

THESIS / THÈSE

MASTER IN COMPUTER SCIENCE

The Trabecular Bone and Morphological Analysis System : a Research and an Educational Training System for Students

Goossens, Pascal; Wauthier, Benoît

Award date:
1995

Awarding institution:
University of Namur

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR

INSTITUT D'INFORMATIQUE

RUE GRANDGAGNAGE, 21, B-5000 NAMUR (BELGIUM)

**The
Trabecular Bone and Morphological
Analysis System : a Research and
an Educational
Training System for Students**

Pascal GOOSSENS - Benoît WAUTHIER

Promoteur : Mr. Professor François BODART, from FUNDP

Co-promoteur : Mr. Professor Tony S. KELLER, from UVM

**Mémoire présenté en vue de l'obtention du grade de
Licencié et Maître en Informatique**

Année Académique 1994 - 1995

Thanks

We wish to thank Professors François BODART, Tony S. KELLER and Eric DUBOIS for their precious advice.

We would also like to thank two members of the TRIDENT Project Team, Messrs Jean VANDERDONCKT and Jean-Marie LEHEUREUX for their help and precious advice concerning the TRIDENT Methodology.

And we would also like to thank the staff of the Biomechanical Engineering Lab of the University of Vermont, USA.

Finally, we would like to thank Mr. M. DEVAUX for his help concerning our english.

Table of Contents

CH. 1 Introduction	1.1
CH. 2 An Introduction to Biomedical Engineering of the Spine	2.1
1. Osteoporosis of the Spine : The Problematic	2.2
1.1 Introduction	2.2
1.2 Epidemiology	2.2
1.3 Basic Bone Physiology	2.5
1.3.1 Modeling and Remodeling	2.5
1.3.2 Stimulus for Skeletal Adaptation	2.7
1.4 Vertebral Strength	2.8
1.4.1 Vertebral Fracture Threshold	2.9
1.4.2 Experimental Assessment of Skeletal Strength and Fracture Risk	2.9
1.4.3 Relationship Between Bone Density and Compressive Strength	2.10
1.4.4 Clinical Estimates of Skeletal Strength and Fracture Risk	2.12
1.4.5 Influence of Vertebral Level on Mechanical and Physical Properties	2.13
1.4.6 Rate of Bone Loss in Aging Trabecular and Cortical Bone	2.14
1.5 Vertebral Morphology	2.15
1.5.1 Structural Indices of Trabecular Bone	2.16
1.5.2 Relationship Between Bone Structural Indices and Mechanical Properties	2.19
1.5.3 Simulations of Osteoporosis in Human Vertebrae	2.20
1.6 Biomechanical Adaptation of the spine	2.22
1.7 Failure Patterns in Osteoporosis	2.24
1.8 Radiographic Assessment And Spinal Osteoporosis	2.26
2. Overview of the Thesis	2.28
CH. 3 The Existing Set of Programs	3.1
1. Functionalities	3.2
1.1 Existing Set of Programs Functionalities	3.2
1.2 The TBMAS Software Functionalities	3.3
1.2.1 The MIL Analysis	3.3
1.2.1.1 Theoretical Introduction to the MIL Analysis	3.3
1.2.1.2 The MIL Computation Algorithm	3.4
1.2.2 The Euler Number	3.5
1.2.2.1 Theoretical Introduction to the Euler number	3.6
1.2.2.2 The Euler Number Computation Algorithm	3.6
1.2.3 The Osteoporosis Simulation	3.7
2. Technical aspect	3.8
2.1 Software setup and display board	3.8
2.2 Drive setup and files	3.9
2.2.1 Drive setup	3.9
2.2.2 Input files	3.9
2.2.3 Output files	3.11

CH. 4 Development Process.....	4.1
1. Requirements Analysis	4.2
1.1 Requirements Model	4.2
1.2 The Different Types of Requirements	4.3
1.2.1 Context Analysis	4.3
1.2.2 Functional Requirements	4.5
1.2.2.1 Functionalities	4.5
1.2.2.2 User Friendly Interface	4.6
1.2.2.2.1 Interface Definition	4.6
1.2.2.2.2 Text Oriented Interface Vs Graphics Oriented Interface	4.7
1.2.2.2.3 A Global Concept : Windows	4.8
1.2.3 Non-Functional Requirements	4.9
1.2.3.1 Performances	4.9
1.2.3.2 Programming Tools and Languages	4.10
1.2.3.2.1 The Watcom C/C++ Compiler 10.0	4.10
1.2.3.2.2 Microsoft Visual Basic 3.0 Professional Edition	4.13
2. Functional Specifications	4.19
2.1 Perform MIL Analysis Functional Specifications	4.19
2.1.1 Perform MIL Function	4.19
2.1.2 Get_Test_Area Function	4.22
2.1.3 Find_Specimen_Centroid Function	4.24
2.1.4 Compute_Rotation Function	4.24
2.1.5 Compute_MIL Function	4.26
2.2 Compute Euler Number Functional Specifications	4.27
2.3 Perform Osteoporosis Simulation Functional Specifications	4.28
3. Implementation Analysis	4.30
3.1 The Software Architecture	4.30
3.1.1 Description of the Software Architecture	4.30
3.1.1.1 Global Architecture	4.30
3.1.1.2 Module Architecture	4.31
3.1.1.2.1 « Perform New Analysis » Module Architecture	4.32
3.1.1.2.2 « Visualize Existing Bitmap » Module Architecture	4.32
3.1.1.3 An Object Oriented Architecture	4.35
3.1.2 Justification of the Software Architecture	4.37
3.1.2.1 Meeting Functional Requirements	4.37
3.1.2.1.1 Functionalities	4.37
3.1.2.1.2 User Friendly Interface	4.37
3.1.2.2 Meeting Non-Functional Requirements	4.38
3.1.2.2.1 Performances	4.38
3.1.2.2.2 Programming Tools and Languages	4.39
3.1.2.3 Conclusion	4.39
3.2 Specific Programming Techniques and Algorithms	4.39
3.2.1 The Utilization of Dynamic-Link Libraries (DLLs)	4.39
3.2.1.1 What is a DLL ?	4.40
3.2.1.2 Why using DLLs ?	4.41
3.2.2 Memory Management	4.42
3.2.2.1 The Bidimensional Memory Arrays	4.42
3.2.2.2 Memory Management Techniques	4.44
3.2.3 Compression of Binary Files	4.45
3.2.3.1 Motivations for a Compression Process	4.45
3.2.3.2 Data Compression Terminology	4.45
3.2.3.2.1 Physical and logical compression	4.46
3.2.3.2.2 Symmetrical and asymmetrical compression	4.46

3.2.3.2.3 Lossy and lossless compressions	4.47
3.2.3.3 The LRLE method	4.47
3.2.3.3.1 The logical compression	4.47
3.2.3.3.2 The physical compression	4.48
3.2.3.3.3 The LRLE algorithm	4.52
3.2.4 Multiple-Document Interface (MDI) Application	4.53
3.2.5 Bitmap Creation and 3D Representation	4.54
3.2.5.1 Bitmap files	4.54
3.2.5.2 The Microsoft Windows Bitmap File Format	4.56
3.2.5.3 Why the Bitmap File Format ?	4.57
3.2.5.4 3D Representation	4.58
3.3 Relations of the TBMAS Program with External Applications	4.61
3.4 Performances Measurements	4.62
3.5 An Empirical Reverse Engineering Method	4.65
3.5.1 Migrating Legacy Information Systems : The Problematic	4.65
3.5.2 Strategies for Migrating Legacy IS	4.67
4. Conclusion	4.73
 CH. 5 Another Approach for the Development Process : the Application of the TRIDENT Methodology	 5.1
1. Introduction	5.2
2. The Methodological Framework	5.3
3. Graphical User Interface Specifications	5.5
3.1 The Task Analysis	5.5
3.1.1 The Box Labeled : « Performing Analysis »	5.7
3.1.2 The Box Labeled : « Visualizing Analysis Results »	5.8
3.1.3 The Box Labeled : « Creating Image »	5.9
3.1.4 The Box Labeled : « Visualizing Image »	5.10
3.1.5 The Box Labeled : « Simulating Osteoporosis »	5.11
3.2 Relations between the Sub-tasks	5.11
3.3 Description of the Interactive Task	5.12
3.3.1 Description of the Parameters	5.13
3.3.1.1 Prerequisites of the Task	5.13
3.3.1.2 The Productivity of the Task	5.13
3.3.1.3 The Objective Environment of the Task	5.13
3.3.1.4 The Reproducibility of the Environment	5.13
3.3.1.5 The Structuration of the Task	5.13
3.3.1.6 The Importance of the Task	5.13
3.3.1.7 The Complexity of the Task	5.14
3.3.2 Application of the Parameters	5.14
3.3.2.1 Prerequisites of the Task	5.14
3.3.2.2 The Productivity of the Task	5.14
3.3.2.3 The Objective Environment of the Task	5.14
3.3.2.4 The Reproducibility of the Environment	5.14
3.3.2.5 The Structuration of the Task	5.15
3.3.2.6 The Importance of the Task	5.15
3.3.2.7 The Complexity of the Task	5.15
3.4 Description of User Stereotypes	5.15
3.4.1 Description of the Parameters	5.15
3.4.1.1 The Experience of the Task	5.15
3.4.1.2 The Experience of Systems	5.16
3.4.1.3 The Motivation	5.16

3.4.1.4 The Experience of a Complex Interaction Mean	5.16
3.4.2 Application of the Parameters	5.16
3.4.2.1 The Beginner Users	5.17
3.4.2.2 The Advanced Users	5.17
3.5 Workplace Description	5.18
3.5.1 Description of the Parameters	5.18
3.5.1.1 Type of Treatment	5.18
3.5.1.2 Capacity of Treatment	5.18
3.5.2 Application of the Parameters	5.18
3.5.2.1 Type of Treatment	5.18
3.5.2.2 Capacity of Treatment	5.19
3.6 Expressing the Product of Task Analysis	5.19
3.6.1 The Object-Oriented Entity-Relationship Model	5.19
3.6.2 Identification of Semantic and Control Functions of the Application	5.20
3.6.2.1 Semantic and Control Functions for the « Performing Analysis » sub-task	5.21
3.6.2.2 Semantic and Control Functions for the « Visualizing Analysis' Results » sub-task	5.24
3.6.2.3 Semantic and Control Functions for the « Creating Image » sub-task	5.27
3.6.2.4 Semantic and Control Functions for the « Visualizing Image » sub-task	5.29
3.6.2.5 Semantic and Control Functions for the « Simulating Osteoporosis » sub-task	5.31
3.6.3 Construction of Activity Chaining Graphs	5.32
3.6.3.1 Activity Chaining Graph for the « Performing Analysis » sub task	5.33
3.6.3.2 Activity Chaining Graph for the « Visualizing Analysis Results » sub-task	5.34
3.6.3.3 Activity Chaining Graph for the « Creating Image » sub-task	5.34
3.6.3.4 Activity Chaining Graph for the « Visualizing Image » sub-task	5.35
3.6.3.5 Activity Chaining Graph for the « Simulating Osteoporosis » sub-task	5.35
3.6.4 Derivation of Dialogue Attributes	5.36
3.6.4.1 Dialogue Mode	5.36
3.6.4.2 Dialogue Control	5.37
3.6.4.3 Function Triggering Mode	5.37
3.6.4.4 Metaphor	5.37
3.7 Conclusion	5.38
4. Presentation Design	5.39
4.1 Methodological Considerations (issued from the TRIDENT Project)	5.40
4.1.1 Presentation Content ^[BODART95]	5.40
4.1.2 Systematic approach for specifying presentation ^[BODART95]	5.40
4.2 Application of the TRIDENT Methodology	5.43
4.2.1 Identification of Presentation Units	5.43
4.2.2 Identification of Windows	5.45
4.2.2.1 Identification of Windows for the PU1	5.45
4.2.2.2 Identification of Windows for the PU2	5.47
4.2.2.3 Identification of Windows for the PU3	5.49
4.2.2.4 Identification of Windows for the PU4	5.51
4.2.2.5 Identification of Windows for the PU5	5.53
4.2.3 Selection of AIOs	5.55
4.2.3.1 Abbreviations used in the following sections	5.56
4.2.3.2 AIOs Selected for the Group « Mother Window and Child Windows »	5.57
4.2.3.3 AIOs Selected for the Group « Open File Dialog Box »	5.57
4.2.3.4 AIOs Selected for the Group « Analysis Choice Dialog Box »	5.58
4.2.3.5 AIOs Selected for the Group « Parameters Dialog Box »	5.59
4.2.3.6 AIOs Selected for the Group « Index Dialog Box »	5.62
4.2.3.7 AIOs Selected for the Group « Information and Error Windows »	5.63
4.2.4 Transformation of AIOs into CIOs	5.64
4.2.5 Placement of CIOs	5.64
4.2.5.1 Group of « Mother Window and Child Windows »	5.66
4.2.5.2 Group of « Open File Dialog Box »	5.68
4.2.5.3 Group of « Analysis Choice Dialog Box »	5.68

4.2.5.4 Group of « Parameters Dialog Box »	5.69
4.2.5.5 Group of « Index Dialog Box »	5.70
4.2.5.6 Group of « Information and Error Windows »	5.71
5. Derivation of the Software Architecture	5.72
5.1 Methodological Considerations^[BODART95]	5.72
5.1.1 Assumptions	5.72
5.1.2 Content of the model	5.73
5.1.2.1 Application Objects (AO)	5.74
5.1.2.2 Control Objects (CO)	5.74
5.1.2.3 Interaction Objects (IO)	5.75
5.1.2.4 Inter-object relations	5.76
5.1.3 Systematic approach to hierarchy building	5.77
5.1.3.1 List of hierarchy objects	5.77
5.1.3.2 Relationships between hierarchy objects	5.77
5.2 Application of the TRIDENT Methodology	5.78
5.2.1 Construction of Functional Description's Hierarchy	5.78
5.2.1.1 « Performing Analysis » Functional Description Hierarchy	5.79
5.2.1.2 « Visualizing Analysis Results » Functional Description Hierarchy	5.80
5.2.1.3 « Creating Image » Functional Description Hierarchy	5.81
5.2.1.4 « Visualizing Image » Functional Description Hierarchy	5.82
5.2.1.5 « Simulating Osteoporosis » Functional Description Hierarchy	5.83
5.2.2 Construction of the Control Objects of the Task Hierarchy	5.84
5.2.3 Construction of Interactive Objects Hierarchy	5.85
6. Dialogue Specifications	5.93
6.1 Methodological Considerations	5.94
6.1.1 Dialogue content	5.94
6.1.2 Systematic approach for specifying a dialogue	5.95
6.2 Application of the TRIDENT Methodology	5.97
6.2.1 Inter-PU Dialogues Specifications	5.97
6.2.2 Intra-PU Dialogues Specifications	5.101
6.2.3 Intra-Window Dialogues Specifications	5.104
7. Transformation of the TRIDENT Architecture in Visual Basic	5.107
7.1 The Visual Basic Architecture	5.107
7.2 Transformation of the TRIDENT Architecture in Visual Basic	5.108
7.3 Comparison with the Method we have used	5.111
8. Conclusion on the Application of the TRIDENT Methodology	5.113
CH. 6 Conclusion	6.1
CH. 7 Further Developments	7.1
1. Windows NT and 32-bit Programming	7.2
2. Providing OLE Support	7.3
3. Texture Mapping	7.5
CH. 8 References	8.1
1. Books	8.2
2. Articles	8.4

Appendix A	A.1
1. Introduction	A.2
2. The Directed Secant Method	A.3
3. Methods	A.5
3.1 Phase Distribution Method	A.7
3.2 Primary Direction Method	A.9
4. Practical Considerations	A.13
Appendix B	B.1
1. Specimen Preparation	B.2
2. Imaging Process	B.4

CHAPTER 1

Introduction

The object of this thesis is to design a **MS Windows application** based on an existing **set of DOS programs**. This application is aimed to serve as both a **research and an educational training system for students**. It allows them to perform **analyses**, i.e. MIL analysis, on digitized trabecular bone specimens and **to display charts** based upon these analyses results. It may also serve as a **diagnostic support** in the field of **Osteoporosis of the Spine**.

The Thesis is divided into four main parts :

- First, we introduce the necessary biomedical principles and we explain in particular the osteoporosis of the spine problematic.
- Then we describe the **existing DOS application**, stressing **requirements** we have been asked to fulfill in order to make up its deficiencies and shortcomings as well concerning functionalities than performances or user friendliness. And, we present the complete **Development Process** of our program as well as the **original programming techniques** we have used and the **original algorithms** we have developed.
- Finally, we will apply the **TRIDENT Methodology** as another approach for the Development Process

After the conclusion, we introduce **further developments** that seem interesting to consider. These take into consideration next generation operating system such as **MS Windows NT** which provide full 32-bit support and which look promising to provide even better performances. We will also present some other improvements susceptible to make **interaction** with other applications easier for the user, and aimed at providing him with a better **three-dimensional representations** of digitized trabecular bone specimen.

CHAPTER 2

An Introduction to Bio-medical Engineering of the Spine

In this chapter, we introduce the basic vocabulary, the basis knowledge and the key points of biomedical engineering of the spine needed to understand the rest of our thesis. We first present the problematic of Osteoporosis defined in [Hansson-Keller95].

<i>1. Osteoporosis of the Spine : The Problematic</i>	2
1.1 Introduction	2
1.2 Epidemiology	2
1.3 Basic Bone Physiology	5
1.4 Vertebral Strength	8
1.5 Vertebral Morphology	15
1.6 Biomechanical Adaptation of the spine	22
1.7 Failure Patterns in Osteoporosis	24
1.8 Radiographic Assessment And Spinal Osteoporosis	26
<i>2. Overview of the Thesis</i>	28

1. Osteoporosis of the Spine : The Problematic

1.1 Introduction

Osteoporosis, which is characterized by a reduction in skeletal bone mass and concomitant change in skeletal structure, produces an increased risk of fracture in patients and thus has a devastating effect in terms of morbidity, mortality and cost of health care in our increasing senile population. Osteoporosis affects both the appendicular and axial skeleton of adults, and is a well recognized public health problem of increasing proportions. Over 1.2 million fractures occur in the United States each year, including over 500,000 cases of vertebral fracture, and 200,000 cases of hip fractures, one-third to one-half of which occur in women over the age of 65. In the United States, the personal and medical costs associated with osteoporotic fractures are expected to increase dramatically in the next two decades, since the number of individuals over the age of 65 has been predicted to double by the year 2010 [1983 United States census].

A close association between bone mineral loss due to osteoporosis and the risk of fracture has been clearly established. Skeletal structures, such as the vertebral bodies and proximal femur, which are comprised primarily of trabecular bone appear to be particularly at risk. Thus, development of clinical diagnostic tools sensitive enough to identify imminent fracture or collapse of vertebral bodies and other weight-bearing tissues is essential. Until these tools are developed, the ability of a clinician to clearly evaluate a patient's bone status, prevent osteoporosis or determine the effect of therapeutic treatments is severely limited.

1.2 Epidemiology

The aging skeleton is characterized by a gradual loss of bone mass which decreases bone strength (force or stress at failure) and increases fracture risk. A more rapid loss of bone mass occurs in post-menopausal women, and collecti-

vely these processes are referred to as primary osteoporosis. At the present time the precise etiology of primary osteoporosis is unknown.

Because of increased morbidity and immobility produced by hip fractures, many epidemiological studies of osteoporosis have focused on hip fractures. Until recently, vertebral fractures were deemed to be of lower incidence and concern than hip fractures. There are, however, no reasons why increases in the incidence of hip fractures should not reflect a similar increasing incidence of osteoporotic spine fractures. A recent Swedish study found that 43% of the subjects who had a hip fracture also had one or more vertebral fractures of an osteoporotic type [Zetterberg et al 1990]. Of note, is the fact that the prevalence of osteoporosis seems to have become more and more common, particularly in industrialized countries. This increase is partly explained by the fact that populations in most industrialized countries are growing older but also by an increased risk. The osteoporotic vertebral fracture is probably the most frequent of all fragility fractures, particularly if every vertebral fracture in the spine is considered.

Vertebral fracture is about four times more common in women than in men, and the risk for a vertebral fracture has been found to increase almost exponentially with age. The frequency of osteoporotic vertebral fracture also increases during menopause in women. From this point on there is a steady increase in vertebral fracture frequency throughout life. In this respect the vertebral fragility fractures differs from fractures of the distal radius. The prevalence of the latter increases at the same age as the vertebral fracture, but levels out after age 60-65. An interesting recent finding is that the increase in risk for a fragility fracture between 1985 and 1991 was almost twice as high for men as for women [Zetterberg et al, 1994]. Depending on the age groups studied (40 to >80 years), the prevalence of osteoporotic vertebral fractures varies from around 5% to somewhat over 50%.

