sites that are directly connected to physical site(s) by transmission edge(s) are assigned to the physical site where the transmission cost in between the virtual

2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

site and the physical site is the highest. If a virtual site is not directly connected to a physical site, then that virtual site can be assigned to any physical site. This step corresponds to line 5, 6 & 7 of figure 4-2. Figure 4-4 gives the results after the initial assignment of virtual sites.

Figure 4—4: Initial Virtual Sites Assignment.

2. **Build Set P of Virtual Sites to be United:** This step creates a set P of pair of virtual sites that can be united. Virtual sites that can be united are pairs of virtual sites connected by a transmission edge or virtual sites which are part of the same update. The set P containing all possible pairs of virtual sites to be united is given in figure 4-5. The variable 'LINK_{ij}' represents a transmission cost directly

saved by uniting a pair of virtual sites. This step corresponds to line 9 of figure 4-2. For example, if in figure 4-3 V1 and V3 are united the transmission cost of 20 in between them would disappear. If V2 and V4 are united (this is possible since they are part of the same update, they are fragment of the same entity), one of the two transmission cost of 5 between V6 and the resulting union would disappear.

(Vi, Vj)	LINKij	(Vi, Vj)	LINKij
(V2, V3)	20	(V3, V5)	15
(V1, V3)	20	(V3, V7)	15
(V5, V7)	15	(V4, V5)	8
(V2, V4)	5	(V4, V6)	5
(V2, V6)	5		

Figure 4—5: Set P of Virtual Sites to Unite.

3. Final Set P of Virtual Site Pair for Union: The final set P of virtual site pair to be kept for union is the pair of virtual sites which has the highest LINKij value. At figure 4-5 there are two possible candidates which are (V1, V3) and (V2, V3). The original algorithm "total-data-allocation" keep one of them only arbitrarily; however a modification has been made to the heuristic so that now all the pairs of virtual sites having the highest LINKij values are remembered. Therefore, a final set F of virtual sites to be evaluated for union has to be created e.i. {(V1,V3), (V2,V3)}. This final set F is due to the modification made to the heuristic. This step corresponds to line 12 to 15 of figure 4-2.

4. **Calculate All Possible Union Costs:** For each pair of virtual sites belonging to the final set P , the total cost of union is calculated. This step corresponds to line 16 to 27 of figure 4-2. Let us show how this cost of union is calculated on the pair of virtual sites ($V1$, $V3$).
 - a. Calculate the cost increase by removing the virtual sites from their assigned physical sites. On the pair ($V1$, $V3$) this increase the transmission cost by $5+0 = 5$. The first term corresponds to the cost increase occurred by the removal of $V1$ from $P1$. The first term corresponds to ' $\max_k \text{LINK}_{ik}$ ' term of line 20 of figure 4-2. This cost is 5 since there is a direct transmission of 5 between $V1$ and $P1$ (See figure 4-3). The second term is 0 since there is no direct transmission between $V3$ and $P1$ (See figure 4-3). This second term corresponds to ' $\max_k \text{LINK}_{jk}$ ' term of line 20 of figure 4-2. Figure 4-6 illustrates the increase in transmission cost by removing $V1$ and $V3$ from physical site 1.

Figure 4—6: Cost Increase Incurred by Removal of V1 and V3 from P1.

- b. Calculate the cost saving by uniting the different virtual sites and assigning the union to the physical site having the highest transmission cost to this resulting union. By uniting V1 & V3 the transmission cost of 20 in between them will disappear, therefore there is a cost saving of 20. This term corresponds to 'LINK_{ij}' term of line 20 of figure 4-2. Figure 4-7 illustrates the decrease in transmission cost by uniting V1 and V3. This term corresponds to ' $\max_k \text{LINK}_{uk}$ ' term of line 20 of figure 4-2. By assigning the resulting union to physical site 1 there is a cost saving of 5. Figure 4-8 illustrates the decrease in transmission cost by assigning the union of V1 and

V3 to physical site 1. On the pair (V1, V3) the total transmission cost is therefore decreased by $(-20) + (-5) = -25$.

Figure 4—7: Decrease in Cost Incurred by Union of V1 and V3.

Figure 4—8: Decrease in Cost Incurred
by Assigning Union of V1 and V3 to P1.

- c. Calculate the total cost by adding the cost increase (a) and the cost saving (b) previously calculated. If the result is negative, there is a cost saving; therefore keep this pair of virtual sites as an option. For our example the cost saving is $5 - 25 = -20$. This total cost corresponds to the 'Cost' term of line 20 of figure 4-2.
- d. Repeat steps a, b and c on each pair of virtual sites member of the final set P. Remember only the pair of virtual sites which results in the highest cost saving. If there is more than one pair with the highest cost saving remember

any of them. This step corresponds to the 'while' loop of line 19 to 27 of figure 4-2. For the example, the union of any of the two pairs of virtual sites (V1, V3), (V2, V3) give a cost saving of 20 for each pair. Any of the two pairs could be chosen; here the pair (V1, V3) will be the chosen one.

5. **Unite Pair of Virtual Sites:** The pair chosen in step 4.d is united. Here the pair of virtual sites (V1, V3) are united and the results are shown in figure 4-8. As a result of the union, the new transmission cost on the edges also has to be recomputed. This step corresponds to line 28 to 31 of figure 4-2. For example, when V_{u13} and V₂ of figure 4-8 are united, the transmission cost between the united virtual sites and the physical sites is the sum of the transmission costs between each virtual sites part of the union and the physical sites. The transmission cost of the union of V_{u13} and V₂ to the physical site 1 is $5+2 = 7$. The union is assigned to the physical site having the highest transmission cost. The same physical site then the one considered at step 4.b.
6. **Repeat step 2 to 5 with the new union of virtual sites just created at step 5.** Iterate until no further cost saving is possible by uniting any of the pairs of virtual sites of set P. This step corresponds to the 'while' loop of line 8 to 34 of figure 4-2. Figure 4-9 illustrates the second last step of the modified heuristic 'total-data-allocation'. It illustrates the final allocation of virtual sites from V1 to V7. For the example, after having applied step 2 to 5 many times, the optimal way to allocate the virtual sites is to put V4 and V6 at physical site 2 and V1, V2, V3, V5 and V7 at physical site 3. The total transmission cost of the system is $7 + 13 = 20$.

Figure 4—9: Final Allocation of Virtual Sites.

7. **Final Union of Virtual Sites:** This final step unites any isolated virtual site to any other virtual site assigned to the same physical site. Further description of this step is given in the next sub-section. This step corresponds to line 35 of figure 4-2.

4.4 Modifications Made to Supported Heuristic.

As previously mentioned different modifications have been made to the heuristic 'total-data-allocation'. These different modifications will be explained in the following subsections.

4.4.1 First Modification.

Let us first explain how model 16A was modified so that the optimality of the final solution is improved. In section 4.3.2, step 4 of the heuristic 'total-data-allocation' it is mentioned that the heuristic is modified to calculate the possible union cost of all pairs of virtual sites having the highest LINK_{ij} value. This is done instead of calculating the cost for only one pair of virtual sites having the highest LINK_{ij} value arbitrarily chosen. This modification corresponds to line 16 to 27 of figure 4-2. With this modification of all the calculated cost only the one which decrease the total cost the most is kept. This modified heuristic compared to the original one will more often give an optimal allocation for the entire system. To illustrate this statement, the following example will be given.

During the application of the heuristic 'total-data-allocation' to figure 4-3, the last union before the final result (final allocation) gives something like figure 4-10. At that stage, the set P of virtual sites to be united would give the figure 4-11 choice. Where Vu12357 is the virtual site resulting from the union of the different virtual sites of V1, V2, V3, V5 and V7.

Figure 4—10: Last Union before Final Allocation.

(V_i, V_j)	LINK _{ij}
$(Vu12357, V4)$	8
$(V4, V6)$	5
$(Vu12357, V6)$	5

Figure 4—11: Set P of Virtual Sites to Unite.

With that choice, the heuristic would first consider uniting V4 and Vu12357 since it has the highest LINK_{ij} value of 8, however uniting them would increase the total

transmission cost by $12 + 15 - (8 + 15) = 4$, therefore that union would not be selected. Then there are two possibilities to consider, the union of (V4, V6) and the union of (Vu12357, V6). Uniting V4 with V6 as per figure 4-12 would decrease the total transmission cost by $12 + 4 - (5 + 16) = -5$. Uniting V6 with Vu12357 as per figure 4-13 would decrease the total transmission cost by $15 + 4 - (5 + 15) = -1$.

Figure 4-12: Union of V4 with V6.

Figure 4—13: Union of V6 with Vu12357.

As it is the original heuristic 'total-data-allocation' could choose arbitrarily to unite(V4, V6) or (Vu12357, V6). If the union of (Vu12357, V6) is chosen then there is no optimum. With the modified heuristic, the two possibilities would be evaluated and as per figure 4-9 only the union of (V4, V6) would be chosen. The modified heuristic therefore improves the final allocation.

4.4.2 Second Modification.

The second modification has been made to apply the heuristic to practical problems. This modification corresponds to line 35 of figure 4-2. The situation at figure 4-14

illustrates when the fragment of a virtual site (such as V8) is only part of an update and is not part of any query. This situation is only treated partially with the original heuristic 'total-data-allocation'. Actually if V8 is updated from more than one site i.e. is connected to different transmission edges, then step 2 of that heuristic assigns the virtual site to the physical site connected with the highest transmission edge cost. Therefore as far as fragment allocation is concerned, the original heuristic considers the above situation. However, an interesting feature of this heuristic is that at the final allocation stage, the total amount of data to be transmitted between the different physical sites is known. For example, in figure 4-9 it can be seen that the total amount of data to be transmitted between P1 and P3 is 7 or between P3 and P2 is 13.

Figure 4—14: Initial Allocation Graph with Virtual Site 'V8' to be Updated Only.

If in a processing-schedule graph an updated virtual site is not connected to any other virtual site (such as V8 figure 4-14), then the original heuristic will never unite that updated virtual site to any other virtual site. If there is no union, the transmission edges from the updated virtual sites to foreign physical sites will not be adjusted as can be seen at figure 4-15. Therefore, for the situation described above from the final allocation it is not possible to read automatically the total amount of data transmitted between the different physical sites. For example, at figure 4-15 the sum of data transmitted between sites P2 and P3 is $13 + 15 = 28$. In this sum 13 is the amount of data transmitted between the union of virtual sites V4, V6 at P2 and the union of virtual sites V1, V2, V3, V5 and V7 at P3 and 15 is the amount of data between V8 at P3 and P2.

Figure 4-15: Final Allocation with Original Heuristic.

The original heuristic has been modified so that after step 7 which gives the final allocation, an extra step is added called 'final-Union-of-Virtual-Sites' (See appendix B for more details). This extra step unites any isolated virtual site to any other virtual site assigned to the same physical site. For example, at figure 4-15 the isolated virtual site V8 is united to the union of virtual sites V1, V2, V3, V5 and V7. At the time of union, the transmission edges to different physical sites are updated. At figure 4-16 there is an addition of $13 + 15 = 28$ done on the transmission edges between P3 and P2. With that last step, the total transmission cost of 28 between P2 and P3 can be read directly from figure 4-16.

Figure 4—16: Final Allocation with Modified Heuristic.

4.4.3 Third Modification.

The last modification was made to the way the virtual sites are united. During a union, the original heuristic suggests the creation of a new virtual site which contains the sum of the two virtual site components (Operations and Fragments) to be united. All the transmission edges of the two virtual sites to be united have to be connected to that newly created virtual sites and the labels (transmission cost) on them have to be adjusted accordingly. Once that new virtual site is created, the two original virtual sites have to be deleted from the processing-schedule graph.

The graphical union of virtual sites has been modified. This modification has been done at line 31 of figure 4-2. The modified union put graphically the united virtual sites on top of each other. There is no new virtual site created and the united virtual sites are not deleted. With the 'adjust-Trans-Edge-Values' method described in appendix B part III, the transmission edges still gets adjusted accordingly. With this modification it is somewhat more complex to adjust the transmission edges than with the original approach. However, the complexity of creating new sites as unions of existing sites as well as connecting this new site to existing transmission edges is gone. With this modified union, the user can find out the members of an union by using a mouse to move the virtual sites piled on top of each other. This last modification was done to ease the implementation of the model 16A heuristic in its supporting environment (ARTTisan's library of objects). Besides, it does not interfere with the way the heuristic 'total-data-allocation' is executed.

Now that model 16A modified algorithm has been described in detail, the next section will describe the CDT design and implementation with its supporting environment.

4.5 CDT General Requirements.

CDT requires as a minimum the following features. The tool will permit users to create and manipulate two dimensional squares, arcs and splines on a specific view. Using a graphical input device such as a mouse, the user should be able to move the created objects. The user will also have control over steps taken by the allocation algorithm which will itself move the created objects. This will permit the user to obtain a printout of each step as well as the initial view of the allocation algorithm.

The CDT development has been done using the objective-C language (See ref [87] for details) and also a library of objects borrowed from an existing CASE tool called ARTTisan. The ARTTisan CASE tool is described in the next section.

4.6 An Overview of ARTTisan.

ARTTisan is a CASE tool being developed at the Department of Systems and Computer Engineering under the auspices of the Advanced Real Time Toolset (ARTT) project. Its environment is based on a network of SUN workstations running Berkeley UNIX.

ARTTisan attempts to support the design needs of programmer developing real-time software. It presents the developer with a hierarchy of notations, each of which plays a different role in the design process. In that hierarchy the initial layer is highly abstract with each of the additional layers becoming more detailed. Symbols/notations in one layer can be related to other symbols/notations in other layer. For example, a procedure within the structure model may be further described by a state machine model.

The user interface for the CASE tool is graphical offering direct manipulation. Icons representing the symbols of the design notations are manipulated by the designer using a

pointing device (mouse) and keyboard. In its present form, the toolset is written entirely in Objective-C using SunView to handle the graphical interface. Although the toolset itself is implemented using an object-oriented language, it should be noted that the design notations supported by the toolset do not support the object-oriented methodology.

4.7 CDT Implementation Design.

A detailed description of the CDT design is given in the three parts of appendix B. The first two parts describe the objects used and created to support the user interfaces. These objects fulfill the functions of the Graphic Interface, Diagramming and Analyser tool included in the CASE tool for data distribution design illustrated at figure 4-1. The first part gives a hierarchical illustration of the seven classes of objects used from the ARTTisan library of objects as well as a brief description of them.

The second part describes the 14 classes of objects created for user interfaces requirements and also to interface with ARTTisan. These different classes of created objects consist of about 800 lines of source code. First the created classes and their inheritance is described. Then the attributes (parameters) of each classes is identified. Finally the created/modified method of each classes are described.

The third part describes 23 methods of the DemoMode class which directly support the algorithm 'total-data-allocation'. The three other DemoMode methods have been created to interface with ARTTisan or with the user. These 23 methods consist of about 2000 lines of code. This DemoMode class can handle 600 different virtual sites, 150 different Forking arcs, 1000 different transaction edges and 25 transaction edges having the same value.

Through the DemoMode, the CDT offers the following two different options to execute the modified heuristic 'total-data-allocation':

1. **Step-by-Step:** This first option allows the user to see the modified heuristic progress one step at a time through the different steps explained in section 4.3 and 4.4. With this option the user has to click the mouse to allow the heuristic to progress one step further and this has to be done up to the end of the heuristic execution. For the user, this option is time consuming, especially for real applications which involves hundreds of fragments which represent numerous unions of virtual sites.
2. **Non-Stop:** This second option, once initiated by the user, goes through all the different steps of the modified heuristic "total-data-allocation" without any stop. This option considerably decreases the execution time of the heuristic since the user does not have to click the mouse after the execution of each steps of the heuristic. This option is very useful, especially for real applications which involves many virtual sites. However, a disadvantage of this option is that the user can not follow the progression of the heuristic.

It is important to note that the second option (Non-Stop option) can also be selected by the user after the execution of any step of the first option (Step-by-Step option). For example, suppose the heuristic is being executed through the step-by-step option. After the execution of the first step which corresponds to the deletion of Forking Arcs the user can still choose the 'Step-by-Step' option which will bring the heuristic execution to the second step or he can choose the 'Non-Stop' option which will bring him to the end of the execution of the heuristic. Appendix B part III illustrates and describes the

sequence of execution of the different DemoMode methods required by option one and two respectively.

4.8 Summary.

This chapter gave first a detailed description of the modified heuristic (model 16A modified) supported by the prototype CASE tool. Then using an object-oriented approach, a summary of the design and implementation of the prototype CASE tool was given (See appendix B for a detailed design description).

In the following chapter, an approach for data distribution will be proposed and applied to a practical problem. This approach covers the application of the modified heuristic. Then the prototype CASE tool will also be applied to that same problem to show how it performs when applied to real size problems and how it can support the proposed approach.

Chapter 5

A Proposed Approach for Data Distribution Design.

5.1 Overview.

This chapter will propose an approach to data distribution during the system design phase. The approach applies at step 21 of IEM described at section 2.5. It can be classified as a TOP DOWN approach and is split into two main parts. The first part is inspired by the data distribution analysis step (21) of IEM. This part is described in section 5.2 and consists of a qualitative approach giving general guidelines and different issues to consider to decide to distribute or centralize the data. Note that some of the centralization versus decentralization issues could also be considered during the previous planning and analysis phases of the software life cycle.

The second part is a fine grained quantitative approach which is applied only if the data has to be distributed. It provides specific guidelines and algorithms to follow for the database fragmentation and for the allocation of the fragments to the different sites. This quantitative approach is explained in section 5.3. Section 5.4 describes the application of the CDT to the appendix C Aerospace Maintenance Information System projects.

As previously mentioned, the proposed data distribution design approach is not geared to enhance IEM only with data distribution design capabilities. The first part i.e. the qualitative approach could be used without constraints to enhance any information system methodologies which support objects of various types as well as relations. However, the second part, i.e. the quantitative approach, could be used to enhance information system methodologies which support objects and relations (e.g. Object Oriented Database) if

and only if the same issues as those supported by the quantitative approach (e.g. Cost and Technical Issues) is optimized, and if the transactions can be specified as steps 1 and 3 of the quantitative approach. Besides, in order to support the feature of query optimization, it would be necessary that query optimization strategies exist to optimize the transactions made with these objects and relations. These different issues as well as object oriented database management systems are currently subject to further research work.

The next subsections will describe each steps of the proposed TOP DOWN approach for data distribution. It will also show where in the data distribution phase can the CDT be useful to the designer. In appendix C, each steps of the proposed approach is applied to the same Aerospace Maintenance Information System project than the one used at appendix A to demonstrate the different steps of IEM at section 2.5. The relational data model produced at step 16 of IEM is used as a starting point to apply the different steps of the data distribution approach. This is done in order to illustrate how the different data distribution steps and the model selected in chapter 3 can be applied to a practical problem. It also demonstrates how the CDT performs when applied to real size problems.

5.2 Qualitative Approach.

As previously mentioned, the objective of this thesis is to treat practical aspects of data distribution; this cannot be done independently of other factors/issues. The next five steps will give an overview of the different issues/factors that the management has to consider to decide if the data should be distributed or not.

Actually these five steps can be applied either to the overall data model or to a subset of it i.e. the subject database. Appendix C applies these steps to a subject database determined at Appendix A. The first three steps of the qualitative approach are borrowed

from IEM. The last two steps are an amalgamation of different issues to consider in order to decide if a database should be distributed or centralized.

5.2.1 Processes versus Subject Databases.

The first step is to develop a process/subject database association matrix. The (i,j) th element of this matrix indicates that the i th subject database is used by the j th process. An illustration of this matrix is given in appendix C.

5.2.2 Processes versus Locations.

This step lists user locations together with the processes that are employed at those locations. It permits a reevaluation of the present distribution of the processes which can be accomplished using a technique developed in [81] which uses factor tables, described below.

For a given process there can be a number of possibly conflicting factors. The factor table shows these at a glance. Management must determine which factors are relevant to centralization or decentralization. Two examples of factors favouring decentralization are a) the different queries/processes originating from different sites require much local knowledge, b) the different queries/processes originating from various sites change frequently due to local causes. More details on these different factors are given in [34], [81]. Figure 5-1 shows an example of a factor table.

Factors	Processes			
	P r o c e s s 1	P r o c e s s 2	...	P r o c e s s N
Factor 'a'	d			D
Factor 'b'		C		d
Factor 'c'	c			

Figure 5—1: Factor Table

In this figure, the (i,j)th elements of this matrix can assume the following values:

1. C = Strong reason for centralization.
2. c = Weak reason for centralization.
3. D = Strong reason for decentralization.
4. d = Weak reason for decentralization.

For example, since process N of figure 5-1 has factors with reasons for decentralization, this process should be decentralized i.e. replicated at multiple sites. Therefore this table could be used to decide if the processes are distributed in an acceptable way and if not appropriate changes can be made. The qualitative approach part of appendix C also gives an example of a factor table.

5.2.3 Subject Database versus Process/Location.

This step determines a rough estimate of the query volumes between process/locations and subject database and specifies whether the queries are interactive or batch. In a batch query the information can be hours old whereas in an interactive query the information can be seconds old. The location of the processes has been determined at step 2. As per figure 5-2 an example of a Subject Database versus Process/Location table is given.

Locations&Processes	Subject Databases		
	Subject DB 'X'	Subject DB 'Y'	Subject DB 'Z'
Location 'X'			
Process 'a'		1000B	
Process 'b'		800B	
Process 'c'	600I		200B
Location 'Z'			
Process 'a'		100B	
Process 'b'			400I

Figure 5—2: Subject Database versus Location/Process Table.

In this figure, the (i,j)th element of this matrix are composed of a number first which correspond to the process' query frequency and is followed by a letter which can assume the following values:

1. I = Interactive Query.
2. B = Batch Query.

For example, the “600I” in column subject database “X” and row process “c” of figure 5–2, means that process “c” has interactive queries which occur 600 times per unit of time. Let suppose here that it is 600 times/month. When these queries occur, data from subject database “X” is required. The qualitative approach part of appendix C shows a subject database versus Location/Process table for the Aerospace Maintenance Information System project.

5.2.4 Effect of Queries and Update Frequencies on Data Distribution.

Using the information given in figure 5–2 on queries and considering the frequency of updates, this step evaluates the effect that the update frequency compared to the query frequency has on data distribution. This is also done considering that the information required by the query is batch or interactive and that the updates and queries are generated from one or more sites. This step applies to long haul networks but not to local area networks.

For each subject database, the frequency of the queries can be taken from figure 5–2. The frequency of updates per subject database can be computed by adding the frequency of updates for each entity composing that subject database. Then using figure 5–3 as a guideline, management has a first idea on the need either to distribute or centralize a subject database.

UPDATES		USERS	
		Users are geographically dispersed.	
Infrequent Updates	Updates from one source	Up-to-the-Second-Info-is-Required.	Info Provided can be hours old.
	Updates from geographically dispersed site	Data can be dispersed with updates distributed via Networks.	Data can be dispersed with updates distributed by batch transmission
Frequent Updates	Updates from one source	Data is centralized or dispersed with updates distributed via a network.	
	Updates from geographically dispersed site	Data is centralized.	Data is centralized or dispersed with updates distributed by batch transmission

Figure 5—3: Effect of Updates on Data Distribution.

In figure 5-3, an 'Infrequent' update implies that the frequency of queries is higher than the frequency of updates. A 'Frequent' update implies the reverse. Using this figure, the user should be aware that there may be no clear cut distinction between 'Frequent' and 'Infrequent' updates. If the update frequency is about the same then the query frequency, the recommendation by this figure to distribute or centralize is not reliable.

Let see how to use figure 5-3. For example, suppose that for a subject database there are infrequent updates; the update as well as the queries come from geographically dispersed sites, and the information provided to the query can be hours old. Use of figure 5-3 shows that the subject database can be distributed with updates distributed by batch transmission. The qualitative approach part of appendix C shows the application of this figure on the different subject database of the Aerospace Maintenance Information System project.

5.2.5 Centralization versus Decentralization Issues.

Many of the important factors in the decision on how to distribute the data are qualitative and not quantitative. J. Martin [14] has noted that "the design of distributed systems cannot be completely reduced to an algorithm".

Besides the update and query frequency, the other issues to consider for centralizing versus decentralizing data can be classified into four distinct categories[39]. They are cost issues, technical issues, application development issues and finally political issues. These will be described briefly to give an intuitive idea of their roles. The qualitative approach part of appendix C shows the considerations of these issues on the different subject database of the Aerospace Maintenance Information System project.

1. Cost Issues: Even if most of the cost issues can be optimized numerically, each cost component should be examined for its effects:
 - a. Transmission Cost: In a long haul network, this cost is an important one. If there is a low update frequency (low compared to the query frequency) from geographically dispersed locations, decentralization will lower transmission cost because of local processing and fewer transmitted messages. This is in accordance with figure 5-3 of the previous sub-section.
 - b. Storage Cost: Centralization decreases storage cost. There exists economies of scale in that the cost per bit is lower with large storage units. Furthermore, distributed data often results in multiple copies of the same data being stored.
 - c. Processor Cost: Decentralization often decreases CPU cost since the cost per instruction may be much lower on microcomputers than on minicomputers and even more on mainframes.

- d. **Database Software and Application Program Cost:** Centralization should decrease software cost. If the software is developed by an outside organization, the customer could have to pay for the extra copies of the software. If the software is developed by the organization itself, the development cost has to be paid only once, the other redundant copies are copies of the master copy. If not well controlled, multiple decentralized system may result in a major duplication of programming efforts. Therefore, an organization will have to be created to ensure configuration control of the different copies of the software.
 - e. **Installation & Maintenance Cost:** Decentralization probably decreases installation cost since most mini and microcomputers do not need air conditioning, false floors or other special building facilities associated with larger centralized computers. Furthermore, most minicomputers do not need highly trained operators and staff to run the installation. On the other hand, decentralization means multiple sites with mini and microcomputers.
 - f. **Execution Time:** As mentioned at chapter 3, this issue can be considered as a cost function. Decentralization has the potential to decrease the response time. If the data is well allocated and if the update rate from remote sites is not too high, local systems can have faster execution time because of smaller workloads and less need for data transmission. Very often the volume of work in most corporation is too great for centralized systems and hence a distributed system would ease the workload.
2. **Technical Issues:** The following are the general impacts that centralization vs decentralization has on different technical issues:

- a. **Fault Tolerance:** Decentralization improves fault tolerance since it increases availability due to the fact that local systems are less affected by network telecommunication failures. It also increases the reliability since the independent loss of several copies of the same information is unlikely. When a copy of data is corrupted, applications can access alternate copies. Besides, during communication link failures more alternative paths (for nonfixed routing) are usually possible for a decentralized system with redundant data than for a centralized one. This is due to the fact that in a decentralized system, on the average there are more paths between the requesting node and the multiple nodes having the data than for a centralized system where alternative paths are possible between only two nodes, i.e. the requesting node and the central node which stores all the data.
- b. **Security:** Decentralization can play in favor or against data security. It gives the possibility of physically separating sensitive data. Such data could be stored at sites with computers running secure software and with external security procedures [83]. Besides, if the privacy and security of the data is a local responsibility, it makes more sense to let local authorities worry about the security of their own data. However, since in a distributed database data may flow among the various sites, the controls at a specific site become irrelevant if data stored at that site is readily available to other sites that do not have effective security controls. Also, data may be exposed to various types of attack while in transit across the network links.
- c. **Modular Growth:** In a centralized system, the system is upgraded to a newer-faster-larger system as the application demand grows. The older-slower-smaller system is retired. In a distributed system, the system can grow

in increments as the demand grows. The existing hardware is not retired, rather it is augmented with additional hardware. Most organizations find it impossible to predict future demand for the system, so modular growth of hardware is a very attractive feature of distributed systems [82].

- d. **Management of Data:** There are more technical issues (e.g., interference between updating transactions or inconsistent reads etc..) involved in a distributed DBMS than for a centralized one.

- 3. **Application Development Issues:** The different factors affecting the centralization or decentralization of the processes/applications have been considered by management at step 2.

Since the issues of the application development have not been treated, the following are the general impacts that centralization versus decentralization has on them.

- a. **Planning:** Multiple decentralized application developments without coordinated control and data design can lead to chaos. Furthermore, centralization argues for a system being designed with a knowledge of overall organizational needs. An advantage of decentralization is that local groups are close to the real end-user problems and can plan to adapt the system to them. Decentralized planning can offer more application innovation from user groups than from a centralized group.
- b. **Developments:** For development, the advantage of centralization over decentralization is that centralized control of development requires fewer programmers. Furthermore, applications which may run identically in multiple locations should be developed once centrally. A central group usually has skilled data-processing management, professional system analysts and pro-

grammers who are highly trained and have experience in structured design and planning for changes. In contrast, user groups may not have this level of knowledge to handle their own data processing.

In a centralized development, the control group may be insufficiently staffed to respond promptly and efficiently to all the end users' needs. In some major organization the backlog is from three to five years [24]. Besides, to comprehend the real end-user problem is difficult, especially with complex applications. End users themselves are uniquely able to understand their own changing requirements and decentralization argues that they should be as involved as possible in application development.

4. Political Issues: The following are the general impacts that centralization vs decentralization has on these different political issues:
 - a. Organization Structure: Increasingly to reflect their organization structure, decentralized organizations are building systems so that each region division or other organizational unit control the equipment that performs its part of the application [82].
 - b. Bureaucratic Reasons: As centralized data processing organizations grow too large they often become entangled in their own management procedures and become slow to move and change. Increasing size also tends to lead to increasingly formal structure which are usually inflexible.
 - c. Quest for Power: Quite often there is conflict between the existing control data processing department which wants to retain control over all the data and end-user groups who wants local autonomy. Centralized data and facilities can give more power to top management or head office staff. Immense

political power rest largely on whoever controls the data processing facility [84]. On the other hand, local management wants to avoid higher management from having too much information too soon. Complete centralization of data may make probing external authorities too easily. Decentralization avoids putting too much power into too few hands.

Due to technical advances, the different cost issues of processor cost, storage cost, transmission cost and execution time are becoming less significant and therefore relatively less important to optimize. As a result other issues such as fault tolerance and security are becoming more important to consider. In this thesis, fault tolerance and security has not been taken into consideration and have been left for future extension to this research.

Considering these different issues and putting the appropriate weight on each of them, management will eventually come up with decision on either to distribute or to centralize the different subject databases. If the decision is not clear, and if that decision has to consider cost as well as technical issues, then a quantitative approach should be used as an aid. If the final decision is to decentralize then the following quantitative approach can be used to optimize the cost as well as the technical issues. The quantitative approach described in the next sub-section presents different steps to be followed and is based on model 16 described in chapter 3.

All the different steps of the qualitative approach have been applied to the appendix C projects. It was found that no critical issues were for centralization and most of the issues were in favour of decentralization. It was therefore recommended to distribute the different subject databases as per the following quantitative approach.

5.3 Quantitative Approach.

As shown in figure 3-3, in addition to the cost of installation and maintenance (item 'e') database software and application programs (item 'd') the four other costs could be covered by some models and could therefore be optimized by one of them using the quantitative approach. Qualitative issues may or may not be quantified. For example, the technical issues of reliability, availability and number of alternate paths could be increased to improve the fault tolerance by using some of the models given in figure 3-3 and therefore are quantitatives. Furthermore if the security is a local responsibility, model 16A would allow tagging the data at specific sites using the quantitative approach. However, the following two qualitative issues of modular growth and management of data as well as the application development issues are not covered by any of the figure 3-3 models and are therefore not quantifiable.

The quantitative approach supports model 16A and is based on the following five different steps:

1. Transaction Specifications;
2. Fragmentation;
3. Construction of Global Processing Schedule Graph;
4. Data Allocation; and
5. Data Dictionary Allocation.

Since the proposed quantitative approach supports the model 16 application, the proposed approach covers as a minimum the different features covered by model 16 such as described at figure 3-3. As for the qualitative approach, appendix C applies the five steps of this approach to the subject database determined at appendix A.

1. **Transaction(Query/Update) Specifications:** As for step 9a. of IEM (as given in section 2.5), this step requires to follow the 20/80 rule. It is a heuristic which assumes that 80 percent of the query instances are obtained by considering the most frequent 20 percent of the applications [44]. Therefore the frequency of the 20 percent most frequent queries has to be determined, its destination site as well as the entities and the attributes of the entities required by each of these queries. Each query has to be specified as depicted in figure 5—4.

Destination Site: Site 'X'
 Frequency: 50 times/month
 Query Number: 1

Figure 5—4: Query Specifications.

In this figure the entity in the top i.e. Entity 'A' represents the first entity required by

the query 1, the entity at the bottom i.e. Entity 'C' represents the last entity required by the query 1. The edges between entities are the relationships existing between the entities of the original data model. The letters in parenthesis to the right of the entities are the attributes (i.e. a,b,c are the attributes of entity 'A'). An underlined attribute represents a key of that entity (i.e. a is the key of entity 'A'). The size in bytes of each attributes also has to be specified (i.e. because of its size, attribute c requires 15 bytes). That information will be required at step 3 of this quantitative approach.

Besides the queries, the updates also have to be specified. The different sites of origin as well as the frequency of updates have to be specified for each entity. The quantitative approach part of appendix C shows different transaction specifications for the Aerospace Maintenance Information System project.

2. Fragmentation: As mentioned in chapter 3, designing fragments consists in grouping tuples or attributes which have the same properties from the viewpoint of their allocation. Since model 16 offers the possibility of allocating data in order to minimize the transmission, storage or processing cost, for horizontal fragmentation the logical as well as the statistical property of the data as explained at section 3.3.1 could be used. If the data has to be allocated in order to minimize the response time, then as mentioned in section 3.3.1 Apers' approach [46], [47] could be used.

Since the appendix C Aerospace Maintenance Information System project is a long haul network and since the CDT supports only model 16A (Model 16A minimizes the transmission cost) described in chapter 4, the data allocation step of appendix C has been done to minimize the transmission cost. With this objective, the horizontal fragmentation could be done using either the logical or the statistical (affinity)

property of the data. Since no tool was available to support the fragmentation done with the statistical property of the data, no affinity analysis was conducted, and therefore the horizontal fragmentation has been done using the logical property of the data.

In that particular project, groups of aircraft of a certain type are located at certain sites. Therefore in order to localize data (minimize transmission cost), all the entities of a specific subject database (Aircraft subject database) were fragmented using that property. All the entities had the attribute 'Aircraft-Type' and there exists 17 different types of aircraft in that project. Therefore each of the entities were partitioned into 17 different horizontal fragments associated with the following predicate:

'Aircraft-Type' = Certain type of Aircraft (There are 17 different types of aircrafts).

In the 'Aircraft' subject database there are six entities, therefore $(17 \times 6) = 102$ fragments were created. For more details, refer to the quantitative approach part of appendix C.

As mentioned in chapter 3 section 3.3.2 for logical and complexity reasons vertical fragmentation is not as frequently applied as horizontal fragmentation. In addition, there is no tool presently available to support either the split approach or the grouping approach, therefore vertical fragmentation is not advisable for now. No vertical fragmentation has been done on the different entities of the appendix C project. However if appropriate tools are developed in the future to support vertical fragmentation, then this issue would have to be considered during the fragmentation step of the quantitative approach.

3. **Construction of Global Processing-Schedule Graph:** This step is a requirement of model 16 selected in chapter 3. If another model is selected for that step such as model 15, this step is not necessarily required. At this step a processing-schedule graph has to be created for the different queries and updates (as per figure 3-5). In order to construct a processing-schedule graph, query optimization has to be done. The selected query processing algorithm has to be able to handle an arbitrary allocation of fragments which correspond to virtual sites (See section 3.6.7 for virtual site definition). The query fragments associated with each virtual site have been determined during the two previous step of this approach. The query optimization part determines which operation to associate with the virtual sites of a query. The criteria for the query processing strategy of the query optimizer has to be the same as the one used in the allocation strategy. For example, if the allocation strategy is to minimize the transmission cost then the strategy of the query optimizer also has to minimize the transmission cost.

This third step can be applied either once or repeated many times during the application of this approach. If the next allocation step is applied with the feature of static query optimization, then this step has to be applied only once, the first time the different steps of the quantitative approach are applied. However, if the next allocation step is applied with the feature of dynamic query optimization, then every time a new pair of virtual sites is considered for union, the corresponding query processing-schedule will be adjusted as a result of dynamic query-optimization.

The two previous paragraphs discussed different issues to consider in order to build a global processing-schedule graph for an allocation algorithm that could cover all the different features of model 16. The next paragraphs describe how to build a

global processing-schedule graph for the model 16A algorithm (described in chapter 4) which uses static query optimization and minimizes transmission cost.

In order to support model 16A as mentioned at section 4.3 a global processing-schedule graph is built by first creating for each query and update a processing-schedule graph as in figure 3-5. Then the individual processing-schedule graph of each query and update are merged together into one graph called the global processing schedule graph. The virtual sites are still composed of the fragments determined in the previous step (one fragment per virtual sites) and the relational operations defined in the processing schedule graphs. The values on the transmission edges represent the amount of data (i.e. number of bytes) that go from one virtual site 'i' to another 'j' in a certain amount of time (e.g. months) for a specific query 'q' (It could also be for a specific update 'u'). These values are calculated as 'f_q' times 'd_{ijq}' where 'f_q' represents the frequency of the query ('f_u' for an update) and 'd_{ijq}' is the amount of data that go from one virtual site 'i' to another 'j' every time a query is executed ('d_{iju}' for an update). If not available from a query optimizer processing schedule (as per figure 3-2) that number 'd_{ijq}' has to be estimated.

For a specific query 'q', that amount of data 'd_{ijq}' can be estimated as follows:

$$d_{ijq} = E_{ijq} \times A_{ijq} \quad (5.1)$$

and

$$A_{ijq} = a + b \dots + x$$

In equation 5.1:

- a. d_{ijq} = total amount of data (in bytes) of a specific query 'q' going from the virtual site of origin 'i' to the virtual/physical site of destination 'j';

- b. E_{ijq} = average instance of fragment/entity of the virtual site of origin 'i' for each instance of fragment/entity of the virtual site of destination 'j' for a specific query 'q'. The E_{ijq} value considers the cardinality of the relationship between the two entities associated with the virtual site of origin 'i' and the virtual site of destination 'j' of a specific query 'q'. If there exists a one-to-one relationship in the E-R diagram between the entity/fragment of the virtual site of origin 'i' and the entity/fragment of the virtual site of destination 'j' of a query 'q', then $E_{ijq} = 1$. If there exists a one-to-many or a many-to-many relationship between these two entities then E_{ijq} can have any values from 1 to N (Where N is any real number greater than 1). The designer will have to estimate what the average E_{ijq} value is;
- c. A_{ijq} = the sum of data (In bytes) for the different attributes (a to x) going from the virtual site of origin 'i' to the virtual/physical site of destination 'j' for a specific query 'q'; and
- d. a, b ... , x = number of bytes per attribute.

For example, for the figure 5–4 query number 1, if we suppose that the entity 'B' fragment is at virtual site 2 and the entity 'C' fragment is at virtual site 3, then the total amount of data going from virtual site 2 to 3 could be estimated as follows:

- a. $A_{231} = 35$ bytes/instance (Where the attributes g = 10 bytes and h = 25 bytes); and
- b. $E_{231} = 5$ instance/query (Here it is assumed that every time the query 1 occurs, there is five instances of entity 'B' for an instance of entity 'C').

then $d_{231} = 35 \text{ bytes/instance} \times 5 \text{ instances/query} = 175 \text{ bytes/query}$.

Therefore in query 1 of figure 5-4, the total number of bytes per month going from entity 'B' to entity 'C' is:

$$f_1 \times d_{231} = 50 \text{ query/month} \times 175 \text{ bytes/query} = 8750 \text{ bytes/month.}$$

For updates, that amount of data ' d_{iju} ' is easier to evaluate since the equation 5.1 can be used by removing the variable ' E_{ijq} '. This equation then becomes:

$$d_{iju} = A_{iju} = a + b + \dots + x \quad (5.2)$$

In equation 5.2:

- a. d_{iju} = total amount of data (in bytes) going from the physical site of origin 'i' to the virtual site of destination 'j' for a specific update 'u';
- b. A_{iju} = the sum of data (In bytes) for the different attributes (a to x) going from the physical site of origin 'i' to the virtual site of destination 'j' for specific update 'u'; and
- c. a, b, ... , x = number of bytes per attribute.

For equation 5.2, it does not make sense to introduce the variable E_{iju} to consider the cardinality of the relationship between entities since in an update there is no data going between different entities. The data (entity/virtual site) is updated directly from a physical site.

For example, if there is an update number 1 done 50 times a month from physical site 1 on entity 'A' of figure 5-4 and if this entity 'A' is at virtual site 4, then the total number of bytes per month going from physical site 1 to the entity 'A' is:

$$f_u \times d_{iju} = f_1 \times d_{141} = 50 \text{ updates/month} \times 50 \text{ bytes/update} = 2500 \text{ bytes/month.}$$

For queries, the processes of merging the individual processing-schedule graph into global processing-schedule graph is straightforward. All the individual queries stay the same and are all put in the same figure without modification. For updates, however, the process of merging all the individual processing-schedule graphs in a global processing schedule graph is more complex. To ensure consistency of data in the overall system, the update done to one entity must be done to all the other redundant entities or fragments. This is illustrated at figure 5-5 a. and 5-5 b.

Figure 5—5: From Individual to Global Query Processing Schedule.

In figure 5.5 a) there are two queries (1) and (2) and one update (3). Figure 5.5 b) is the result of merging the three transactions of figure 5.5 a). In the process of merging, it can be seen in figure 5.5 b) that the two queries (1) and (2) of figure 5.5 a) stayed the same. The figure 5.5 a) update got modified in the sense that in figure 5.5 b) there is data transmitted from V6 to V2 and V5. This has been done supposing that the V2 and V5 entity/fragment are the same/part of the V6 entity. The amount of data transmitted from V6 to V2 and V5 is the total amount of the data that come from the different sites to update the V6 entity. Note that every time the V6 entity is updated the V2 and V5 entity/fragment also has to be updated to ensure consistency of data. The added forking arc means that V2 and V5 have to be updated at the same time.

This step has been applied to the Aerospace Maintenance Information System project in appendix C. Since no query optimizer was available, the different processing-schedule graph for the query/updates have been done with the object of minimizing the transmission cost using an estimated amount of data value 'd' as described in the previous paragraphs. Then the different processing-schedule graph of each queries and updates have been merged together into the same global processing-schedule graph.

4. Data Allocation: From the Global Processing-Schedule graph developed at the previous step, the different allocation developed under model 16 can be applied to allocate data in order to minimize the different cost/execution time issues selected during the qualitative approach. For such applications in which hundreds of fragments and tens of sites are involved it is vital to have a CASE tool that supports these heuristics automatically.

Since the CDT supports only one model 16A heuristic described in chapter 4, the data allocation performed by the CDT will only minimize the transmission cost of the system. The next sub-section will describe how the CDT performed when applied to the appendix C Aerospace Maintenance Information System projects.

5. Data Dictionary Allocation: As discussed at chapter 3, there is at least five different ways to allocate data dictionaries. In this step, it is up to the designer to consider each of them and also to consider the advantages and disadvantages associated with them in order to decide which approach to select.

For the Aerospace Maintenance Information System project of appendix C, the fifth approach (SDD-1) described in chapter 3 is selected. In this approach the entire data dictionary is treated as a single entity. Like any other relation, it is fragmented and allocated. There is a higher-level data dictionary which contain only the information about where are located the different fragments of the data dictionary. This approach is selected since the reliability of the overall system is an important issue. To have the total data dictionary stored only at one site such as in the approach one or four would not be a good choice for a military application where any of these sites could be subject of an attack of any kind or a natural disaster.

5.4 Application of the CDT.

The practical problem to which the CDT is applied to at appendix C is the same one then the appendix A ADAM project. In addition, it is also applied to the AMMIS project. AMMIS is included since this system retrieves a subset of the overall ADAM conceptual schema data developed at appendix A.

The ADAM project consists of building an information system distributed over 17 different air force bases located in Canada and Germany (from Vancouver to Baden Germany). This information system will assist the different Base Aircraft Maintenance Officer (BAMEO) in managing their aircraft resources by automating the current technical administrative tasks associated with the aircraft maintenance.

The AMMIS project consists of building an information system located in the national defence headquarter in Ottawa (NDHQ). This information system will use ADAM data to provide NDHQ aircraft manager with the maintenance information necessary to make decisions. These two projects require therefore data from different distant sites.

The overall data model of these two projects was developed at appendix A and consists of 19 entities. Since the CDT was applied to only a subset of that entire data model, the Aircraft subject database, six entities only were involved for the application of the CDT during the quantitative approach.

5.4.1 Data Traffic Without Distribution.

Figure 5-6 illustrates what would be the data traffic of the Aircraft subject database (subset of the overall data model) when all its data is centralized in one location. In this case it is assumed that all the data is centralized in NDHQ i.e. site 17.

In this figure, only the major sites of the appendix C projects are illustrated. The lines joining the different sites illustrate the major channel of communication required between the sites. The labels on the lines represent the amount of data to be transmitted during a year e.g. 13977K between site 12 and 17 means that 13977 kilo-bytes/year has to be transmitted between CFB Cold Lake and NDHQ. This amount of data has been estimated at appendix C using equation 5.1 and 5.2. It is important to note that some

of the data required to derive that amount of data transmitted between sites was not available. Different approximations therefore had to be made on that data. For example, the value of the E_{ijq} variable (see equation 5.1) of some queries as well as the frequency of occurrence ('f_u' or 'f_q' variables) of some queries and updates were not well known and had to be estimated.

Figure 5—6: All Data Centralized at NDHQ Ottawa.

Figure 5—7 gives the breakdown of the data to be transmitted in a year between the 16 different bases and NDHQ, when all the data is centralized at NDHQ. It also gives

the total amount of data to be transmitted in a year i.e. 76,142 Kilo-Bytes/Year with this option.

<u>Site Number :</u>	<u>K-Bytes/Year from Site to NDHQ :</u>
1- CFB Baden	6429
2- CFB North Bay	1552
3- CFB Moose Jaw	5778
4- CFB Winnipeg	1748
5- CFB Ottawa (S)	3053
6- CFB Shearwater	6835
7- CFB Valcartier	1098
8- CFB Gagetown	994
9- CFB Summerside	2996
10- CFB Greenwood	7677
11- CFB Bagotville	2494
12- CFB Cold Lake	13977
13- CFB Portage	1645
14- CFB Edmonton	2058
15- CFB Trenton	14801
16- CFB Comox	3007
Total :	<u>76,142K-Bytes/Year</u>

Figure 5—7: Data Traffic with Data Centralized at NDHQ.

Figure 5—8 gives the breakdown of the data to be transmitted in a year between the 16 different bases and NDHQ, when a copy of all the data is centralized at NDHQ and when a copy of the data generated at the base is kept at the base. It also gives the total amount of data to be transmitted in a year with this option i.e. 76,106 K-Bytes/Year.

<u>Site Number :</u>	<u>K-Bytes/Year from Site to NDHQ :</u>
1- CFB Baden	6427
2- CFB North Bay	1550
3- CFB Moose Jaw	5774
4- CFB Winnipeg	1747
5- CFB Ottawa (S)	3051
6- CFB Shearwater	6833
7- CFB Valcartier	1096
8- CFB Gagetown	992
9- CFB Summerside	2995
10- CFB Greenwood	7676
11- CFB Bagerville	2493
12- CFB Cold Lake	13973
13- CFB Portage	1642
14- CFB Edmonton	2056
15- CFB Trenton	14798
16- CFB Comox	<u>3003</u>

Total: 76,106 K-Bytes/Year

Figure 5—8: Data Traffic with Data Centralized at NDHQ and One Copy of the Data Generated Locally Stored Locally.

Since the qualitative approach recommended to distribute the different subject databases of the appendix C projects, the quantitative approach was applied to one of them, the Aircraft subject database. The next sub-section will describe the application of the CDT to the Aircraft subject database during the quantitative approach. This will be done in order to minimize the transmission cost between the distant sites of the appendix C.

5.4.2 CDT Application Results.

The result of the construction of the Global Processing-Schedule graph at step 3 of the quantitative approach for the Aircraft subject database was the creation of 194 virtual sites to be allocated over 17 different physical sites. These 194 virtual sites are composed

of 102 redundant fragments created at the step 2 fragmentation (see quantitative approach step 2 fragmentation or appendix C for further details). These 17 different physical sites correspond to the 17 different Canadian Air Force bases involved with the appendix C information system projects.

With these 194 virtual sites and 17 physical sites, using the 'Non-Stop' option of the prototype CDT (this option is described in chapter 4 section 4.7) and using a SUN Workstation 3/60 (two Mips), the optimal allocation was computed in about two hours. Note that a significant amount of this time was spent on graphics interface (e.g. virtual sites moving on the screen from one physical site to another). The final result shows graphically to which physical site each virtual site (fragment) should be assigned in order to minimize the total transmission cost of the system. It also shows the total amount of data that will travel in a year between the different sites with that assignment. The detailed results are shown at the quantitative part of appendix C. Figure 5-9 illustrates the data traffic of the Aircraft subject database when the data has been allocated using the CDT in order to minimize the transmission cost.

Figure 5—9: Data Allocated to Minimize Transmission Cost.

As for figure 5-6, in this figure only the major sites of the appendix C project are illustrated and the lines joining the different sites illustrate the major channel of communication required between physical sites. With this new allocation of data, it looks like some of the sites such as site 3, 17 and 6 are isolated from the rest of the network. However, from a practical view this is not the case since figure 5-9 illustrates only the major channel of communication required between the different sites. It has to be remembered that the first step of the quantitative approach has been done using the 20/80 rule, therefore only 80 percent of the query instances are considered. Besides, the CDT is applied to only a subset of the appendix A data model, the Aircraft subject

database (6 entities out of 19). For the Equipment or Personnel subject database, there is probably major channel communication between the isolated sites and the network.

The next figure, figure 5-10 gives the breakdown of the data to be transmitted in a year between the possible pairs of sites, when the data is allocated in order to minimize the overall transmission cost of the system. It also gives the total amount of data to be transmitted in a year i.e. 18,019 K-Bytes/Year with this configuration.

<u>Site 1:</u>	<u>Site 2:</u>	<u>K-Byte/Year Between Site 1 and 2 :</u>
CFB Bagotville	CFB Cold Lake	2494
CFB Baden	CFB Cold Lake	6429
CFB North Bay	CFB Cold Lake	787
CFB Comox	CFB Cold Lake	749
CFB Comox	CFB Greenwood	750
CFB Comox	CFB Summerside	750
CFB Comox	CFB Trenton	750
CFB Edmonton	CFB Trenton	1025
CFB Edmonton	CFB Ottawa (S)	1025
CFB Moose Jaw	NDHQ	107
CFB Moose Jaw	CFB Portage	411
CFB Moose Jaw	CFB Winnipeg	1748
CFB Valcartier	CFB Gagetown	994
Total :		18,019 K-Bytes/Year

Figure 5—10: Data Traffic when Transmission Cost is Minimized.

By comparing the total data traffic without distribution i.e. option 1 and 2 (76,142 K-Bytes/Year and 76,106 K-Bytes/Year respectively) to the total data traffic with optimal data distribution (18,019 K-Bytes/Year), it can be seen that the application of the CDT can reduce the total data traffic considerably and therefore the transmission cost of the overall system.