Radiographically detectable compression fractures of the spine for most clinicians has verified the presence of osteoporosis or bone fragility. Without any

known pathomorphological aberrations distinguishing osteoporotic bone from non-osteoporotic bone tissue, the fracture itself defines pathology. Since the occurrence of a fracture is not only the result of the mechanical properties of the bone, but is also a function of the fracturing trauma, both factors must be considered when defining osteoporosis. In the presence of a patient with a recent fracture, knowing nothing or very little about the patient's bone quality or the forces involved in the trauma, the most practical way for clarifying whether a fracture is osteoporotic or not, is Harold Frost's criteria of the « everyday trauma ». Frost stated that a fracture occurring as a consequence of an everyday trauma indicates that the patient has osteoporosis or bone fragility. Even if the technology today allows us to determine, for example, the amount of bone mineral in different parts of the human skeleton we still lack practical techniques for measuring the fracture generating forces. Therefore the « everyday trauma » definition is still a practical measure for estimating bone fragility [Frost 1993].

Without any distinct differences between the bone tissue in the osteoporotic versus the normal subject there are, however, apparent difficulties in assessing the limits for normality. Since demineralization of the human skeleton is usually a more or less continuous process from relatively early in life, weakening of the skeleton is a part of normal life and aging. An osteoporotic or fragility fracture occurs in those subjects in which the demineralization progresses to a level where the spine or other parts of the skeleton no longer can resist an everyday trauma. In many subjects with spinal osteoporosis the vertebrae may become so demineralized that they can not resist the spinal loads accompanying everyday life. Since the amount of bone mineral in combination with the loading conditions determines the occurrence of a fracture, a subject with a *low amount of bone mineral*, but *no fracture*, has *osteopenia*. A subject with a *low amount of bone mineral* and a *fracture* sustained during a minor « everyday » trauma is likely to have *osteoporosis*.

1.3 Basic Bone Physiology

Bone is a two-phase, porous, directional composite material, comprised of hydroxyapatite (inorganic or mineral phase) and collagen (organic phase). In the normal adult skeleton, hydroxyapatite constitutes approximately 2/3 of the weight or about 50% of the volume of dry bone tissue. Bone composition can be described by several histologic variables, including mineral content, porosity and density. The density (mass/volume) may refer to either the wet or dry tissue density (mass/unit volume of solids) or the wet or dry bulk density (mass per unit volume of a region of bulk bone). Bone devoid of pores has a tissue density or specific gravity of approximately 2 g/cm³. Bulk density or apparent density (ρ_a), however, is a measure of both the porosity and mineral content of bone and range from < 0.1 g/cm³ to approximately 2.0 g/cm³. All of these histologic variables have been used to describe the composition of bone. From a morphological point-of-view, two principal types of bone are recognized : **cortical** and **cancellous**. In the adult skeleton, both cortical and cancellous bone have roughly the same amount of mineral, except in metabolic diseases such as osteogenesis imperfecta for which the mineral content is significantly reduced. Cortical or « compact » bone is generally distinguished from cancellous or « trabecular » bone by its lower porosity (< 30% pores by volume) and higher apparent density (> 1.7 g/cm³), and is most prevalent in the shafts of long bones. The ends of long bones and the axial skeleton (spine) are comprised primarily of trabecular bone, which in the case of the axial skeleton has a porosity greater than 70% or an apparent density less than 0.6 g/cm³. By virtue of its inherent porosity, trabecular bone has an extremely complex structure or « architecture ». Decreases in bone mass associated with aging, inactivity and menopause have profound effects on the architecture of trabecular bone. Collectively, the changes or « adaptations » in skeletal mass and architecture are referred to as modeling and remodeling processes.

1.3.1 Modeling and Remodeling

Modeling and remodeling are two different biologic activities which can affect the architecture of cortical and trabecular bone.

Modeling or structural adaptation can change the shape and size of the surfaces of the bone via resorption and formation processes. Adaptation can take place as macro-change in the shape or size of an entire skeletal part, mini-change in the shape and size of an individual trabecula, or mini-modeling- a change which determines the type of bone formed. Modeling is postulated to have an aging component as well as loading component, and in terms of the latter structural adaptation seems to be the response to the strain history experienced by the tissue. Modeling can increase bone mass, but can not decrease bone mass.

Remodeling takes place through basic multicellular units or BMUs. Each unit remodels a microscopic quantum or packet of bone. The remodeling sequence is an orderly cascade including activation, resorption and formation or ARF. Activation involves the initiation of the resorptive ability of the BMU. Resorption by osteoclast cells create cavities when bone and matrix is removed. Formation is the repair phase by which the resorption cavities are filled with new tissue via osteoblast cells. The end result of the Activation-Resorption-Formation BMU sequence is a new « packet » of bone or so-called Basic Structural Unit (BSU).

The amount of bone replaced in one remodeling sequence (about four months) is roughly 0.05 mm^3 . There are three possible outcomes for the BSU during one remodeling sequence : a net increase in the amount of bone, net decrease in the amount of bone, or equal amount of bone. Remodeling takes place on all surfaces of the bone-periosteal, cortical-endosteal, haversian and trabecular. Without pharmacological help, however, a net gain of bone can only appear on the periosteal surface. On the trabecular and cortical endosteal surfaces the remodeling sequences are generally negative. The trabecular and cortical endosteal surface bone loss during one ARF sequence is estimated to about -0.003 mm^3 . Bone remodeling is a lifelong process, and is considered to be a somewhat faster process in children than in adults. Remodeling events explain skeletal changes such as the expansion of the marrow cavities in long bones, the widening of these bones (increased outer diameter) as well as the concomi-

tant thinning of the cortices with increasing age. Bone remodeling also explains why skeletal structures proportionally rich in spongy bone (vertebrae, the hip region, proximal humerus, distal radius, the pelvic bones, the knee region) are more prone to loose bone mass and throughout life.

Microfracturing or fatigue fracture formation is another means in which both modeling and remodeling can be affected. Fatigue fracture formation is considered to be a normal occurrence in the human skeleton. Indeed, healing microfractures have been found in many locations within the skeleton, including the spine and hip. Hence, one must assume that both modeling and remodeling will respond to microfracturing in order to prevent the accumulation of microtrauma, which if unchecked will produce gross fracture. Remodeling processes will replace the damaged bone with new bone, while modeling processes will modify bone structure in a manner which may make it more resistant to failure during periods of overloading.

1.3.2 Stimulus for Skeletal Adaptation

Although numerous theories regarding the stimuli for skeletal adaptation have been proposed, tissue strain (deformation per unit length) is considered by many investigator to be of central importance to this processes [Keller et al 1989b]. A normal bone fractures at strains corresponding to about $25,000 \mu\epsilon$ (microstrain, or 10^{-6} mm/mm). During normal daily activity, bone strains do not usually exceed about $1500 \mu\epsilon$. Strains above this magnitude are likely to initiate modeling processes - woven bone formation in the immature skeleton, and lamellar bone formation in the mature skeleton. Continuous exposure to strains lower than $1500 \mu\epsilon$ presumably do not elicit a modeling response. However, during exposure of the skeleton to strains below $50 \mu\epsilon$, such as that occurring during episodes of acute disuse, skeletal remodeling will be initiated through the activation of more BMU's than normal. Consequently, the trabecular and endosteal surfaces will experience a deficit in the formation of new bone in the ARF sequence. Since no compensatory mechanisms (modeling and microfracturing) are present during the disuse period, the result will be a net loss of bone.

During normal activity, strains in the range 50 - 1500 $\mu\epsilon$ are hypothesized to keep resorption and formation at a similar magnitude. This implies that, over a long period, bone will be lost since the ARF remodeling sequence tends to keep the balance between resorption and formation towards the negative side. Physical exercise producing strains of sufficient magnitude and/or changes in skeletal loading which alter the distribution of strains within a skeletal structure can activate modeling processes to form new lamellar bone and thereby preserve or even increase skeletal mass, which in turn limit the range of strain in the skeleton. The overall result of a « strain limiting » strategy, therefore, is to preserve or maintain a « safety factor » within the bone material. Given the complexity of the skeleton's functional loading environment, it is not obvious how remodeling and modeling, which are cellular processes, could determine just how close to deleterious strain levels in the skeleton comes during activity, and adjust mass accordingly. Consequently, some researchers have proposed that bone cells are predominantly sensitive to strains within specific frequency bands and react to increases or decreases in the strain energy within that band. Another possibility is simply that the stimulus « threshold » at which bone cells respond in a positive or « osteogenic » manner is mediated by the stimulus frequency; certain frequencies eliciting a lower stimulus threshold in the bone cell transduction process.

1.4 Vertebral Strength

The spine is a weight bearing structure which, besides protecting the spinal cord and offering exceptional flexibility and range of motion, must continually support the weight of the torso and head. Together with everyday activities, these structures must support to a significant degree, axial compressive forces on the vertebrae and intervening disc tissues. In the L1-L4 lumbar spine this amounts to approximately 50-60% of the subjects body weight. Consequently, numerous investigators have examined the axial, compressive strength properties of cadaveric human thoracolumbar vertebrae. Ultimate strength values ranging from about 1 to 15kN have been recorded in these experimental studies, most of which have examined tissues from more age subjects (eg. > 40

years). To which extent the inability to obtain specimens representative of the entire population has influenced these strength values is hard to estimate. However, it is reasonable to assume that the compressive strength of vertebrae is grossly underestimated for ages below 50 years. Experimentally, as well as clinically, large variations in bone strength have made it very difficult to define a specific threshold or even a range with which to differentiate normal bone from osteoporotic bone. The latter also requires knowledge of the physiologic forces and stresses which act on the vertebral structures.

1.4.1 Vertebral Fracture Threshold

In vivo disc pressure measurements are still the most realistic reference values for estimating the actual physiologic loads acting on the human lumbar spine. By simply extrapolating Nachemson's lumbar disc pressure results corresponding to « everyday activities », one can predict that ultimate compressive stress values for the L3-L4 spine should be in the range 150-175 N/cm². Presumably this might represent the threshold at or below which the risk for a vertebral fracture is increased during « everyday » activities. In lieu of the very few clinical studies in which the magnitude of fracture trauma has been considered, however, it seems likely that these values are much too narrow in range to represent a realistic physiologic range with which to define vertebral fracture risk. As a result, considerable effort has been devoted to establishing reliable experimental and clinical measures of skeletal strength and fracture risk.

1.4.2 Experimental Assessment of Skeletal Strength and Fracture Risk

In vitro studies of cadaver spines have long ago revealed close relations between measures of the amount of bone (ash weight, wet and dry weight, apparent density, etc.), and the compressive strength of vertebrae. Initially, interest in characterizing the mechanical and physical properties of vertebrae and other skeletal structures originated in the need to evaluate fracture risk associated with aging and disease. More recently, this interest has intensified due to the need to understand implant-bone interactions and biological fixation. Thus, numerous *in vitro* studies of the physical and mechanical behavior of bony tissues can be found in the literature. Some of these studies have established si-

gnificant mathematical relationships between clinical, non-invasive measures of bone density and bone mechanical properties [Keller et al 1989a]. One of the most striking features of both the *in vitro* and *in vivo* data is the large variation in density, modulus and strength reported.

1.4.3 Relationship Between Bone Density and Compressive Strength

Both linear and non-linear mathematical relationships have been used to characterized the dependence of modulus and strength on trabecular bone density, with current sentiment among investigators tending to favor the power law formulations originally described for human and bovine trabecular and cortical bone tested in compression and tension. In describing trabecular bone compressive mechanical behavior, most investigators have found skeletal strength to be proportional to apparent density squared, but significant variations of data are commonly found among different studies. Variations in mineral content (ash fraction, calcium and phosphorus fractions) such as those which are evident during growth also influence bone mechanical properties. In the normal adult skeleton, however, variations in mineral content are generally very small and contribute very little to the mechanical response of cortical and trabecular bone, except in metabolic diseases such as osteogenesis imperfecta.

In a recent comprehensive study of human vertebral trabecular and femoral cortical bone, the apparent density of 496 specimens ranged from 0.05 to 1.89 g/cm³ resulting in over a 3000-fold difference in bone strength over this range [Keller 1994]. Bone strength was closely correlated to tissue apparent dry density (ρ_a), mineral fraction (α) and apparent ash density (ρ_α = mineral fraction x apparent dry density). The variation in bone strength was best described by power functions ($y=ax^b$) of bone apparent ash density. Bone strength (S, Mpa) was approximately proportional to the square of the apparent ash density (g/cm³) in the case of the vertebral bone specimens (n=200, $\rho_\alpha < 0.1$ g/cm³):

$$S = 284 \rho_\alpha^{2.27 \pm 0.09} (R^2 = 0.79) \quad (\text{eq. 1})$$

and to the square of the apparent ash density in the case of the femoral bone specimens ($n=296$, $0.1 < \rho < 1.22 \text{ g/cm}^3$) :

$$S = 116 \rho_{\alpha}^{2.03 \pm 0.03} (R^2 = 0.93) \quad (\text{eq.2})$$

each model explaining roughly 80-90% of the variation in bone strength. Theoretical analyses of the mechanisms of cell wall deformations of idealized open and closed cell porous engineering materials support the use of a squared power law to describe the relationship of bone strength to apparent density.

Over 90% of the variance in skeletal strength was explained by apparent density when the vertebral and femoral bone data were combined ($n=496$, $0.03 < \rho < 1.22 \text{ g/cm}^3$) :

$$S = 117 \rho_{\alpha}^{1.93 \pm 0.02} (R^2 = 0.97) \quad (\text{eq. 3})$$

Based upon the 97.5% confidence intervals associated with these models of bone strength, predictions obtained by extrapolating equation (1) high density values ($\rho_{\alpha} = 1.22 \text{ g/cm}^3$), resulted in gross underestimation of bone strength (approximately 3-fold) in comparison to the values obtained from the appropriate regression equation. Closer agreement between femoral model extrapolations of strength to the lowest density specimens was obtained. Similar predictions based upon the values obtained from the combined data (eq.3), however, resulted in close agreement to the 97.5% confidence intervals for strength computed separately from the vertebral (eq.1) and femoral (eq.2) bone data. These results support the notion that extrapolations do not yield valid results, and that reliable predictions can only be obtained over the data range evaluated.

This remarkable difference in mechanical properties reflects, at least in part, the skeleton's adaptive response to its mechanical environment. In addition to environmental and testing conditions, much of this variation can also be attributed to the anatomical origin and age of the specimens.

Recognizing that there is a close association between bone apparent density and bone mechanical properties has prompted numerous *in vivo* investigations focusing on establishing mathematical relationships between bone strength and clinical measures of skeletal density.

1.4.4 Clinical Estimates of Skeletal Strength and Fracture Risk

The advent of non-invasive techniques for determination of the amount of bone mineral in the spine, e.g. dual (DPA) and triple photon energy absorptiometry, quantitative computerized tomography (QCT), double (DEXA) and triple (RAT) energy x-ray absorptiometry, have made non-destructive bone quantifications possible *in vitro* as well as *in vivo*. Each of these techniques provide a precision of about 1-4 percent and are accurate to about 3-10 percent in comparison to direct measurements of bone ash content (inorganic weight). These techniques also allow quantitative measurement of bone density in relationship to average density for both age matched and younger individuals and can be used as a criterion for the definition of osteoporosis. The definition of osteoporosis has varied but is generally defined in the range of 2.0-2.5 standard deviations below the mean density for an average 30 year old adult of the same sex. Since all of these measurement approaches can be used both *in vitro* and *in vivo*, each have provided great advantages in spinal research.

In general non-invasive measurements of bone density - bone mineral content (BMC, g/cm), bone mineral density (g/cm²), and trabecular density (ρ , g/cm³)¹ - have explained roughly 25 to 80 percent of the variation in thoraco-lumbar vertebrae trabecular bone axial compressive strength when expressed as linear ($y=bx + a$) relationships or power ($y = ax^b$) relationships [Keller et al 1989a]. Vertebral trabecular bone strength (S, Mpa) predictions based upon BMC (g/cm) obtained from the latter study :

$$S = 0.13\text{BMC}^{2.25} \quad (R^2 = 0.76) \quad (\text{eq. 4})$$

¹ Trabecular density can only be derived from radiographic measures of bone mineral using *in vitro* BMC/BMD-density correlations or *in vivo* calibration phantoms.

are more consistent with the aforementioned *in vitro* studies reporting similar squared relationships between trabecular strength and apparent density or apparent ash density, but like other clinical estimates do not yield the same high correlations found using histologic measures of bone quality.

Significant positive correlations between ultimate strength and BMC have also been found when the vertebral specimens have been tested in flexion-compression, distraction and distraction-extension. Testing of vertebral specimens in flexion-compression did not reveal any unique relations between the amount of bone and strength or stress in comparison to testing under axial compression. In addition, except for osteopenic spines' general fragility, there are no indications that osteopenia increases the spines susceptibility to fail in any special loading direction. Studies which have compared trabecular bone failure patterns in osteopenic specimens to failure patterns in specimens with normal amounts of bone mineral have not identified any unique differences. In contrast to most other fragility fractures, the osteoporotic vertebral fracture is less frequently related to a distinct trauma, like a slip or a fall. More often, trauma causing the vertebral fracture is what at least seemed to be that associated with normal activities, eg. a moderate or light lift when bending forward, a change of position, a sudden twist of the trunk or even coughing.

Non-linear relationships between vertebral strength and density as such as those which have been noted implies that, in already osteopenic vertebrae, a further reduction of the amount of bone would bring about a proportionally greater strength reduction. For example, a 50 percent reduction in bone density (analogous to bone losses associated with chronic bedrest, paralysis and senile osteoporosis) would correspond to a 4-fold decrease in bone strength.

1.4.5 Influence of Vertebral Level on Mechanical and Physical Properties

The compressive mechanical strength (stress at failure) of vertebral trabecular bone specimens does not appear to vary significantly at different levels within the lumbar spine [Keller et al 1989a]. Since the vertebral levels Th12 and L1 have the highest risk for an osteoporotic fracture of any region of the human

spine, one hypothetical explanation for this observation could be that these levels are intrinsically more susceptible to fracture. A more plausible explanation for an increased fracture risk at these two levels may be the fact that the intervertebral joint between Th12 and L1 is the fulcrum for motions between the relatively stiff thoracic cage and the more freely movable lumbar spine. Therefore, stress concentrations, especially in flexion, will predispose these vertebrae to a higher risk of fracture in an osteopenic person.

1.4.6 Rate of Bone Loss in Aging Trabecular and Cortical Bone

Age-related bone losses are purported to occur earliest in trabecular bone. The earliest changes observed in the spine are a decrease in the amount of osseous tissue, a loss of the horizontal trabeculae and a thickening of the vertical trabeculae. Of note is the finding that bone losses are 8-10 fold greater in the trabecular skeleton (2-3% / year) than in the cortical skeleton (0.0-0.5% / year). This has been hypothesized to be related to functional differences in the time course of the remodeling process. Another explanation, which deserves further attention, is the hypothesis that measured differences in the rate of trabecular and cortical bone loss reflects the greater surface/volume ratio of trabecular bone ($S/V \approx 4$) in comparison to whole cortical bone ($S/V \approx 0.4$) [Keller et al 1992a]. Assuming a constant rate of remodeling for both cortical and trabecular bone, a ten-fold greater amount of exposed trabecular bone surface would result in a ten-fold greater loss of trabecular bone in comparison to cortical bone. This observation correlates well with the reported differences in the pattern of bone loss for these tissues. Hence, skeletal structures, such as the vertebral bodies, which have trabecular bone as the primary constituent are particularly susceptible to fracture.

Since trabecular bone is a cellular material, changes in the internal architecture of vertebrae associated with age- and disease-related bone loss will also have a profound effect on vertebral mechanical strength. Thus, accurate assessment of fracture risk in patients must include, in addition to traditional estimates of density, considerations of skeletal architecture. At the present time, however, clinical considerations of skeletal architecture are limited to radiographic « signs » such as loss of horizontal trabeculae and concomitant accentuation of

vertical oriented trabeculae which may ultimately produce a pattern of bioconcavity or « codfish » vertebrae in elderly individuals.

1.5 Vertebral Morphology

Although non-invasive measures of bone density are now considered the most effective method known for predicting fracture risk, these techniques appear to be only about 70% accurate. Presumably, other material features of bone and supporting structures are needed to explain the additional 30% of causes of fracture risk. These additional factors play a greater role in the spine, making bone density less predictable in the spine than in other regions of the skeleton which are comprised of more dense bone. Trabecular bone researchers currently attribute the unexplainable variation in mechanical properties to differences in the morphological features of this tissue.

In vertebrae, large variations in trabecular density and mechanical properties have been noted within adjacent regions separated by only a few millimeters [Keller et al 1989, 1992a]. Five morphologically distinct regions of trabecular bone are found in the vertebral centrum : a superior, 1st transitional, center, 2nd transitional and inferior level (**Figure 2.1**). The superior and inferior sections each occupy approximatively 30-35% of the total segment height and exhibit patterns of orientation distinct from the center and transitional sections. The transitional and middle portions of the centrum consisted primarily of plate-like trabeculae forming a closed cell structure in contrast to the superior and inferior sections of the centrum which consisted primarily of rod-like trabeculae forming an open cell structure. Trabecular bone structure is more dense in the inferior and superior sections than in the central sections of the lumbar centrum. Plate-like trabeculae are associated with the central regions of less dense bone. The central, plate-like regions of the lumbar spine, therefore, appear to be somewhat unique in terms of its trabecular architecture. The functional significance of this finding remains to be determined.

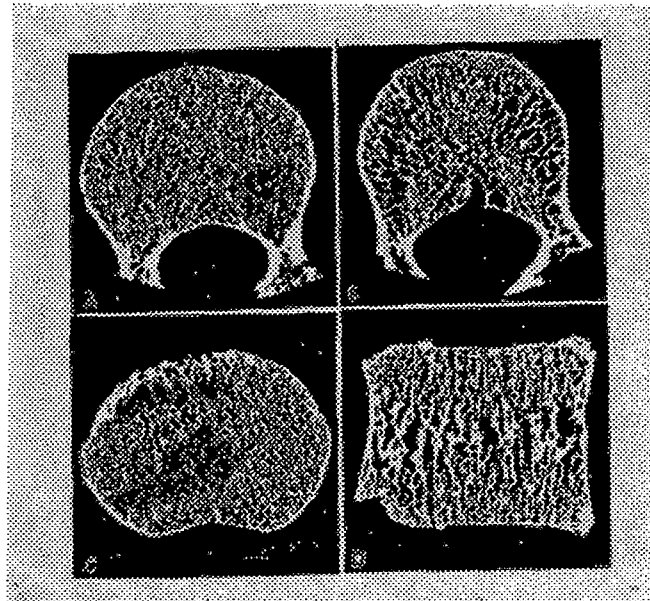


Figure 2.1 Photomicrographs of transverse and sagittal sections of the lumbar vertebral centrum obtained from an 18-year-old male

The complex organization and distribution of vertebral trabeculae and trabeculae in other regions of the skeletal support the generally accepted hypothesis that function directly influences the structure and strength of bone, a relationship known as Wolff's Law [Wolff 1892]. From a mechanical engineering standpoint, trabecular bone behaves similarly to porous engineering materials due to its cellular structure and large energy absorption capabilities. The distribution of trabecular bone density and mechanical properties within vertebrae varies along the axis and within the cross section of vertebrae. Some investigators have reported a variable or heterogenic distribution of trabecular bone tissue physical and mechanical properties for the vertebral centrum [Keller et al 1989a, 1992a, 1993]. Most of these studies have noted that anterior regions of the vertebral centrum are generally less dense and less strong than posterior regions. Keller and associates (1992a) noted that the superior and inferior regions of lumbar vertebrae are denser than the central and transitional regions.

1.5.1 Structural Indices of Trabecular Bone

In anisotropic materials such as bone, the organization of individual material components may be more important than actual amount of material present. Indeed, the dependence of mechanical properties on the structure of trabecular

bone has been noted by numerous investigators, but difficulties in adequately characterizing the two-dimensional and three-dimensional structure of trabecular bone have severely limited the development of predictive models relating structure to mechanical properties. Many anatomical studies have provided general descriptions of trabecular bone architecture, and it is commonly accepted that variations in trabecular architecture can be attributed to the magnitude and distribution of functional stresses or strains. In some skeletal structures such as the vertebral body, trabecular bone is the primary load bearing constituent, and is surrounded by only a thin layer of cortical bone. Some researchers noted that the number and directions of trabeculae within the spine varied with spinal curvature, which they also attributed to adaptations to functional stresses. Others have reported a decrease in vertebral trabecular plate thickness and tissue volume with increasing age. Knowledge of variations in trabecular morphology have important clinical implications in terms of the type and severity of injuries to the spine, and the sites where functional disturbances are potentially most harmful.

One of the first detailed studies of vertebral structure was conducted by Amsutz and Sissons (1969), who created stacked celluloid models from serial transverse histological sections of the third lumbar vertebrae of a young woman. Based upon an analysis of four 6x7x1 mm thick models from the superior, central (neural arch) and inferior sections, they reported an increased density of bone in superior and inferior surfaces of the vertebrae. Trabeculae were preferentially oriented in the vertical and horizontal planes and, at the level of the vertebral arch, exhibited an anterior radial orientation which condensed toward the vertebral arch. Using a scanning electron microscope, Whitehouse, Dyson and Jackson (1971a) described the general three-dimensional arrangement of trabeculae and distribution of trabecular bone within the lumbar vertebral body of a young male. Their analysis of relative bone area, volume and trabecular boundary estimates from twelve areas of a transverse and a sagittal section of the vertebral centrum indicated a relatively high degree of tissue heterogeneity, most notably in the superior and inferior sections near the intervertebral discs. These and other similar studies, while providing quantita-

tive data on the distribution of bone, have been primarily descriptive in terms of the orientation of trabecular bone.

During the past two decades, investigators have used quantitative imaging techniques, based upon stereologic theory to characterize structural anisotropy in trabecular bone. Using histologic sections and high-resolution micro-CT imaging techniques, stereologic analysis can provide planar mathematical indices of skeletal structure, including trabecular number (TM), trabecular width (TW), trabecular orientation (Θ), trabecular anisotropy (MIL_{ratio}), and bone volume fraction (V_f). Each of these structural parameters has been shown, to a certain degree, to predict of the mechanical behavior of trabecular bone, but none have proved to be more effective than apparent density and mineral content. Some of the ambiguities associated with these parameters may be related to the fact that these are two-dimensional descriptions of a complex three-dimensional structure.

In a recent morphological study, Zhu and associates (1994) described the multi-planar structural variations of vertebral trabecular bone. These investigators examined a series of 200+ high resolution (20 micron voxels - volume pixels) serial images of lumbar vertebral trabeculae. The results of this study indicated that the surface planar structural properties of the vertebral specimens could not adequately describe the complex and heterogeneous distribution of bone present ; an accurate description of the « bulk » structural properties of vertebral trabecular bone required multiple serial plane sections distributed at least every 100 microns along a given axis. They also found that the vertebral bone specimens were characterized by smooth and repetitive transitions of plane structural indices (TW, TN, MIL_{ratio} , Θ) along the superior-inferior axis of the specimens, suggesting a high degree of connectivity within the vertebral trabecular bone. In addition, these authors noted that the trabecular bone area centers (BAC) within the vertebral specimens exhibited a helical variation. This helical variation in the BAC was hypothesized to reflect the notion that vertebral trabecular bone has an intrinsic « spring-like » struc-

ture that provides an efficient means to absorb energy and withstand external impact during loading.

Quantitative information from these and other studies have provided insight into the role of structural parameters in mechanical properties, and have been used to characterize changes in skeletal architecture associated with bone remodeling, implant loosening, and pathological changes to the skeletal.

1.5.2 Relationship Between Bone Structural Indices and Mechanical Properties

Researchers often attribute the unexplained variance in bone mechanical properties to bone structure, but few studies have performed detailed analysis of trabecular bone structure in comparison to mechanical properties. Surface planar structural indices have been used, together with traditional measures of apparent density or porosity, in order to improve predictions of the mechanical properties of trabecular bone. Using a high-resolution micro-CT imaging technique, some noted that surface planar structural indices could reliably predict the bulk mechanical properties of trabecular bone as long as the bone specimens were relatively homogeneous in structure. Others noted a close correlations between trabecular structural indices (TN, TW) and calculated strengths of trabecular bone specimens from L1 vertebral bodies were within 20% of values predicted by previous experimental studies for similar density specimens. Others reported that approximatively 93% of the variance in the Young's modulus of human tibial and femoral cancellous bone and non-human tibial and femoral cancellous bone could be explained using mineral volume fraction, apparent density and two structural variables. However, given the fact that the architecture of vertebral trabecular bone is highly heterogenous [Zhu et al. 1994], such high correlations would not be expected for human vertebral trabecular bone. Others have established relationships which explain roughly 60 to 90% of the variance in the elastic modulus trabecular bone using both apparent density and three-dimensional descriptions of trabecular bone « fabric ». Three-dimensional connectivity parameters, such as the Euler number (EN), are hypothesized to be good indicators of osteopenia [Feldkamp et al. 1989]. Additional quantitative structural analyses of trabecular bone is

needed and may provide insight into mechanical behavior not explained by tissue physical properties and gross structure alone.

1.5.3 Simulations of Osteoporosis in Human Vertebrae

With progressive demineralization the vertical trabeculae become thinner and the horizontal cross-linkings become fewer. Since the buckling strength of a slender column is largely dependent on its diameter, length and the distance between cross-linkings (Euler buckling), a loss of the bone in certain regions will reduce the total number of cross-linkings. Consequently, a small drop in bone mineral will at this point produce a proportionally much greater drop in strength.

Simulations of osteoporosis have recently been reported in which planar and multi-planar structural indices were quantified using three-dimensional binary image arrays obtained from high resolution serial histologic sections of lumbar vertebral trabecular bone [Keller et al. 1992b]. In this study, structural measurements of the image arrays were repeated several times following sequential removal of layers of bone pixels from the surfaces of the trabeculae at 20 μm increments (**Figure 2.2**). Here, the sequential removal of bone pixels « simulates » age-related bone loss associated with osteoporosis (assuming a uniform loss of bone in both space and time). After three iterations of the computer simulation of aging (removal of 60 μm), there was a 60% loss in bone volume fraction (V_f) which was accompanied by significant fragmentation of the horizontal trabeculae. In terms of the structural indices computed after each iteration, the connectivity (EN) decreased in direct proportion to V_f whereas the structural indices (TN and TW) decreased at a slightly lower rate than V_f and EN.

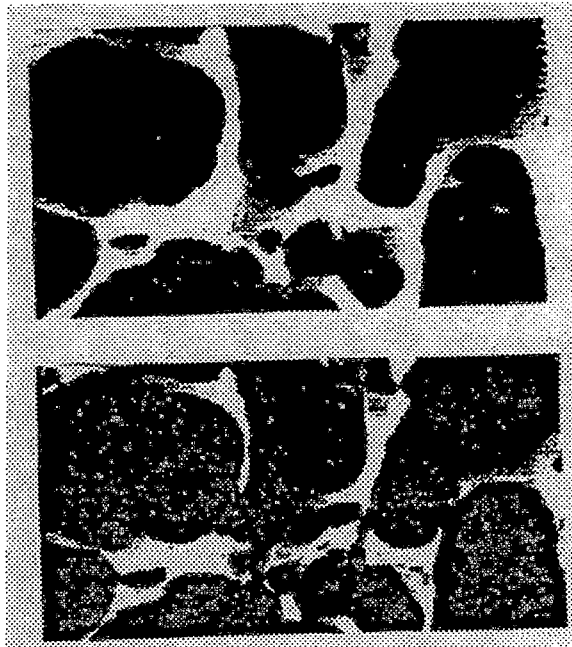


Figure 2.2 Reconstructed, three-dimensional image arrays for a 7.00x1.00x4.02 mm region (x,y,z) of L4 vertebral trabecular bone from a 60 year old male subject

The biomechanical implications of these variations in trabecular structure and relative density (V_f) can be estimated from this analysis. After 50% of the bone has been resorbed (comparable to losses associated with severe osteoporosis or chronic bed rest), there is an estimated 4-fold decrease in bone strength (assuming $S \propto \rho_a^2 \propto V_f^2$). At this point, there is also a concomitant 50% and 30% decrease in EN and TW, respectively (see dashed line in figure). Assuming a constant length for the trabeculae, the decrease in TW is equivalent to an approximately 50% (0.7^2) decrease in the critical elastic buckling load, indicating that there is a markedly greater risk of fracture (eg. two-fold) associated with bone loss than one would predict from density or mineral content changes alone. One would also expect that the trabecular fragmentation seen in Figure 2 would further decrease the ability of the trabecular network to resist loading. This highly simplified analysis does not take into consideration bone stress-adaptive changes which could be used to model local rather than global changes in the remodelling process.

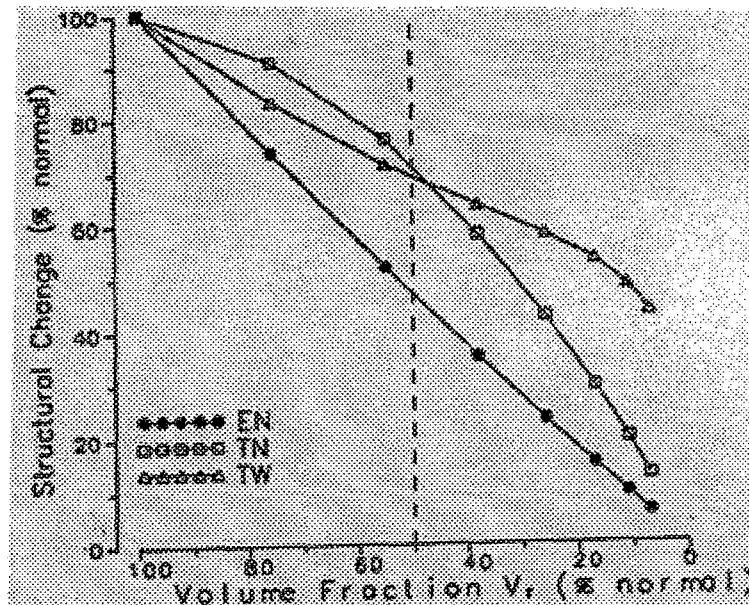


Figure 2.3 Change in vertebral trabecular bone structure as a function of bone volume fraction.

1.6 Biomechanical Adaptation of the spine

A link between degenerative changes in the intervertebral disc and spinal ligaments was originally suggested by Harris and MacNab (1954). More recently Keller et al. (1989a, 1993) noted that there is a similar link between the intervertebral disc and the underlying trabecular bone. This was hypothesized to reflect the fact that the properties of ligaments, discs and bone in the spine are interrelated, and stems from biomechanical considerations of load sharing by these structures.

Consider the following. During aging, degenerative changes in the intervertebral disc (IVD) which reduce the water content of the nucleus also lower the nucleus pressure. Ordinarily, this pressure creates higher compressive stresses in the nucleus region in comparison to peripheral regions of the IVD, and allows the IVD to retain flexibility and resilience during extreme loading conditions. Hence, additional density and strength is needed for trabecular bone underlying high stress regions in the normal disc if the stress-adaptive properties of bone [Wolff's law] are to be observed. This implies that a heterogeneous distribution of physical and mechanical properties in trabecular bone tissue underlying the disc should be present. Indeed, heterogeneous distributions of trabecular structural and physical properties have been described for the

young lumbar vertebral centrum [Keller et al. 1989, 1992, 1993]. In contrast, a more uniform pressure or stress distribution across the IVD and adjacent end-plates and sub-discal bone is predicted for the severely degenerated IVD, in which case the sub-discal variations in mechanical and physical properties should be more homogeneous. This notion has been previously validated using mathematical models. Recent findings by Keller et al. (1993) provide further support for this view. In this study, physical-chemical properties of the disc (FCD) were well correlated to sub-discal bone mechanical properties. Age-related changes in the disc physical-chemical properties also closely paralleled age-related changes in sub-discal bone mechanical properties.

The notion that IVD, spinal ligaments and vertebral bone properties are interdependent has important implications for degenerative processes in the spine as well as the etiology of vertebral osteoporosis. Alterations in disc function which somehow change disc properties will propagate changes in bone properties via disturbances in spine mechanical function, which in turn produces further alterations in disc function. A similar scenario for alterations in bone properties arises, creating a vicious circle of events which facilitate degeneration of the spine (**Figure 2.4**). The consequences of alterations in disc properties, however, are probably more damaging than alterations in bone properties, since the adult IVD lacks a direct blood supply, and hence has limited regenerative capabilities. The amount of bone mineral in the spine determined with the DPA has also been found to be an accurate predictor of tensile properties of the spinal ligaments. This also seems to confirm the fact that the spine is a functional entity in which all structures are interdependent, and respond to remodelling stimuli in a similar manner.

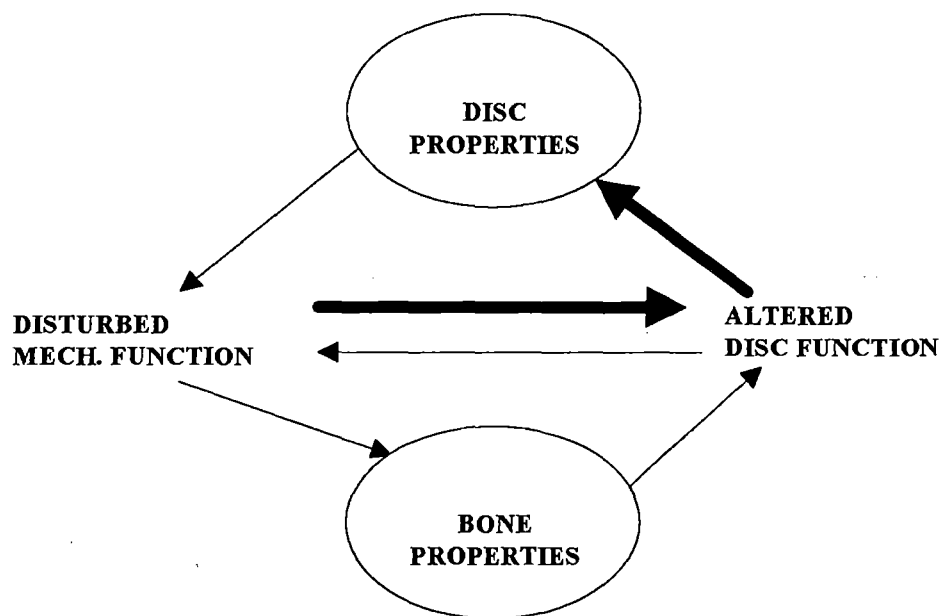


Figure 2.4 Schematic diagram of the biomechanically controlled regulation of intervertebral disc and sub-discal bone properties.

Each of these studies demonstrate how the properties of the vertebral disc and/or spinal ligaments determine the load-sharing conditions between two neighbouring vertebrae, which in turn also control the remodelling of the trabecular bone beneath the disc. Although there is little knowledge of how disc or ligament degeneration is related to the long term development of osteopenia and osteoporosis, one can speculate that the relation between the properties of the disc, ligaments and/or the adjacent bone in some way determines the failure pattern in spinal osteoporosis. Consequently, it is reasonable to assume that discrepancies between the properties of the discs and the vertebral bodies (endplates and the adjacent trabecular bone) may contribute to the formation of the central compression fracture type.

1.7 Failure Patterns in Osteoporosis

Perey (1957) was perhaps the first to report that the weakest part of a compressed thoracolumbar vertebra was the central endplate and the underlying trabecular bone. Numerous studies have confirmed these results, and there is some evidence to which indicates that the amount of bone mineral does not alter this failure pattern. Under combined compression-flexion, quasi-static

loading (< 10 mm/sec) of lumbar vertebrae produces a typical central endplate fracture, together with an anterior wedge type of fracture. Axial compression testing at very high strain rates (200 mm/sec), which are more like the body motions that occur during normal activities, also produced central endplate failures. These higher and but more physiological strain rates, however, nearly doubled the ultimate compressive strength of the spine in comparison to the corresponding values obtained during quasi-static compression conditions.

Central endplate fractures are also often the first sign of failure during *in vitro* repetitive loading experiments conducted on normal and osteopenic human lumbar vertebral motion segments. Some researchers also noted that the type of endplate fracture formed (Schmorl's node or central endplate) were correlated to the age and degree of disc degeneration of the specimens. Schmorl's nodes were predominantly associated with younger, less degenerated discs, whereas central endplate fractures were predominantly associated with older more degenerated specimens (**Figure 2.5**). This study also demonstrated how remarkably close to failure the human spine is when it comes to repetitive loading-compressive stresses as low as 50% of the failure strength of the vertebrae produce fatigue fractures after fewer than 1000 cycles.

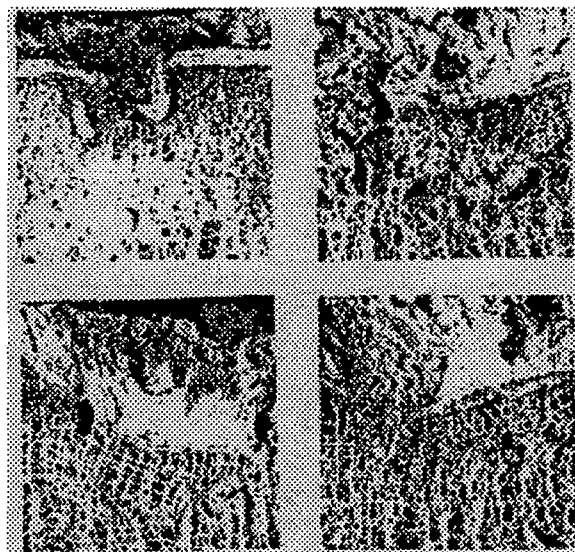


Figure 2.5 Typical vertebral endplate fractures resulting from dynamic axial compressive fatigue loading experiments.