As previously mentioned, estimations had to be made to derive the amount of data transmitted between sites. This is reflected in the accuracy of the total data traffic before (option 1 and 2) and after the application of the CDT. However, these results give an idea of the transmission cost savings that can be made by applying the CDT to minimize the data traffic of real size systems.

5.5 Summary.

In this chapter, an approach for data distribution was proposed and applied to a practical problem at appendix C. The developed CDT was also applied to that same problem to demonstrate how it performs when applied to real size problems, how it can support the proposed approach and how it can reduce the data traffic (which is proportional to the transmission cost) for real size systems.

It is important to note that the approach used in this section is applicable to problems in which a global processing schedule graph can be developed. This approach involves transactions and their frequencies, fragments and transmission cost. The next chapter will conclude by describing the contributions of this research and the future developments to be done to the CDT as well as the proposed approach for data distribution design.

Chapter 6

Conclusion.

6.1 Contributions of this Research.

The major contribution of this research work is the development of two interrelated issues i.e. an approach for data distribution design as well as its supporting prototype CASE tool i.e. the CDT. As demonstrated in chapter 5, for real size problems (Information Systems with hundreds of data fragments and tens of sites) effective data distribution design requires a CASE tool to automate some of its time consuming steps such as data allocation. In addition, the application of this CASE tool helps the designer to follow different phases of a distribution methodology.

The usefulness of the developed approach was demonstrated for realistic problem by applying it to the Aerospace Maintenance Information System projects described in appendix C. The data distribution capabilities developed in this thesis can be incorporated in a methodology such as the Information Engineering Methodology (IEM), and under certain constraints specified in section 5.1 are general enough to be applied to other information system methodologies which support data in various data models such as relational and network data models. The qualitative approach identified the major issues that management should consider to decide either to distribute or to keep data centralized. The data allocation part of the quantitative approach supports model 16 selected in chapter 3. The advantage of having the proposed approach separated in two halves (i.e. qualitative and quantitative) is that if it is decided during the qualitative approach to distribute the data by considering criteria other than the ones covered by model 16 (such as reliability),

then only the quantitative approach would have to be modified in order to adapt to the new selected model

The analysis of the different data allocation models given in chapter 3 identified different allocation features (such as transmission cost, response time etc...) that could possibly be evaluated and/or optimized by existing data allocation models during the data distribution design. These different allocation features are also useful for the selection of a model to be supported by a CASE tool. In chapter 3, model 16 was selected to be implemented in the CDT since it offers heuristics to support the largest number of data allocation features i.e. 12 different ones and if appropriately modified, it has the potential to cover four extra features. This model is the only one which cover two important groups of features, namely, cost and performance, as well as the duality between data allocation and query optimization. However it is not determined how the features of reliability and availability could be covered by this model.

Of the different functions and components (described in section 4.2) which are part of the architecture of a CASE tool for the data distribution design, the CDT covers the function of data/fragment allocation and is composed of the following components: graphics interface tool, diagramming tool and limited design checker and analyser tool. The function of fragment allocation of the CDT supports a subset of model 16 i.e. model 16A. As described in Section 4.3.1, model 16A minimizes the communication cost which is considered as important in long haul networks. Section 4.4 presents three modifications made to this model. The first modification enhance the heuristic optimality, the second one adapts the heuristic so that it can be applied to practical problems, and the third one is a simple modification made to ease the model implementation.

The application of the CDT to a real size problem (i.e. a subject database of the Aerospace Maintenance Information System projects consisting of 194 virtual sites or fragments and 17 real sites described at appendix B) performed well on a two Mips workstation (Sun 3/60) since it computed the optimal data allocation in a reasonable amount of time (i.e. in about two hours). The final result giving the optimal data allocation as well as the total amount of data transmitted between the different sites can easily be seen graphically from the CDT (See appendix C for a graphical display of the optimal data allocation). Although fault tolerance issues as well as the extra data traffic incurred by some of the distributed DBMS functions such as concurrency control and recovery have not been considered in this research (in section 5-4), the 422 % data traffic reduction (from 76 Mega-Bytes/Year to 18 Mega-Bytes/Year) obtained for the example given in appendix C shows that the CDT has the potential to reduce the transmission cost of data. Therefore because of its performance, ease of use and potential to reduce the transmission cost of data, the CDT could become a useful tool for a database administrator in order to perform allocation of data during design as well as operational life cycle phases.

The second contribution of this thesis is to demonstrate the efficiency of the object oriented approach for a CASE tool development environment. This approach proved to be efficient for the CDT development since most of the programming required by the graphic interface, the diagramming and the design checker and analyser tool was inherited from ARTTisan. This considerably speeded up the development process and allowed to develop the overall prototype in approximatively three months.

6.2 Future Developments.

The following are the different enhancements that could be made to the CDT as well as the different extensions that could be made to the data distribution approach:

1. The fragment allocation function of the CDT presently supports the model 16A heuristic which allocates data in order to minimize the transmission cost. As mentioned in section 3.6, since the processing-schedule graph represents operation as well as data at sites, the model 16A heuristic could be modified to consider I/O and CPU cost as well as storage cost. On the global processing-schedule graph, the information about the amount of data transmitted is on the transmission edges between the different physical sites. If the heuristic is modified accordingly then the channel capacity could be considered as a constraint during the allocation process. The modified heuristic could then be implemented and included in the CDT.
2. As mentioned in section 3.6, model 16 offers a heuristic which consider queuing delay to compute the minimum operation processing and data transmission time. This heuristic could be incorporated into the CDT to cover the execution time feature.
3. Even with these two previous modifications model 16 does not cover the features of availability as well as reliability (See figure 3-3). If the modification of model 16 with these two features proved to be unfeasible, then another model such as model 15 could be incorporated into the CDT to cover these features. With a new model, the quantitative approach would probably have to be modified. For example, the step of the construction of the Global Processing-Schedule graph in the quantitative approach is not required by model 15.

4. The selected model 16 requires that multi or individual query optimization be done before (for static query optimization) or during (for dynamic query optimization) allocation of data. In order to support that model, as shown in figure 4-1 a query optimizer could be implemented as a separate modular part of the CDT with the same components (Graphic interface tools etc...) as the fragment allocator. This query optimizer would support the step of the construction of the Global Processing-Schedule graph in the quantitative approach. From the results of the query optimization (Figure 3-2 Processing Schedule) the individual processing-schedule graph could be generated and merged together automatically. As mentioned at section 4.6 model 16, the user should be aware that the objective of the query processing strategy must comply with the objective of the data allocation strategy. For example, if the objective is to optimize the transmission cost, then the strategy of both the query optimizer and the data allocator should be to minimize that cost. The tool could be built to constrain the user to select query processing strategies and data allocation strategies with the same objective.
5. Major efforts could be spent to develop a CASE tool for data distribution design with the architecture and all others functions and components of figure 4-1. This CASE tool could be developed as a whole (a Toolkit) or integrated into an existing Workbench CASE tool having the components of figure 2-1.
6. A study of existing and emerging standards for the effective management of information resources such as IRDS mentioned in section 2.4.3 could be made. Part of this study could determine the degree to which data distribution is covered by these standards. If it is not, then recommendations could be made to the relevant standardization committees to explain how to incorporate this issue.
7. The qualitative approach describes the effect that different issues have on the

decision to distribute or centralize the data. The effect that each issue has on data distribution is described individually and there are no specific guidelines that describe how to use these issues together. For a given application environment, some issues are probably more important than others. A weighting factor could be attached to these different issues. Further work could be done to give management more specific guidelines on how to consider these different issues together in order to decide to distribute or centralize the data.

8. Investigation of the applicability of the data distribution approach used in this thesis for development of a data distribution algorithm for object oriented databases.

Bibliography

- [1] R.S. Pressman, "Software Engineering A Practitioner's Approach", McGraw-Hill Book Co., California, 1987.
- [2] W. Olle, "Information Systems Methodologies. A framework for Understanding". Addison-Wesley, U.S.A., 1988.
- [3] L. Peters, "Advances Structured Analysis and Design". Prentice-Hall, New Jersey, 1987.
- [4] Bo-Shoe Chen, Raymond T. Yeh, "Formal Specification and Verification of Distributed Systems", IEEE Distributed Computing Systems, 1982, pp. 380-387.
- [5] B. Yadav, "Determining an organization's information requirements: a state of the art survey", Database, Vol 14 No 3, Spring 1983, pp. 3-20.
- [6] P. Mimno, "The James Martin Productivity Series", Computer-Aided Software Engineering", Special Report for the CASE Symposium, Feb. 1987.
- [7] D. Teichroew and E. Hershey, "PSL/PSA: a computer-aided technique for structured documentation and analysis of information processing systems", IEEE Transactions on Software Engineering, Vol 3 No 1, 1977, pp. 326-345.
- [8] M.Jackson, "System Development", Englewood Cliffs, New Jersey, 1983.
- [9] T.W. Olle & A.A. Verijn-Stuart, "Information Systems Design Methodologies: Improving the Practice", Proceedings of the IFIP WG 8.1 Working Conference, North-Holland, Amsterdam, 1986, pp. 209-223.
- [10] M. Jackson, "Principles of Program Design", Academic Press, 1975
- [11] J. Martin, "Information Engineering", Lancaster:Savant 1987.
- [12] J. Martin, "An Information Manifesto", Prentice-Hall, New Jersey, 1984
- [13] J. Martin, "Information Engineering: A Management White Paper", Knowledge-Ware, Atlanta, 1986.
- [14] J. Martin, "Strategic Data-Planning Methodologies", Prentice-Hall, New Jersey,

1982.

- [15] G. McDonald, "Information Engineering- a methodology to match fourth generation tools", Pergamon Infotech Ltd, U.K., 1984.
- [16] R. Jones, "Automated Tools for the analyst", Data Processing, Vol 28 No 5, June 1986, pp. 241-243.
- [17] O. Laurido-Santos, "A Course on Information Engineering", ACM, Vol 18 No 1, 1986, pp. 194-202.
- [18] E. Yourdon and L. Constantine, "Fundamentals of a Discipline of Computer Programs and System Design", Prentice—Hall, New Jersey, 1979.
- [19] E. Yourdon, "Managing the System Life Cycle", Yourdon Press, New York, 1988.
- [20] M. Page-Jones, "The Practical Guide to Structured Systems Design", Yourdon Press, New York, 1980.
- [21] P.T. Ward & S.J. Mellor, "Structured Development for Real Time Systems", Yourdon Press, New York, 1985.
- [22] E.J. Chikofsky, "Software Technology People can Really Use", IEEE Software, Vol. 5, No. 2, March 1988, pp. 8-10.
- [23] W. Suydam, "CASE makes strides toward automated software development", Computer Design, Jan 1987, pp. 49-70.
- [24] R.H. Sprague, "Information Systems Management In Practice", Prentice—Hall, Toronto, Canada, 1986.
- [25] Digital Consulting Inc., "Computer-Aided Software Engineering Symposium", Spring 1988
- [26] C.L. McClure, "Software Workbenches: The new software development environment", Extended Intelligence Inc, Chicago, 1987.
- [27] C. McClure, "Computer-Aided Software Engineering", System Development, 1988, pp. 736-748.

- [28] J. Desmond, "Working on the Connections". Software News, April 87, pp. 60-73.
- [29] J. Martin & C. McClure, "The Latest Look in Programmer Productivity Tools", Business Software Review, May 1986, pp. 38-44.
- [30] R. C. Johnson, "Front End, Back End of CASE Meet in the Middle", Electronic Engineering Times, March 7, 1988, pp. 78-107.
- [31] H. Falk, "CASE Tools Emerge to Handle Real-Time Systems", Computer Design, Jan 1988, pp. 53-74.
- [32] R. Hurst, "CASE systems near fruition", ComputerWorld, July 87, pp. 27-29.
- [33] Margaret Henderson Law, "Guide to Information Resource Dictionary System Applications: General Concepts and Strategic Systems Planning", U.S. Department of Commerce, National Bureau of Standards, MD, April 1988.
- [34] J. Martin, "Strategic Data-Planning Methodologies", Prentice-Hall, New Jersey, 1982.
- [35] J. Martin, "Application Development Without Programmers", Prentice-Hall, New Jersey, 1982.
- [36] J. Martin, "Managing the Database Environment", Prentice-Hall, New Jersey, 1980.
- [37] J. Martin, "Managing the Database Environment", Prentice-Hall, New Jersey, 1984.
- [38] J. Martin & C. McClure, "Action Diagram", Prentice-Hall, New Jersey, 1985.
- [39] J. Martin, "Design and Strategy for Distributed Data Processing", Prentice-Hall, New Jersey, 1981.
- [40] B. Davis, "Strategies for information requirements determination", IBM Systems, Vol 21, No. 1, 1982, pp. 4-30.
- [41] J.F. Rockart, "Critical success factors", Harvard Business Review 57, No. 2, April 1979, pp. 81-90.

- [42] S. Ceri, S. Navathe, G. Wiederhold, "Distribution Design of Logical Database Schemas". IEEE Transaction on Software Engineering, Vol SE-9 No 4, July 1983, pp. 487-503.
- [43] S. Ceri & G. Pelagatti, "Distributed Database Principles and Systems", McGraw-Hill, U.S.A., 1984.
- [44] S. Ceri, B. Pernici and G. Wiederhold, "Distributed Database Design Methodologies", IEEE Proceeding, Vol 75 No 5, May 1987, pp. 533-546.
- [45] S. Ceri, M. Negri & G. Pelagatti, "Horizontal Data Partitioning in Database Design", in Proc. ACM-SIGMOD Int. Conf. Management of Data, 1982, pp. 128-136.
- [46] P.M.G. Apers, "Query Processing and Data Allocation in Distributed Database System", PhD. Dissertation, Computer Science Dept, Vrije Univ. , Amsterdam, Sept. 1982.
- [47] P.M.G. Apers, "Data Allocation in Distributed Database System", ACM Transaction on Database Systems, Vol 13 No 3, Sept 1988, pp. 263-304.
- [48] A. Albano, V. DeAntonellis, A. DiLeva, "Computer-Aided Database Design, The DataidProject", North-Holland, Amsterdam, 1985.
- [49] D. Maier and J. Ullman, "Fragments of relations", ACM SIGMOD, May 1983, pp. 15-22.
- [50] J. Paredaens and DeBra, "On horizontal decompositions", in Proc.XP2 Workshop, 1981, pp. 111-125.
- [51] S. K. Chang and W. H. Cheng, "A methodology for structured database decomposition", IEEE Trans. Software Eng., Vol SE-6 No 2, Mar 1980, pp. 205-218.
- [52] S. Navathe, S. Ceri, G. Wiederhold, J. Dou, "Vertical Partitioning Algorithms for Database Design", ACM Trans. Database System, Vol 19 No 4, Dec. 1984, pp. 680-710.
- [53] M. Hammer, B. Niamir, "A heuristic approach to attribute partitioning", In Proceedings ACM SIGMOD International Conference on Management of Data, Boston, 1979, pp. 93-101.
- [54] B. Gavish, "Models for Configuring Large-Scale Distributed Computing Systems",

AT&T Technical Journal, Vol 64 No 2, Feb 1985, pp. 491-532.

[55] B. Gavish, H. Pirkul, "Computer and Database Location in Distributed Computer Systems", IEEE Transaction On Computers, Vol C-35 No 7, July 1986, pp. 583-590.

[56] B. Gavish, "Optimization Models for Configuring Distributed Computer Systems", IEEE Transaction On Computers, Vol C-36 No 7, July 1987, pp. 773-793.

[57] K. P. Eswaran, "Placement of Records in a File and File Allocation in a Computer Network", Information Processing 74 (IFIPS), 1974, pp. 304-307.

[58] B.W. Wah, "File Placement on Distributed Computer Systems", IEEE Computer, Jan 1984, pp. 23-32.

[59] P. Bodarik, "Query Processing Strategies in a Distributed Database", PhD Dissertation, Carleton University, Canada, Sept. 1985.

[60] L. W. Dowdy, D.V. Foster, "Comparative Models of the File Assignment Problem", Computing Surveys ACM, Vol 14 No 2, June 1982, pp. 287-313.

[61] A.S. Tanenbaum, "Computer Networks", Prentice-Hall, New Jersey, 1981.

[62] H.L. Morgan, K.D. Levin, "Optimal program and data locations in computer networks", Commun. ACM, Vol. 20, May 1977, pp. 315-322.

[63] C.J. Date, "An Introduction to Database Systems", The systems programming series, Vol II, Addison-Wesley Pub. Comp, U.S.A., 1984.

[64] R.G. Casey, "Allocation of copies of a file in an information network", Proc AFIPS 1972 Spring Jt. Computer Conf., Vol. 40, AFIPS Press, 1972, pp. 617-625.

[65] K.M. Chandy & J.E. Hewes, "File allocation in distributed systems", in Proc. Int. Symp. on Computer Performance Modeling, Measurement and Evaluation, Cambridge, Mass., March 1976, pp. 10-13.

[66] H. L. Morgan & K. D. Levin, "Optimal program and data locations in computer networks", Commun. ACM, Vol. 20, May 1977, pp. 315-322.

[67] P.P.S. Chen and J. Akoka, "Optimal design of distributed information systems",

IEEE Trans. Comput., Vol C-29 No 12, Dec 1980, pp. 1068-1078.

- [68] W. W. Chu, "Performance of file directory system for database in star and distributed networks", in Proc AFIPS 1976 Nat. Comp. Conf., Vol 45, Va, 1976, pp. 577-587.
- [69] S. Mahmoud and J.S. Riordon, "Optimal allocation of resource in distributed information networks", ACM Trans. Database Syst., Vol. 1, March 1976, pp. 66-78.
- [70] K. B. Irani and N.G. Khabbaz, "A model for a combined communication network design and file allocation for distributed databases." Proc. 1st Conf. Distributed Comp. Syst. , 1979, pp. 15-21.
- [71] C.V. Ramamoorthy & K.M. Chandy, "Optimization of memory hierarchies in multiprogrammed systems", ACM, Vol 17, July 1970, pp. 426-445.
- [72] A.S.R. Arora & A. Gallo, "Optimal sizing, loading and reloading in a multilevel memory systems", in Proc AFIPS, Vol 38, U.S.A., 1971, pp. 337-344.
- [73] P.H. Hughes and G. Moe, "A structural approach to computer performance analysis", in Proc. AFIPS, Va, 1973, pp. 109-120.
- [74] P. P-S. Chen, "Optimal file allocation in multilevel storage systems", in Proc. AFIPS, Vol 42, Va, 1973, pp. 277-282.
- [75] L. Kleinrock, "Queuing Systems —Vol 2: Computer Applications", Wiley, New York, 1976.
- [76] L. G. Jones, D.V. Foster & P. D. Krolak, "A goal programming formulation used in the file assignment problem", in Proc, Pittsburg, July 1978, pp. 108-121.
- [77] L.W. Dowdy, "Optimal branching probabilities and their relationships to computer network file distribution", PhD Dissertation, N.C. U.S.A., 1977.
- [78] R. Timmons, "Distributed Database Design Methodologies and Tools", Carleton University, Canada, April 1988.
- [79] S. Ceri, B. Pernici & G. Wiederhold, "An Overview of Research in the Design of Distributed Databases", Database Engineering, Vol 3, U.S.A., March 1985, pp. 227-238.

- [80] K.C. Watson, "ARTTisan Class Definition", Carleton University, Canada, 1989.
- [81] J.F. Rockart, C.V. Bullen and J.L. Levantor, "Centralization versus Decentralization of Information Systems: A Preliminary Model for Decision Making", Alfred P. Sloan School of Management, MIT, Cambridge, Mass. 1976.
- [82] N.G. James & M. Anderton, "Distributed Computer Systems: Four Case Studies", Proceedings of the IEEE, Vol 75 No 5, May 1987, pp. 719-726.
- [83] E. Bertino & L.M. Hans, "Views and Security in Distributed Database Management Systems", In Advances in Database Technology-EDBT 88, Lecture Notes in Computer Science, No 303, Italy, March 1988, pp. 155-169.
- [84] J. Akoka, "Centralization versus Decentralization of Information Systems: A decision support model", Management of Distributed Data Processing, North-Holland publishing Co., Amsterdam, 1982, pp. 1-18.
- [85] Das Eng 5-2, "Automated Data for Aerospace Maintenance System Requirements Definition", NDHQ, Ottawa, Canada, Jan 1984.
- [86] Das Eng 5-4, "AMMIS CF349 & CF543 Submissions Statistics", Report ID:DM 01, NDHQ, Ottawa, Canada, 30 Jun 88.
- [87] "Objective-C Reference Manual", Release 3.3 Productivity Products International Inc. , U.S.A., Nov. 1986.

Appendix A

IEM AND IEW APPLIED TO AN AEROSPACE MAINTENANCE INFORMATION SYSTEM PROJECT.

Application of IEM and IEW to ADAM.

The real world problem to which IEM and IEW is applied to a project called "ADAM". It stands for Automated Data for Aerospace Maintenance. This project consists of building an information system which will assist the Base Aircraft Maintenance Officer (BAMEO) in managing his aircraft resources by automating the current technical administrative tasks associated with aircraft maintenance. The following are the different steps of IEM applied to ADAM. The detailed information required by each step was taken from ref[85].

Planning Phase.

This phase of IEM focuses on building the overall "TOP-DOWN" plan for the entire organization. The following paragraphs will describe the different steps of the planning phase:

1. **Step (1)-> Enterprise Model:** This step consists of decomposing in a hierarchical manner the organization chart (organizational unit) into sub-units (figure A-1).

BAMEO

July 21 1989 16 43 11

Figure A-1: BAMEO Organization Chart

Organizational Unit: AMADO

Definition
TECHNICAL ADMINISTRATION OFFICER

PROPERTY VALUE

Organizational Unit: AMCRO

Definition
RECORD KEEPING AND CONTROL FUNCTION OFFICER

PROPERTY VALUE

Organizational Unit: ARMSO

Definition
ARMAMENT SUPPORT OFFICER

PROPERTY VALUE

Organizational Unit: ARO

Definition
AIRCRAFT REPAIR OFFICER

PROPERTY VALUE

Organizational Unit: ASO

Definition
AIRCRAFT SERVICING OFFICER

PROPERTY VALUE

Organizational Unit: AVSO

Definition
AIRCRAFT AVIONIC SUPPORT OFFICER

PROPERTY VALUE

Figure A-1: BAMEO Organization Chart (Cont'd)

Information Engineering Workbench Report
Object Summary Report
July 21, 1989 17:03:11 NEWUSER ADAM

Organizational Unit: BAMEO

Definition
BASE AIRCRAFT ENGINEERING OFFICER

PROPERTY	VALUE
----------	-------

Organizational Unit: MSO

Definition
MECHANICAL SUPPORT OFFICER

PROPERTY	VALUE
----------	-------

Organizational Unit: PHOTOSO

Definition
PHOTO SUPPORT OFFICER

PROPERTY	VALUE
----------	-------

Figure A-1: BAMEO Organization Chart (Cont'd)

This organization chart is the decomposition of the different 8 basic sections of the current Base Aircraft Maintenance Engineering Officer's (BAMEO) organization. Also during this step, the functional area of the organization have to be decomposed into functions and then the functions are decomposed into processes (figure A-2). This functional area decomposition is the decomposition of the 10 base aircraft maintenance functions into their respective processes.