1.8 Radiographic Assessment And Spinal Osteoporosis

Radiographically there are at least three different types of vertebral fractures related to osteoporosis :

1. Osteoporotic compression, collapse or crush fractures which involve the entire vertebral body including the anterior as well as the posterior cortices.
2. Wedge-type fracture in which the height of the posterior aspect of the vertebral body remains relatively intact.
3. Central compression type with a ballooning, concave disc, giving the spine a codfish like appearance. The deformity can occur on one or both side of the same vertebrae. The anterior and posterior cortices are relatively unaffected.

Both central endplates fractures and the anterior wedge type of fractures observed during *in vitro* ultimate strength testing have been difficult to visualize radiographically, even when all the surrounding soft tissues had been excised. Thus, it is likely that a conventional radiographic examination of the spine in an osteoporotic patient may overlook fractures such as those caused by *in vitro* strength testing. Typically fractures, which are readily detectable on an A-P or lateral radiograph, are actually gross fractures involving major wedging, bi-concavity or collapse (crush) of the entire vertebral body. Considerable controversy remains concerning : 1) the precise relationship between vertebral fracture type and the quality of bone, and 2) the precise amount of reduction of the anterior cortex which should be classified as a fracture. Radiologic criteria, such as translucency, sparse trabecular pattern and thickness of the vertebral cortices are difficult to quantify reliably, at least using conventional radiographic techniques. For this reason a lot of interest has been focused on the possibility of finding accurate and reproducible radiographic measures which describe the shape of the human vertebrae.

Currently, digital x-rays, CT and MRI image analysis offer promise in terms of improved definition vertebral structure and/or trabecular structure, and may ultimately improve the chances to detect minor vertebral fractures, such as the central endplate fracture. Quantitative analysis of these images may also assist in the clinical assessment of vertebral fracture risk. With regards to quantita-

tive computed tomography (QCT), Chevalier et al. (1992) used a spatial filter to obtain enhanced two-dimensional CT images of lumbar vertebral trabecular networks. From the resulting digital images they computed a so-called trabecular fragmentation index (TFI), defined as the ratio of the number of network discontinuities and the total length of the network. This index was able to separate normal and osteoporotic women subjects regardless of age, and was significantly correlated ($R=-0.60$) with BMD values obtained for the same subjects. High resolution multiplanar models of trabecular structures derived from micro-CT have also been used to describe the « connectivity » of trabecular bone [Feldkamp et al. 1989]. Reliable radiographic techniques may also improve the ability to assess the effects of various pharmacologic and physical treatment regimens on prevention and treatment of osteoporosis.

2. Overview of the Thesis

In the respect of what we have presented above, the trabecular connectivity index which is computed in our TBMAS (Trabecular Bone and Morphological Analysis System) software may offer a significant improvement in fracture risk assessment.

CHAPTER 3

The Existing Set of Programs

In this chapter, we describe the internals of the existing set of programs. We first address the functionalities offered by these programs, then we discuss the functionalities that we have implemented in our program and, at last, we present some technical considerations.

1. Functionalities	2
1.1 Existing Set of Programs Functionalities	2
1.2 The TBMAS Software Functionalities	3
2. Technical aspect	8
2.1 Software setup and display board	8
2.2 Drive setup and files	9

1. Functionalities

In this section we first present briefly the functionalities that were offered by the existing set of programs. Then we present with more details the main functionalities that we have implemented in our program. Other functionalities such as bitmap pictures creation and compression of binary files are not discussed in this chapter as they are detailed in chapter 4.

1.1 Existing Set of Programs Functionalities

We discuss below the **functionalities** that are offered by the **package of programs** used in the biomechanics lab of the University of Vermont. This package consists of a program set containing some ten of **executables**, some are simple **batch** programs, others have a simple **text interface**. The program interface, when existing, is composed of a text menu containing different options one can choose from. This is one of the most problematic aspects of the existing programs and one of the **motivations** to create a new program (see chapter 4).

We show in **Table 3.1** a summary of these functionalities along with their inputs and outputs for each programs composing the package introduced above.

Program	Display	Threshold	Enter MHL	Osteo	Input	Output
Binread	*	*	*	-	-	zip Targa Binary file
Bincheck	*	-	-	-	-	Binary file ASCII report
Bineuler	*	-	*	-	-	Binary file Microstat file
Binosteo	*	-	-	-	*	Binary file Binary files
Binmorph	*	-	-	*	-	Binary file Microstat file

Binoseul	*	-	*	-	*	Binary files	Microstat files Binary files
Bindisp	*	-	-	-	-	Binary file	Targa
Binshow	*	-	-	-	-	Targa zip Targa	Targa zip Targa
Vistamil	*	*	-	*	-	Targa	Microstat file Targa
Batchmil	*	*	-	*	-	Targa	Microstat file Targa

Table 3.1 Existing set of programs functionalities

1.2 The TBMAS Software Functionalities

We now present the main functionalities implemented in our program. We detail in this chapter the most important ones (see chapter 4 for additional functionalities covering).

1.2.1 The MIL Analysis

We now discuss the MIL analysis. We first give a short theoretical introduction which explains the MIL analysis goal in this context, then we explain the algorithm used.

1.2.1.1 Theoretical Introduction to the MIL Analysis

A structure is **isotropic** if it has **no preferred orientation** or, more rigorously, if the perpendicular to any element of surface has an equal probability of lying within any element of solid angle. That **cancellous bone is anisotropic** is evident from simple inspection, but a **numerical estimate** of the degree of anisotropy is **crucial** to understanding the biomechanical properties of bone

and is also important for optimizing sampling strategies for bone histomorphometry.

As described in **Appendix A**, departure from isotropy is expressed as a polar diagram of mean intercept length (MIL) and angle of orientation. Such a diagram consists of an elliptical plot in which the ratio of the major and minor axis indicates the degree of anisotropy and the angle between the axes indicates the overall orientation relative to an arbitrarily chosen reference direction. As perfect isotropy is approached, the ellipse becomes a circle. An illustration of this process is shown below in **Figure 3.1**.

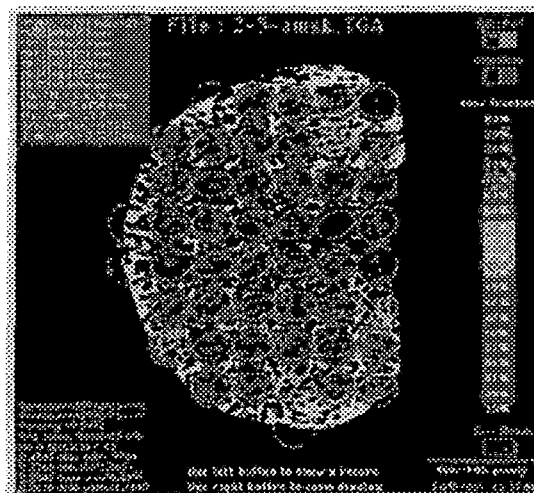


Figure 3.1 MIL computation of a lumbar centrum

1.2.1.2 The MIL Computation Algorithm

The method and the resulting algorithm that we address below were originally implemented in the existing set of programs. We have extracted corresponding C code thanks to a reverse engineering approach (see chapter 5).

The MIL computation algorithm :

- Get center coordinates of test areas (described below)
- Each test area is a circle with a radius about the center, keep only the circles where the specimen's percentage is at least 75% (= 75% of bone)

- For each of the test area do :
 - Rotate the current test area at different angle of rotation (0-180)
 - For each rotation do :
 - Smoother the area
 - Get the number of intercepts, that is to say the number of changes between bone and marrow
 - Compute and store :
 - $mil = (2.0 * bf * length) / intercepts$
 - $tw = tw + (intercepts / length)$
- where :
- bf = bone fraction of test area
 tw = trabecular width
 length = the area of the test area
 intercepts = the number of intercepts
- compute tw of all the test areas : $tw = (2 * bf) / ((pi * tw) / (2 * nr))$
- where :
- nr = the number of rotation display
 pi = 3.1415

Getting test areas :

- Find the specimen centroid coordinates (described below) : (xctr,yctr)
- Find the xmin and ymin from the centroid :
 - $xmin = xctr - radius$
 - $ymin = yctr - radius$
 - while (xmin > diameter) xmin = xmin - diameter
 - while (ymin > diameter) ymin = ymin - diameter
- Save the coordinates of each area

Finding specimen centroid :

- Compute qx and qy :
 - $qx = \sum x$ where x is the x-axis coordinate of a bone pixel
 - $qy = \sum y$ where y is the y-axis coordinate of a bone pixel
 - Compute xctr and yctr :
 - $xctr = (qx / area)$
 - $yctr = (qy / area)$
- where area equal the area of the specimen, that is to say the number of bone pixels

1.2.2 The Euler Number

We now discuss the Euler number, as for the MIL analysis, we first give a short theoretical introduction to the Euler number, then we discuss its algorithm.

1.2.2.1 Theoretical Introduction to the Euler number

The **Euler number** is a **topological quantity** and as such carries no information regarding the efficacy of the positions of connections, the size of connections, or the strength of the material comprising the connections. Hence, by itself, it cannot be expected to be an index of mechanical performance. On the other hand, it appears to be **promising for distinguishing** between **structures** that are equivalent according to more primary measures, such as total bone mass.

For an open network structure, the Euler number may be calculated from the number of nodes n and the number of branches b :

$$N^{(3)} = n - b$$

In the case of network in which at least one path exists between any two nodes, the quantity $1 - N^{(3)}$ can be interpreted as the **maximum number of branches that could be removed without breaking the network into parts**. A **highly connected structure**, such as **healthy cancellous bone**, has a **negative Euler number** of several thousands in a volume of 1 cm^3 . Severely osteoporotic bone has a markedly smaller value of $1 - N^{(3)}$. On a detailed level, the breaking of a single connection increases the Euler number by 1, but the addition of a connection decreases it by 1.

1.2.2.2 The Euler Number Computation Algorithm

This algorithm, just as the precedent one, was originally implemented in the existing set of program and has been made available for us thanks to reverse engineering.

Euler number computation :

- While scanning specimen's pixels from left to right and from top to bottom:
 - Compute the area of the specimen ($area_h$)
 - Compute the number of intercepts (same as described for the MIL algorithm) : $intercepts_h$
 - Compute the number of bone pixels (bn)
- While scanning specimen's pixels from top to bottom and from left to right:
 - Compute the area of the specimen ($area_v$)
 - Compute the number of intercepts ($intercepts_v$)
- Compute the total area : $area = area_v + area_h$
- Compute the total intercepts : $intercepts = intercepts_v + intercepts_h$
- Compute the Euler number : $en = -1.0 * bn$
- Compute the area fraction : $af = bn / area_h$
- Compute the trabecular number : $tn = intercepts / area$
- Compute the trabecular width : $tw = af / tn$

1.2.3 The Osteoporosis Simulation

We do not introduce here the concept of the osteoporosis anymore as its problematic has been largely addressed in chapter 2. Hence we just present below the basic principle of the simulation which, indeed, is relatively simple.

Simulating **osteoporosis** means **reducing** the **width** of bone structures. To perform this task, edge pixels of bone structures are removed. An **edge bone pixel** is a pixel whose neighbors are not all bone pixels. If we want to perform a one-step simulation, then we remove the edges one time, if we want a n -step simulation, then we remove the edges n times.

2. Technical aspect

We now describe the hardware and software specifications of the existing set of programs. We have based our text on [Moeljanto91].

2.1 Software setup and display board

The existing set of programs was created by using the **Microsoft C compiler 6.0** and **MiPS¹ library routines**. Hence it **requires** to be run in the MiPS directory so that the file *COMMON.MIP*² and the display driver files can be found by the program.

One of **MiPS main characteristics** is that it needs to use a display device capable of using two different monitors. One is called the **text screen** and is used to display textual output such as menu items, the other, called the **image screen**, is used to display images or pictures. That system is represented below in **Figure 3.2**. Due to that characteristic, the program can only be used on a computer with a proper display board. This is a **serious limitation** and, as we say in chapter 4, one of the main reasons motivating the creation of a new program.

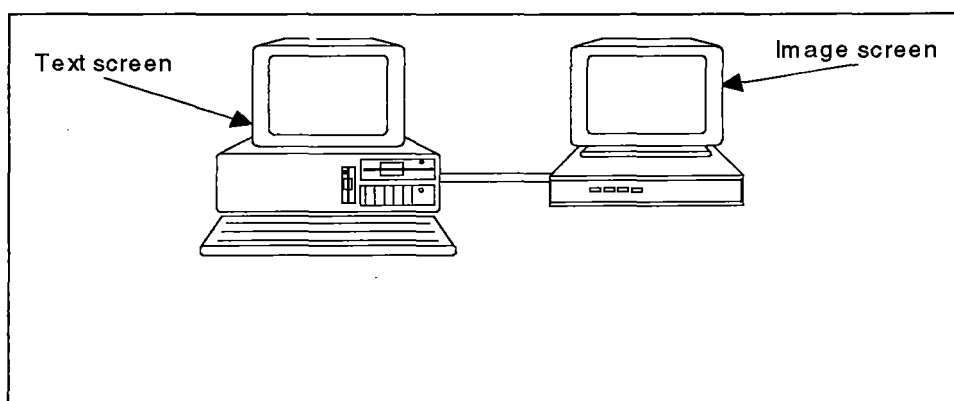


Figure 3.2 Schematic representation of the two monitors MiPS hardware configuration

¹ Maps and Image Processing System by Microimage, inc.

² MiPS reference file

The display board used at the biomechanics lab of the University of Vermont is the **vista board**. We present, to conclude this point, its characteristics and the MiPS setup associated in **Table 3.2**.

	Vista Board and MiPS setup
Video memory	4 MB
MiPS specs	8, 16, 24 bit display mode
MiPS setup	display mode : 8, 16, 24 Interrupt : 61, 63, 64 Software bit mode : 0, 1, 2

Table 3.2 The Vista board and the MiPS setup^[Moeljanto91]

2.2 Drive setup and files

In this point we describe the drive setup, which consists indeed of a RAM drive setup needed by the existing set of programs and the different input and output files it uses or generates.

2.2.1 Drive setup

The programs use a **RAM drive**, the size needed may vary depending on how large the input file is, to perform the desired computations. In any case the RAM drive size needed is at least 2 MB and this is **another limitation** of these programs. Indeed, using RAM drive instead of memory **decreases** dramatically the level of **performances** obtained. We discuss how we have solved this problem and what improvements we have obtained in chapter 4 when introducing how we have managed memory in our program.

2.2.2 Input files

The programs use 3 types of input files. The first is the **binary file** type, its structure is shown below in **Table 3.3** and it consists in a three dimensional

representation of a series of serial section images³. Various analyses can be conducted on that type of file, especially Euler and MIL analyses which were discussed in the previous section.

The second type is the **global variable file** whose structure is presented below in **Table 3.4**. This file is **an essential complement of the binary file** presented above. Each binary file should have its corresponding global variable file as it contains needful information which is necessary to make proper use of it.

	Binary file structure
Microsoft C format	struct S {int x, y, z}
Meaning	the file basically consists of 3 integer number which correspond to the coordinate of a one bone pixel in a three dimensional space (x,y,z)

Table 3.3 The structure of a binary file^[Moeljanto91]

	Global variable file structure
Format	[xmin], [xmax] <CR> [ymin], [ymax] <CR> [zmin], [zmax] <CR> [xy-calibration], [z-calibration] <CR>
Meaning	[xmin], [xmax], [ymin], [ymax], [zmin], [zmax] : integer values representing the dimensions of the image coded inside the binary file [xy-calibration] : a real/float value representing the calibration across the image (in mm/pixels) [z-calibration] : a real/float representing the calibration along the z axis or the cut (in mm/pixels)

Table 3.4 The structure of the global variable file^[Moeljanto91]

³ The creation process of the binary files is addressed in Appendix B

The third input file type is the **Targa image file**⁴. The Targa format is « used widely in paint, graphics and imaging application that require the storage of image data containing up to 32 bits per pixel »^[Murray&vanRyper94]. Thus it is clear that the choice of this format was obvious ; even more, the **MiPS library routines** contain several procedures which make it **easy to manipulate** such pictures. Nevertheless, we have decided to adopt another image file format for our program, the **Microsoft Windows Bitmap format**. The justification of this choice is quite simple ; indeed, as our program is aimed at being used with **the Microsoft Windows interface**, it was then wiser to use an image file format which **fits well** with that configuration. Hence the choice of the **BMP** image file format which is described in chapter 4.

2.2.3 Output files

The existing set of programs generates a certain number of files. Two different **parameter files**, one used for morphological computation of binary files and another which is used for Targa files. Those two are not related, thus not interchangeable.

The two **parameter files** are aimed at **saving information** on the current calculation, including the name of the input file and the step reached into the analysis process. Thus, being retrieved, it serves to display information on the already executed computations and to take back the ongoing analysis. This is a feature we have **not included in our program** because it was **not useful** anymore. Indeed, as we have **increased** dramatically the level of **performances** obtained with the existing set of programs, an analysis should never last more than a few hours at the most, instead of a few days. Hence, it can be conducted in one step and does not need to be divided into several stages.

Another important type of output files, is the **Microstat format text files** which contains the results of the analysis performed. A problem with those files was the **absence** of the DOS *EOF* (end of file) character which is due to the use of the Microsoft C compiler 6.0 and which had to be corrected by

⁴ Targa is a registered mark of Truevision Inc., Indianapolis, In, USA

using the *edlin* DOS shell command. This can be regarded as quite benign but it is not because **old files** were susceptible to be **appended** with new data, an operation which is **impossible** if the file does not contain the *eof* character. Several more applications would not recognize properly the file because of the same problem.

In our program we have decided to **keep** the same **Microstat file format** as a facility for Mr. Keller who is used to working with the Microstat program to produce statistics and graphics. Nevertheless we have corrected the bug discussed above.

A last sort of output files that is produced is the **binary files** themselves. Indeed, performing an **osteoporosis simulation**, as discussed in the first section of this chapter, generates **a new set of binary files**. Each of these being compatible with the format addressed in this section and thus, being susceptible to serve as **input for new analysis** to be performed. That is a characteristic that we have, of course, left unchanged in our program.

CHAPTER 4

Development Process

In this chapter, we introduce the development process we have used. We first discuss the key points we have been asked to fulfill and the notations defined in [Mylopoulos95] that we have used to describe the requirements. Then we presents the functional specifications and implementation analysis of our program. Finally, we conclude with a conclusion on the development process discussed.

1. Requirements Analysis	2
1.1 Requirements Model	2
1.2 The Different Types of Requirements	3
2. Functional Specifications	18
2.1 Perform MIL Analysis Functional Specifications	18
2.2 Compute Euler Number Functional Specifications	26
2.3 Perform Osteoporosis Simulation Functional Specifications	27
3. Implementation Analysis	29
3.1 The Software Architecture	29
3.2 Specific Programming Techniques and Algorithms	38
3.3 Relations of the TBMAS Program with External Applications	60
3.4 Performances Measurements	60
3.5 An Empirical Reverse Engineering Method	63
4. Conclusion	71

1. Requirements Analysis

1.1 Requirements Model

We can express the definition of requirements as « ... *a careful assessment of the needs that a system is to fulfill ... must say **why** a system is needed, based on current and foreseen conditions, which may be internal operations or an external market ... must say **what** system features will serve and satisfy this context ... must also say **how** the system is to be constructed ... »^[Mylopoulos95].*

Upon this definition we can split requirements in three different categories : the **context analysis**, the **functional requirements** and the **non-functional requirements**.

- **Context analysis**¹ can be defined as « *the reasons **why** the system is to be created and why certain technical, operational and economic feasibility are the criteria that form boundary conditions for the system* »^[Mylopoulos95].
- **Functional requirements** can be defined as « *a description of **what** the system is to do* »^[Mylopoulos95].
- **Non-functional requirements** can be defined as « *global constraints on how the system is to be constructed and function* »^[Mylopoulos95].

Below² we express the requirements we have been asked to fulfill using the categories explained above.

We think that splitting requirements is important because it helps understanding the scope of each type of requirements and its impact on implementation techniques. Further, getting the requirements right is a problem of crucial importance. Nevertheless, « *finding and fixing a fault after software delivery is*

¹ Context analysis even though it is a new concept can be assimilated to suitability investigation which is a concept known and used for a long time

² See point 1.2

100 times more expensive than finding and fixing it during requirements or early design phases »^[Mylopoulos95]. This makes very clear the importance of an accurate requirements definition.

Another question of general interest that we address here briefly, is how can requirements be made use of ? Four different usages can be obtained : **problem statement**, **contract**, **documentation** and **tool** « *to support design validation* »^[Mylopoulos95]. Such a tool could be a **consistency checker** that proves that invariant are preserved by activities assuming that « *a requirement model consists of state invariants, ..., and conditions defined by (pre,post) conditions* »^[Mylopoulos95]. For our program, the most obvious and important usages were **problem statement** and **documentation** in the sense of support for « ... *communication between designer, customer and end-users* »^[Mylopoulos95]. We do not deepen the subject as the relations between us, the designers, and Mr. Keller, the customer, are obvious.

A last challenge is to deal with **non-functional requirements** which can be defined as « ... *global constraints on a software system, such as development costs, operational costs, performances, reliability, maintainability, portability, robustness etc.* »^[Mylopoulos95]. These should not be confused with functional requirements which impose constraints on the function of a system. That is why we insist on this aspect below³ as performance achievement was of paramount important to us.

1.2 The Different Types of Requirements

In this point we describe all the requirements that we had to fulfill using the different categories described above. Then we specify on the design and implementation decisions we made to satisfy the requirements.

1.2.1 Context Analysis

As presented in chapter 3, the original set of programs could only be executed on a particular machine. That machine, represented in **Figure 3.1**, has two

³ See point 1.2.3

monitors and a specific display board, the vista board (Truevision, Doraville, IN). Further, the set of programs requires the MiPS (Microimage, Lincoln, Nebraska) library routines to be present in the same directory.

It is very important to be aware of this utilization context because it is the main reason **why** a new program was needed. Indeed, as we say in chapter 1, our program is aimed to serve as both a research and an educational training system for students. Therefore it is obvious that our program could not depend on the **specific configuration** described above. This then was perhaps the most important task we had to address. Thus, our main goal has been to develop a program which would not be tied to any specific configuration and, even though it has been **specifically developed for the Microsoft Windows platform**, could be migrated more easily, thanks to our **software architecture** which is described in section 3.1, to an **other platform**.

To succeed in making our program as independent from a specific configuration as possible, we had to change the way in which the **computations** were executed, this is largely detailed later on in this chapter.

Another decision was to separate the user interface from the computational tool, the user interface is more platform dependent and separating both aspects led to a **software architecture**⁴ which makes a possible platform migration easier. In this way, the choice of the Watcom C++ compiler (Waterloo, Ontario) was very important because it covers a large panel of current platforms. We refer the reader to point 1.2.3 in this chapter for more details.

A last point is the extensive use of **object oriented programming** which makes maintenance and revisions much easier. Nevertheless, **generic objects** can remain almost unchanged, e.g. if we change the platform, and the more specific aspects are kept in a few **specific objects** which are the only ones to undergo modifications.

⁴ The reader may refer to section 3.1 for an in depth look at the architecture of the program

In conclusion for this type of requirement we present in table 4.1 the configuration needed for a proper use of our program.

1.2.2 Functional Requirements

We split the functional requirements in two main aspects : the **functionalities** aspect and the **user friendly interface** aspect. We deal with both aspects below.

Hardware requirements	a PC or compatible, with 386DX or higher microprocessor & VGA+256-color display
Memory requirements	4 MB of memory and at least 5 MB of available hard disk space (depending on the size of the binary files to use with the program)
Operating system requirements	MS-DOS® OS version 3.1 or later and Microsoft Windows™ 3.1 or later
Pointing device requirements	Microsoft Mouse or compatible pointing device

Table 4.1 Hardware and software configuration needed for a proper use of the TBMAS program

1.2.2.1 Functionalities

We first had to duplicate some analytic functionalities present in the original set of morphology programs. This includes the **MIL** analysis, the **Euler** analysis and the **osteoporosis simulation** which are explained in chapter 3. Note that certain aspects of these analyses were improved to enhance performance. An important feature of the program was a visualization tool to display **three dimensional representations** of a trabecular bone sections. To fulfill this we have chosen to adopt the Microsoft BMP image file format instead of the Targa image file format because it was more appropriate to use under the Microsoft Windows™ graphical interface⁵.

⁵ The justification of this choice is explained in section 3.2

Further more we had to add some new functionalities such as the possibility to **graphically represent the analysis results**, as well as **three dimensional modeling** of trabecular bone. All this is described in more detail in following sections.

1.2.2.2 User Friendly Interface

One of the most critical aspects of the contract was the creation of a **user friendly interface** which was mostly lacking in the original set of programs (see chapter 3).

We will begin with a general **definition of an interface**, the distinction between **text oriented interface** and **graphics oriented interface** and finally the improvements obtained by the use of the **Microsoft Windows™** graphical interface.

1.2.2.2.1 Interface Definition

An interface can usually be defined using two different approaches. First, we can define an interface as a **program layer** standing between the user and the machine^[Tomsdorf&al92]. Secondly, it can be defined as a **computer system** used by a person to fulfill a task^[Bodart89]. We have tried, as explained below, to take into accounts both aspect both aspects in the preparation of our program.

As we have chosen to separate the user interface from the computational part of the program, the user interface can be viewed as a **layer** which comes on top of the analytic and modelization routines. More, it serves as an **intermediate** between the user and the machine, that is to say the rest of the program which interact itself with the computer via the operating system. So our **software architecture** takes into accounts the first definition we have given above of an interface.

Further more, we have tried to take into counts as much as possible the **human aspect** of the interface, thus respecting the second interface definition. To guarantee this, the interface requirements were developed in close **collaboration** with the user (Mr. Keller) and underwent several revisions. We have

also taken care to use **ergonomic recommendations** found in [Sacre&al92] and [Vanderdonckt92]. Ergonomic considerations and the interaction with the user at a design level should help **solve** typical interface related problems such as : **performances fall, long training period** and **under utilization**. These are typical limitations that user interface ergonomics tries to deal with^[Bodart89]. The graphic in figure 4.1 which is called the **B. Senach diagram**^[Bodart89] explains goals pursued by ergonomic considerations which consists of minimizing the points a, b, c and d on the graph and maximizing the point e.

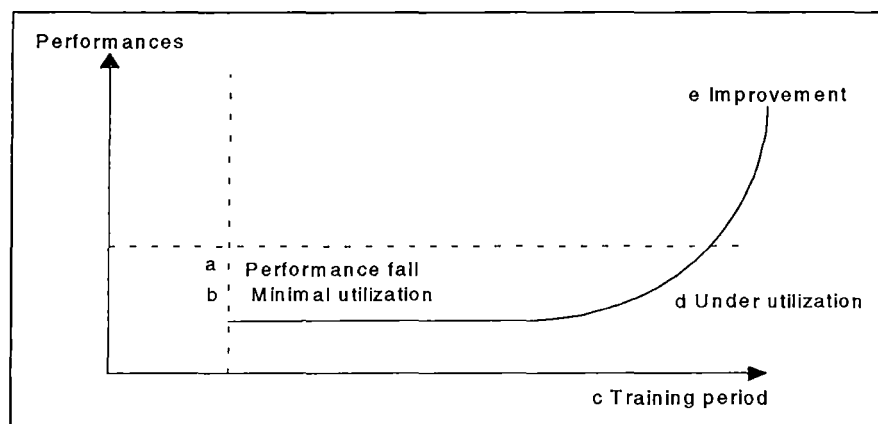


Figure 4.1 B.Senach diagram^[Bodart89]

1.2.2.2.2 Text Oriented Interface Vs Graphics Oriented Interface

Text oriented interface can be defined as an interface where program commands are accessible by some **lines of text**^[Tomsdorf&al92]. Typical examples of such an interface is the well known DOS prompt. The program modules executed in batch mode (see chapter 3) belong to the same category. For such programs, commands plus a certain number of parameters are needed to obtain the desired result.

More recent improvements to the command interface include **pull-down menus** which presents available options to the user (e.g. Lotus 1-2-3). The modules associated with MiPS menus structure (see chapter 3) belong to this early evolution of the text oriented interface. Base upon the menu functionalities, one can begin to determine the degree of user friendliness of an inter-

face^[Tomsdorf&al92]. In this regard, the MiPS menu interface can be considered to be poorly convivial because most options are presented on the same menu level with little separation or logical order. In the MiPS case, almost no ergonomic considerations have been taken into account for the realization of this particular program. That is one particular weakness we have tried to address by using ergonomic recommendations stated previously.

A graphics oriented interface can be defined as an interface using **symbols** (icons) and other **graphical elements** to represent the different commands and program functions^[Tomsdorf&al92]. The use of such symbols coupled with a new peripheral input device : the **mouse**, helps new users to deal with an application functionalities more easily. However, a new kind of problem may confront the user : namely the presence of numerous different program interfaces all pretending to be the most convivial. We discuss this problem below while justifying our choice of the Microsoft Windows™ graphical interface.

1.2.2.2.3 A Global Concept : Windows

As noted previously, a common problem for the user is the **lack of uniformity** among programs interfaces. Thus, even someone who's not new to the use of a program may have some difficulty when confronted with procedures for selecting commands. This can become a real headache if one has to work with programs created by different editors which require significant adjustments to their fashions of utilization. This is one reason why we have chosen to base our program on the Microsoft Windows™ interface.

Windows can be regarded as a **concept** which defines the utilization of an **interface** in terms of permanent elements, but also the **integration of programs** concerning various tasks^[Tomsdorf&al92]. Applications created for Windows have a **style guide** similar to Windows and **share** tools provided by the interface such as screen, mouse and printer management. This is called the **SAA concept** (System-Application-Architecture) where Windows serves as an interface to **establish links** between applications, the operating-system and the hard-

ware configuration along with being an interface between the user and the machine, that is to say a **human-machine interface**^[Tomsdorf&al92].

In view of the above considerations, the Windows interface was the **logical choice** to base our program on as it gathers both aspects that were important for us : an easiness to implement **ergonomic** considerations and a standard structure widely considered as the market **standard**. The first aspect is provided by the SAA concept itself which concerns programming by the use of libraries which make it easier to obtain high level interface requirements. The second aspect makes it easier for the user to take full advantage of the application by significantly reducing the training period required to master applications standard commands since they are already familiar.

1.2.3 Non-Functional Requirements

In this point, we deal with two main types of non-functional requirements : the **level of performances** and the use of **high level programming tools and languages**.

1.2.3.1 Performances

Our main goal here was to significantly **increase** the performance level provided by the existing set of programs (see chapter 3). This was maybe a more secondary aspect as a good user interface and a less specific configuration were our two main objectives. However, we exceeded our expectations and increased the performance level dramatically. The reader can learn more about this in section 3.9.

To obtain such results, we used **state of the art programming techniques** such as **extensive use of memory** in dynamically allocated two-dimensional memory arrays and a **unique file compression algorithm**⁶ together with other techniques like object oriented programming and dynamically linked libraries (DLL).

⁶ The memory arrays and the file compression algorithm are discussed in section 3.2

It is not the aim of this chapter to address technical details about the implementation techniques we have developed since all these aspects are detailed in section 3.2. Nevertheless, we present here a concrete disk memory saving result we have achieved as an illustration of performance improvement. **Table 4.2** compares gains in file size obtained by different compression techniques. Our compression method is an adapted RLE encoding technique⁷ coupled with an index, it has the advantage to be simple to implement, fast and, as shown in table 4.2, it can be combined with other compression methods, such as PKZIP®, to obtain dramatic size reduction.

Indeed, the binary file Z60IM compressed using our technique and then composed again using PKZIP® can be stored on a double density disk although the original file size was almost 8 mega bytes, as shown in table 4.2.

Type of file	Size (in bytes)	Gain (in percent)
Original binary file	7.783.284	0
Original file zipped	2.725.227	65
Original file LRLE compressed	835.773	89
Original file LRLE compressed & zip-ped	253.057	97

Table 4.2 Comparison of gains in size obtained for the file Z60IM.BIN

1.2.3.2 Programming Tools and Languages

In this point we describe the two main languages and the associated tools we were asked to use for programming our application.

1.2.3.2.1 The Watcom C/C++ Compiler 10.0

The Watcom company is perhaps not as popular as Borland or Microsoft, but its C/C++ compiler 10.0 has many advantages over its competitors.

First of all, it is widely recognized as being the best compiler for **code optimization** available in its price range.

⁷ The method we have called Logical & Run-Length Encoding is detailed in section 3.2

Second, Watcom has a really **attractive price tag**, especially for students and universities, and is a key point in its phenomenal success. Thirdly, it provides an **integrated development environment** for Windows which incorporates many interesting features inside of convivial user interface as shown in figure 4.2. This development environment is **multi-target**, which permits a project to include EXEs and DLLs. The IDE is hosted under OS/2 2.x, Windows 3.x, and Windows NT.

An other important feature of the Watcom compiler is its **multi-platform** capability. Watcom IDE makes it easy to edit, compile, link, debug and build applications for 16-bit systems like DOS, OS/2 1.x, and Windows 3.x and 32-bit systems like extended DOS, Novell NLMS, OS/2 2.x, Windows 3.x, Win32s, and Windows NT. This is important in terms of considerations we made in the context analysis (see point 1.2.1). More, the core tools in the package permit **cross-platform development** and allows developers to exploit the advanced features of today's popular 32-bit operating systems, including OS/2 2.x and Windows NT. Cross-platform support allows one to develop programs on a host development environment which can be executed on a different target system.

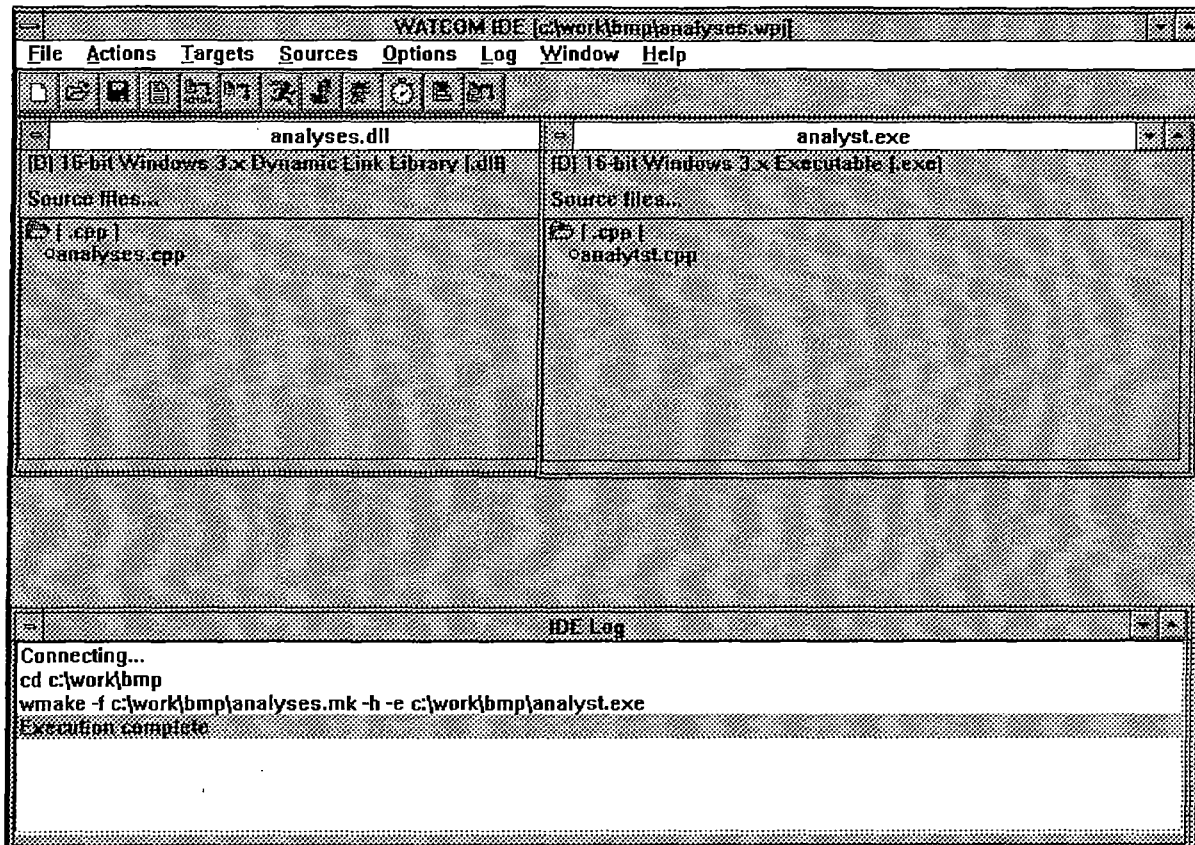


Figure 4.2 The Watcom Windows IDE

Finally, the Watcom compiler comes with a set of **powerful tools** such as a multi-platform debugger, a class browser which lets you visually navigate the object hierarchies, functions, variable types, and constants of C/C++ applications, text code and resource editors, a heap walker which displays memory usage for testing and debugging purposes, and a spy which monitors messages passed between applications and Windows, etc. ...

Table 4.3 presents the range of Intel x86 host and target platforms covered by the Watcom compiler.

	Host platforms	16-bit target platforms	32-bit target platforms
AutoCAD ADS	-	-	x
DOS	command line	x	using extended DOS
Novell NLMs	-	-	x

OS/2 2.x	IDE and command line	x	x
Win32s	IDE	-	x
Windows 3.x	IDE	x	using Watcom 32-bit extender technology
Windows NT	IDE and command line	-	x

Table 4.3 Host and target platforms of the Watcom compiler

1.2.3.2.2 Microsoft Visual Basic 3.0 Professional Edition

In contrast to other languages such as C or FORTRAN, the Basic language is not strictly standardized. This fact explained the appearance of different dialects years after year. Visual Basic for Windows is distinguishable from its predecessors by its ability to deal with objects.

Visual Basic is truly an evolutionary language. It combines procedural and structural programming with two innovative techniques : **object-oriented** and **event-driven programming**. In most fundamental ways, Visual Basic is similar to popular languages such as Basic, C, and Pascal. The structure of the Visual Basic language (loops, data types, procedures, operators, input/output, and so on), however, differs somewhat from language implementations such as QuickBasic, Turbo Pascal, and Turbo C in syntax, but differs primarily in the details. For example, a data type to hold more than one kind of data is called a *record* in Pascal, a *struct* in C, and a *type* in Basic. Any programmers versed in these popular languages will have little trouble understanding Visual Basic. The largest conceptual leap from these languages to Visual Basic is the leap to Windows (not to the language itself) and to a graphical conceptualization of application development rather than a code-oriented conceptualization.

In Visual Basic, for example, you create a default application window (called a **form**), and you select a control from the Visual Basic toolbox and position it

on a form without writing any code. In Turbo Pascal or C++ you must create the windows and the control and position both with code. On the screen in Visual Basic, you drag windows (forms) and controls with the mouse to size and position them.

« Visual Basic is an evolutionary language (utilizing both event- and object-oriented programming techniques) and an evolutionary programming system. It is much easier to create Visual Basic applications because less time is spent coding, and more time spent focusing on the problem you want to solve, not on coding the user interface. The ability to create sophisticated applications has never been so available to such a broad group of people. Granted, Visual Basic can't turn every user into a programmer, but it can greatly simplify programming. » [Entsminger92].

Visual Basic's **visual** and **event-oriented** approach are the keys to simplifying Windows development. In a nutshell, you (the programmer) can develop applications graphically. You point and click, selecting **objects**, **controls**, **properties**, and so on from menus as illustrated below in **Figure 4.3**.

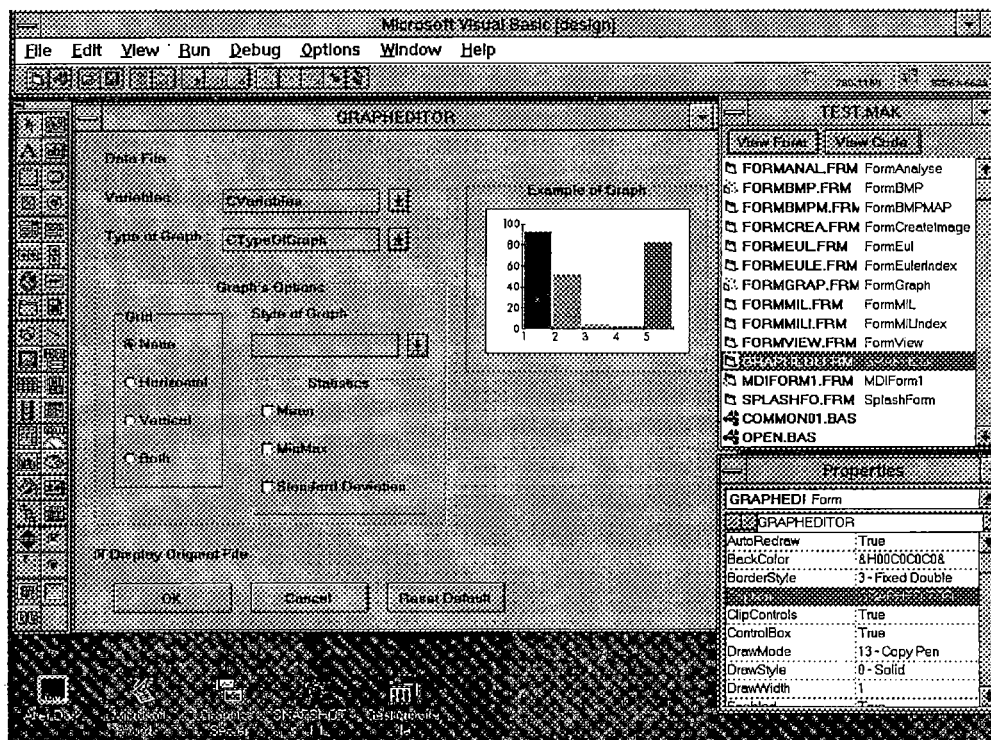


Figure 4.3 The Visual Basic Windows IDE

Programming the interface in Visual Basic is much closer to **user** level than the traditional programming level. You must, of course, add code to the windows (called **forms** in Visual Basic) that you create by pointing to and clicking menu items, but even adding code is simplified by **templates**, **automatic indentation**, **super-fast syntax checking**, and so on, as shown in **Figure 4.4**.