Figure A-2: BAMEO Functions Decomposition

Function AMMIS

PROPERTIES:

ASSOCIATIONS:

IS COMPOSED OF
 Process P9.1
 Process P9.2
 OCCURS IN
 Function BASE AIRCRAFT MAINTENANCE
 IS RESPONSIBILITY OF
 Organizational Unit BAMEO
 Organizational Unit ASO
 Organizational Unit ARO
 Organizational Unit AVSO
 Organizational Unit MSO
 Organizational Unit ArmSO

Function BASE AIRCRAFT MAINTENANCE

PROPERTIES:

ASSOCIATIONS:

IS COMPOSED OF
 Function SERVICING
 Function MAINTENANCE
 Function PERSONNEL QUALIFICATIONS
 Function LOGISTIC SUPPORT
 Function REPORTS
 Function LOG SET MAINTENANCE
 Function PROJECTS
 Function MAINTENANCE SCHEDULING
 Function AMMIS
 Function ISSUE
 IS RESPONSIBILITY OF
 Organizational Unit BAMEO
 Organizational Unit ASO
 Organizational Unit ARO
 Organizational Unit AVSO
 Organizational Unit MSO
 Organizational Unit ArmSO
 Organizational Unit PhotoSO
 Organizational Unit AMAdO
 Organizational Unit AMCRO

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Function: ISSUE**PROPERTIES:****ASSOCIATIONS:****IS COMPOSED OF**

Process P10.1
 Process P10.2
 Process P10.3

OCCURS IN

Function BASE AIRCRAFT MAINTENANCE

IS RESPONSIBILITY OF

Organizational Unit BAMEO
 Organizational Unit ASO
 Organizational Unit ARO
 Organizational Unit AVSO
 Organizational Unit MSO
 Organizational Unit ArmsO
 Organizational Unit PhotoSO
 Organizational Unit AMAdO
 Organizational Unit AMCRO

Function LOG SET MAINTENANCE**PROPERTIES:****ASSOCIATIONS:****IS COMPOSED OF**

Process P6.1
 Process P6.2
 Process P6.3
 Process P6.4
 Process P6.5
 Process P6.6

OCCURS IN

Function BASE AIRCRAFT MAINTENANCE

IS RESPONSIBILITY OF

Organizational Unit BAMEO
 Organizational Unit ASO
 Organizational Unit AMCRO

Function LOGISTIC SUPPORT**PROPERTIES:****ASSOCIATIONS:**

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Function PERSONNEL QUALIFICATIONS

PROPERTIES:

ASSOCIATIONS:

IS COMPOSED OF

Process P3.1
 Process P3.2
 Process P3.3
 Process P3.4
 Process P3.5
 Process P3.6

OCCURS IN

Function BASE AIRCRAFT MAINTENANCE

IS RESPONSIBILITY OF

Organizational Unit BAMEO
 Organizational Unit ASO
 Organizational Unit ARO
 Organizational Unit AVSO
 Organizational Unit MSO
 Organizational Unit ArmSO
 Organizational Unit PhotoSO
 Organizational Unit AMAdO
 Organizational Unit AMCRO

Function PROJECTS

PROPERTIES:

ASSOCIATIONS:

IS COMPOSED OF

Process P7.1
 Process P7.2
 Process P7.3
 Process P7.4
 Process P7.5
 Process P7.6

OCCURS IN

Function BASE AIRCRAFT MAINTENANCE

IS RESPONSIBILITY OF

Organizational Unit BAMEO
 Organizational Unit AVSO
 Organizational Unit MSO
 Organizational Unit AMCRO

Figure A-2: BAMEO Functions Decomposition (Cont'd)

IS COMPOSED OF
Process P4.1
Process P4.2
Process P4.3
Process P4.4
Process P4.5
Process P4.6
OCCURS IN
Function BASE AIRCRAFT MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO
Organizational Unit AMCRO

Function MAINTENANCE

PROPERTIES:

ASSOCIATIONS:

IS COMPOSED OF
Process P2.1
Process P2.2
Process P2.3
OCCURS IN
Function BASE AIRCRAFT MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO
Organizational Unit ARO
Organizational Unit AVSO
Organizational Unit MSO
Organizational Unit ArmsO
Organizational Unit PhotoSO

Function MAINTENANCE SCHEDULING

PROPERTIES:

ASSOCIATIONS:

IS COMPOSED OF
Process P8.1
Process P8.2
OCCURS IN
Function BASE AIRCRAFT MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO
Organizational Unit ASO
Organizational Unit ARO

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Function REPORTS

PROPERTIES:

ASSOCIATIONS:

IS COMPOSED OF

Process P5.1
 Process P5.2
 Process P5.3
 Process P5.4
 Process P5.5

OCCURS IN

Function BASE AIRCRAFT MAINTENANCE

IS RESPONSIBILITY OF

Organizational Unit BAMEO
 Organizational Unit ASO
 Organizational Unit ARO
 Organizational Unit AVSO
 Organizational Unit MSO
 Organizational Unit ArmsO
 Organizational Unit PhotoSO
 Organizational Unit AMCRO

Function SERVICING

PROPERTIES:

ASSOCIATIONS:

IS COMPOSED OF

Process P1.1
 Process P1.2
 Process P1.3
 Process P1.4

OCCURS IN

Function BASE AIRCRAFT MAINTENANCE

IS RESPONSIBILITY OF

Organizational Unit BAMEO
 Organizational Unit ASO

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Process P1.1**PROPERTIES:**

DEFINITION:
DETERMINE SERVICING REQUIREMENT

ASSOCIATIONS:

OCCURS IN
Function SERVICING
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P1.2**PROPERTIES:**

DEFINITION:
SERVICING INSPECTION

ASSOCIATIONS:

OCCURS IN
Function SERVICING
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P1.3**PROPERTIES:**

DEFINITION:
LOAD CHANGES

ASSOCIATIONS:

OCCURS IN
Function SERVICING
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P1.4**PROPERTIES:**

DEFINITION:
FUEL AIRCRAFT

Figure A-2: BAMEO Functions Decomposition (Cont'd)

ASSOCIATIONS:

OCCURS IN
Function SERVICING
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P10.1

PROPERTIES:

DEFINITION:
ISSUE EQUIPMENT

ASSOCIATIONS:

OCCURS IN
Function ISSUE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P10.2

PROPERTIES:

DEFINITION:
UPDATE EQUIPMENT STATUS

ASSOCIATIONS:

OCCURS IN
Function ISSUE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P10.3

PROPERTIES:

DEFINITION:
UPDATE USAGE

ASSOCIATIONS:

OCCURS IN
Function ISSUE
IS RESPONSIBILITY OF

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Organizational Unit BAMEO**Process P2.1****PROPERTIES:**

DEFINITION:
OPEN MAINTENANCE ACTION

ASSOCIATIONS:

OCCURS IN
Function MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P2.2**PROPERTIES:**

DEFINITION:
INVESTIGATE MAINTENANCE ACTION

ASSOCIATIONS:

OCCURS IN
Function MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P2.3**PROPERTIES:**

DEFINITION:
CONDUCT MAINTENANCE ACTION

ASSOCIATIONS:

OCCURS IN
Function MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P3.1**PROPERTIES:**

Figure A-2: BAMEO Functions Decomposition (Cont'd)

DEFINITION:
ISSUE QUALIFICATION

ASSOCIATIONS:

OCCURS IN
Function PERSONNEL QUALIFICATIONS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P3.2

PROPERTIES:

DEFINITION:
ROTATION AND POSTING

ASSOCIATIONS:

OCCURS IN
Function PERSONNEL QUALIFICATIONS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P3.3

PROPERTIES:

DEFINITION:
ATAT QUARTERLY REPORT

ASSOCIATIONS:

OCCURS IN
Function PERSONNEL QUALIFICATIONS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P3.4

PROPERTIES:

DEFINITION:
ELIGIBILITY DATES

ASSOCIATIONS:

OCCURS IN

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Function PERSONNEL QUALIFICATIONS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P3.5

PROPERTIES:

DEFINITION:
TRADE EXAMS

ASSOCIATIONS:

OCCURS IN
Function PERSONNEL QUALIFICATIONS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P3.6

PROPERTIES:

DEFINITION:
COURSE LOADING

ASSOCIATIONS:

OCCURS IN
Function PERSONNEL QUALIFICATIONS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P4.1

PROPERTIES:

DEFINITION:
DETERMINE AVAILABILITY

ASSOCIATIONS:

OCCURS IN
Function LOGISTIC SUPPORT
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P4.2

Figure A-2: BAMEO Functions Decomposition (Cont'd)

PROPERTIES:

DEFINITION:
GENERATE MACR

ASSOCIATIONS:

OCCURS IN
Function LOGISTIC SUPPORT
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P4.3**PROPERTIES:**

DEFINITION:
ENTER NEW ENTITLEMENT

ASSOCIATIONS:

OCCURS IN
Function LOGISTIC SUPPORT
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P4.4**PROPERTIES:**

DEFINITION:
COMPLETE CF2302

ASSOCIATIONS:

OCCURS IN
Function LOGISTIC SUPPORT
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P4.5**PROPERTIES:**

DEFINITION:
ENTER EXPECTED DATE

Figure A-2: BAMEO Functions Decomposition (Cont'd)

OCCURS IN
Function LOGISTIC SUPPORT
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P4.6

PROPERTIES:

DEFINITION:
ENTER AS RECEIVED

ASSOCIATIONS:

OCCURS IN
Function LOGISTIC SUPPORT
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P5.1

PROPERTIES:

DEFINITION:
MAINTENANCE REPORT

ASSOCIATIONS:

OCCURS IN
Function REPORTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P5.2

PROPERTIES:

DEFINITION:
MOD/SI/PROJECTS REPORT

ASSOCIATIONS:

OCCURS IN
Function REPORTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Process P5.3**PROPERTIES:**

DEFINITION:
AIR INCIDENT/ACCIDENT REPORT

ASSOCIATIONS:

OCCURS IN
Function REPORTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P5.4**PROPERTIES:**

DEFINITION:
INTERNAL MANAGEMENT REPORT

ASSOCIATIONS:

OCCURS IN
Function REPORTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P5.5**PROPERTIES:**

DEFINITION:
PERSONNEL REPORTS

ASSOCIATIONS:

OCCURS IN
Function REPORTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P6.1**PROPERTIES:**

DEFINITION:
RECORD LIFE USAGE

Figure A-2: BAMEO Functions Decomposition (Cont'd)

ASSOCIATIONS:

OCCURS IN
Function LOG SET MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P6.2**PROPERTIES:**

DEFINITION:
RECORD SIGNIFICANT EVENT

ASSOCIATIONS:

OCCURS IN
Function LOG SET MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P6.3**PROPERTIES:**

DEFINITION:
RECORD CONFIGURATION CHANGES

ASSOCIATIONS:

OCCURS IN
Function LOG SET MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P6.4**PROPERTIES:**

DEFINITION:
RECORD LIFE AND INSPECTION DATA

ASSOCIATIONS:

OCCURS IN
Function LOG SET MAINTENANCE
IS RESPONSIBILITY OF

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Organizational Unit BAMEO**Process P6.5****PROPERTIES:****DEFINITION:**

RECORD MOD & SPECIAL INSPECTION DATA

ASSOCIATIONS:**OCCURS IN**

Function LOG SET MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P6.6**PROPERTIES:****DEFINITION:**

PREDICT LIFE EXPIRY

ASSOCIATIONS:**OCCURS IN**

Function LOG SET MAINTENANCE
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P7.1**PROPERTIES:****DEFINITION:**

DETERMINE UNSATISFACTORY CONDITION

ASSOCIATIONS:**OCCURS IN**

Function PROJECTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P7.2**PROPERTIES:****Figure A-2: BAMEO Functions Decomposition (Cont'd)**

DEFINITION:
ENTER UCR AND DATA

ASSOCIATIONS:

OCCURS IN
Function PROJECTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P7.3

PROPERTIES:

DEFINITION:
RESEARCH SOLUTION
730

ASSOCIATIONS:

OCCURS IN
Function PROJECTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P7.4

PROPERTIES:

DEFINITION:
COMPLETE DOCUMENTATION
745

ASSOCIATIONS:

OCCURS IN
Function PROJECTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P7.5

PROPERTIES:

DEFINITION:
RECORD APPROVAL

Figure A-2: BAMEO Functions Decomposition (Cont'd)

ASSOCIATIONS:

OCCURS IN
Function PROJECTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P7.6

PROPERTIES:

DEFINITION:
ENTER AMENDMENTS

ASSOCIATIONS:

OCCURS IN
Function PROJECTS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P8.1

PROPERTIES:

DEFINITION:
DETERMINE INSPECTION RANKING

ASSOCIATIONS:

OCCURS IN
Function MAINTENANCE SCHEDULING
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P8.2

PROPERTIES:

DEFINITION:
MAKE SCHEDULE

ASSOCIATIONS:

OCCURS IN
Function MAINTENANCE SCHEDULING
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Figure A-2: BAMEO Functions Decomposition (Cont'd)

Information Engineering Workbench Report
Object Summary Report
April 11, 1989 14:25:25 CHRIS ADAM

Process P9.1

PROPERTIES:

DEFINITION:
COMPILE-AMMIS-DATA

ASSOCIATIONS:

OCCURS IN
Function AMMIS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Process P9.2

PROPERTIES:

DEFINITION:
UPDATE AMMIS REQUIREMENTS

ASSOCIATIONS:

OCCURS IN
Function AMMIS
IS RESPONSIBILITY OF
Organizational Unit BAMEO

Figure A-2: BAMEO Functions Decomposition (Cont'd)

2. **Step (2)-> Business System Plan (BSP) Study:** This step consist of the following different parts:

- a. The first 2 parts are step 1&3 of IEM,
- b. The third part is the analysis of the current business and system relationships.

Four different kind of association matrices can be done:

- i. Organization units vs business process/function (figure A-3),
- ii. Organization units vs DP systems,
- iii. DP systems vs business process/function, and
- iv. DP systems vs Data files.

	ISSUE	AMMIS	MAINTENANCE SCHEDULING	PROJECTS	LOG SET MAINTENANCE	REPORTS	LOGISTIC SUPPORT	PERSONNEL QUALIFICATIONS	MAINTENANCE	SERVICING	BASE AIRCRAFT MAINTENANCE	BAMEO
BAMEO	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ASO			✓	✓		✓			✓	✓		✓
ARO					✓	✓					✓	✓
AVSO					✓	✓			✓	✓		✓
MSO					✓	✓			✓	✓		✓
ARMSO					✓	✓						✓
PHOTOSO					✓	✓						✓
AMADO						✓						✓
AMCRO						✓	✓					✓

Organizational Unit is responsible for Function

Figure A-3: BAMEO Org. Units versus Funct/Process

For the last 3 matrices no examples were given since presently there are no existing DP systems for the aircraft maintenance at base level. Figure A-3 could be used to know who to ask for the required data by each process/function. It could also be used to know who to see for the next step which are the interviews. The 3 other association matrices are basically there to understand the existing DP systems, since there is no existing one, this part is not required. The next 2 steps consist of doing interviews with executives and reduce as well as organize the interviewed data. These 2 steps were not done since they are much too time consuming (at least 4 to 6 months) and the required information was already at ref [85].

- c. The heart of the BSP study consist of interview with high level executives to understand their perspectives about the business, their information needs and problems they have. The interviewed executives will normally be no more than 2 levels below the BAMEO.
- d. The next step is to reduce and organize the interviewed data. The interviewed data will be in the figure A-4 format.

Problem	Solution	Value Statement	Information Syst. Needs	Business Process Affected
Business Process Causing the Problem				

Figure A-4: Format of Interviewed Data

As we can see from this format the data is used to identify the problems.

Only the problems related to possible future information systems will be of interest. With these problems, only the information system needs with their corresponding processes are identified. This is useful for the next step which is the design of the information architecture.

- e. The final step is to do 2 things in parallel. The first one is to review the present information system management and does not apply to the ADAM project. The second one is to define an information architecture which is done at step 7 of IEM.

3. Step (3)-> Subject Database Study:

- a. The basic subject area that the BAMEO organization works with have been identified as follow:
 - i. Aircraft,
 - ii. Equipment, and
 - iii. Personnel.

- b. The next step is to look at the list of processes and write down what data classes(Entities) are used as input and output to each processes. The figure A-5 list of processes has been limited to process 1.1 to 3.6 and 5.1 to 7.6. In this figure, an External Agent is defined as an external world object which is not a part of the information system. For example, for process LP 3.2, NDHQ is an external agent.

<u>Process:</u>	<u>Input:</u>	<u>Output:</u>
LP1.1	Flying-Time, Aircraft Equipment.	Forms
LP1.2	Forms	Defect, Manhours-of-Repairs, Forms
LP1.3	Forms	Forms
LP1.4	Forms	Forms, Defect, Manhours-of-Repairs
LP2.1	Flying-Time, Forms, Defect, Manhours-of-Repair, Log-Set.	Forms, Defect, Manhours-of-Repair
LP2.2	Forms, Defect, Tool-Kit, Pers-Info, Manhours-of-Repair, Modification, Pers-Qualif, Pers-Qualif-Info.	Forms, Defect, Tool-Kit, Manhours-of-Repair
LP2.3	Log-Set, Forms, Defect, Tool-Kit, Manhours-of-Repair, Pers-Info, Pers-Qualif-Info, Modifications.	Forms, Defects, Tool-Kit, Manhours-of-Repair.
LP3.1	External Agent (Operator)	Pers-Qualif-Info, Pers-Qualif
LP3.2	External Agent (NDHQ)	Pers-Info
LP3.3	External Agent (Operator)	Pers-Info
LP3.4	Pers-Qualif, Pers-Info	Pers-Info
LP3.5	External Agent (Command HQ)	Pers-Qualif-Info, Pers-Qualif
LP3.6	External Agent (Base Services)	Pers-Qualif-Info
LP5.1	Log-Set, Forms, Defect, Tool-Kit, Manhours-of-repair, Flying-Time, Abnormal Occurences.	External Agents.(Command HQ, Operators)
LP5.2	Log-Set, Forms, Defect, Manhours-of-Repairs, Modifications, Tool-Kit	External Agents (Command HQ, NDHQ, Operators)
LP5.3	Modifications, Flying-Time, Personnel, Forms, Defect, Tool-Kit, Manhours-of-Repairs	External Agents (Base Services)
LP5.4	Pers-Info, Forms, Defect, Tool-Kit, Manhours-of-Repairs, Log-Set	External Agents (Operators)
LP5.5	Pers-Info	External Agents (Command HQ)
LP6.1	Log-Set	Flying-Time
LP6.2	Modification, Forms, Defect, Tool-Kit, Manhours-of-Repairs	Transfer, Flying-Time, Depot-Level-Inspection
LP6.3	Forms, Defect, Tool-Kit, Manhours-of-Repairs, Flying-Time	Equipment
LP6.4	Depot-Level-Inspection, Equipment	Depot-Level-Inspection, Equipment
LP6.5	Forms, Defect, Tool-Kit, Manhours-of-Repairs, Modifications	Modifications, Abnormal-Occurences
LP6.6	Equipment	Forms, Defect, Manhours-of-Repair

Figure A-5: Input and Output Data Classes of each Processes

<u>Process:</u>	<u>Input:</u>	<u>Output:</u>
LP7.1	External Agent (Operator)	Modifications
LP7.2	External Agent (Fault)	Forms
LP7.3	Modifications, Forms	Forms
LP7.4	Forms	Modifications, Forms
LP7.5	External Agent (NDHQ)	Forms, Modifications
LP7.6	External Agent (NDHQ)	External Agent (Technical Orders)

Figure A-5: Input and Output of each Processes

- c. The last step is to cross-refer that list of data classes to the different subject area of the BAMEO organization. This is done to ensure that the step "a" subject area cover all the possible step "b" data classes, to see if no subject area are missing. As per figure A-6 all the data classes required by process 1.1 to 2.3 and 7.1 to 7.6 are covered by the 3 subject databases.

	EQUIPMENT		
	AIRCRAFT		
	PERSONNEL		
PERS-QUALIF-INFO	✓		
PERS-QUALIF	✓		
MODIFICATIONS			✓
PERS-INFO	✓		
LOG-SET			✓
HANHOUS-OF-REPAIR	✓		
TOOL-KIT			✓
DEFECT			✓
FORMS			✓
AIRCRAFT-EQUIPMENT			✓
FLYING-TIME		✓	

Entity Type is involved in Subject Area

Figure A-6: Cross-Reference Between Data Classes and Subject Area

MOD", and "PERS-INFO" is personnel and therefore should be classified as confidential.

The access to those entities should be controlled at base level and if this data has to leave the base, then appropriate action should be performed so that the security of this data can not be violated.

6. Step (6)-> Entity Analysis: Using the information developed at the previous 5 steps, it is now possible to derive all the entities and their corresponding relationships to create an E-R diagram. Figure A-8 has thus been created. In that first figure one-to-one, one-to-many and many-to-many relationships are allowed.

4. **Step (4)-> Critical Success Factor (CSF) Analysis:** Figure A-7 identifies 2 critical success factors within the BAMEO organization with their corresponding objectives.

CONTROL ACCESS TO CLASIFIED DATA		
SUPPORT A/C SERVICING SET		
TIME CONSTRAINT	✓	
CLASSIFIED INFORMATION		✓

Critical Success Factor impacts Goal

Figure A-7: Critical Success Factor versus Objectives

Critical Success Factor: CLASSIFIED INFORMATION**Definition**

THE INFORMATION OF CERTAIN DATA ELEMENTS OF THE ORGANIZATION IS CONFIDENTIAL TO SECRET AND THE ACCESS TO IT COULD JEOPARDIZE THE CONFIDENTIALITY OF ITS PERSONNEL OR THE SUCCESS OF ITS OPERATIONS.

PROPERTY	VALUE	
ASSOCIATION	TYPE	NAME
Impacts	Goal	CONTROL ACCES TO CLASSIFIED DATA

Critical Success Factor: TIME CONSTRAINT**Definition**

UNDER CERTAIN TIME OF HEIGHTENED HOSTILITY AND/OR IN TIMES OF INCREASED OPERATIONAL COMMITMENT, THE SYSTEM SHOULD OPERATE UNDER CERTAIN TIME CONSTRAINTS TO OUTPERFORM THE ENEMY.

PROPERTY	VALUE	
ASSOCIATION	TYPE	NAME
Impacts	Goal	MAX A/C AVAIL BY SUP A/C MAINT

Figure A-7: Critical Success Factor versus Objectives (Cont'd)

Goal: CONTROL ACCES TO CLASSIFIED DATA**Definition**

ACCESS TO CLASSIFIED DATA ELEMENTS HAS TO BE CONTROLLED AT BASE LEVEL.

PROPERTY	VALUE	
ASSOCIATION	TYPE	NAME
Is Impacted by	Critical Success Factor	CLASSIFIED INFORMATION

Goal: MAX A/C AVAIL BY SUP A/C MAINT**Definition**

ANY INFORMATION PERTAINING TO MAINTENANCE FUNCTION OF THE AIRCRAFT (A/C) IS REQUIRED TO BE ACCURATE WITHIN ONE DAY OF OCCURENCE.

PROPERTY	VALUE	
ASSOCIATION	TYPE	NAME
Is Impacted by	Critical Success Factor	TIME CONSTRAINT

Figure A-7: Critical Success Factor versus Objectives (Cont'd)

5. Step (5)-> Management by Objectives: There are various levels of objectives. In figure A-7 only the objectives (or sub-objectives) which falls within the scope of responsibility of the BAMEO organization are considered. Let us show what data is needed for the measurement of the 2 objectives/goals indicated in figure A-7. The first objective is a subsidiary of the general objective 'Maximize aircraft availability'. However, only the sub-objective 'Maximize aircraft availability by supporting aircraft maintenance' falls within the scope of responsibility of the BAMEO organization. This sub-objective can be translated as follows: Support the aircraft servicing set composed of CF335, 336, 338, 339, 343, 349, and 543 forms. (These forms already exist in the BAMEO organization.).

Now what data is needed for the measurement of this objective? Under the ADAM project, it has been decided to automate the CF349 and 543 forms and to validate the entered data (Validate the data entry and accuracy). Each field of the form could correspond to an attribute as well as the form type itself. All the form types together could be an entity.

The second objective falls within the scope of responsibility of the BAMEO organization and could also be translated as follow: Access to classified data elements has to be controlled at base level.

What data is needed for the measurement of this objective? First to answer that question, the classified data has to be identified. The data related to the entity "FLYING-TIME" could have strategic impact on the operations and therefore should be classified as secret.

The data related to the entities "PERS-QUALIF", "PERS-QUALIF-INFO", "QUAL-

MOD", and "PERS-INFO" is personnel and therefore should be classified as confidential.

The access to those entities should be controlled at base level and if this data has to leave the base, then appropriate action should be performed so that the security of this data can not be violated.

6. Step (6)-> Entity Analysis: Using the information developed at the previous 5 steps, it is now possible to derive all the entities and their corresponding relationships to create an E-R diagram. Figure A-8 has thus been created. In that first figure one-to-one, one-to-many and many-to-many relationships are allowed.

Figure A-8: E-R Diagram

7. **Step (7)-> Business System Chart:** This step shows how to group subject database into implementable systems and subsystems. Reference [14] proposes a method which order subject databases according to the occurrence of process into the life cycle. Then the subject databases are grouped into implementable systems since it would be almost impossible to implement subject databases for the whole enterprise at once.

In this project, there are only 3 subject databases involved, therefore it is reasonable to put them all in the same system. To implement 3 subject database in parallel is definitely possible.

8. **Step (8)-> Structured Entity Chart:** This step is to determine the depth of the different entities in an entity chart. To be able to determine the depth to put order in an entity chart, only one-to-one and one-to-many relationships are allowed. Besides that in many-to-many relationships 2 possible problems can occur [35]:
 - a. **Difficulties of physical representations:** One-to-many relationship can be represented by a chain of pointers linking each "A" parent entity to its associated "B" child entities. This simple method cannot be used with the many-to-many mapping. Instead, there is to be a file of A entities and a file of B entities, and separate information showing how they are related.
 - b. The second problem is that in practice when a many-to-many association is used there will usually be intersection data associated with it sooner or later.

Because of these 2 possible problems, it is usually best to avoid many-to-many relationships and instead create an extra record having concatenated key A+B when the database is first implemented. Therefore figure A-8 with many-to-

many relationships has been transformed to figure A-9 where only one-to-many relationships exists.

Figure A-9: E-R Diagram with only One-to-Many Relationships

Figure A-11: Structured Entity Chart

9. Step (9)-> Entity Cluster Analysis: Three approaches exist to cluster entities

Figure A-10: IEW Structure Entity Chart

Entity Model

April 11 1985 5 00 33

Note: We can see that IEW does not really support the design of structure entity chart. IEW does not support it well graphically and the depth of each entity is not computed automatically. Figure A-11 show how a real structure entity chart should appear.

Figure A-11: Structured Entity Chart

9. Step (9)-> Entity Cluster Analysis: Three approaches exist to cluster entities

together:

- a. **Frequency of Association Path Between Entities:** To be able to do this approach, the 20/80 rule have been followed. It is a heuristic which assures that 80 percent of the accesses are obtained by analyzing the most frequent 20 percent of the applications [44].

Therefore the frequency of the 20 percent most frequent transactions would have to be found as well as the required entities. Due to the amount of required information this approach has not been done. For example the transaction "Give the total man-hours of repairs done by Cpl Jones during the last week. " would have to be specified as per figure A-12.

Frequency: 52 times/year

Figure A-12: Example of Transaction Specification.

- b. **Affinity Analysis:** For the affinity analysis approach, first using the IEW association matrix, the entities used by each process would have to be mapped as per figure A-13.

		Flying-Time				
		Aircraft-Equipment				
		Forms				
		Log-Set				
		Equipment				
Process						
P 2 . 1		✓	✓	✓	✓	
P 1 . 4		✓		✓		
P 1 . 3		✓		✓		
P 1 . 2				✓		
P 1 . 1		✓			✓	✓

Process involves Entity Type

Figure A-13: Process versus Entity Association Matrix

The affinity is computed as per figure A-14.

$a(\text{Aircraft-Equipment}) = \text{Number of processes using Entity (Aircraft-Equipment)} = 2.$

$a(\text{Aircraft-Equipment, Forms}) = \text{Number of processes using entity (Aircraft-Equipment) \& (Form)} = 1$

$a(\text{Form}) = \text{Number of processes using entity (Form)} = 4.$

$$\left[\frac{a(\text{Aircraft-Equipment, Form})}{a(\text{Aircraft-Equipment})} + \frac{a(\text{Aircraft-Equipment, Form})}{a(\text{Form})} \right] / 2$$

multiplied by 100 = Affinity.