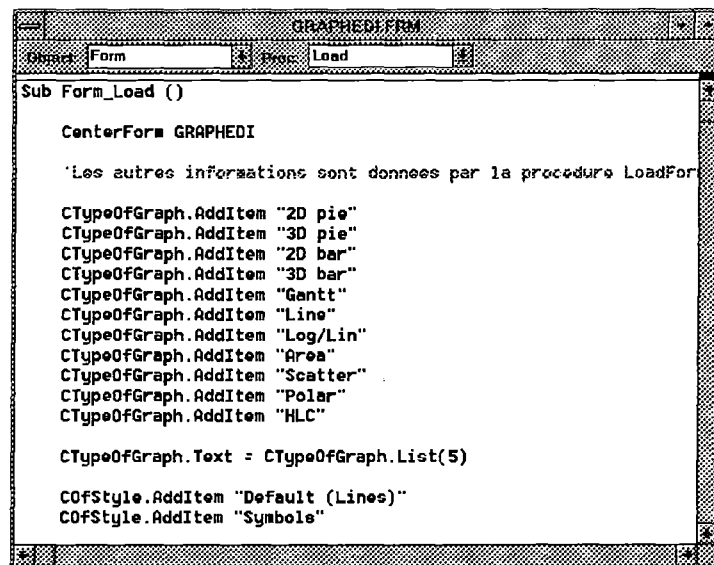


Figure 4.4 Editing code in Visual Basic

Visual Basic is a complete programming environment within the Windows environment and it has several advantages :

- Because its core is **compatible** with previous versions of QuickBASIC (with a few exceptions), only a small learning investment is required by the millions of current QuickBASIC programmers.
- It abstracts many of the 600 Windows **application program interface** (API) functions to a high level, and therefore enables a programmer to use existing Windows functionalities (buttons, dialog windows, menus, and so on) without using the Windows Software Development Toolkit (SDK). Programmers can still call low-level Windows API functions through a **dynamic link library** (DLL).
- It lets a programmer write, compile, run, and debug applications within the Windows environment, without exiting to DOS. No other popular Win-

dows compiler (including Borland's C++ or Turbo Pascal for Windows) does this as well. The Turbo Pascal for Windows debugger, for example, despite running as a Windows application, awkwardly switches from graphics to text screen during the debugging process.

- It incorporates resource development on-site -- that is, within the Visual Basic environment. You don't have to run a resource editor or compiler to add menus, dialogs, controls, and so on to an application (see **Figure 4.5**).

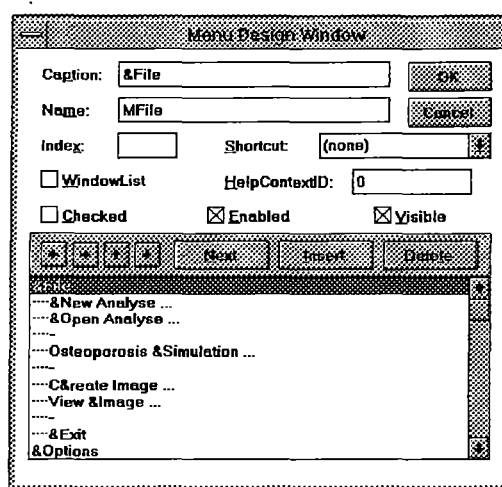


Figure 4.5 Menu Design Window in Visual Basic

In addition, « *thanks in part to an interpretive syntax checker, compiling Visual Basic applications is many times faster than both C and C++ compilers. Therefore, if you're trying to get a Windows application up and running quickly, or a large application prototyped, Visual Basic is clearly the system of choice. Because Visual Basic applications run as fast, or nearly as fast in most cases, as C or C++, you sacrifice very little by developing in Visual Basic* »^[Entsminger92].

The Visual Basic system is also being enhanced at an exceptional pace by way of Microsoft and third-party **custom control support**, **Dynamic Data Exchange (DDE)**, **Dynamic Link Library (DLL)** and **OLE client control**.

OLE enables Windows programmer to create applications that can both display data from other applications and edit the data from within the application

that created it. OLE, in other words, is DDE with user or application control of the data in other applications.

« The use of these sophisticated extensions establishes what Microsoft calls a « document-centered » view of computing. Typically, programmers and users have taken an application-centered view in that a complete task is usually a single application. The document-centered view lets any number of tools work on a document. This is multitasking at a very high level. »^[Entsminger92]

For example, your application (created in Visual Basic) might use Microsoft Excel, or another spreadsheet or database to hold and manipulate data; a graphics application or tool to plot the data (such as Paintbrush); a text editor to transpose some aspect of the data; the Windows recorder to run a macro on your data ; and your Visual Basic application to filter the data and control operations. This document-centered perspective, which Visual Basic supports, is a giant step forward in the world of multitasking and complex processing.

And finally, *« Corporations are increasingly adopting Visual Basic as a core information system development tool because of fast turnaround times, and the wealth of off-the-shelf components. To use Visual Basic effectively in a mature information system (IS) environment, you need to start judging your effort by the following criteria :*

- *maintainability,*
- *reusability,*
- *simplicity,*
- *testability,*
- *size, and*
- *speed.*

One way to accomplish these goals is with an « architectural approach ». For example, a layered paradigm has many benefits that help meet objectives in application development. The following is the short list :

- **Maintainability** : Code is organized in a recognizable manner. Task-oriented code is centrally located.
- **Reusability** : Task-oriented code is easily developed for reuse, specifically for tasks that cross application boundaries. The functions that support a task are also good candidates for reuse.
- **Simplicity** : Modular design removes the use of « spaghetti code » and fosters elegance, not « hacks ».
- **Testability** : Modules can be tested easily. Modularization breaks up the code coverage task into smaller, manageable units.
- **Size** : The memory footprint can be significantly reduced. Also, code size optimizations are much easier to implement and localize in a modular design.
- **Speed** : Modular code can be safely optimized without affecting the calling procedures. »^[MDNN94]

2. Functional Specifications

In this section we discuss the functional specifications of the functionalities that we have implemented in our program. We have based our work on the specification language introduced in [Dubois93].

To specify a software problem is to define it as precisely as possible but without focusing on its solution ([Meyer85]), that is why we only discuss the three main functionalities implemented in our program, i.e. the MIL analysis, the computation of the Euler number and the Osteoporosis simulation.

Furthermore we have only developed one function in full detail because we thought it was not relevant to perform the same task with all functionalities.

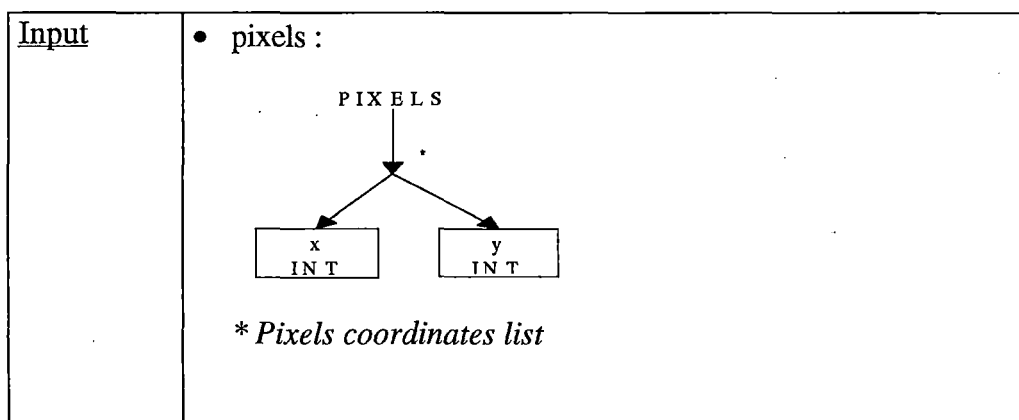
2.1 Perform MIL Analysis Functional Specifications

The goal of this function is to perform a MIL analysis as described in chapter 3 and to specify the associated results. To specify it we had to decompose it into a certain number of smaller functions. All those functions are specified below.

2.1.1 Perform_MIL Function

This is the main function of the performed MIL analysis function, it deals with initializations, calls to other functions and makes the final computations.

A. Interface and Main Data Structures



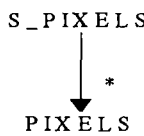
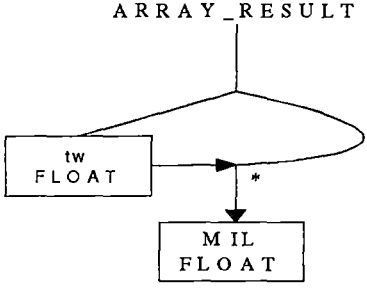
	<ul style="list-style-type: none"> globals : <div style="text-align: center;"> <pre> graph TD GLOBALS[G L O B A L S] --> xm_in["x m i n
I N T"] GLOBALS --> ym_in["y m i n
I N T"] GLOBALS --> xm_max["x m a x
I N T"] GLOBALS --> ym_max["y m a x
I N T"] </pre> </div> <ul style="list-style-type: none"> * <i>Pixels coordinates list boundaries</i> increment : INTEGER <ul style="list-style-type: none"> * <i>Rotation increment angle</i>
<u>Output</u>	<ul style="list-style-type: none"> result : <div style="text-align: center;"> <pre> graph TD RESULT[R E S U L T] --> MIL_min["M I L m i n
F L O A T"] RESULT --> MIL_max["M I L m a x
F L O A T"] RESULT --> tw["t w
F L O A T"] </pre> </div> <ul style="list-style-type: none"> * <i>Computation results</i>

B. Processing Rules

<u>Preconditions</u>	<ul style="list-style-type: none"> not (empty?(pixels)) <ul style="list-style-type: none"> * <i>Pixels coordinates list is not empty</i> $\forall i : (1 \leq i \leq \text{length}(\text{pixels})) \Rightarrow$ $(\text{xmin}(\text{globals}) \leq \text{x}(\text{pixels}_i) \leq \text{xmax}(\text{globals})) \wedge$ $(\text{ymin}(\text{globals}) \leq \text{y}(\text{pixels}_i) \leq \text{ymax}(\text{globals}))$ <ul style="list-style-type: none"> * <i>Values in globals represent the true boundaries of pixels coordinates list</i> $30 \leq \text{increment} \leq 180$ <ul style="list-style-type: none"> * <i>Boundaries for increment value</i>
<u>Postconditions</u>	<ul style="list-style-type: none"> diameter = $\text{xmax}(\text{globals}) - \text{xmin}(\text{globals})$ <ul style="list-style-type: none"> * <i>Compute the diameter of the circular test area from pixels coordinates boundaries</i> radius = diameter / 2 <ul style="list-style-type: none"> * <i>Compute radius of the circular test area based on diameter value</i>

- $\text{pixels}' = \text{Get_test_area}(\text{pixels}, \text{radius}, \text{diameter})$
** Get the circular test area from **pixels** and store it into **pixels'***
- $\text{area} = \eta * \text{radius}^2$
** Compute the surface-area of the circular test are*
- $\text{fraction_area} = \text{length}(\text{pixels}') / \text{area}$
** Compute bone fraction ratio from the number of bone pixels and the surface-area of the circular test area*
- $\text{pixels}'' = \text{Compute_Rotation}(\text{pixels}', \text{increment})$
** Compute rotations of the test area and store them into **pixels''***
- $\text{array_result} = \text{Compute_MIL}(\text{pixels}'', \text{area}, \text{fraction_area}, \text{globals})$
** Compute **MIL** indice for each rotations and the global **tw** value and store them into **array_result***
- $\text{MIL}_{\max}(\text{result}) = \text{MIL}(\text{array_result}_i) :$
 $(1 \leq i \leq \text{length}(\text{array_result})) \wedge$
 $(\forall j : (1 \leq j \leq \text{length}(\text{array_result})) \Rightarrow$
 $\text{MIL}(\text{array_result}_j) \leq \text{MIL}(\text{array_result}_i))$
** Get the maximum **MIL** value from **array_result***
- $\text{MIL}_{\min}(\text{result}) = \text{MIL}(\text{array_result}_i) :$
 $(1 \leq i \leq \text{length}(\text{array_result})) \wedge$
 $(\forall j : (1 \leq j \leq \text{length}(\text{array_result})) \Rightarrow$
 $\text{MIL}(\text{array_result}_i) \leq \text{MIL}(\text{array_result}_j))$
** Get the minimum **MIL** value from **array_result***
- $\text{nr} = (180 / \text{increment}) + 1$
** Compute the number of rotations*
- $\text{tmp_tw} = (\eta * \text{tw}(\text{array_result})) / (2 * \text{nr})$
** Compute temporary value for the computation of the **tw** value*
- $\text{tw}(\text{result}) = (2 * \text{fraction_area}) / \text{tmp_tw}$
** Compute final **tw** value*

C. Intermediate Data structures

<u>Intermediate data</u>	<ul style="list-style-type: none"> • pixels' : PIXELS • diameter : INTEGER • radius : INTEGER • area : FLOAT • fraction_area : FLOAT • pixels'' : <div style="text-align: center;">  <pre> graph TD S_PIXELS -- "*" --> PIXELS </pre> </div> <ul style="list-style-type: none"> • array_result : <div style="text-align: center;">  <pre> graph TD ARRAY_RESULT --> J1(()) J1 --> TW[tw
FLOAT] J1 --> MIL[MIL
FLOAT] TW -- "*" --> MIL MIL --> J1 </pre> </div> <ul style="list-style-type: none"> • nr : INTEGER • tmp_tw : FLOAT
--------------------------	---

2.1.2 Get_Test_Area Function

This function computes the test area on which the MIL analysis is performed.

A. Interface and Main Data Structures

<u>Input</u>	<ul style="list-style-type: none"> • pixels : PIXELS * <i>Pixels coordinates list</i> • radius : INTEGER * <i>Radius of the test area</i> • diameter : INTEGER * <i>Diameter of the test area</i>
<u>Output</u>	<ul style="list-style-type: none"> • pixels' : PIXELS

	<i>* Pixels coordinates list of the test area</i>
--	---

B. Processing Rules

<u>Preconditions</u>	<ul style="list-style-type: none"> • not (empty?(pixels)) <i>* Input pixels coordinates list not empty</i>
<u>Postconditions</u>	<ul style="list-style-type: none"> • not(empty?(pixels')) <i>* Output pixels coordinates list not empty</i> • (xctr,yctr)=Find_specimen_Centroid(pixels) <i>* Compute the centroid coordinates from pixels and store them into xctr and yctr</i> • $xmin = xctr - radius - (n * diameter) \Rightarrow$ $\exists n : ((n \geq 0) \wedge (xmin \leq diameter))$ <ul style="list-style-type: none"> <i>* The center of the test area x coordinate is equal to the x coordinate of the centroid minus the radius and n times the diameter (until it is inferior to the diameter value)</i> • $ymin = yctr - radius - (n * diameter) \Rightarrow$ $\exists n : ((n \geq 0) \wedge (ymin \leq diameter))$ <ul style="list-style-type: none"> <i>* Same as above but for the y coordinate</i> • $\forall i : (1 \leq i \leq \text{length}(\text{pixels}')) \Rightarrow$ $((xmin - radius) \leq x(\text{pixels}'_i) \leq (xmin + radius)) \wedge$ $((ymin - radius) \leq y(\text{pixels}'_i) \leq (ymin + radius))$ <ul style="list-style-type: none"> <i>* Each pixel coordinates are within a circle whose center coordinates are (xmin,ymin) and whose radius is radius</i>

C. Intermediate Data Structures

<u>Intermediate data</u>	<ul style="list-style-type: none"> • xctr : INTEGER • yctr : INTEGER • xmin : INTEGER • ymin : INTEGER • n : INTEGER
--------------------------	---

2.1.3 Find_Specimen_Centroid Function

This function computes the coordinates specimen centroid whose pixels coordinates are given.

A. Interface and Main Data Structures

<u>Input</u>	<ul style="list-style-type: none"> pixels : PIXELS * <i>Pixels coordinates list</i>
<u>Output</u>	<ul style="list-style-type: none"> xctr : INTEGER * <i>x coordinate of the specimen centroid</i> yctr : INTEGER * <i>y coordinate of the specimen centroid</i>

B. Processing Rules

<u>Preconditions</u>	<ul style="list-style-type: none"> not(empty?(pixels)) * <i>Pixels coordinates list not empty</i>
<u>Postconditions</u>	<ul style="list-style-type: none"> $xctr = \left(\sum_{i=1}^{length(pixels)} x(pixels_i) \right) / length(pixels)$ * <i>Specimen centroid x coordinate equal the sum of all pixels x coordinates divided by the number of pixels</i> $yctr = \left(\sum_{i=1}^{length(pixels)} y(pixels_i) \right) / length(pixels)$ * <i>Same as above but for the y coordinate</i>

2.1.4 Compute_Rotation Function

This function computes rotations of the original pixel coordinates and stores rotated pixel coordinates in a data structure. The number of rotations depends on the value of *increment* and determines the number of pixel coordinate lists inside the new data structure. This function uses another function

that we suppose already defined⁸ and which computes the rotated coordinates of a point given its current coordinates (x,y) and a rotation angle.

A. Interface and Main Data Structures

<u>Input</u>	<ul style="list-style-type: none"> • pixels' : PIXELS * Pixels coordinates to be rotated • increment : INTEGER * Rotation increment angle
<u>Output</u>	<ul style="list-style-type: none"> • pixels'' : S_PIXELS * Data structure containing the rotated pixels coordinates

B. Processing Rules

<u>Preconditions</u>	<ul style="list-style-type: none"> • not(empty?(pixels')) * Pixels coordinates list not empty • $30 \leq \text{increment} \leq 180$ * Increment value must be between 30 and 180 degrees
<u>Postconditions</u>	<ul style="list-style-type: none"> • $\text{length}(\text{pixels}'') = (\text{length}(\text{pixels}') / \text{increment}) + 1$ * The number of pixels coordinates lists contained in pixels'' • $\forall i : (1 \leq i \leq \text{length}(\text{pixels}'')) \Rightarrow$ $\text{length}(\text{pixels}(\text{pixels}'', i)) = \text{length}(\text{pixels}')$ * The number of pixels in each list contained in pixels'' is equal the pixels number of the original pixels list • $\forall i : (1 \leq i \leq \text{length}(\text{pixels}'')) \Rightarrow (\text{tmp_pixels} = \text{pixels}'') \wedge$ $(\forall j : (1 \leq j \leq \text{length}(\text{tmp_pixels}))) \Rightarrow$ $(x(\text{tmp_pixels}_j), y(\text{tmp_pixels}_j)) =$ $\text{Rotate}(x(\text{pixels}'_j), y(\text{pixels}'_j), ((i-1) * \text{increment}))$ * Each pixel coordinates of the pixels lists contained in pixels'' comes from a rotation of the corresponding

⁸ We suppose the *Rotate* function defined. We do not think it would relevant to specify it formally because it would overload this section. Furthermore we have extracted this function as such from the existing code thanks to the reverse engineering method developed later on in this chapter.

	<i>pixel coordinates contained in pixels' with a specified angle</i>
--	---

C. Intermediate Data Structures

<u>Intermediate data</u>	<ul style="list-style-type: none"> • tmp_pixels : PIXELS
--------------------------	---

2.1.5 Compute_MIL Function

This function computes the MIL value for each rotated test area computed by the *Compute_Rotation* function. It also computes the global trabecular width and store these values in a data structure. It uses a function (*Compute_Intercepts*) that we suppose already defined⁹ and which, given a pixel coordinate list and its boundaries, computes the number of intercepts which can be defined as the number of times that a pixel succeeds to a hole¹⁰ and inversely. For example, if we suppose that holes are represented by 0 and pixels by 1, then the number of intercepts for : 0001111011010101100 would be : 10.