$(1/2 + 1/4) / 2$ multiplied by 100 = 38.

Figure A-14: Example of Affinity Analysis

For illustration purpose the affinity analysis has been done only on the five processes and five entities of figure A-13. The results are at figure A-15.

Association from which the Affinity Analysis is based

Process involves Entity Type

Affinity Processing Limits

Minimum New Group Affinity	= 75
Minimum Merge Group Affinity	= 75
Minimum Add To Group Affinity	= 65
Minimum Usable Affinity	= 64

Report Printing Flags

Usages Matrix Printed?	Yes
Affinity Matrix Printed?	Yes
Grouping Progress Printed?	Yes

Affinity Processing Flags

Average Affinity for Pairs?	Yes
Include Non-Leaf Objects?	No

Project Template

None entered

Figure A-15: Affinity Analysis

Leaf Level Right Object Instances

- 1 FLYING-TIME
- 2 AIRCRAFT-EQUIPMENT
- 3 FORMS
- 4 LOG-SET
- 5 EQUIPMENT

Leaf Level Left Object Instances

- 1 DETERMINE-SERVICING-RQMNTS (P1.1)
- 2 SERVICING-INSPECTIONS (P1.2)
- 3 LOAD-CHANGES (P1.3)
- 4 FUEL-AIRCRAFT (P1.4)
- 5 OPEN-MAINT-ACTION (P2.1)

Figure A-15: Affinity Analysis (Cont'd)

Affinity Matrix

	1	2	3	4	5
1	=	75	.	.	63
2	75	=	38	75	75
3	.	38	=	63	75
4	.	75	63	=	63
5	63	75	75	63	=

Usage Matrix

	1	2	3	4	5
1	*	*	.	.	*
2	.	.	*	.	.
3	.	.	*	.	*
4	.	.	*	.	*
5	.	*	*	*	*

Figure A-15: Affinity Analysis (Cont'd)

Cycle Number	Affinity	Between Objects	Action Taken	Group Containing
1	75.00	1 & 2	Created	1
2	75.00	2 & 4	Add To	1
3	75.00	2 & 5	Add To	1
4	75.00	3 & 5	Add To	1

All remaining affinities are below the new group, add, or merge affinity thresholds you have chosen

The following Objects will be added to the Miscellaneous group:

Group 1 (containing object 1):

- 1 FLYING-TIME
- 2 AIRCRAFT-EQUIPMENT
- 3 FORMS
- 4 LOG-SET
- 5 EQUIPMENT

Figure A-15: Affinity Analysis (Cont'd)

- c. The last approach is to cluster entities together by groups of subject databases determined at step 1.3. It can be determined logically which entity belong to which group of subject databases.

Note: The only approach not supported by IEW is the "frequency of association path between entities" .

10. Partitioned Entity Chart (10): The relationships still uniting the different groups of entity formed during the entity cluster analysis are removed to create independent group of entities as per figure A-16 for the aircraft subject database, figure A-17 for the equipment subject database and figure A-18 for the personnel subject database.

Figure A-16: Aircraft Subject Database

AIRCRAFT

April 11, 1983 15:19 20

EQUIPMENT

Apr. 11, 1989 15 22 24

Figure A-17: Equipment Subject Database

PERSONNEL

APR 11 1985 15 35 14

Figure A-18: Personnel Subject Database

Business Analysis Phase.

The business analysis phase of IEM focus on building a “BOTTOM-UP” plan for the entire BAMEO organization. The following paragraphs will describe the different steps of the business analysis phase:

1. Step (11)-> Transition from “TOP-DOWN” to “BOTTOM-UP”: As mentioned at section 1.4.2, using the partitioned entity charts of the planning phase, the analyst has a way to develop the overall data of the enterprise in a partitioned group of entities of reasonable size.
2. Step (12)-> User Views: The attributes of the different user Views are associated to the appropriate entity of the entity chart.
 - a. View 1: Details about the different equipments are required. See associated figure A-19.

Serial #	Equipment Part #	Equipment -Type	W.U.C.	Life-Usage- Since-New	Life-Usage- Since-Overhaul	Inspector
		Rank	Manufac tured-by	NATO-Stock- Number	Overhauled-By	Date- Overhauled

Figure A-19: View 1

- b. View 2 : Some equipment are composed of one or many aircraft equipment. Details of these aircraft equipment are required. An aircraft equipment can have many defects. Information about these defects is required. See associated figure A-20.

Figure A-20: View 2

- c. **View 3** : When a defect is identified information about the defect is required and a CF 349 form is raised. If supplement work has to be done then a CF349 Supp form is raised. If the work has to be done off the aircraft then a CF543 form is raised. The required tool-kit is identified. See associated figure A-21.

Figure A-21: View 3

- d. View 4: When a defect is corrected, details of repairs has to be kept. The required tool-kit is closed. See associated figure A-22.

Figure A-22: View 4

Note: IEW does not support user views.

3. Step (13)-> Users Data Flow Diagrams: Using ref [85] data a Data Flow Diagram has been built for the maintenance function. See figure A-23.
4. Step (14)-> Compare DFDs and User Views: Using the duality between DFD and E-R diagrams, the content of the maintenance function DFD of figure A-23 is compared to the content of the 4 user views of figure A-19 to A-22. For example, it can be verified that the content of the data store "Maintenance Required" of figure A-23 and A-24 includes the entities "Defect" and "Form" of figure A-20 to A-22. The content of the different views is compared to the content of the different DFD diagrams to ensure that nothing is missing on both sides.

Figure A-23: DFD of Maintenance Function

MAINTAIN

Apr 11 12. 1998 18:22:44

Figure A-24: Data Store "Maintenance Required"

5. Step (15)-> Canonical Synthesis: The representation of the different user view of step 12 should be in third normal form, which means the following:

- a. The entities have no transitive dependencies,
- b. If a concatenated key is used, this key is regarded as one data item, and
- c. The key completely identifies the attributes.

The canonical synthesis consist of first normalizing each view individually and then by following different steps, the different views are merged together to give a final normalized schema. To illustrate that synthesis, these different steps as described at [36] will be applied to the 4 views of figure A-19 to A-22.

- a. Each user's view have to be transformed to a bubble chart graph where the link/dependency between attributes is represented by 2 types of arrow, one for single and the other for many dependencies. During this transformation, the resulting views have to be in third normal form. For example, where a concatenated key is used, ensure that all single-head arrow leaving that key go to an attribute which is dependent on that full concatenated key, not only part of it. Applied to the 4 views, this step will give the following results:
 - i. View 1: in the transformation the key "Serial#" and "Equip. Part#" have to be concatenated together. See results at figure A-25.

Figure A-25: View 1

- ii. **View 2:** In the transformation the concatenated key of "Serial# + Equip.Part#" and "Serial# + Aircraft Equip. Part#" have to be created. The 2 keys called "Equip. Part#" are homonyms, the second is transformed to "Aircraft Equip. Part#". See results at figure A-26.

Figure A-26: View 2

- iii. View 3: A many-to-many relationship exists, therefore a concatenated key composed of "Tool-Kit-Serial# + Defect-ID" is created. See results at figure A-27.

Figure A-27: View 3

- iv. **View 4:** As for View 3 the same many-to-many relationship exists, therefore the same concatenated key is created. See results at figure A-28.

Figure A-28: View 4

The next 4 steps have to be applied every time user views are merged together:

- a. Merge the user's view together. Check for any synonyms or homonyms, removing or renaming them if they appear,
- b. In the resulting graph, distinguish between the attributes nodes and the primary-key nodes. Mark the primary keys in some way,

- c. For each association between keys, add the inverse association if it is not already on the graph. If this results in a many-to-many association between keys, determine whether the inverse association would ever be used in real life. If it could be used at any time in the future, replace it by introducing an extra concatenated key incorporating the key data items that were linked, and
- d. For any associations that are candidates for removal, check carefully that their meaning is redundant, if so, remove them.

Following the 4 steps described above to our for Views would give the following results:

- a. Merging View 1 & View 2: The 2 attributes "Equip-Type" and "Type" are synonyms, therefore attribute "Type" has been removed. See results at figure A-29,

Figure A-29: Merging View 1 & View2

- b. Merging Result to View 3: The 2 attributes "Defect-Info" and "Defect-Description" are synonyms, therefore attribute "Defect-Info" has been removed. See results at figure A-30, and

Figure A-30: Merging Result to View 3

c. Merging Result to View 4: See results at figure A-31.

Figure A-31: Merging Result to View 4

Figure A-31: Merging Result to View 4 (Cont' d)

The next step is to identify the root keys. Where a root key is a key with no single arrow leaving it to another key (This is the key of an entity of depth 1). The root key entity is putted at the top of the diagram. As for step 8 of the planning phase (Structure Entity Chart) an entity of depth 2 is putted below a depth 1 entity and so on. See results at figure A-32.

Figure A-32: Final Result

Figure A-32: Final Result (Cont'd)

This step is to observe if the graph contains any isolated attributes. An isolated attribute is a bubble with no single-arrow association entering or leaving it. If there is, that attribute has to be linked to the appropriate group of attributes (By logical reason). Figure A-32 has no isolated attribute.

The next step is to adjust the graph to avoid any intersecting attributes. An intersecting attribute is an attribute with more than one single-arrow association entering it. It can be handled in 3 different ways.

- a. All but 1 link to it may be replaced with equivalent links via an existing key,
- b. Redundant versions of it may be connected to each associated key, and
- c. It may be made into a key with no attributes.

Figure A-32 has no intersecting attributes.

The next step is to redraw the attributes arranged into groups (entities), each having one primary key and its associated attributes. The entities may be offset from the left to indicate their “depth” under the root entity. See figure A-33 for results.

Figure A-33: Corresponding Entities

Then the last step of the canonical synthesis is to identify secondary key path. A secondary key is defined as an attribute (not a key) with one or more double-arrow associations leaving it. Figure A-33 has no secondary path.

Once the canonical synthesis is completed the next thing to be done is to apply the stability analysis. This analysis consists of the following 5 steps:

- a. The first step consists of looking for all the possible candidate primary keys. Each attribute which is not by itself a primary key should be evaluated to see if it can become a primary key in the future. Figure A-33 has no candidate primary keys.
- b. The next step is to identify intersecting attributes. These attributes are handled as in the canonical synthesis. Figure A-33 has no intersecting attributes.
- c. If the model contains the attribute A which identifies B and B which identifies A then these attributes/key are functionally equivalent. If this equivalence is likely to remain in the future then these keys should be concatenated if not they should remain separated. Figure A-33 has no functionally equivalent attributes.
- d. The cycles have to be removed by the addition of a cross-association which causes 2 or more associations in the cycle to be deleted. Figure A-33 has no cycles.
- e. With high-volume secondary-key paths the model may be split into 2 disjoint model for machine performance reasons.

It is to note that every steps of the canonical synthesis and the stability analysis has to be done manually since IEW does not support any of them.

6. Step (16) & (16A) -> Structured Data Model: Having applied the different steps of the stability analysis to figure A-33 (data model), the result is a stable structured data model. Step (16A) requires to compare the step (8) structured entity chart (Figure A-11) to figure A-33. This comparison demonstrates that figure A-33 is

not complete since numerous entities are missing. Due to the time and resources required, the view integration of all the possible views was not done in the scope of this thesis. Instead, using the entities of the step (8) structured entity chart and determining their associates attributes from [85], a complete structured data model was derived at figure A-34.

7. Step (17)-> Overall DFDs: Merging together the DFDs of the figure A-2 different processes which occur in the same function, an overall DFD is created for each function. Figure A-35 represents a DFD for the Maintenance Function.
8. Step (18)-> Compare Overall DFDs with Overall Data Model: After having compared the overall DFD of each function to the overall Data Model some adjustments can be made. For example, if we compare the figure A-35 DFD for the "Maintenance Function" to the figure A-34 Data Model some adjustments have to be made to figure A-35. The data store "Parts In" (Parts Inventory) of figure A-35 can not be covered by any of the figure A-34 entities. Therefore the "Parts-In" data store has to be transformed as an external agent, this also apply to the "Tech Ord" and "Maintenance Schema" data stores. The results are at figure A-36.

ASSOCIATION	TYPE	NAME
Is Involved in	Subject Area Process	EQUIPMENT FUEL-AIRCRAFT (P1.4) LOAD-CHANGES (P1.3) SERVICING-INSPECTIONS (P1.2) OPEN-MAINT-ACTION (P2.1) DEFECT FORM-TYPE/NUMBER DEFECT-ID RECTIFIED-BY RECTIFICATION-HOURS INSPECTED & PASSED-BY DATE-INSPECTED INDEPENDANT-CHECK-REQUIRED CERTIFIED-SERVICEABLE-BY ENTERED-BY PARTS-PAGE# AMMIS-REPORTABLE PUT-U/S-BY AREA FIXER-UNIT WORK-ORDER SEQUENCE TRADE ORIGINATOR-UNIT AIRCRAFT-TYPE
RAISE Is Described by	Entity Type Attribute Type	
Entity Type: LOG-SET		
ASSOCIATION	TYPE	NAME
Is Involved in	Subject Area Process	EQUIPMENT OPEN-MAINT-ACTION (P2.1)
IDENTIFIED-BY Is Described by	Entity Type Attribute Type	EQUIPMENT LOG-SET# SERIAL# EQUIP-PART# CF351-DATA CF357-DATA CF359-DATA CF354-DATA CF363-DATA CF352-DATA CF353-DATA CF358-DATA CF350-DATA

Figure A-34: BAMEO Data Model Description (Cont'd)

Entity Type: ABNORMAL-OCCURENCES

ASSOCIATION	TYPE	NAME
Is Involved in	Subject Area	EQUIPMENT
OCCUR	Entity Type	AIRCRAFT-EQUIPMENT
Is Described by	Attribute Type	DATE-OF-OCCURENCE
		AIRCRAFT-EQUIP-PART#
		EVENT
		SERIAL#
		DAILY-APU-STARTS
		TYPE-OF-OCCURANCE
		AIRCRAFT-TYPE

Entity Type: AIRCRAFT

ASSOCIATION	TYPE	NAME
IS-COMPOSED-OF	Entity Type	AIRCRAFT-EQUIPMENT
HAS	Entity Type	PERIODICS
Is Involved in	Subject Area	AIRCRAFT
FLY-FOR	Entity Type	FLYING-TIME
GO-ON	Entity Type	TRANSFERS
SEND-ON	Entity Type	DEPOT-LEVEL-INSPECTIONS
Is Described by	Attribute Type	AIRCRAFT-TYPE
		AIRCRAFT-NUMBER
		DATE-OF-ACCEPTANCE
		ACCEPTED-BY
		FLYING-HOURS-ON-ACCEPTANCE
		PERIODIC-DUE
		CURRENT-FLYING-HOURS
		CURRENT-NORMAL-LANDING
		CURRENT-ARRESTED-LANDING
		SUPP-DUE
		MANUFACTURED-BY
		AIRFRAME-HOURS

Entity Type: AIRCRAFT-EQUIPMENT

ASSOCIATION	TYPE	NAME
SUFFERS	Entity Type	ABNORMAL-OCCURENCES
IS-COMPOSED-OF	Entity Type	EQUIPMENT
HAS	Entity Type	DEFECT
Is Involved in	Subject Area	EQUIPMENT
	Process	DETERMINE-SERVICING-RQMNTS (P1.1)
		OPEN-MAINT-ACTION (P2.1)
MODIFIES	Entity Type	MODIFICATIONS

Figure A-34: BAMEO Data Model Description (Cont'd)

COMPOSE Is Described by	Entity Type Attribute Type	AIRCRAFT AIRCRAFT-EQUIP-PART# SERIAL# AIRCRAFT-TYPE AIRCRAFT-NUMBER PERIODIC-DUE SUPP-DUE DATE-INSTALLED LOCATION LIFE-USAGE-INDICE INSTALLATION-349#
Entity Type: DEFECT		
ASSOCIATION	TYPE	NAME
RAISE	Entity Type	FORMS
REPAIRED-FOR	Entity Type	MANHOURS-OF-REPAIR
HAS	Entity Type	TOOL-KIT-DEFECT
Is Involved in	Subject Area	EQUIPMENT
ON	Entity Type	AIRCRAFT-EQUIPMENT
Is Described by	Attribute Type	DEFECT-ID AIRCRAFT-EQUIP-PART# SERIAL# RECTIFICATION SUPP-WORK-RECTIFICATION SUPP-DATA DEFECT-DESCRIPTION HOW-FOUND-CODE MISSION-EFFECT-CODE SUPP-WORK-DEFECT AIRCRAFT-TYPE
Entity Type: DEPOT-LEVEL-INSPECTIONS		
ASSOCIATION	TYPE	NAME
HAS-BEEN	Entity Type	AIRCRAFT
HAS	Entity Type	DEPOT-LEVEL-MODIFICATIONS
Is Involved in	Subject Area	AIRCRAFT
Is Described by	Attribute Type	DATE-DLI-COMPLETED AIRCRAFT-TYPE AIRCRAFT-NUMBER DLI-INSPECTOR DLI-CONTRACTOR
Entity Type: DEPOT-LEVEL-MODIFICATIONS		

Figure A-34: BAMEO Data Model Description (Cont'd)

ASSOCIATION	TYPE	NAME
Is Involved in DONE-CN Is Described by	Subject Area Entity Type Attribute Type	AIRCRAFT DEPOT-LEVEL-INSPECTIONS DATE-DLM-EMBODIED CERTIFICATION-OF-EMBODIEMENT AIRCRAFT-TYPE EMBODIED-BY CONTRACTOR-REF-NUMBER CF-MODIFICATION-NUMBER DATE-DLI-COMPLETED MODIFICATION-DESCRIPTION
Entity Type: EQUIPMENT		
ASSOCIATION	TYPE	NAME
HAS Is Involved in COMPOSE Is Described by	Entity Type Subject Area Entity Type Attribute Type	LOG-SET EQUIPMENT AIRCRAFT-EQUIPMENT SERIAL# EQUIPMENT-PART# PERIODIC-DUE LIFE-USAGE-INDICIE EQUIP-TYPE WORK-UNIT-CODE LIFE-USAGE-SINCL-NEW LIFE-USAGE-SINCE-OVERHAUL INSPECTOR RANK INSTALLED-ON
Entity Type: FLYING-TIME		
ASSOCIATION	TYPE	NAME
BEING-FLIGHT-FOR Is Involved in Is Described by	Entity Type Subject Area Process Attribute Type	AIRCRAFT AIRCRAFT DETERMINE-SERVICING-RQMNTS (P1.1) FLYING-DATE AIRCRAFT-TYPE AIRCRAFT-NUMBER DAILY-FLYING-HOURS CURRENT-FLYING-HOURS DAILY-ARRESTED-LANDINGS DAILY-NORMAL-LANDINGS
Entity Type: FORMS		

Figure A-34: BAMEO Data Model Description (Cont'd)

ASSOCIATION	TYPE	NAME
Is Involved in	Subject Area Process	EQUIPMENT FUEL-AIRCRAFT (P1.4) LOAD-CHANGES (P1.3) SERVICING-INSPECTIONS (P1.2) OPEN-MAINT-ACTION (P2.1) DEFECT FORM-TYPE/NUMBER DEFECT-ID RECTIFIED-BY RECTIFICATION-HOURS INSPECTED & PASSED-BY DATE-INSPECTED INDEPENDANT-CHECK-REQUIRED CERTIFIED-SERVICEABLE-BY ENTERED-BY PARTS-PAGE# AMMIS-REPORTABLE PUT-U/S-BY AREA FIXER-UNIT WORK-ORDER SEQUENCE TRADE ORIGINATOR-UNIT AIRCRAFT-TYPE
RAISE Is Described by	Entity Type Attribute Type	
Entity Type: LOG-SET		
ASSOCIATION	TYPE	NAME
Is Involved in	Subject Area Process	EQUIPMENT OPEN-MAINT-ACTION (P2.1)
IDENTIFIED-BY Is Described by	Entity Type Attribute Type	EQUIPMENT LOG-SET# SERIAL# EQUIP-PART# CF351-DATA CF357-DATA CF359-DATA CF354-DATA CF363-DATA CF352-DATA CF353-DATA CF358-DATA CF350-DATA

Figure A-34: BAMEO Data Model Description (Cont'd)

CF355-DATA
CF356-DATA
AIRCRAFT-TYPE

Entity Type: MANHOURS-OF-REPAIR

ASSOCIATION	TYPE	NAME
PER	Entity Type	VALID-ENTRY
Is Involved in	Subject Area	PERSONNEL
REPAIR	Entity Type	DEFECT
Is Described by	Attribute Type	DATE-OF-ENTRY
		TOTAL-MANHOUR
		NUMBER-OF-PERSONNEL
		TRADE
		TYPE-OF-ENTRY
		WORK-ORDER-NUMBER
		WORK-ORDER-TYPE
		PARENT-UNIT

Entity Type: MODIFICATIONS

ASSOCIATION	TYPE	NAME
MODIFIED-BY	Entity Type	AIRCRAFT-EQUIPMENT
Is Involved in	Subject Area	EQUIPMENT
Is Described by	Attribute Type	LEAFLET#
		SERIAL#
		AIRCRAFT-EQUIP-PART#
		DATE-OF-LEAFLET-RECEIPT
		LEAFLET-DATE
		DATE-MOD/SI-COMPLETED
		AIRCRAFT-TYPE

Entity Type: PERIODICS

ASSOCIATION	TYPE	NAME
Is Involved in	Subject Area	AIRCRAFT
HAS	Entity Type	AIRCRAFT
Is Described by	Attribute Type	DATE-COMMENCED
		AIRCRAFT-TYPE
		AIRCRAFT-NUMBER
		HOURS-PERIODIC-DUE
		DATE-DUE
		HOURS-COMMENCED
		DATE-COMPLETED
		TYPE-OF-MODIFICATION

Figure A-34: BAMEO Data Model Description (Cont'd)

Entity Type: PERS-INFO

ASSOCIATION	TYPE	NAME
HAS	Entity Type	QUAL-MOD
OCCUPY	Entity Type	POSITION
Is Involved in	Subject Area	PERSONNEL
COMPOSED-OF	Entity Type	PERS-QUALIF-INFO
Is Described by	Attribute Type	SIN
		NAME
		INITIALS
		RANK
		TRADE
		MOC-CODE
		PARENT-UNIT

Entity Type: PERS-QUALIF

ASSOCIATION	TYPE	NAME
HAS	Entity Type	PERS-QUALIF-INFO
Is Involved in	Subject Area	PERSONNEL
HAS	Entity Type	VALID-ENTRY
Is Described by	Attribute Type	SIN
		TRADE
		WORK-ORDER-TYPE
		AIRCRAFT-TYPE
		WORK-UNIT-CODE
		TYPE-OF-ENTRY
		TITLE
		FROM-DATE
		EXPIRY-DATE
		PARENT-UNIT

Entity Type: PERS-QUALIF-INFO

ASSOCIATION	TYPE	NAME
HAS	Entity Type	PERS-INFO
Is Involved in	Subject Area	PERSONNEL
HAS	Entity Type	PERS-QUALIF
Is Described by	Attribute Type	SIN
		TRADE
		PARENT-UNIT

Entity Type: POSITION

Figure A-34: BAMEO Data Model Description (Cont'd)

ASSOCIATION	TYPE	NAME
Is Involved in HAS	Subject Area	PERSONNEL
Is Described by	Entity Type	PERS-INFO
	Attribute Type	POSITION
		SIN
		PARENT-UNIT
		SECTION
		CREW
Entity Type: QUAL-MOD		
ASSOCIATION	TYPE	NAME
Is Involved in CONSIST-OF	Subject Area	PERSONNEL
Is Described by	Entity Type	PERS-INFO
	Attribute Type	TYPE-OF-ENTRY
		SIN-AFFECTED
		DTG-ENTERED
		SIN-ENTERED-BY
		NAME-ENTERED-BY
		RANK-ENTERED-BY
		PARENT-UNIT
Entity Type: TOOL-KIT		
ASSOCIATION	TYPE	NAME
Is Involved in REPAIR	Subject Area	EQUIPMENT
Is Described by	Entity Type	TOOL-KIT-DEFECT
	Attribute Type	TOOL-KIT-SERIAL#
		TOOL-KIT-DESCRIPTION
		AIRCRAFT-TYPE
Entity Type: TOOL-KIT-DEFECT		
ASSOCIATION	TYPE	NAME
BEING-REPAIRED-BY	Entity Type	TOOL-KIT
Is Involved in	Subject Area	EQUIPMENT
PART-OF	Entity Type	DEFECT
Is Described by	Attribute Type	DEFECT-ID
		TOOL-KIT-SERIAL#
		TOOL-KIT-OPENED-BY
		TOOL-KIT-CLOSED-BY
		TOOL-KIT-INSPECTED-BY
		AIRCRAFT-TYPE

Figure A-34: BAMEO Data Model Description (Cont'd)

Entity Type: TRANSFERS

ASSOCIATION	TYPE	NAME
ON	Entity Type	AIRCRAFT
Is Involved in	Subject Area	AIRCRAFT
Is Described by	Attribute Type	DISPATCH-DATE
		AIRCRAFT-TYPE
		AIRCRAFT-NUMBER
		DISPATCH-UNIT
		RECEIPT-DATE
		RECEIPT-UNIT
		AUTHORITY-FCR-TRANSFER

Entity Type: VALID ENTRY

ASSOCIATION	TYPE	NAME
LOVE-BY	Entity Type	PERS-QUALIF
Is Involved in	Subject Area	PERSONNEL
CONSISTS	Entity Type	MANHOURS-OF-REPAIR
Is Described by	Attribute Type	TOTAL-MAN-HOURS
		DATE-OF-ENTRY
		QUALIF
		SIN
		TYPE-OF-ENTRY
		WORK-ORDER-NUMBER
		PARENT-UNIT

Figure A-34: BAMEO Data Model Description (Cont'd)

Figure A-35: Maintenance Function DFD

TRAINING

44-1 13. 2445 16 23. 44

Figure A-36: Maintenance Function DFD after Correction

System Design Phase.