A. Interface and Main Data Structures

<u>Input</u>	<ul style="list-style-type: none"> • pixels'' : S_PIXELS * <i>Data structure containing the rotated pixels coordinates</i> • area : FLOAT * <i>Surface-area of the test are</i> • fraction_area : FLOAT * <i>Bone fraction ratio of the test area</i> • globals : GLOBALS * <i>Pixels coordinates list boundaries</i>
<u>Output</u>	<ul style="list-style-type: none"> • result_array : ARRAY_RESULT * <i>Data structure to store computed values</i>

⁹ Same as for the *Rotate* function

¹⁰ Here a hole is considered as an absence of pixels

B. Processing Rules

<u>Preconditions</u>	<ul style="list-style-type: none"> • not (empty?(pixels'')) * <i>Data structure containing the rotated pixels coordinates not empty</i> • $\forall i : (1 \leq i \leq \text{length}(\text{pixels}'')) \Rightarrow (\text{tmp_pixels} = \text{pixels}''_i) \wedge$ $(\forall j : (1 \leq j \leq \text{length}(\text{tmp_pixels})) \Rightarrow$ $(\text{xmin}(\text{globals}) \leq \text{x}(\text{tmp_pixels}_j) \leq \text{xmax}(\text{globals})) \wedge$ $(\text{ymin}(\text{globals}) \leq \text{y}(\text{tmp_pixels}_j) \leq \text{ymax}(\text{globals})))$ * <i>Values in globals represent the true boundaries of pixels coordinates lists contained in pixels</i> • $0 < \text{fraction_area} < \text{area}$ * <i>Fraction_area superior to 0 and inferior to area</i>
<u>Postconditions</u>	<ul style="list-style-type: none"> • $\text{length}(\text{array_result}) = \text{length}(\text{pixels}'')$ * <i>There are as many values in array_result as there are pixels coordinates lists in pixels</i> • $\forall i : (1 \leq i \leq \text{length}(\text{pixels}'')) \Rightarrow$ $(\text{intercepts} = \text{Compute_Intercepts}(\text{pixels}''_i, \text{globals})) \wedge$ $(\text{MIL}(\text{array_result}_i) = ((2 * \text{fraction_area} * \text{area}) / \text{intercepts}))$ $\wedge (\text{tw} = \text{tw} + (\text{intercepts} / \text{area}))$ * <i>Formula to compute MIL values and trabecular width value</i>

C. Intermediate Data structures

<u>Intermediate data</u>	<ul style="list-style-type: none"> • intercepts : INTEGER • tmp_pixels : PIXELS
--------------------------	---

2.2 Compute Euler Number Functional Specifications

The goal of this function is to compute the Euler number as described in chapter 3. We do not specify it formally but we use below natural language as we thought it would not be relevant to specify all functions using formal language because of the overload it would cause to this chapter.

<u>Input</u>	<ul style="list-style-type: none"> • A data structure containing the pixels coordinates • Pixels coordinates boundaries
<u>Output</u>	<ul style="list-style-type: none"> • Euler number • Trabecular width
<u>Preconditions</u>	<ul style="list-style-type: none"> • The data structure containing the pixels coordinates is not empty • The boundaries of the pixels list are not corrupted
<u>Postconditions</u>	<ul style="list-style-type: none"> • The Euler number and the trabecular width of the specimen whose pixels coordinates were given as input have been computed following the method discussed in chapter 3

2.3 Perform Osteoporosis Simulation Functional Specifications

The goal of this function is to perform an osteoporosis simulation as described in chapter 3. The same remark like the preceding one concerning the fact that we specify this function in natural language is still appropriate.

<u>Input</u>	<ul style="list-style-type: none"> • A data structure containing the specimen pixels coordinates • Pixels coordinates boundaries • A simulation indice
<u>Output</u>	<ul style="list-style-type: none"> • A series of data structures containing the new specimen pixels coordinates (one data structure per simulation step)
<u>Preconditions</u>	<ul style="list-style-type: none"> • The data structure containing the pixels coordinates is not empty • The boundaries of the pixels list are not corrupted
<u>Postconditions</u>	<ul style="list-style-type: none"> • A data structure containing new pixels coordinates is produced for each simulation step • The number of simulation steps equal the number given in input as the simulation indice

	<ul style="list-style-type: none">• The number of pixels coordinates in the data structure corresponding to the first simulation step is inferior to the number of original pixels coordinates according to the principle discussed in chapter 3• The number of pixels coordinates of each simulation step is inferior to the number of pixels coordinates of the preceding step according to the same principle
--	---

3. Implementation Analysis

In this section we discuss the software architecture we have implemented in our program and the aftereffects it had on the behavior of our program. Then we discuss the specific programming techniques and algorithms we have developed and/or implemented.

3.1 *The Software Architecture*

3.1.1 **Description of the Software Architecture**

In this section we discuss our software architecture. We first present the global architecture of our program ; then we present the module architecture of two tasks performed and finally we show the mapping between the module architecture and an Object Oriented architecture.

3.1.1.1 *Global Architecture*

We now present below in **Figure 4.6** the **global architecture** of our program. It consists of **two main layers** : the **interface** layer and the **functionality** layer. They interact respectively with the user, the permanent data and with each other. The interface layer is, naturally enough, responsible for the user's dialogue management. The justification of this architecture is discussed later in this chapter¹¹.

¹¹ See section 3.1.2

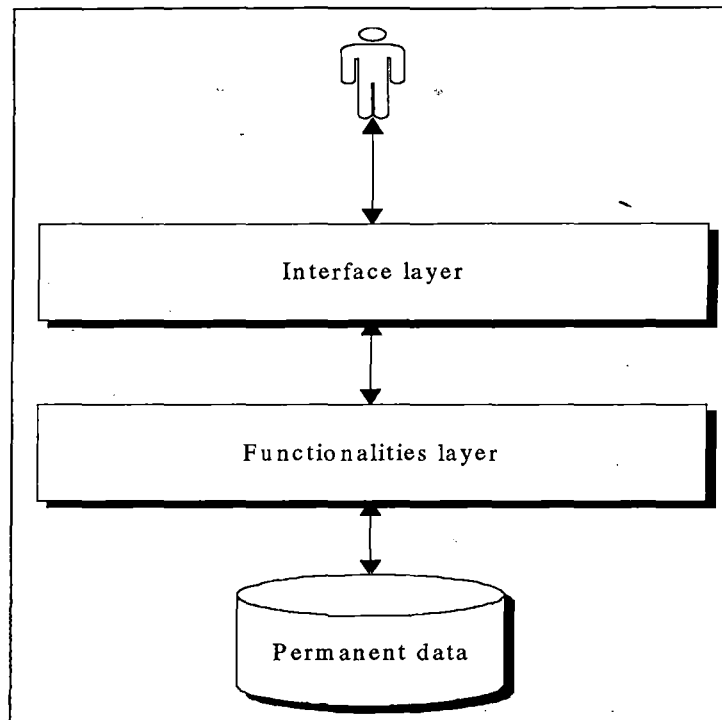


Figure 4.6 Global architecture

3.1.1.2 Module Architecture

We describe below the **module architecture** concerning the tasks : creation of a new analysis and visualization of an existing bitmap. We do not present the complete architecture of all tasks because we think it would not be of any interest.

The following architecture is based on the method described in [Dubois93] and the following considerations of [Parnas72]. Nevertheless we have not used the **traditional level hierarchy** proposed by that methodology. Indeed, we have decided to adopt a **level hierarchy** where **interface level modules** are placed **on top** of the hierarchy. This is often used when interface complexity outdoes functionality complexity. But this is not our motivation. Indeed, we have chosen such a hierarchy because we think it better fits with a **MS Windows application architecture** where functionalities are released by **events** fired in the **user interface** (i.e., when the user presses the OK button, then the analysis starts).

The **module levels** that we have chosen to represent are :

- **Level 1** : Operating System modules
- **Level 3** : Data access level
- **Level 4** : Functionalities level
- **Level 5** : Interface level

In the next figures, all relations between a level- n module and a level- m module such as $m < n$ imply that the level- n module **uses** services provided by the level- m module.

3.1.1.2.1 « Perform New Analysis » Module Architecture

The goal of the *perform new analysis* task, whose module architecture is shown on the next page in **Figure 4.7**, is to perform a selected **analysis** (MIL or Euler), on a selected **binary file** with selected **parameters**.

3.1.1.2.2 « Visualize Existing Bitmap » Module Architecture

We present below in **Figure 4.8** the module architecture of the *visualize existing bitmap* task whose goal is to display on the screen one or more selected **bitmaps** and corresponding **information**.

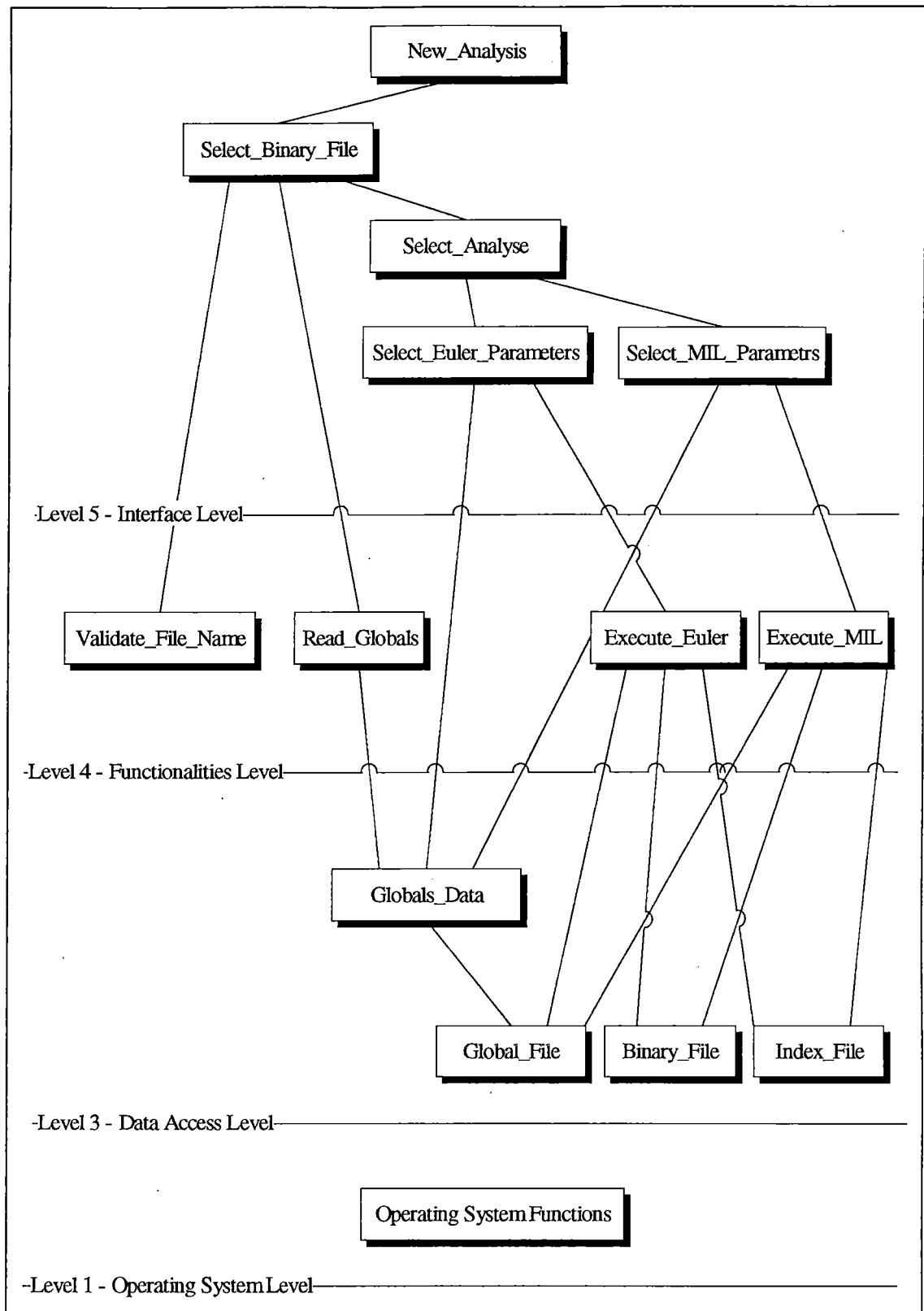


Figure 4.7 Modules architecture of the create new analysis task

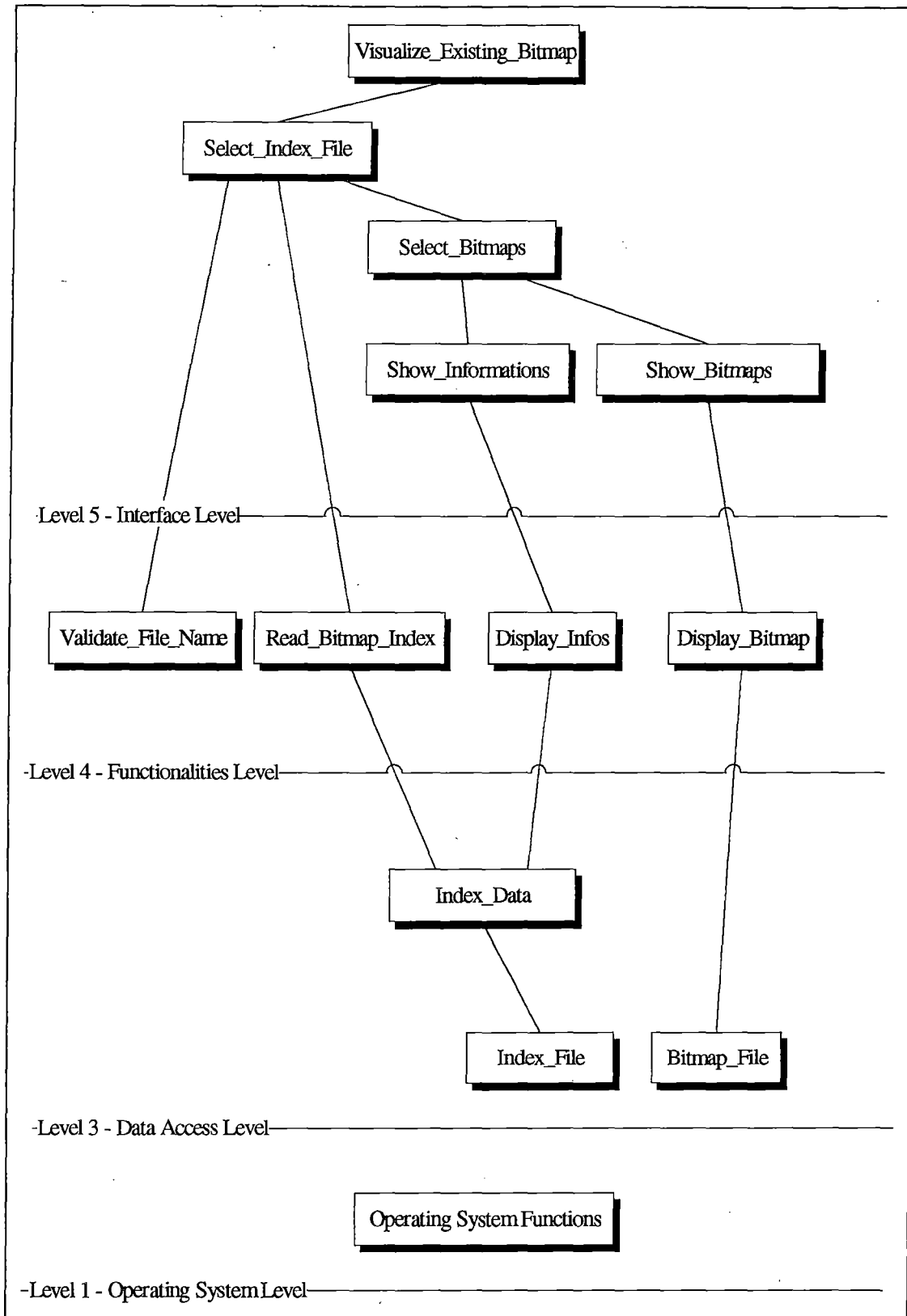


Figure 4.8 Modules architecture of the visualize bitmaps task

3.1.1.3 An Object Oriented Architecture

We now present the functionalities libraries Object Oriented architecture based upon the method described in [Booch94] and whose fundamentals are presented below.

A **class diagram** is used to show **classes** existence and their **relationships**, it represents the structure of the classes which compose the system architecture ([Booch94]). The most important elements of a diagram class are the classes themselves and their principal relationships. We show below, in **Figure 4.9**, the graphical notation used for classes and relationships.

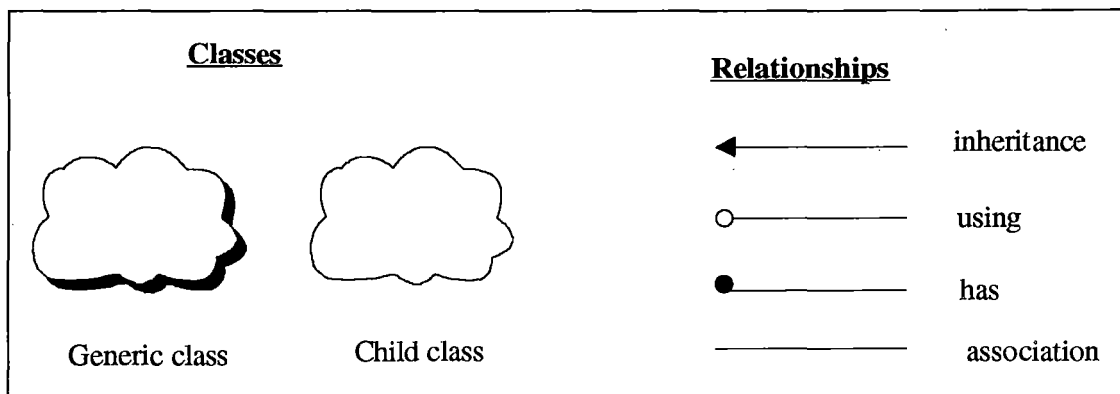


Figure 4.9 Representations used in [Booch94]

We think that using **class diagrams** present a good **illustration** of a **module architecture implementation**. Indeed, the mapping between both is quite immediate when looking at the two figures. For example, if we look at the level 3 modules shown in **Figure 4.7** (Global_File, Binary_File and Index_File modules), their Object Oriented implementation is clearly shown below (**Figure 4.10** and **Figure 4.11**), where all three modules are child objects of a main generic object. The goal of this generic object is to serve as a base container object for implementing all level 3 modules. Further more, the reader will find in the two figures below the objects corresponding to concepts described in section 3.2 of this chapter and also a concrete **illustration** of what is implemented inside each **dynamic libraries** (DLLs).

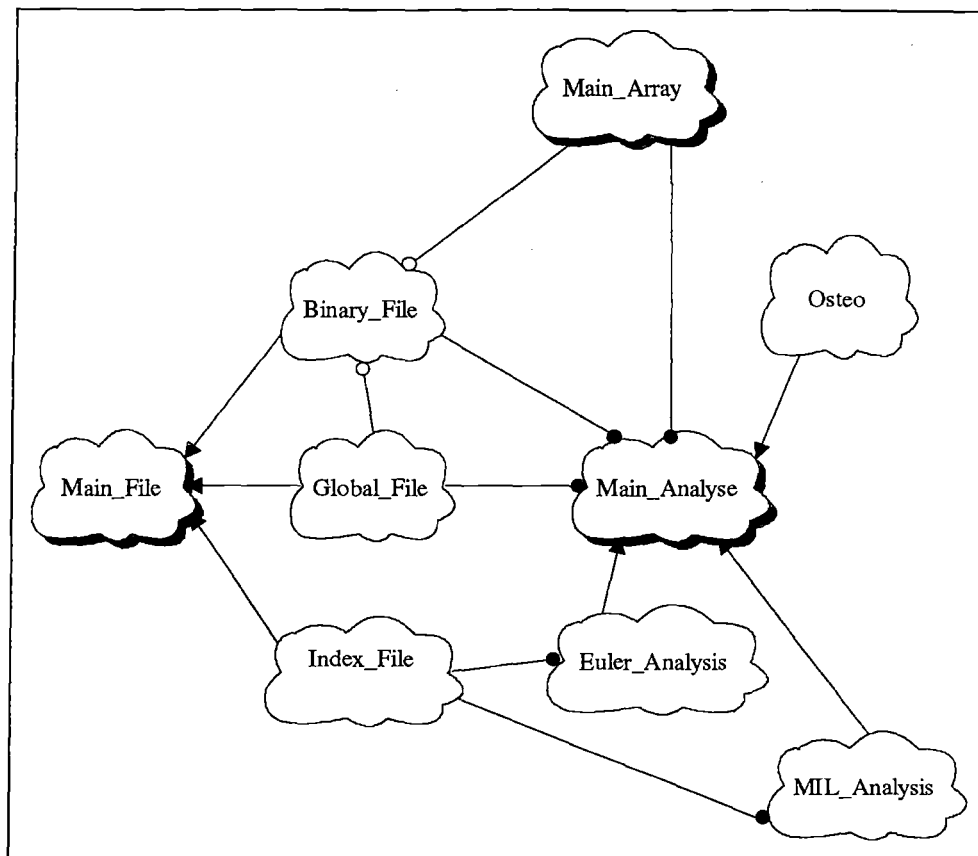


Figure 4.10 Class diagram of the Analyses DLL

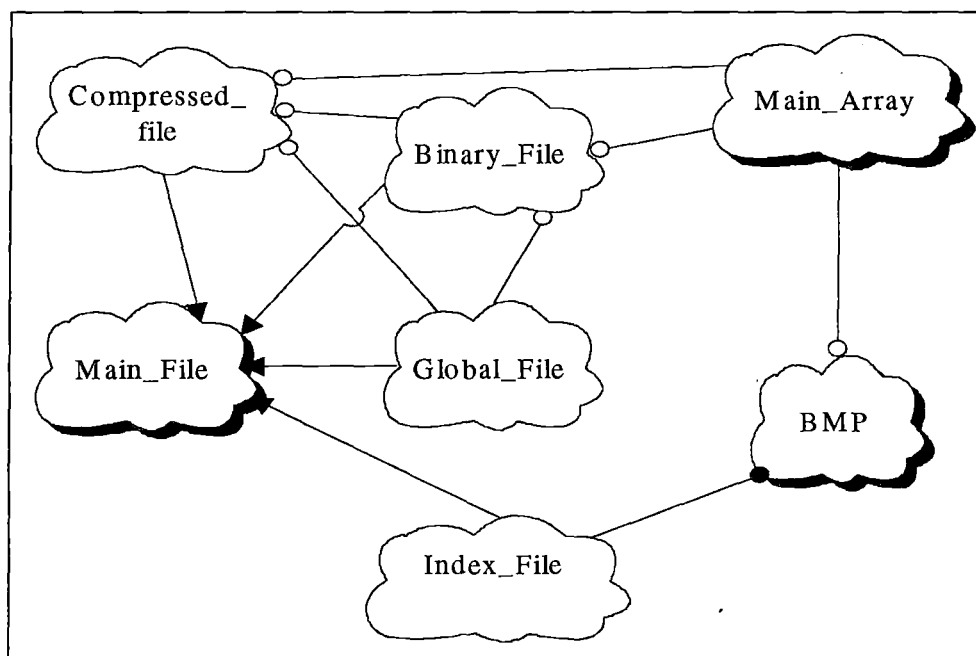


Figure 4.11 Class diagram of the Bitmap DLL

3.1.2 Justification of the Software Architecture

In this section we present **why** the software architecture we have described above helped us meeting the **functional** and **non-functional** requirements that have been expressed the first section of this chapter.