The following paragraphs will describe the different steps of the IEM design phase applied to the BAMEO organization:

1. Step (19)-> Logical Access Map: For each sub-processes belonging to a process, a logical access map has to be designed [36]. Lets apply it to process LP5.5 "Personnel Report" to illustrate how logical access map are derived. Suppose process LP5.5 consists of the following sub-process. "Give the total man-hours of repair done by each Cpl during last week"

The following 5 steps of the logical access map would have to be applied to the above sub-processes:

- a. Establish the main entities with which the procedure is concerned.
- b. Examine the neighborhood of these entities, from the overall data model.
- c. Eliminate the unneeded records from the neighborhood and print the resulting subset data model.
- d. Establish the sequence of accesses through this subset data model. Draw this on the model.
- e. Print the logical access map vertically ready for annotation. Check that all required data items are present.

This would give the following results:

- a. Step 1: The main entities required by the application are "PERS-INFO" and "MANHOURS-OF-REPAIRS".
- b. Step 2: From figure A-9 it can be determined that the neighborhood of these entities include the following: "POSITION" , "PERS-INFO", "QUAL-

MOD", "PERS-QUALIF-INFO", "PERS-QUALIF", "VALID-ENTRY", "MANHOURS-OF-REPAIR" and "DEFECT".

- c. Step 3: Eliminating the unnecessary entities figure A-37 can be created. Here the main entity "MANHOURS-OF-REPAIR" is not required since the "total man-hours-of-repair" attribute is already in "VALID-ENTRY" entity.

Figure A-37: Logical Access Map
Main Entities

- d. Step 4&5: The last 2 steps are done together where the sequence of the accessed entities as well as the required attributes are marked as per figure A-38.

Figure A-38: Logical Access Map

IEW does not support logical access maps.

2. Step (20)-> Action Diagram: From the Logical Access Map of step(19), an Action Diagram can be derived as per figure A-39.

Figure A-39: An Example of an Action Diagram

IEW supports Action Diagrams as per figure A-40.

Figure A-40: IEW Action Diagram

3. **Step (21)-> Data Distribution Analysis:** IEM offers 11 different steps to follow for distribution analysis. The first 7 steps provide a framework for the management to decide at a subject database level if a subject database should be distributed or not. The last 4 steps provide only general guidelines to distribute entities they briefly indicate issues such as transmission cost, response time, availability etc... to be considered when it is time to allocate data, but none of them provide choices for algorithms to decide how the data should be distributed. These steps will not be further described here since the data distribution design is treated at chapter 5. IEW does not support any of the 11 steps of the IEM distribution analysis step.

Appendix B

CDT Implementation Design.

The following three parts will describe the design of the CDT. The first two parts will describe the objects used and created in order to support the user interfaces. These objects will fulfill the functions of the Graphics Interface, Diagraming and Analyzer tool part of a CASE tool as illustrated at figure 4-1. The third part describes the implementation of the supported model 16A algorithm. It fulfills the function of the fragment allocator illustrated at figure 4-1.

Part I: Objects Required from ARTTisan.

The following is a hierarchical illustration of the classes of objects used from the ARTTisan library of objects as well as a brief description of them. If a detail description of one of these classes is required see reference [80] for more details.

1. Figure B-1 illustrates the inheritance of the ARTTisan classes 'TArc' and 'TNode'. As can be seen, 'TArc' and 'TNode' inherits from 'ToolComponent' class. The 'ToolComponent' class responds to the user events. All events, either generated directly through the terminal or artificially through one of the system objects must pass through this object. This class main responsibility is to handle events directly from the keyboard and/or a mouse separating them into logical and graphical events for further processing. The TArc class inherits from the ToolComponent class. It is used to describe the most general form of an Arc. It

is used to handle all user input for arcs. It separates events for further processing by LArc and GArc classes. The TNode class inherits from the ToolComponent class. It is used to describe the most general form of a Node. It is used to handle all user input for Nodes. It separates events for further processing by LNode and GNode classes.

Figure B-1: TArc and TNode Classes Inheritance.

2. Figure B—2 illustrates the inheritance of the ARTTisan classes "LNode" and LArc". As can be seen, 'LNode' and 'LArc' inherits from 'Logical Object' class. The 'Logical Object' class is the logical representation of the system. It ensures that the relationship between objects in the system is maintained. The LArc class inherits from the Logical Object class. This class ensures that the logical property of an Arc is maintained. For example, it ensures that an arc is connected to a Node only. The LNode class inherits from the Logical Object class. This class ensures that the logical property of a node is maintained. For example, it ensures that a node is connected to arcs only.

Figure B-2: LNode and LArc Classes Inheritance.

3. Figure B—3 illustrates the inheritance of the ARTTisan class 'GBSpline'. As can be seen 'GBSpline' inherits from 'GArc' and 'GVector' class. The GVector class is meant to be the basic building block for arcs. This class provides the bases to draw, erase and determine if the line intersects an area. Besides the properties inherited from GVector, GArc has three variables used to manage whether it is drawn on the screen and how it is drawn on the screen. GBSpline inherits the properties from the GArc except that instead of drawing itself as a straight line, it uses a spline function to draw itself as a curve (a spline).

Figure B-3: GBSpline Class Inheritance.

4. Figure B—4 illustrates the inheritance of the ARTTisan class "GNode". The GNode class, in conjunction with the Rectangle class, provides bases to quickly and easily build icons. This class provides the bases to draw, erase and force a point to the edge of any four side figure.

Figure B-4: GNode Class Inheritance.

5. Figure B—5 illustrates the inheritance of the ARTTisan class "SMEModes". The SMEModes in conjunction with the Modes class provides a means for imposing a global state on all the icons. Each instances of Mode is associated with one System Window or View and when that instance becomes the 'mode in control' the view will pass all events directly to it.

Figure B-5: SMEModes Class Inheritance.

Part II: Created Objects.

This sub-section will describe the classes of objects created for the CASE tool user interface requirements and to interface with ARTTisan. These different classes of created objects consist of about 800 lines of source code. The following three steps used to describe the object-oriented design are inspired from reference [1]:

1. Identify the classes of the system and apply inheritance where appropriate.
 - a. GNode → GVirtSite: For the following class description, the class at the right of the arrow inherits from the class at left of it. Here it means that GVirtSite inherits from GNode. This class creates virtual sites Icons. No method peculiar to that class has been created. (Filename: GVirtSite.m)
 - b. GNode → GPhySite: This class creates physical sites icons. One inherited GNode method has been modified. (Filename: GPhySite.m)
 - c. GArc → GTransEdge: This class creates edges between pair of node of the same type or between different type of nodes. These edges represent transmission costs/edges between sites. Some inherited GArc methods have been modified. (Filename: GTransEdge.m).
 - d. GBSpline → GForkingArc: This class creates a spline between pair of transmission cost/edges only. These splines represent forking arcs. Some inherited GBSpline methods have been modified. (Filename: GForkingArc.m)
 - e. LNode → LPhySite: This class ensures that the logical property of a physical site is kept, i.e., there is a set of arcs that can enter it and a set of arcs that can leave it. There is however no constraint on the type of arcs therefore the user has to realize that these arcs have to be of the TransEdge class. Some inherited LNode methods have been modified. (Filename: LPhySite.m).

- f. LNode → LVirtSite: This class ensures that the logical property of a virtual site is kept i.e. there is a set of arcs that can enter it and a set of arcs that can leave it. There is however no constraint on the type of arcs and therefore the user has to realize that these arcs have to be of the TransEdge class. No method peculiar to that class has been created. (Filename: LVirtSite.m).
- g. LArc → LForkingArc: This class ensures that the logical property of a Forking arc is kept, i.e., there is one node that the arc leave and one node that the arc enters. There is however no constraint on the type of node, therefore the user has to realize that these node have to be of the TransEdge class. No method peculiar to that class has been created. (Filename: LForkingArc.m).
- h. LArc → LTransEdge: This class ensures that the logical property of a transmission cost/edge is kept. The transmission cost/edge behaves like a node and an arc at the same time i.e. there is one node that the transmission edge can leave or enter and there is one set of arcs that can enter or leave it. There is however no constraint on the type of node leaving or entering or on the type of arcs that can enter or leave it. Therefore the user has to realize that these nodes have to be of the virtual or physical class and these arcs have to be of the ForkingArc class. Since this class has to behave as a node as well as an arc, all the LNode methods have been added to the inherited LArc methods. (Filename: LTransEdge.m).
- i. TNode → TVirtSite: This class receives events from the user and translates them into logical and graphical events for further processing by GVirtSite and LVirtSite classes. Some new methods have been added and some other inherited methods have been modified. (Filename: TVirtSite.m).
- j. TNode → TPhySite: This class receives events from the user and translates

them into logical and graphical events for further processing by GPhySite and LPhySite. Some new methods have been added and some other inherited methods have been modified. (Filename: TPhySite.m).

- k. TArc —> TForkingArc: This class receives events from the user and translates them into logical and graphical events for further processing by GForkingArc and LForkingArc. No method peculiar to that class has been created. (Filename: TForkingArc.m).
- l. TArc—> TTransEdge: This class receives events from the user and translates them into logical and graphical events for further processing by GTransEdge and LTransEdge. Since the received events are translated into both logical and graphical events and the logical part of a transmission cost/edge behaves like a node as well as an arc all the TNode methods have been added to the inherited TArc methods. (Filename: TTransEdge.m).
- m. SMEModes —> Add-Process-Icon-Mode: This class is activated when the user selects the "Edit" button from the ARTTisan panel. When the selection of that button is made, the created Add-Process-Icon-Mode takes over the ARTTisan SMEMode.(Filename: AddProcessIconMode.m)

The created Add-Process-Icon-Mode panel offers the same button selection as the ARTTisan Add-Process-Icon-Mode panel , except that a "Demo" button has been added which permits to transfer to the "DemoMode". The menu is completely new and offers the choice of transfer to any of the "T" class (Such as: TVirtSite, TPhySite, etc...) previously described, which upon user request will create a new instance of a physical site, virtual site, transmission edge, or forking arc class. Some inherited SMEModes method have been modified (Filename: AddProcessIconMode.m).

- n **Add-Process-Icon-Mode—> DemoMode:** This class executes the algorithm “total-data-allocation”. It is activated when the user select the “Demo” button from the “Edit” panel. When the selection of that button is made, the created DemoMode takes over the Add-Process-Icon-Mode. All methods had to be created (Filename: DemoMode.m).

2. Identify the attributes for each classes.

- a. **GVirtSite:** All the methods inherited from the GNode class keep the same attribute values. The only attribute to be specified is the virtual site dimension of 25 by 25 units.
- b. **GPhySite:** All the methods inherited from the GNode class keep the same attribute values. The first attribute to be specified is the physical site dimension of 65 by 65 units. The second attribute is the radius of three units and the black color of the physical site circle to be drawn at the center of the physical site.
- c. **GTransEdge:** All the methods inherited from the GNode class keep the same attribute values. The only attribute to be specified is the position of the arrow in the middle of the transmission edge.
- d. **GForkingArc:** No special attribute has been created.
- e. **LPhySite:** No special attribute has been created.
- f. **LVirtSite:** No special attribute has been created.
- g. **LForkArc:** No special attribute has been created.
- h. **LTransEdge:** No special attribute has been created.
- i. **TVirtSite:** A collection of united icon has been created to know if that icon is united with another icon or not.

- j. TPhySite: No special attribute has been created.
 - k. TForkingArc: No special attribute has been created.
 - l. TTransEdge: Since the transmission edge behaves like a node as well as an arc, a collection has been created to keep track of the number of source and destination arcs.
 - m. Add-Process-Icon-Mode: No special attribute has been created.
 - n. DemoMode: For the algorithm "total-data-allocation" many different attributes had been created. A collection has been created to keep track of the instances of each virtual sites, physical sites, forking arcs and transmission edges. An array was created to keep track to which physical site a virtual site is assigned. Different flags have been created, one for transmission edges part of the same forking arc, one when a virtual site is assigned to a physical site, one when a set of virtual site is part of the same forking arc, one when a transmission edge is between two united virtual sites, one when a transmission edge has to be redrawn and one when the union of two virtual sites connected by a transmission edge increase the total cost function. Finally a set of virtual sites to consider for union has been created.
3. Describe the operations for each abstractions and identify them by message name.
- a. GVirtSite: The following factory method has been created:
 - i. new: A new graphical instance of a virtual site is created where the dimension is 25 by 25 units.
 - b. GPhySite: The following methods have been created/modified:
 - i. new: A new graphical instance of a physical site is created where the dimension is set to 65 by 65 units.

- ii. **draw:** This method inherits everything from the GNode class draw method, except that a black circle of radius three is specified to be draw at the center of the physical site.
- c. **GTransEdge:** The following methods have been created/modified:
 - i. **new:** A new graphical instance of the GTransEdge class is created.
 - ii. **force-To-Edge:** This method is used by the ForkingArc class and returns only the same point value as the one sent to it.
 - iii. **draw-Arrow-Head:** This method overwrites the same method in the GArc class to draw the arrow head in the middle of the transmission edge.
 - iv. **erase-Arrow-Head:** This method overwrites the same one in the GArc class in order to erase the arrow head in the middle of the transmission edge.
 - v. **clear-And-Erase-Arrow-Head:** This method overwrites the same one in the GArc class in order to erase the arrow head in the middle of the transmission edge.
- d. **GForkingArc:** The following methods have been created/modified:
 - i. **new:** A new graphical instance of the GForkingArc class is created.
 - ii. **draw-Arrow-Head:** This method neutralize the same one in the GArc class so that now no arrow head is drawn.
 - iii. **clear-And-Erase-Arrow-Head:** This method neutralizes the same one in the GArc class so that now nothing is done.
- e. **LPhySite:** No new methods had to be created/modified.
- f. **LVirtSite:** No new methods had to be created/modified.
- g. **LForkingArc:** No new methods had to be created/modified.

h. LTransEdge: The following methods have been created/modified:

- i. **owner:** A new logical instance of a transmission edge is created which inherits everything from the LArCs class. In its creation two collections are created, one of source arcs and another one of destination arcs.
- ii. **can-You-Accept-Starting-Arc:** This method overwrites the same one in the LArc class so that now the starting of an arc may be connected to a transmission edge.
- iii. **can-You-Accept-Ending-Arc:** This method overwrites the same one in the LArc class so that now the ending of an arc may be connected to a transmission edge.
- iv. **free:** This method overwrites the same one in LArc so that now in addition of the parameters previously free, the two collection of source arcs and destination arcs can be free as well.

All the succeeding methods have been created so that the transmission edges have the same logical properties of a node in addition of having the ones of an arc. These methods have very similar source code then the ones of the same name of the TNode class.

- i. **is-A-Src-Arc:** Return YES when an arc (Forking Arc) is one of the transmission edge source arc and NO otherwise.
- ii. **is-A-Dst-Arc:** Returns YES when an arc (Forking Arc) is one of the transmission edge destination arc and NO otherwise.
- iii. **add-Dst-Arc:** This method adds an arc (Forking Arc) to the list of destination arcs that are currently connected to the transmission edge. The request is ignored if the arc is already in the list of destination arcs.

- iv. **add-Src-Arc:** This method adds an arc (Forking Arc) to the list of source arcs that are currently connected to the transmission edge. The request is ignored if the arc is already in the list of source arcs.
- v. **remove-All-Arcs:** This method remove all the destination and source arcs (Forking Arcs) from the transmission edge.
- vi. **remove-Src-Arc:** This method removes an arc (Forking Arc) from the list of source arcs currently attached to the transmission edges.
- vii. **remove-Dst-Arc:** This method removes an arc (Forking Arc) from the list of destination arcs currently attached to the transmission edges.
- viii. **first-Dst-Arc:** This method returns the first arc (Forking Arc) in the transmission edge list of destination arcs.
- ix. **first-Src-Arc:** This method returns the first arc (Forking Arc) in the transmission edge list of source arcs.
- x. **src-Arc-At:** This method returns the transmission edge source arc (Forking Arc) that is located at a certain offset in the list of source arcs of that transmission edge.
- xi. **dst-Arc-At:** This method returns the transmission edge destination arc (Forking Arc) that is located at a certain offset in the list of destination arcs of that transmission edge.
- xii. **srcArcs:** This method returns an ordered collection of all source arcs (Forking Arcs) that are currently attached to that transmission edge.
- xiii. **dstArcs:** This method returns an ordered collection of all destination arcs (Forking Arcs) that are currently attached to that transmission edge.
- xiv. **allArcs:** This method returns an ordered collection of all destination and source arcs (Forking Arcs) that are currently attached to that transmission

edge.

- xv. **src-Arc-After**: This method returns the id of the arc (Forking Arc) that is currently located in the transmission edge list of source arcs after a certain arc.
 - xvi. **dst-Arc-After**: This method returns the id of the arc (Forking Arc) that is currently located in the transmission edge list of destination arcs after a certain arc.
 - xvii. **num-Of-Arcs**: This method returns the total number of source and destination arcs (Forking Arcs) currently attached to the transmission edge.
 - xviii. **num-Of-Dst-Arcs**: This method returns the total number of destination arcs (Forking Arcs) currently attached to the transmission edge.
 - xix. **num-Of-Src-Arcs**: This method returns the total number of source arcs (Forking Arcs) currently attached to the transmission edge.
 - xx. **is-Your-Arc**: This method returns YES if a certain arc (Forking Arc) is either in the list of source or destination arcs and NO otherwise.
- i. **TVirtSite**: The following methods have been created/modified:
- i. **new**: A new instance of a virtual site class is created. It also triggers the creation of its graphical and logical part (GVirtSite & LVirtSite) and attach labels as well as an Artisan menu to itself.
 - ii. **first-United-Icon**: This method returns the first Icon (virtual site) in the list of icon(s) united to itself.
 - iii. **United-Icon-After**: This method returns the icon (virtual site) that is currently located in the list of united Icon after aIcon.

- iv. num-Of-United-Icon: This method returns the total number of icon (virtual site) currently united to that icon.
 - v. force-To-Edge: This method overwrites the same one in the TNode class so that now aPoint is forced to the center of that virtual site
 - vi. draw: This method overwrites the same one in the TNode class so that now the virtual site is drawn as before and in addition all its source and destination arcs are redrawn.
 - vii. erase: This method overwrites the same one in the TNode class so that now the virtual site is erased as before and in addition all its source and destination arcs are redrawn.
 - viii. redraw: This method overwrites the same one in the TNode class so that now the virtual site is redrawn as before and in addition all its source and destination arcs are redrawn
 - ix. uniteWith: This method add anIcon (virtual site) in the list of icon united to itself.
 - x. ununiteWith: This method remove anIcon (virtual site) in the list of icon united to itself.
 - xi. remove-All-Unions: This method remove all icon of the list of icon united to itself.
 - xii. move-United-Icon-By: This method move an Icon using the moveBy method of the TNode class . In addition all the icons in the list of united icon do the same move.
- j. TPhySite: The following methods have been created/modified:
- i. new: A new instance of a physical site class is being created. It

also triggers the creation of its physical and logical part (GPhySite & LPhySite) and attach labels as well as an ARTTisan menu to itself.

- ii. num-Of-United-Icon: This method returns the total number of Icon currently united to itself. This number is always zero.
- iii. all-Sub-Icons: This method is required by the TNode class for its method auto-Find-Free-Sub-Area (For description see ref [78]) and returns an ordered collection of all Icons (virtual sites) that can be contained within boundaries of the physical site.
- iv. force-To-Edge: This method overwrites the same one in the TNode class so that now aPoint is forced to the center of the physical site.
- v. push-To-Edge: This method inherits from the TNode method force-To-Edge. It forces aPoint to the closest edge of the receiver.
- vi. draw: This method overwrites the same one in the TNode class so that now the physical site is drawn as before and in addition all its source and destination arcs are redrawn.
- vii. erase: This method overwrites the same one in the TNode class so that now the physical site is erased as before and in addition all its source and destination arcs are redrawn.
- viii. redraw: This method overwrites the same one in the TNode class so that now the physical site is redrawn as before and in addition all its source and destination arcs are redrawn.
- ix. first-United-Icon: This method returns the first icon (physical site) in the list of icon(s) united to itself.
- x. United-Icon-After: This method returns the icon (physical site) that is currently located in the list of united icon after aIcon.

- xi. num-Of-United-Icon: This method returns the total number of Icon (virtual site) currently united to that Icon.
 - xii. uniteWith: This method add anIcon (virtual site) in the list of icon united to itself.
 - xiii. ununiteWith: This method remove anIcon (virtual site) in the list of icon united to itself.
 - xiv. remove-All-Unions: This method remove all icon united to itself.
 - xv. move-United-Icon-By: This method move an icon using the moveBy method of the TNode class. In addition all the icons in the list of united icon do the same move.
- k. TForkingArc: The following factory method has been created:
- i. new: A new instance of a forking arc class is being created. It also trigger the creation of its logical and graphical part (GForkingArc & LForkingArc) and attach labels as well as an ARTTisan menu to itself.
- l. TTransEdge: The following methods have been created:
- i. new: A new instance of a transmission edge class is being created. It also trigger the creation of its logical and graphical part (GTransEdge & LTransEdge) and attach labels as well as an ARTTisan menu to itself.
 - ii. force-To-Edge: This method uses the force-To-Edge method of the GTransEdge class.
 - iii. attachLabels: This method overwrites the same one in the TNode class . It does the same thing than before (For more details see ref [78]) except that it now display the label “Enter Transmission Cost”.

- iv. **prepare-To-Delete:** This method overwrites the same one in the TNode class. It does the same thing than before (For more detail see ref [78]) and in addition the arcs attached to it will prepare for its deletion.

All the succeeding methods have been created so that the transmission edges have the same properties of a node in addition of having the ones of an arc. These methods have very similar source code then the ones of the same name of the TNode class.

- i. **add-Dst-Arc:** This method add an arc (Forking Arc) as a destination arc to the transmission edge.
- ii. **add-Src-Arc:** This method add an arc (Forking Arc) as a source arc to the transmission edge.
- iii. **dstArc:** This method returns the identification (id) of the destination arc (Forking Arc) at a certain offset in the transmission edge list of destination arcs.
- iv. **srcArc:** This method returns the id of the source arc (Forking Arc) at a certain offset in the transmission edge list of source arcs.
- v. **add-Dst-Arc:** This method add an arc (Forking Arc) as a destination arc to the transmission edge.
- vi. **add-Src-Arc:** This method add an arc (Forking Arc) as a source arc to the transmission edge.
- vii. **remove-Dst-Arc:** This method remove an arc (Forking Arc) from the transmission edge list of destination arcs.
- viii. **remove-Src-Arc:** This method remove an arc (Forking Arc) from the transmission edge list of source arcs.

m. **Add-Process-Icon-Mode:** The following methods have been created:

- i. **create:** As a standard create method of an ARTTisan mode class it informs itself that it has been created and performs appropriate initialization. It initializes the content of a standard menu and appropriate parameters.
- ii. **activate:** As a standard activate method of an ARTTisan mode class, it creates all items in the panel.
- iii. **changeModels:** This method has the same function then the changeModels method of the ARTTisan modes class. This method is basically used for changing the model that the receiver is working with. For more details see ref [80].
- iv. **doNothing:** This method has the same function then the doNothing method of the ARTTisan modes class. However it has been adapted for our own CASE tool. This is the default event handler. It determines which icon the event belongs to and passes the event on to that icon.
- v. **push-Selectors-On-Arc (aArc, aView):** It is a procedure which could easily be replaced by a method of the same name. This procedure automatically puts in a definite order in a selector stack different methods defined in the ARTTisan TArc, LArc & GArc classes.
- vi. **push-Selectors-On-Node (aNode, aView):** As for v. , this procedure could easily be replaced by a method. This procedure puts in a definite order in the selector stack different methods defined in the ARTTisan TNode, LNode & GNode classes.

n. **DemoMode:** The following methods have been created:

- i. **create:** As a standard create method of an ARTTisan mode class it informs itself that it has been created and perform appropriate parameters initialization.
- ii. **activate:** As a standard activate method of an ARTTisan mode class it display and manage all items in the panel.
- iii. **button-Been-Hit:** As a standard button-Been-Hit of an ARTTisan mode class, depending of the hited button it transfers the control to the appropriate method of the DemoMode.

The succeeding 23 methods have not been created to interface with ARTTisan or for user interface requirements. They directly support the algorithm “total-data-allocation”. For that reason these methods will be describe in the next sub-section. These methods could easily be replaced by procedures.

Part III: “Total-Data-Allocation” Algorithm.

Because of the supporting environment (Objective-C and ARTTisan) the algorithm “total-data-allocation” was implemented within the class DemoMode described in the previous section. 23 of the 26 DemoMode methods directly support that algorithm. The three other methods have been created to interface with ARTTisan or with the user. These 23 methods consist of about 2000 lines of code. As previously mentioned these methods could easily be replaced by procedures. This program was created to handle 600 different virtual sites, 150 different Forking Arcs, 1000 different transaction edges and 25 transaction edges having the same value.

1. **Step-By-Step and Non-Stop Options:** Figure B—6 and B—7 illustrate the sequence of execution of the different 23 methods of the DemoMode class required

by the step-by-step and non-stop option respectively. A detailed description of each of these methods is given later in this sub-section. By looking at figure B—6 and B-7, it can be seen that these two DemoMode options are initiated from ARTTisan through the Add-Process-Icon-Mode. When the DemoMode is entered, the 'Create' method is executed automatically and then the control is send to the 'Activate' method. In that method, the user can select six buttons which correspond to six different selections. The first three selections (i, ii &iii) correspond to the ARTTisan classes of RedrawMode, SaveMode and PrintMode. Once executed, these modes give the control back to the 'Activate' method (The Activate method belongs to the 'DemoMode' class). The fourth selection (iv) corresponds to the ARTTisan class of UpMode. The execution of this class send the control back to ARTTisan. The fifth selection (v) corresponds to the second 'NonStop' option and will be described later in this sub-section. The sixth selection (vi) corresponds to the 'Step-by-Step' option and its sequence of execution will be explained in the next paragraphs:

- a. Sequence of Execution of the 'Step-by-Step' Option: This option is illustrated at figure B-6. The 'Step-by-Step' option is initiated by clicking the 'St-by-St' button in the six choices of buttons offered to the user. Once this button is selected, the user has the choice of having part of the second step of the algorithm executed through the selection of the 'Rm-Fk-Arc' button. As explained at section 4.3 this part of the second step deletes the forking arcs. Once the 'Rm-Fk-Arc' button has been selected by the user, the case 'Rm-Fk-Arc' option of figure B—6 which consists of the 'Init-Up-Trs-Edge' and 'Delete-Forking-Arcs' methods is executed, the 'Stop' flag is set to 1 and the

user has the choice to have the last part of the second step executed through the selection of the 'Assign' button. As explained at section 4.3 that last part of the second step does an initial assignment of virtual sites to physical ones. Once the 'Assign' button has been selected by the user, the case 'Assign' option of figure B—6 which consists of the 'Max-Trs-Cost-Between-Virt-Phy-Sites', 'assign-Virt-Site' and 'assign-Remaining-Virt-Sites' methods is executed, the 'Step' flag is set to 2 and the user has the choice to have the union part of the algorithm executed through the selection of the 'Unite' button.

Figure B-6: Step-by-Step Through Algorithm 'total-data-allocation'.

The union part of the algorithm is covered by step 3 to 6 explained at section 4.3. Once the 'Unite' button has been selected by the user, the case 'Unite'

option of figure B—6 which consists of the 'Build-SetP-Of-Virt-Site-To-Be-United', 'final-SetP-Virt-Site-Pair-for-Union', 'calculate-All-Possible-Union-Cost' and 'calculate-Total-Cost-Of-Unions' methods are executed as many times as required (Through the 'Next' flag) to execute step 3 to 6 of section 4.3. During the execution, when the flag 'Next' equals NO, the user has the choice of having the seventh step of the algorithm executed through the selection of the 'Return' button. As explained at section 4.3 that last part of the algorithm decides if step 3 to 6 have to be repeated. These steps have to be repeated as long as the optimal solution is not reached. Once the 'Return' button has been selected by the user, the case 'Return' option of figure B—6 which consists of the verification of the 'End' flag value is executed. The 'End' flag value has been set during the execution of the previous steps 3 to 6 of section 4.3. If the 'End' value equals NO, then the user has the choice to have the union part of the algorithm executed once again through the selection of the 'Unite' button. If the 'End' value equals YES, the 'Step' flag is set to 3, then the user has the choice to have the final union part of the modified algorithm executed through the selection of the 'LstUnion' button.

The final union part of the modified algorithm was made to adapt the original algorithm to practical problems. That final union is described at section 4.4. Once the 'LstUnion' button has been selected by the user, the case 'LstUnion' option of figure B—6 which consists of the 'final-Union-of-Virtual-Sites' method is executed, the 'Step' flag is set to 4 and in order to have a better view of the final allocation, the user has the choice to delete non-required transmission edges through the selection of the 'DelTrs' button. The non-

required transmission edges are edges between virtual sites part of the same union or between virtual and physical sites. If non-deleted, these edges deteriorate the visual interpretation of the final allocation. Once the 'DelTrs' button has been selected by the user, the case 'DelTrs' option of figure B—6 which consists of the 'Delete-United-Trs-Edges' and 'Delete-Phy-Site-Trs-Edges' method is executed, the 'Step' flag is set to 5 and the user has the choice to reset all the global variables of the DemoMode class. The DemoMode global variables have to be freed so that if the user exit the instance of that DemoMode, the memory space required by these variables can be used by other applications. Once the 'END' button has been selected by the user, the case End option of figure B—6 which consists of the 'Reset-All-Glob-Variables' method is executed, the 'Step' flag is set to 6 and the user has no other choice as far as the execution of the algorithm 'total-data-allocation' is concerned.

- b. Sequence of Execution of the Non-Stop Option: This option is illustrated in figure B-7. The Non-Stop option is initiated by clicking the 'NonStop' button in the six choices of buttons offered to the user. Once this button is selected, if the 'Step' variable has a value of '0' then all the different steps (Methods 'Init-Up-Trs-Edge' to 'Reset-All-Glob-Variables') of the modified algorithm 'total-data-allocation' are executed. A value of '0' given to the 'Step' variable means that none of the methods of the 'Step-by-Step' options from 'Init-Up-Trs-Edges' to 'Reset-All-Glob-Variables' have been executed.

Figure B-7: NonStop Execution of Algorithm 'total-data-allocation'.

If the 'Step' variable has any integer value from '1' to '6', then part of the algorithm 'total-data-allocation' would be executed. For example, if through the 'Step-by-Step' option, the user had previously selected the 'Assign' button, then the variable 'Step' would be set to a value of 2. As can be seen in figure B—7 for the 'NonStop' option, a value of 2 assigned to the variable 'Step' means that the first two decisions (if Step < X) would be negative and the last four would be positive. Therefore, only the methods of 'Build-SetP-Of-Virt-Site-To-Be-United' to 'Reset-All-Glob-Variables' would be executed. This corresponds to the execution of the steps starting from the union of virtual sites to the end of the modified algorithm 'total-data-allocation'. The first steps of the algorithm which correspond to the removal of forking arcs and the assignment of virtual sites would not be executed.

2. DemoMode Methods: The following paragraphs will give a description of the 23 different DemoMode methods which directly support the modified algorithm 'total-data-allocation' :
 - a. init-Up-Trans-Edge: This method sets a flag for transmission Edges which are connected by a forking arc.
 - b. delete-Forking-Arcs: This method deletes all forking arcs from a view
 - c. Max-Trs-Cost-Between-Virt-Phy-Sites: This method considers all the transmission edges/costs which connect virtual sites to physical sites and assigns (calling the assign-Virt-Site method) the virtual sites to the physical sites connected by the highest transmission cost. This method with the next one does step "2" of the algorithm "total-data-allocation" explained at section 4.3.
 - d. assign-Virt-Site: This method is called by the Max-Trs-Cost-Between-Virt-

Phy-Sites and **assign-Remaining-Virt-Sites** methods to assign a virtual site to a physical site.

- e. **assign-Remaining-Virt-Sites**: When a virtual site is not already assigned, this method will assign it (calling the **assign-Virt-Site** method) to the same physical site to which one of its neighbour virtual site is assigned to. Where a neighbour virtual site is a virtual site connected by a transmission edge. This method with the previous one does step “2” of the algorithm “total-data-allocation” explained in section 4.3.
- f. **Build-SetP-Of-Virt-Site-To-Be-United**: This method will build the set P of all the pairs of virtual sites that can be united. This method does step “3” of the algorithm “total-data-allocation” explained at section 4.3.
- g. **final-SetP-Virt-Site-Pair-For-Union**: This method will keep only the pair of virtual sites which have the largest LINK-value. If there are 'n' pairs having this high value then these 'n' pairs will be remembered. This method does step “4” of the algorithm “total-data-allocation” explained at section 4.3 and 4.4.
- h. **calculate-All-Possible-Union-Cost**: This is a recursive method which will call the **calculate-Total-Cost-Of-Union** method as long as the largest LINK-value of the previous **final-Set-Virt-Site-Pair-For-Union** method is greater than zero. If the only largest LINK-value left to consider is equal to zero then it is the end of the 'total-data-allocation' algorithm. This method does step “7” of the algorithm “total-data-allocation” explained at section 4.3.
- i. **calculate-Total-Cost-Of-Unions**: This method calculates the cost of uniting two virtual sites in the pairs of virtual sites which have the largest LINK value. It first considers the increase in transmission cost by removing the vir-

tual sites from their assigned physical sites. It then considers the decrease in transmission cost due to the assignment of the union of virtual sites to a physical site and the removal of the transmission edge in between the two united virtual sites. Calculating the union cost for all the pair of virtual sites part of the final set P only the pair decreasing the cost the most will be remembered. Figure B—8 gives a further breakdown of the method 'calculate-Total-Cost-Of-Unions' illustrated at figure B—6 & B-7. As illustrated in these two figures, the method 'calculate-Total-Cost-Of-Unions' is called by the method 'calculate-All-Possible-Union-Cost'. As illustrated in that figure, if the variable 'TotUnionCost' is greater than '0', then the execution of the method 'Calculate-Total-Union-Cost' go back to the calling method 'calculate-All-Possible-Union-Cost'. The variable 'TotUnionCost' represents the total cost of uniting a pair of virtual sites as defined in section 4.3 step 5.c. If the variable 'TotUnionCost' is less than or equal to '0', then an union of virtual sites occur by the execution of the 'Unite-Virtual-Sites' method. During its execution, the 'Unite-Virtual-Sites' method calls the 'adjust-Trans-Edge-Values' method which call the 'Comput-Connected-Site' method. During its execution, the 'Comput-Connected-Site' method call the 'are-Trs-Edge-United', 'is-Mb-Of-Same-Union', 'add-Trs-Edge-Cost', 'replace-Trs-Edge-Cost' and 'return-Rel-Trs-Edge-Num' methods. These smaller methods do the union and appropriate adjustments on the pair of virtual sites remembered for union. Once completed, the 'Compute-Connected-Site' gives the control back to the 'adjust-Trans-Edges-Values' method which at its turn, gives the control back to the 'Unite-Virtual-Sites' method. Once the 'United-Virtual-Sites' method is completed, the 'Assign-Unit-Virt-Sites' method is executed,

bered in the method calculate-Total-Cost-Of-Unions. It will first move them outside their assigned physical site, unite them and the resulting union will be assigned to connected physical site with the highest transmission cost. This method does part of step "6" of the algorithm "total-data-allocation" explained at section 4.3.

- k. **adjust-Trans-Edge-Values:** This method adjust the labels on specific transmission edges. If before an union a single virtual site or a union of virtual sites on one side of the union is connected (by a transmission edge) with any of the same physical or virtual site to which the other side of the union is connected to and if this same physical or virtual site is not part of any of these two side of the union, then the label on the transmission edge between the resulting union and that same physical or virtual site is the sum of the two transmission edges labels. For example at figure B-9(a), before being united virtual sites V1 and V2 are connected to the same other virtual site V3. As a result of the union, their common transmission edges going to the same virtual site V3 has to be adjusted as the sum of their two transmission edges e.i. $5+10 = 15$. The result can be seen at figure B-9(b).

Figure B-9: Adjustment of Transmission Edges Values as Result of an Union.

Actually this method takes care of all the possible cases, even complex cases where both virtual sites to be united are part of an union of virtual sites before being united together and where the sites to which they are connected in common is also part of an union. For example, at figure B-9(a) it would mean that virtual site V1 as well as V2 would be part of an union of virtual sites before being united. Besides, virtual sites V3 would also be part of an union of virtual sites. The update of transmission edges in this situation is relatively complex. This method does part of step "6" of the algorithm "total-data-allocation" explained at section 4.3.

1. **comput-Connected-Site:** This method updates the pair of transmission edges which are come from two virtual sites or the union of virtual sites to unite

and which is connected to the same third site. The values of the updated transmission edge is the sum of the labels on these two transmission edges. This method is used by the 'adjust-Trans-Edge-Value' method.

- m. **return-Rel-Trs-Edge-Num**: This method returns the relative number where a transmission edge resides in the global collection of transmission edges. This method is used by the 'comput-Connected-Site' method.
- n. **replace-Trs-Edge-Cost**: This method replaces a TrsEdge2 label by a TrsEdge1 one. This model is used by the 'comput-Connected-Site' method.
- o. **are-Trs-Edge-United**: This method returns a "0" value if the two transmission edges passed to it are not between two united virtual sites. This method is used by the adjust-Trans-Edge-Values method.
- p. **is-Mb-Of-Same-Union**: This method returns a "1" value if the two icons are member of the same union of Icons. This method is used by the adjust-Trans-Edge-Values method.
- q. **add-Trs-Edge-Cost**: This method adds the transmission cost on two transmission edges and the result is written back on these edges. This method is used by the adjust-Trans-Edge-Values method.
- r. **assign-Unit-Virt-Sites**: This method will determine the highest transmission cost between the union of virtual sites and any of the physical sites to which it is connected. It will assign the union (using the assign-United-Virt-Site method) to the physical site connected by the highest transmission cost. This method does part of step "6" of the algorithm "total-data-allocation" explained at section 4.1.
- s. **assign-United-Virt-Site**: This method will move the union of virtual sites to a specified physical site. This method is used by the assign-Unit-Virt-Sites

method.

- t. **find-Union-Of-Virtual-Sites:** This method considers isolated virtual sites (Virtual Sites part of updates only) and union of virtual sites. If these two groups (Isolated virtual sites and union of virtual sites) are assigned to the same physical site then this method unite them together. As a result of that union, all the transmission edges between pair of physical sites are adjusted accordingly. This method is one of the modifications made to the algorithm 'total-data-allocation' described at section 4.4.
- u. **delete-United-Trs-Edges:** This method deletes transmission edges between united Virtual sites.
- v. **delete-Phy-Site-Trs-Edges:** This method deletes transmission edges between virtual sites and physical sites to where that virtual site is assigned.
- w. **reset-All-Global-Variables:** This method resets the DemoMode global parameters. This is done so that if the user exits the instance of the DemoMode class, the memory space required by its global variables is free to be used by any other applications.

4

OF/DE

4

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

Appendix C

PROPOSED DATA DISTRIBUTION APPROACH APPLIED TO AEROSPACE MAINTENANCE INFORMATION SYSTEM PROJECTS.

Proposed Data Distribution Approach Applied to ADAM & AMMIS.

The practical problem to which the data distribution method proposed at chapter 5 is applied to is the same one then appendix A Aircraft Maintenance Information System i.e. ADAM. In addition, it is applied to Aircraft Maintenance Management Information System (AMMIS). AMMIS is considered in this part only (It was not considered at appendix A) since this system only retrieves subset data of the overall ADAM conceptual schema data developed at appendix A.

It is important to note that the final allocation result is only as good as the data that was putted into the allocation algorithm. Since some of the required data was not directly available, many assumptions had to be made. To have better allocation results, more reliable input data (transaction specifications etc...) would be required.

The following are the different steps of the data distribution methodology applied to ADAM & AMMIS, the information required by each steps was taken from ref[85] & [86].

Qualitative Approach.

This approach gives a general framework for the management to decide at the subject database level if a subject database should be distributed or not. The following paragraphs will apply the different steps of this approach to the ADAM & AMMIS project.

1. **Process versus Subject Databases:** This step is to develop a process/subject database association matrix to know which subject database each process uses. The figure C-1 list of processes has been limited to process 1.1 to 3.6 and 5.1 to 7.6 due to the large number of processes. Actually using the information of appendix A (figure A-5) this can be done relatively easily.

Subject Databases					
Personnel			Personnel		
Equipment			Equipment		
Aircraft			Aircraft		
Processes					Processes
P1.1					P5.3
P1.2					P5.4
P1.3					P5.5
P1.4					P6.1
P2.1					P6.2
P2.2					P6.3
P2.3					P6.4
P3.1					P6.5
P3.2					P6.6
P3.3					P7.1
P3.4					P7.2
P3.5					P7.3
P3.6					P7.4
P5.1					P7.5
P5.2					P7.6

Figure C-1: Processes versus Subject Databases Association Matrix

2. **Processes versus Locations:** For the AMMIS part, all the processes are generated at one central location which is CFB Ottawa(N). For the ADAM part (See ref [85]) it is already decided how each processes has to be distributed. It shows which processes should occur under the BAMEO organization and since this organization is replicated at every sites each processes specified under that organization is replicated/distributed over these different sites. However, if a reevaluation of the distribution of the processes is wanted, then figure C-2 shows how this would be done.

Processes							
Factors	P 1.1	P 1.2	P 1.3	P 1.4	P 2.1	P 2.2	P 2.3
Most data can be stored locally	d	d	d	d	d	d	d
Application is simple	d	d	d	d			
..							
...							
..							

Figure C-2: Factor Table for ADAM Processes

Where:

- a. C = Strong reason for centralization.
- b. c = Weak reason for centralization.
- c. D = Strong reason for decentralization.
- d. d = Weak reason for decentralization.

Considering only the 2 factors of figure C-2, it could be decided that all the processes (P1.1 to P2.3) should be distributed.

3. Subject Database versus Process/Locations: Let estimates the transaction volumes between process/locations and subject database for the ADAM project and indicate whether the queries are interactives (I) or batch (B). It is important to note that the query frequencies developed during the next paragraphs are based on assumptions. This had to be done since the exact frequency number of these queries is presently unknown. Since this step is only to give the management a rough idea of the query distribution and their frequencies, the total number of processes as well as the number of sites had to be limited. Due to the numerous different processes to consider only the most representatives have been selected. The following 6 processes with their corresponding queries have been selected.
 - a. Conduct Maintenance Action (Process P2.3): Give the rectification description, the name of the person who did the rectification and the fixer unit for all the defect of a certain type done on an aircraft of a certain type.
 - Site: All sites except CFB Ottawa (N).

- Frequency: Proportional to the number of aircraft on a base and on the aircraft complexity. This has a high frequency of occurrence, it can occur thousand times/year.
- b. Maintenance Report (Process P5.1): What kind of tool-kit was/were used to repair a certain type of abnormal-occurrence on a certain type of aircraft.
- Site: All sites except CFB Ottawa (N).
 - Frequency: Proportional to the aircraft complexity and number of aircraft on a base. This has a low frequency of occurrence, it can occur hundred times/year.
- c. Internal Management Report (Process P5.4): Give the name of the Corporal (Private or Sergeant) who did the most manhours of repair at CFB 'X' for this week.
- Sites: All sites except CFB Ottawa (N).
 - Frequency: 50 weeks/year times 3 different ranks = 150 times/year.
- d. Record Significant Event (Process P6.2) : How many flying-hours elapsed on an aircraft of a certain type and a certain number since that airplane is back from its Depot-Level-Inspection?
- Site: All sites except CFB Ottawa (N).
 - Frequency: Proportional to the number of aircraft on a base times 12 months.
- e. AMMIS (Q1): Give all the AMMIS-Reportable, Parts-Page# which were inspected and passed during the last year on an aircraft of a certain type and a certain number.
- Site: CFB Ottawa (N).
 - Frequency: According to historical data it is 1000 times/year.
- f. AMMIS (Q2): Give the contractor modifications done on an aircraft of a certain type during the last 3 years.
- Site: CFB Ottawa (N).
 - Frequency: On the average, twice a year.

Due to the numerous different bases to consider only the most important ones (As far as update frequency number is concerned) have been selected. All the different queries are batch since with the present manual system the information can be days to weeks old. Actually using [86] the query frequency has been estimated using the figure C-3 information.

Site # / Description	Number of Plane-Type	Update#/Year	Rel. Freq Z
1- CFB Baden	48 of CF-18	15,123	15.1
2-CFB North Bay	21 of CT-133, 7 of CE-144	3,692	3.7
3-CFB Moose Jaw	89 of CT-114	13,587	13.6
4-CFB Winnipeg	7 of CT-114	4,108	4.1
5-CFB Ottawa (S)	4 of CH-144, 7 of CC-104	7,164	7.2
6-CFB Shearwater	35 of CH-124	16,077	16.1
7-CFB Valcartier	6 of CH-135, 8 of CH-136	2,577	2.6
8-CFB Gagetown	11 of CH-135, 10 of CH-136	2,336	2.3
9-CFB Summerside	15 of CP-121	6,860	6.9
10-CFB Greenwood	14 of CP-140	18,060	18.1
11-CFB Bagotville	24 of CF-18	5,867	5.9
12-CFB Cold Lake	63 of CF-18, 12 of CT-133	32,971	33
13-CFB Portage	11 of CT-114, 21 of CT-134, 15 of CH-139	3,872	3.9
14-CFB Edmonton	10 of CC-130, 6 of CC-138, 4 of CH-147	7,232	7.2
15-CFB Trenton	6 of CC-115, 14 of CC-130, 5 of CC-137	19,876	19.9
16-CFB Comox	3 of CC-115, 3 of CP-121, 4 of CT-133, 4 of CP-140	7,047	7.0
Total Updates for all bases: 166,449 updates/year			

Figure C-3: ADAM Update Statistics

The figure C-3 relative frequency 'Z' is proportional to the number of aircraft at a base as well as the complexity of these aircrafts. This number 'Z' is calculated as follow:

Number of updates per base divided by 100 = 'Z', where the remainder part is removed.

The relative number 'Z' is used to evaluate the frequency of some of the queries. For example, since the frequency of some queries is proportional to the aircraft number

at a base as well as its complexity, the frequency of P2.3 at a base is equal to 'Z' times 1000. For example, for CFB Cold Lake this would give:

33 times 1000 = 33,000 times/year.

This relative number 'Z' is required by queries of processes P2.3 & P5.1. The subject database versus Process/Location table showing the frequency of the 6 previous process' queries with their respective sites is illustrated at figure C-4.

Subject Database									
Locations & Processes				Personnel		Equipment		Locations & Processes	
CFB Shearwater								CFB Trenton	
P2.3			16100B			19900B		P2.3	
P5.1			1610B			1990B		P5.1	
P5.4				150B	150B			P5.4	
P6.2		420B					300B	P6.2	
CFB Greenwood								CFB Cold Lake	
P2.3			18100B			33000B		P2.3	
P5.1			1810B			3300B		P5.1	
P5.4				150B	150B			P5.4	
P6.2		168B					900B	P6.2	
CFB Moose Jaw								CFB Borden	
P2.3			13600B			15100B		P2.3	
P5.1			1360B			1510B		P5.1	
P5.4				150B	150B			P5.4	
P6.2		1068B					576B	P6.2	
CFB Ottawa (N)									
AMMIS (Q1)		1000B						Where:	
AMMIS (Q2)			2B					I = Interactive Queries	
								B = Batch Queries	

Where:

I = Interactive Queries

B = Batch Queries

Figure C-4: Subject Database versus Process/Locations Table

- Effect of Queries and Update Frequencies on Data Distribution: Since the Aerospace Maintenance Information System project will be implemented in a long haul network environment, this step apply. In order to use figure 5-3, the update as well as the query frequency has to be known. The query frequency for each subject database can be found from figure C-4. Lets adjust the rate of update of figure C-3 to the rate of update of each subject databases. The update frequency of figure C-3 apply directly to the entities of the 'Aircraft' and 'Equipment' subject database. The update

frequency of the 'Personnel' subject database entities is proportional to the number of personnel under the BAMEO organization for each bases. Assuming that for all the bases each personnel does on the average the same number of update, it can be said that the number of personnel on the base under the BAMEO organization is proportional to the frequency of update per bases. By doing the following:

Total #Updates for all bases divided by 20 = 166,449 divided by 20 = 8325 people.

which is close to the total number of personnel in the Air Force under the different BAMEO organizations. Therefore, for each base the frequency of updates done on the 'Personnel' subject database could be calculated as follow:

Number of Updates per base divided by 20

For the 6 bases considered at figure C-4 we would have the update rate of figure C-5.

Location	Update#/Year	Location	Update#/Year
CFB Shearwater	803	CFB Trenton	994
CFB Greenwood	903	CFB Cold Lake	1,649
CFB Moose Jaw	679	CFB Baden	756

Figure C-5: Frequency of Updates for Subject Database Personnel

If we compare the rate of updates from figure C-3 & figure C-5 to the rate of query of figure C-4 and if we compare the results to figure 5-3 we have the following:

- Equipment Subject Database: Considering that not all the different query are considered at figure C-4, the rate of query for the Equipment Subject Database is higher then the rate of updates. Besides, the information required can be hours old. Therefore using figure 5-3, the data of the Equipment Subject Database can be dispersed with update distributed by Batch transmissions.
- Aircraft Subject Database: Even by considering that not all the different query are considered at figure C-4, the rate of update is probably higher then the rate of query. Besides, the information required can be hours old. Therefore using figure 5-3, the data of the Aircraft Subject Database can either be centralized or dispersed with update distributed by Batch transmissions.
- Personnel Subject Database: By considering that not all the different query are considered at figure C-4, the rate of query is probably the same order of magnitude then the rate of updates. Besides, the information required can be hours old. Therefore using figure 5-3, the data of the Personnel Subject Database can be dispersed with update distributed by Batch transmissions.

5. **Centralization versus Decentralization Issues:** Using the information gathered during the first 3 steps and considering the impact that the different issues discussed at chapter 5 have on decentralization of the data, the decision has to be taken as to distribute or centralize the different databases.

- a. **Cost Issues:**

- i. **Transmission Cost:** Since we are in a long haul network this cost is an important one and has to be considered. As mentioned at section 5.2.5 since the query rate is higher than the update rate for the Equipment subject database, this subject database should be decentralized. For the Aircraft subject database the update rate is higher than the query rate, therefore neither centralization nor decentralization can be recommended as far as transmission cost is concerned. For the Personnel subject database, the update rate is the same order of magnitude than the query rate, therefore neither centralization nor decentralization can be recommended as far as transmission cost is concerned.
- ii. **Storage Cost:** As mentioned at section 5.2.5 centralization decreases storage cost. However with the advances in technology, this cost is getting less considerable.
- iii. **Processor Cost:** As mentioned at section 5.2.5 decentralization decreases processor cost.
- iv. **Database Software and Application Program Cost:** As mentioned at section 5.2.5 centralization decrease database software and application program cost.
- v. **Installation and Repair Cost:** As mentioned at section 5.2.5 decentralization probably decrease installation & repair cost.
- vi. **Response Time :** As mentioned at section 5.2.5 decentralization decrease response time. Even if the information of the query can be hours old it is always nice for the user to have the answer of a request within a reasonable amount of time.

- b. **Technical Issues:**

- i. **Availability:** As mentioned at section 5.2.5 decentralization increases availability.
- ii. **Reliability:** As mentioned at section 5.2.5 decentralization increase reliability. Besides, the reliability of the overall system is an important issue since any of the sites could be subject of an attack of any kind.
- iii. **Security:** As mentioned at appendix A , for objective number 2 of the fifth step of the planning phase of IEM the access to classified data elements has to be controlled at base level. Decentralization of the data offers the

possibility to let local authorities control the security of their own data. This directly apply to the entities of the Personnel subject database.

- iv. Modular Growth: As mentioned at section 5.2.5 distributed system is in favour of modular growth.
 - v. Database Management Systems (DBMS): As mentioned at section 5.2.5 there is not as many commercial DBMS for distributed database as there is for centralized one. ORACLE offers distributed DBMS. Further investigation could be required to see if an existing DBMS exists to support the specific requirements of that particular application.
- c. Application Development Issues:
- i. Planning: As mentioned at section 5.2.5 centralization as well as decentralization has advantages as well as disadvantages as far as planning is concerned. Not enough information is available to recommend one or the other.
 - ii. Developments: Not enough information is available to recommend centralization or decentralization.
- d. Political Issues:
- i. Organization Structure: To decentralize would reflect the organization structure of the BAMEO organization at each bases.
 - ii. Bureaucratic Reasons: Not enough information is available to recommend centralization or decentralization.
 - iii. Quest for Power: Not enough information is available to recommend centralization or decentralization.
- e. Recommendations: Since no critical issue is in favor of centralization and most of the issues are for decentralization it is recommended that the Equipment, Aircraft and Personnel subject database be distributed.

Quantitative Approach:

Since the purpose of this exercise is only to demonstrate that this approach works in real life application and because to do a full scale application would be too time consuming, this quantitative approach is only applied to the Aircraft subject database.

As for the qualitative approach not all the required information was available therefore some assumptions had to be made. The following paragraphs will apply the different steps of this approach to the Aircraft subject database considering the ADAM and AMMIS requirements.

1. Transaction Specifications: The usual way to specify the transaction is by following the 20/80 rule described at section 5.3. However since no real data exists describing

the different transactions and the frequency of those transaction on the different entities of the Aircraft subject database, only the queries described at step 3 of the qualitative approach have been considered. Out of these 6 queries, only process P6.2 and AMMIS Q2 apply to the entities of the Aircraft subject database, therefore only these 2 queries will be considered. Process P6.2 is specified as per figure C-6.

Figure C-6: Process P6.2 Transaction Specification

AMMIS query Q2 is specified as per figure C-7.

Figure C-7: AMMIS (Q2) Transaction Specification

As mentioned at section 5.2.5 of the qualitative approach, the update frequency of figure C-3 apply directly to the entities of the Aircraft subject database.

2. Fragmentation: As explained at chapter 3 fragmentation is tightly related to allocation. Since the developed methodology companion allocate data only in order to minimize the transmission cost, the fragmentation has to be done also in order to minimize the same criteria, the transmission cost. Therefore we are left with 2 options for horizontal fragmentation, the logical as well as the statistical property of the data. Since no tool are available for the statistical option, the logical property of the data had to be selected. The fragmentation will be restricted only to the entities of the Aircraft subject database.

By looking at the nature of the process P6.2 query and AMMIS query Q2 and by looking at the plane distribution at figure C-3 it can be seen that from a certain site there is a higher probability to have a query or an update done on aircrafts of a certain type. Therefore fragmenting the entities of the Aircraft subject database by 'Aircraft-Type' = Certain-Type would increase the locality of the data. Besides, all the entities of the Aircraft subject database already have an attribute of the type 'Aircraft-Type'. Therefore the entities of the Aircraft subject database have to be fragmented by 'Aircraft-Type' = Certain-Type.

As mentioned at section 5.3 due to its complexity and since there is no tool presently available, vertical fragmentation has not been done.

3. Construction of Processing-Schedule Graph: A processing-schedule graph will be done at first for Process P6.2 and AMMIS Q2 queries and then for the different updates. Since no query optimizer is available, the query optimization has been done by hand with the intention of minimizing the transmission cost.

1- Processing-Schedule Graph for Queries:

- a. The following figure C-8 represents process P6.2 processing-schedule graph.

Figure C-8: Process P6.2 Processing-Schedule

Where:

- Vx = Represents Virtual Site 'X'.
- Ph = Represents a Physical Site.

- $f \cdot dx$ = Represents the frequency (f) of the query times the amount of data from one virtual site to another (dx).

The content of each virtual sites will be explained within the next 3 steps, the following notation will be used:

- O_x = Represents operation 'x'.
- PJ = Represents a projection.
- SL = Represents a selection.
- NJN = Represents a natural join.
- $RSLO_x$ = Result of operation 'x'.
- SJ = Represents a semi-join.
- F_x = Represents fragment 'x'.

i. V1:

- $F1$ = Depot-Level-Inspection (Aircraft-Type = 'CF-XXX').
- $O1 = SL$ (aircraft-number = certain number) $F1 = RSLO1$.
- $O2 = PJ$ (Date-DLI-Completed, Aircraft-Type, Aircraft-Number) $RSLO1 = RSLO2$.
- Where $RSLO2 = \text{Date-DLI-Completed} + \text{Aircraft-Type} + \text{Aircraft-Number}$ (Attribute Length) = 15 bytes + 5 bytes + 5 bytes = 25 bytes.
- $f \cdot d1 = f \cdot RSLO2$

If we take CFB Shearwater as an example: $f \cdot d1 = f \cdot RSLO2 = \# \text{times/year} * \# \text{planes} * RSLO2 = 12 * 35 * 25 = 10,500 \text{ bytes/year}$.

ii. V2:

- $F2$ = Aircraft (Aircraft-Type = 'CF-XXX').
- $O3 = RSLO2 NJN F2 = RSLO3$.
- $O4 = PJ$ (Aircraft-Type, Aircraft-Number, Date-DLI-Completed) $RSLO3 = RSLO4$.
- Where $RSLO4 = \text{Date-DLI-Completed} + \text{Aircraft-Type} + \text{Aircraft-Number}$ (Attribute Length) = 15 bytes + 5 bytes + 5 bytes = 25 bytes.
- $f \cdot d2 = f \cdot RSLO4$

For CFB Shearwater: $f \cdot d2 = f \cdot RSLO4 = 12 * 35 * 25 = 10,500 \text{ bytes/year}$.

iii. V3:

- $F3$ = Flying-Time (Aircraft-Type = 'CF-XXX').
- $O5 = RSLO4 SL$ ($RSLO4$.Date-DLI-Completed before $F3$.Flying-Date) $F3 = RSLO5$.
- $O6 = PJ$ (Daily-Flying-Hours) $RSLO5 = RSLO6$.
- $O7 = \text{Add all the instances of attribute Daily Flying-Hours} = RSLO7$.

- Where RSLO7 = Daily-Flying-Hours attribute length = 4 bytes.
- $f \cdot d3 = f \cdot RSLO7$

For CFB Shearwater: $f \cdot d3 = f \cdot RSLO7 = 12 \cdot 35 \cdot 4 = 1,680$ bytes/year.

Figure C-9 gives a summary of the processing-schedule graphs for query P6.2 for each of the different 16 sites of figure C-3. Where K = kilo-bytes.

Military Base	Plane Type	$f \cdot d1$		$f \cdot d2$		$f \cdot d3$	
1- CFB Baden	48 → CF-18	V1	14K	V2	14K	V3	2K
2-CFB North Bay	21 → CT-133	V4	4K	V5	4K	V6	1K
	7 → CE-144	V97	4K	V98	4K	V99	1K
3-CFB Moose Jaw	89 → CT-114	V7	27K	V8	27K	V9	4K
4-CFB Winnipeg	7 → CT-114	V10	2K	V11	2K	V12	1K
5-CFB Ottawa (S)	4 → CH-147	V13	2K	V14	2K	V15	1K
	7 → CC-109	V179	2K	V180	2K	V181	1K
6-CFB Shearwater	35 → CH-124	V16	11K	V17	11K	V18	2K
7-CFB Valcartier	6 → CH-135	V19	2K	V20	2K	V21	1K
	8 → CH-136	V182	2K	V183	2K	V184	1K
8- CFB Gagetown	11 → CH-135	V22	3K	V23	3K	V24	1K
	10 → CH-136	V185	3K	V186	3K	V187	1K
9-CFB Summerside	15 → CP-121	V25	5K	V26	5K	V27	1K
10-CFB Greenwood	14 → CP-140	V28	4K	V29	4K	V30	1K
11-CFB Bagotville	24 → CF-18	V31	7K	V32	7K	V33	1K

Figure C-9: P6.2 Processing-Schedule Graph Summary

Military Base	Plane Type	f * d1		f * d2		f * d3	
CFB Cold Lake	63 → CF-18	V34	20K	V35	20K	V36	3K
	12 → CT-133	V55	3K	V56	3K	V57	1K
CFB Portage	11 → CT-114	V37	3K	V38	3K	V39	1K
	21 → CT-134	V188	6K	V189	6K	V190	1K
	15 → CH-139	V191	6K	V192	6K	V193	1K
CFB Edmonton	4 → CH-147	V40	2K	V41	2K	V42	1K
	6 → CC-138	V194	2K	V195	2K	V196	1K
	10 → CC-130	V197	2K	V198	2K	V199	1K
CFB Trenton	6 → CC-115	V43	2K	V44	2K	V45	1K
	14 → CC-130	V200	4K	V201	4K	V202	1K
	5 → CC-137	V203	2K	V204	2K	V205	1K
CFB Comox	3 → CC-115	V84	2K	V83	2K	V82	1K
	3 → CP-121	V73	2K	V74	2K	V75	2K
	4 → CT-133	V46	2K	V47	2K	V48	2K
	4 → CP-140	V66	2K	V65	2K	V64	1K

Figure C-9: P6.2 Processing-Schedule Graph Summary

By looking at figure C-9 and the different 3 steps previously explained we can derive the fragment associated to a virtual site. For example, for CFB Cold Lake for 'V35' we have the fragment Aircraft (Aircraft-Type = CF-18), for 'V36' we have the fragment Flying-Time (Aircraft-Type = CF-18).

- b. The following figure C-10 represents AMMIS Q2 processing-schedule graph.

Figure C-10: AMMIS Q2 Processing-Schedule Graph

The content of each virtual sites will be explained within the next 2 steps. The same notation than process P6.2 will be used:

i. V1:

- F1 = Depot-Level-Inspection (Aircraft-Type = 'CF-XXX').
- O1 = SL(Date-DLI-Completed = Within last 5 years) F1 = RSLO1.
- O2 = PJ (Date-DLI-Completed) RSLO1 = RSLO2.
- Where RSLO2 = Date-DLI-Completed (Attribute Length) = 15 bytes times the amount of DLI completed per aircraft within the last 3 years. (Let assume that 2 DLI were completed within the last 3 years per aircraft).The result times #Aircraft per base.
- $f*d1 = f*RSLO2$.

If we take CFB Greenwood as an example: $f*d1 = f*RSLO2 = 2 * 15 \text{ bytes} * 2 \text{ DLI} * 18 \text{ Aircrafts} = 1,080 \text{ bytes/year}$.

ii. V2:

- F2 = Depot-Level-Modification (Aircraft-Type = 'CF-XXX').
- O3 = RSLO2 NJN F2 = RSLO3.
- O4 = PJ(Modification-Description) RSLO3 = RSLO4.
- Where RSLO4 = Modification-Description attribute length = 50 bytes times the amount of depot level modification (DLM) per depot level inspection (DLI). (Let assume that there is on the average 10 DLM per DLI).
- $f*d2 = f*RSLO4$.

For CFB Greenwood: $f*d2 = f*RSLO4 = 2 * 50 \text{ bytes} * 10 \text{ (DLM/DLI)} * 18 \text{ Aircrafts} = 18,000 \text{ bytes/year}$.

Figure C-11 gives a summary of the processing-schedule graphs for the query AMMIS Q2 for each of the different aircraft types of figure C-3. All the query are required by the same site CFB Ottawa (N).

Plane Type	f * d1		f * d2	
18 → CP-140	V206	1K	V207	18K
35 → CH-124	V208	2K	V209	35K
18 → CP-121	V210	1K	V211	18K
107 → CT-114	V212	6K	V213	107K
21 → CT-134	V214	1K	V215	21K
15 → CH-139	V216	1K	V217	15K
9 → CC-115	V218	1K	V219	9K
24 → CC-130	V220	1K	V221	24K
6 → CC-138	V222	1K	V223	6K
4 → CH-147	V224	1K	V225	4K
37 → CT-133	V226	2K	V227	37K
136 → CF-18	V228	8K	V229	136K
7 → CE-144	V230	1K	V231	7K
17 → CH-135	V232	1K	V233	17K
18 → CH-136	V234	1K	V235	18K

Figure C-11: AMMIS Q2 Processing-Schedule Graph Summary

By looking at figure C-11 and the 2 different steps previously explained the fragment associated to the virtual sites can be derived. For example, for 'V212' we have the fragment Depot-Level-Inspection (Aircraft-Type = CT-114) , for 'V213' we have the fragment Depot-Level-Modification (Aircraft-Type = CT-114).

II-Processing-Schedule Graph for Updates:

The processing-schedule graph for updates done on the different entities of the Aircraft subject database will be described in the next paragraphs:

- a. Aircraft entity: Since the entity Aircraft is fragmented by aircraft-type, the appropriate fragment only is associated to its corresponding site. Figure C-3 can be used to find which fragment associates to which site. Figure C-12 illustrates the update done to the Aircraft fragment at CFB Gagetow.a.

Figure C-12: Update Aircraft Fragment Processing-Schedule Graph

Where:

- $f \cdot d1$, $f \cdot d2$, $f \cdot d3$ represent process/transaction P6.2.
 - $f \cdot d4$ represent the update done to the 'V134' which contain fragment Aircraft(Aircraft-Type = CH-135). Where f is the update frequency adjusted (if required) from figure B-3 and $d4$ is the total number of bytes obtained by adding the number of bytes for each individual attribute of the Aircraft entity i.e. $5 + 5 + 8 + 15 + 4 + 8 + 4 + 4 + 4 + 8 + 15 + 4 = 84$ bytes . The update frequency f had to be adjusted since there is 11 CH-135 and 10 CH-136 at CFB Gagetown (f was divided by 2). For CFB Gagetown $f \cdot d4 = 84 \text{ bytes} \cdot 1,168 \text{ a year} = 98,000 \text{ bytes/year}$. The results for the update of the different Aircraft fragments at each different sites are illustrated at figure B-14.
 - $f \cdot d4'$ represent the update done from another site to the 'V134'. Here the update comes from CFB Valcartier and $f \cdot d4' = 108,000 \text{ bytes/year}$.
 - $f \cdot d5$ is computed by adding all the updates done on 'V134' coming from the different sites. Here $f \cdot d5 = 98,000 + 108,000 = 206,000 \text{ bytes/year}$. This represent the amount of updated data that is done on 'V23'.
- b. Flying-Time entity: The way to compute the update of the fragment Flying-Time(Aircraft-Type = CH-135) of 'V130' is the same then as previously explained for the Aircraft fragments. $f \cdot d4$ has to be adjusted by considering the total length of the entry Flying-Time which is now equal to $8 + 5 + 5 + 4 + 4 + 4 + 4 = 34$ bytes. $f \cdot d4$ still represents the update done from another site to the 'V130'. $f \cdot d5$ is still computed by adding $f \cdot d4$ to $f \cdot d4'$. The results for the update of the different Flying-Time fragments at each different sites are illustrated at figure C-14.

- c. Depot-Level-Inspection entry: The way to compute the update of the fragment Depot-Level-Inspection (Aircraft-Type = CH-135) of 'V136' is the same then as previously explained for the Aircraft fragments. $f*d4$ has to be adjusted by considering the total length of the entity Depot-Level-Inspection which is now equal to $8 + 5 + 5 + 15 + 15 = 48$ bytes. $f*d4'$ still represents the update done from another site to the 'V136'. $f*d5$ is still computed by adding $f*d4$ to $f*d4'$. The results for the update of the different Depot-Level-Inspection fragments at each different sites are illustrated at figure C-14.
- d. Transfers entity: Since the entity Transfers is fragmented by aircraft-type, the appropriate fragment only is associated to its corresponding site. Figure C-3 can be used to find which fragment associates to which site. Figure C-13 illustrates the update done to the Transfer fragment at CFB Gagetown.

Figure C-13: Update Transfer Fragment Processing-Schedule Graph

Where:

- $f*d4$ represents the update done to the 'V138' which contain fragment Depot-Level-Modification (Aircraft-Type = CH-135). Where f is the update frequency from figure C-3 adjusted (if required) and $d4$ is the total number of bytes obtained by adding the number of bytes for each individual attribute of the Depot-Level-Modification entity i.e. $5 + 15 + 5 + 5 + 8 + 50 = 88$ bytes. For CFB Gagetown $f*d4 = 88 \text{ bytes} * 1,168 \text{ a year} = 103,000 \text{ bytes/year}$. The result for the update of the different Depot-Level-Modification fragments at each sites are illustrated at figure C-14.
- $f*d4'$ represents the update done from another site to the 'V138'. Here the update comes from CFB Valcartier and $f*d4' = 113,000 \text{ bytes/year}$.

- e. Periodics entity: The way to compute the update of the fragment Periodics (Aircraft-Type = CH-135) of 'V140' is the same then as previously explained for the Depot-Level-Modification fragments. f*d4 has to be adjusted by considering the total length of the entity Periodics which is equal to 92 bytes. The results for the update of the different Periodics fragments at each different sites are illustrated at figure C-14.
- f. Transfer entity: The way to compute the update of the fragment Transfer (Aircraft-Type = CH-135) of 'V132' is the same then as previously explained for the Depot-Level-Modification fragments. f*d4 has to be adjusted by considering the total length of the entity Transfer which is equal to 79 bytes. The results for the updates of the different Transfer fragments at each different sites are illustrated at figure C-14.

Military Base:	Plane Type	Flying-Time		Trans-fers		Aircraft		Depot-Le vel-Insp.		Depot-Le vel-Mod.		Periodics	
1 - CFB Baden	48 CF-18	V49	514K	V52	1192K	V50	1270K	V51	726K	V53	1331K	V54	1391K
2- CFB North Bay	21 CT-133	V60	63K	V61	146K	V59	155K	V58	89K	V62	163K	V63	170K
	7 CE-144	V91	63K	V92	146K	V93	155K	V94	88K	V95	162K	V96	170K
3- CFB Moose Jaw	89 CT-114	V100	462K	V101	1073K	V102	1141K	V103	652K	V104	1196K	V105	1250K
4- CFB Winnipeg	7 CT-114	V100	140K	V101	325K	V102	345K	V103	197K	V104	362K	V105	378K
5- CFB Ottawa(S)	4 CH-147	V112	98K	V114	226K	V116	241K	V118	138K	V120	252K	V122	264K
	7 CC-109	V113	146K	V115	340K	V117	361K	V119	206K	V121	378K	V123	395K
6- CFB Shearwater	35 CH-124	V124	547K	V125	1270K	V126	1350K	V127	772K	V128	1415K	V129	1479K
7- CFB Valcartier	6 CH-135	V130	44K	V132	102K	V134	108K	V136	62K	V138	113K	V140	119K
	8 CH-136	V131	44K	V133	102K	V135	108K	V137	62K	V139	113K	V141	119K
8- CFB Gagetown	11 CH-135	V130	40K	V132	92K	V134	98K	V136	56K	V138	103K	V140	107K
	10 CH-136	V131	40K	V133	92K	V135	98K	V137	56K	V139	103K	V141	107K
9- CFB Summerside	15 CP-121	V78	233K	V79	541K	V77	576K	V76	329K	V80	604K	V81	631K
10- CFB Greenwood	14 CP-140	V69	614K	V70	1427K	V68	1517K	V67	867K	V71	1589K	V72	1662K
11- CFB Bagotville	24 CF-18	V49	119K	V52	463K	V50	493K	V51	282K	V53	516K	V54	540K
12- CFB Cold Lake	63 CF-18	V49	1000K	V52	2200K	V50	2300K	V51	1300K	V53	2600K	V54	2600K
	12 CT-133	V60	121K	V61	400K	V59	470K	V58	282K	V62	300K	V63	400K

Figure C-14: Processing-Schedule Graph Summary
for Updates of Aircraft Subject Database Fragments

Military Base	Plane Type	Flying-Time		Trans-fers		Aircraft		Depot-Le vel-Insp.		Depot-Le vel-Mod.		Periodics	
14- CFB Edmonton	4 CH-147	V112	82K	V114	190K	V116	202K	V118	116K	V120	212K	V122	222K
	6 CC-138	V162	82K	V165	190K	V168	202K	V171	116K	V174	212K	V177	222K
	10 CC-130	V163	82K	V166	190K	V169	202K	V172	116K	V175	212K	V178	222K
15-CFB Trenton	6 CC-115	V87	169K	V88	392K	V86	417K	V85	238K	V89	437K	V90	457K
	14 CC-130	V163	338K	V166	785K	V169	835K	V172	477K	V175	875K	V178	914K
	5 CC-137	V161	169K	V164	392K	V167	417K	V170	238K	V173	437K	V176	457K
16-CFB Comox	3 CC-115	V87	60K	V88	139K	V86	148K	V85	85K	V89	155K	V90	162K
	3 CP-121	V78	60K	V79	139K	V77	148K	V76	85K	V80	155K	V81	162K
	4 CT-133	V60	60K	V61	139K	V59	148K	V58	85K	V62	155K	V63	162K
	4 CP-140	V69	60K	V70	139K	V68	148K	V67	85K	V71	155K	V72	162K
13- CFB Portage	11 CT-114	V100	33K	V101	76K	V102	81K	V103	46K	V104	85K	V105	89K
	21 CT-134	V144	66K	V147	153K	V150	162K	V153	93K	V156	170K	V159	178K
	15 CH-139	V145	33K	V148	76K	V151	81K	V154	46K	V157	85K	V160	89K

Figure C-14: Processing-Schedule Graph Summary
for Updates of Aircraft Subject Database Fragments.

4. Data Allocation: By putting together on the same graph all the individual processing-schedule graph developed at step 3 and documented at figure C-9 , C-11 and C-14 , it results in a graph called the initial processing schedule graph for the Aircraft subject database. This is illustrated at figure C-15. Out of all the different possible fragments to consider for the query AMMIS Q2 at figure C-11 only the one with fragment 'Aircraft-Type = CT-114' was considered. Figure C-16 is a subset of figure C-15 and it shows on a bigger scale the processing-schedule for physical site 1, site 11 and site 12.(Ref figure C-3 for corresponding site description).The number on the arcs are the traffic of data between the different sites (virtual or Physical) in kilo-bytes/year. As for figure C-16 , figure C-17 to figure C-25 increases the scale of figure C-15.

Once the initial processing schedule graph is loaded into the developed methodology companion then the next step is to start the model 16A allocation algorithm described at chapter 4. The original results are at figure C-26. This figure shows virtual sites piled on

top of each other at their assigned physical sites. These piles represent union of virtual sites. Since this figure is hard to read the designer can move the virtual sites besides each other to give figure C-27. This figure shows much better the final allocation of the virtual sites to their assigned physical sites. The total traffic of data (kilo-bytes/year) between the different physical sites is written on the arcs between these physical sites. For example, between physical site 11 and 12 there will be a total of 2494 kilo-bytes/year. By referring back to figure C-9, C-11 and C-14 the fragments corresponding to each virtual sites can easily be identified. Figure C-28 to figure C-36 are there to increase the scale of figure C-27.

5. Data Dictionary Allocation: This part is not done on the real world application, however the SDD-1 approach described at chapter 3 is the selected one. That approach has been selected since the reliability of the overall system is considered as an important issue. The way to implement this approach is to treat the data dictionary as a single relation(entity) which can be fragmented and is allocated as any other normal entity. Once this is done there is a higher-level data dictionary which is fully replicated at each sites. This data dictionary contains only the information as to where each fragment of the data dictionary are allocated.

Figure C-15: Aircraft Subject Database Initial Processing Schedule Graph

Figure C-16: Enlargement of Figure C-15 for PH1, PH11 and PH12

Figure C-17: Enlargement of Figure C-15 for PH2, PH12 and PH16

Figure C-18: Enlargement of Figure C-15 for PH10, PH12 and PH16

Figure C-19: Enlargement of Figure C-15 for PH9, PH10 and PH16

Figure C-20: Enlargement of Figure C-15 for PH15 and PH16

Figure C-21: Enlargement of Figure C-15 for PH14 and PH15

Figure C-22: Enlargement of Figure C-15 for PH3, PH13 and PH17

Figure C-23: Enlargement of Figure C-15 for PH3, PH4, PH13 and PH17

Figure C-24: Enlargement of Figure C-15 for PF5, PH6 and PH14

Figure C-25: Enlargement of Figure C-15 for PH7 and PH8

Figure C-26: Aircraft Subject Database Final Allocation

Figure C-27: Aircraft Subject Database Final Allocation

Figure C-28: Enlargement of Figure C-27 for PH1, PH11 and PH12

Figure C-29: Enlargement of Figure C-27 for PH2, PH12 and PH16

Figure C-30: Enlargement of Figure C-27 for PH10, PH12 and PH16

Figure C-31: Enlargement of Figure C-27 for PH9, PH10 and PH16

Figure C-33: Enlargement of Figure C-27 for PH14 and PH15

Figure C-34: Enlargement of Figure C-27 for PH3, PH4, PH7 and PH13

Figure C-35: Enlargement of Figure C-27 for PH5, PH6 and PH14

Figure C-36: Enlargement of Figure C-27 for PH6, PH7 and PH8

END

2002/9 1

FIN