3.1.2.1 Meeting Functional Requirements

We discuss now the functional aspect of the requirements. As presented in section 1, we have splitted this point in two separate aspects : functionalities and user friendly interface. We present below both aspects separately to respect that distinction.

3.1.2.1.1 Functionalities

In fact, using the **C++** language provided us with a **powerful** and **modern** language which was necessary to fully implement desired functionalities in an object oriented fashion. It also provided us the capacity to implement **complex data structures** such as binary arrays that are discussed below.

Further more it permitted us to take full advantage of the **reverse engineering work** (discussed in section 3.5) we have accomplished by providing a **common base**, the **C** programming language, between our program and the existing set of programs.

Finally the use of **C++** allowed us to **access** more easily **binary files** at the necessary level to perform the compression and the manipulations that are discussed later in this chapter.

3.1.2.1.2 User Friendly Interface

Using the Microsoft **Visual Basic**, which is widely considered as the quickest and easiest way to create applications for Windows, helped us for two different aspects of the interface coding and design :

- it helped us making a **good looking** user friendly interface
- it helped us **manipulating graphical** items such as bitmap and graphic charts

Indeed, Visual Basic allows the programmer to spend **more time** focusing on **problems** to be solved and **less time on coding**. According to [Entsminger92] « the ability to create sophisticated applications has never been so available to such a broad group of people. Granted, Visual Basic can't turn every user into a programmer, but it can greatly simplify programming ». This is an important characteristic because we hadn't any practice of this language but **its easiness helped** us greatly **learning** it over a very short period.

Then we have been able to **spend more time on the interface design** because its coding was much easier and we had plenty of time to focus on the user will. More, as programming the interface was then much **closer to user level** than the traditional programming level, it helped us having a quick understanding of Mr. Keller's considerations as user.

Further more, the use of **high level functions** to display bitmaps and to create and manipulate graphic charts has been very helpful for the realization of all functionalities associated with those graphical elements.

Finally, the complete **set of appropriate tools** for the different aspects of **GUI¹² development** provided us with the facility to refine appearance and behavior of the interface objects.

3.1.2.2 Meeting Non-Functional Requirements

We discuss below the non-functional requirements that have been presented in section 1. We first talk about performances and then about programming tools and languages.

3.1.2.2.1 Performances

As presented below, we have developed **state of the art programming techniques** such as dynamically allocated bidimensional arrays, an original compression algorithm and an object oriented implementation. This has been

¹² Graphical User Interface

made possible by the use of the C++ programming language and those techniques are the reasons we have been able to provide high level performances.

Indeed it wouldn't have been possible to perform the same programming with **Visual Basic** as it is a **less powerful** programming language. Nevertheless it is more appropriate for developing the interface and C++ is more appropriate for manipulating complex data and implementing complicated algorithms.

3.1.2.2.2 Programming Tools and Languages

It is **obvious** that we have respected the requirements on this particular aspect as we have used the tools and languages that we were asked to use. There is then no more to say about that.

3.1.2.3 Conclusion

By making use of both, Visual basic and C++, key points, it helped us achieving a good **mix between user friendliness and high level performances**. If we had to stay with one of the two, we couldn't have offered the desired result and this is the best justification we can put forward to justify our software architecture. We indeed think that it consists of a **modern and useful architecture** that makes it possible and easier to provide both characteristics that every Windows application should provide : power and user friendliness.

3.2 Specific Programming Techniques and Algorithms

3.2.1 The Utilization of Dynamic-Link Libraries (DLLs)

As presented in the first part of this chapter (The Software Architecture), we have extensively used **DLLs** in our application. In this section, we first say **what** a DLL is, then we say **why** we have chosen to develop part of our application using this concept and what where the resulting advantages.

3.2.1.1 What is a DLL ?

Three different **meanings** can be associated with the term Link-Libraries according to [Myers&Hamer93], but all libraries contain groups of functions that are gathered together in the same file. The difference comes from their organization and their addressing.

Languages such as **C** rely on **libraries** for the implementation of their functionalities. The linker includes in executable code functions that have been extracted from the libraries. These can then be used by the program as its own functions. The library is then aimed at preventing the programmer from recording basic functions. Such libraries are often called « *static* » **libraries** because they are included inside the executable code when it has been compiled and linked.

Static libraries present a big problem for Windows. Indeed, as a multitasking operating system, Windows tends to run more than one application at the same time. Thus, if two programs use the same library, it is then present twice in memory causing a **waste of memory** which is maybe the most critical Windows resource. The concept of **dynamic libraries** even though it has not been invented for Windows is widely used thanks to the Microsoft OS. With dynamic-link libraries, functions are linked dynamically, that is to say that there is no more incorporation into the executable code but it is the program itself which calls the desired functions dynamically. Furthermore, these functions can be simultaneously used by several programs even though only one copy of the function is present in memory, thus DLL can contribute to save memory.

The third and last sort of library is the **incorporation library**. It is a small file that contains the reference points to the corresponding DLL. It indicates what functions are present and where to find them. For example, the file *user32.lib* contains references to all the functions of the *user32.dll* library.

3.2.1.2 Why using DLLs ?

Dynamic-Link Libraries (DLLs) are a key feature of Microsoft Windows. As their name suggests, **DLLs** are libraries of **procedures** that applications can link to and **use at run time** rather than link to statically at compile time. This means that the libraries can be updated independently of the application, and many applications can share a single DLL.

DLL present a certain amount of general **advantages** according to [Myers&Hamer93] :

- It is possible to **personalize a program** by using several DLLs. For example, it could be possible to include messages displayed by a program in DLLs so that the user may choose the desired language when installing the program. Let us this means that the messages in French would be included inside the file *french.dll* and the messages in English would included inside the file *english.dll*. The installation program would then just have to copy the appropriated DLL to provide messages in the selected language.
- Using DLLs can lead to **share functions between several applications** which means a reduction in development costs.
- Developers with a **specific knowledge** can create **DLLs** for others whose knowledge in the same domain is not that good. For example, animation or 3D graphics libraries.
-

Using DLLs presented for us the following advantages :

- As we said when justifying our software architecture, it provided us with the capacity of **combining** key points of the C++ programming language with the key points of the **Visual Basic** programming language.
- It has made it possible for us to **split the work into modules** whose development has been parallel.
- It has been easier to test, modify, recompile and debug the application as **functionalities were interface independent**.

- As we said previously, DLLs are also important to **save memory** which we have used intensively¹³.
- Finally, it provides an easy path for further **evolutions** and maintenance work.

3.2.2 Memory Management

In this section we first say what usage we have made of the memory and then we explain what programming techniques and operating system capacities helped us manage it.

3.2.2.1 The Bidimentional Memory Arrays

As described in chapter 3, the existing set of programs did a **poor usage of memory**. Indeed, all computations were directly made on the screen (the display screen) using color tags and commands such as *putpixel*¹⁴ and *getpixel*¹⁵. Hence the low performance level which was obtained by those programs.

As **increasing performances** is a key aspect of our work, taking full advantage of **memory** has consequently been one of our main subjects. We think it is not necessary to explain here the advantages obtained when using memory for computations instead of screen displays as it should be obvious for everybody. Nevertheless, using memory means finding a good data representation that would fit with both the computation needs and the memory size restrictions. That is why we have chosen to implement **dynamically allocated bidimentional arrays**.

Concretely, these arrays are arrays of **pointers** to unsigned short integers and their size, in pointer number, correspond to the size of the screen, in pixel number, that would be necessary to display the image represented in the input binary file. **Bone pixels** (white pixels on **Figure 4.12**) are assigned the **value 1** to the corresponding unsigned short integer whose pointer coordinates in the

¹³ See section 3.2.2

¹⁴ The *putpixel* command puts a pixel on the screen at a specified location and with a specified color

¹⁵ The *getpixel* command reads the current color of the pixel at the specified location

array correspond to a projection of the screen coordinates. For example, if we assume that the image screen landmarks are (100, 500) for X and (200, 400) for Y, then the size of the array would be 400 ($=500-100$) for X and 200 ($=400-200$) for Y. Thus, if the pixel whose screen coordinates are (200,350) is a bone pixel, the array square (200-100,350-200) will contain value 1. **Marrow pixels** (black pixels on **Figure 4.12**) are assigned **value 0**. This process is illustrated below in **Figure 4.12**.

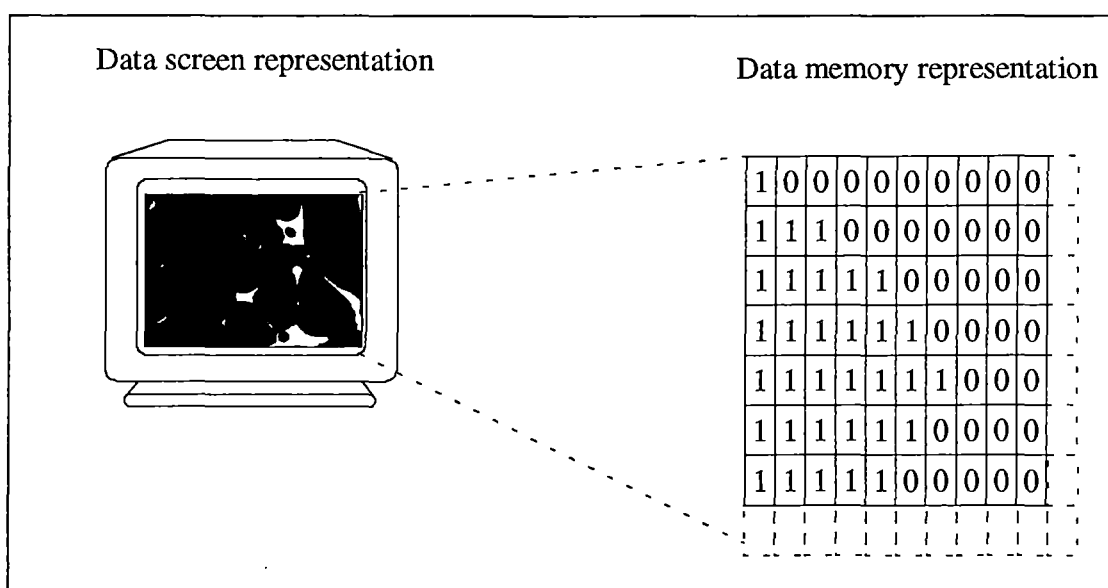


Figure 4.12 Data screen representation and its projection to data memory representation

Using the described arrays gave us the following advantages :

- A data structure **compatible** with screen characteristics used by the existing set of programs. Hence it enabled us to use existing computation algorithms that we have obtained thanks to our reverse engineering work¹⁶.
- A data structure **simple** enough to be manipulated with relative ease but complex enough to fulfill our needs.
- A dynamic characteristic that helped us **sparing memory** as we were able to use an array that would fit exactly our needs for a specific task, thus preventing waste of memory that would have been caused by fixed size arrays.

¹⁶ See section 3.5

3.2.2.2 Memory Management Techniques

We have used a certain number of **programming techniques** that helped us **managing memory**. We discuss below what programming techniques and what were the resulting advantages :

- As we explained previously¹⁷, we have implemented a **data structure** (dynamically allocated bidimensional memory arrays) that would fit our memory needs for a specific task, thus minimizing memory waste.
- We have also used an **Object Oriented approach** that provided features for a proper use of memory. Indeed, we have implemented our memory arrays within an object that allowed us to use the *new* and *delete* operators. These operators are used to allocate and to free the necessary memory for an object. It was then possible to manage dynamically the memory so for its allocation that for its freeing.
- Finally, using **Dynamic-Link Libraries** (DLL) also helped us saving and managing memory as we have explained it above¹⁸.

Further more, the **Microsoft Windows** operating system provides **features** for managing memory, we quote some of them below without describing them in details as it is not the goal of this thesis¹⁹ :

- **EMS** (expanded memory specifications) and **XMS** (extended memory specifications) support
- **Virtual memory** support
- Utilization of **segmented memory**
- **Discardable** and fixed memory
- ...

¹⁷ See section 3.2.2.1

¹⁸ See section 3.2.1

¹⁹ The reader may find detailed informations about the subject in [Norton&Yao92].

3.2.3 Compression of Binary Files

« **Compression** is the **process** used to **reduce** the physical size of a block of information »^[Murray&vanRyper94]. Compression is useful to fit a larger block of information in a given storage space or in a given block of memory.

In the following sections, we introduce our motivations for developing a compression algorithm, then we discuss the data compression basic concepts and terminology and finally we describe the unique method we have developed for our program.

3.2.3.1 Motivations for a Compression Process

The **binary files**, which have been introduced in chapter 3 and which are a **necessary input** for analysis and bitmap creations, have the disadvantage to **require a large amount of storage space**. Indeed, the size of a binary file can range from a few megabytes to **several tens of megabytes** depending on the size of the trabecular bone digitized. Thus and eventhough storage cost has decreased dramatically over the last years, the utility of a a high performance compression algorithm is obvious.

This consideration and the fact that the compression method used by the existing set of programs, that is to say the utilization of the **PKZIP** compression utility, **was not enough powerful**, persuaded us of developing **our own method**.

3.2.3.2 Data Compression Terminology

A compressor, naturally enough, performs compression and a decompressor reconstruct the original data. Although this may seem obvious, a decompressor can operate only by using knowledge of the compression algorithm used to convert the original data into its compressed form.

The terms *unencoded data* and *raw data* describe data before it has been compressed. The terms *encoded data* and *compressed data* describe the same information after it has been compressed.

3.2.3.2.1 Physical and logical compression

« Compression algorithm are often described as squeezing, squashing, crunching, or imploding data, but these are not very good descriptions of what is actually happening » ^[Murray&vanRyper94]. Of course, the major use compression is to make data use less disk space , but it **does not physically cram the data into a smaller size package** in any meaningful sense. Instead, « compression algorithms are **used to reencode data into a different, more compact presentation** » ^[Murray&vanRyper94]

The distinction between physical and logical compression methods has to be made on the basis of how the data is compressed. « **Physical** compression is performed **on data exclusive of the information** it contains » ^[Murray&vanRyper94]. That is to say that it only translate a series of bits to one pattern to an other one more compact. The **relationship** between the compressed data and the raw data is **mechanical** and will not appear as obvious to somebody which does not know about it. The physical compression algorithm removes the **redundancy** that existed in the data itself, thus making it smaller.

« **Logical** compression is accomplished through the process of **logical substitution** » ^[Murray&vanRyper94]. That is to say replacing a symbol, or a data representation by another one. Changing « United States of America » by USA is a good example of a logical compression. Logical compression works only on data at the character level or higher and is **based exclusively on informations contained within the data**.

3.2.3.2.2 Symmetrical and asymmetrical compression

In one hand, « **symmetric** compression method uses roughly the **same algorithms** and performs the **same amount of work** for compression as it does for decompression » ^[Murray&vanRyper94]. In the other hand, « **asymmetric** methods require substantially **more work** to go in one direction than they require in the other » ^[Murray&vanRyper94]

The **algorithm** choice has to be **related to the application nature**. For example, if we write a backup file routine, then we can expect that some files will never be read. In such an example, a fast compression algorithm that is expensive to decompress might be enough. On the contrary, if we make an image database where a picture will be compressed once for storage but will be decompressed often for viewing, then an asymmetric algorithm which uses much CPU time for compression, but is quick to decode would work well.

3.2.3.2.3 Lossy and lossless compressions

Some compression algorithms are called **lossless**. What does this mean ? « When a chunk of data is compressed and then decompressed, the original information contained in **the data is preserved** » ^[Murray&vanRyper94] That is to say that no data has been lost. Those algorithms are used when the data must imperatively remain unchanged. For example, compression of programs must be lossless since a difference of a single bit may cause a failure.

Lossly compression methods however, « **throw away some of the data** » ^[Murray&vanRyper94] This is in order to achieve compression ratios better than those of most lossless compression methods. Algorithms of this kind are often used to compress high resolution images where a single pixel is barely visible anyway.

3.2.3.3 The LRLE method

In this section, we describe the unique method and the associated algorithm we have developed for our program.

We have called it the **LRLE** method, or logical and run-length encoding method, because it is a two steps method which **combines a logical** compression of the original data with a **physical RLE** compression.

Below, we discuss both parts of the compression process separately, pointing out general characteristics we have presented above and original aspects.

3.2.3.3.1 The logical compression

Logical compression naturally enough has to be the **first step** of the compression process. We say naturally enough because as it is aimed to substitute a new representation for the data, it must be performed before the physical compression after which that substitution would not be possible anymore.

The most important aspect of this logical compression is the performing of a **logical substitution** that **helps** compression process next step, the **physical compression**, achieving good results. Eventhough we discuss the physical compression algorithm below, we can already point out the fact that it works well with black and whites pictures. This is important to understand why the following substitution is relevant.

Indeed, we have used the **bidimensionnal binary arrays** that have been described in a previous section²⁰. Using these provided us with all the desired **characteristics** :

- It is a **logical substitution** in the sense that we have replaced a data representation with an other one.
- There is not any **loss** of the original information.
- The new data representation **fits** perfectly the **physical** compression.
- Bidimensional binary arrays are used throughout the application, thus we have been able to use existing objects and less specific code had to be written for the compression.

3.2.3.3.2 The physical compression

The **physical** compression algorithm we have used is a typical **run-length encoding** (RLE) algorithm which is supported by most bitmap file formats.

RLE which may not achieve the high compression ratios of the more advanced compression methods is both **easy** to implement and **quick** to execute. These

²⁰ See section 3.2.2.1

are very important characteristics as we wanted the compression to be relatively fast since a too slow process would reduce for the user the gain obtained.

More, RLE is a **symmetric** compression algorithm which means, as we said above, that it does not require a lot more work to go in one direction than to go in the other. The compression and uncompression times described in a following section, are a direct consequence of this and fit well with the high performances care we have developed throughout our program.

RLE is suited for **compressing** any type of **data regardless of its information content** but it affects the compression ratio. This is the main limitation of the method but it is also the key aspect which lead us to adopt the RLE algorithm. Indeed, it works by **reducing the physical size** of a repeating string of character, called a **run**. This process works well if the length of the **run** is quite long. Thus, compressing a black-and-white image, for example, would work well as there are only two possible values which makes long **run** existence credible. If we recall the **logical substitution** we have described in the previous section, it is then obvious that a RLE method fits perfectly to our needs.

Now that we have justified the choice of the compression method, we describe its standard algorithm and how we have implemented it.

As we said above, RLE works by reducing the physical size of a repeating string called the **run**. It is encoded in two bytes. The first byte representing the number of characters in the **run**, it is called the **run count**, the second byte representing the value of the character in the **run**, it is called the **run value**. For example, if we consider the following **run**,

AAAAAAAAAAAAAAAAAAAA

then it would be encoded with the following two bytes :

15A

15 being the *run count* and A being the *run value*. The 15A code generated is called an **RLE packet**.

A new packet is generated each time the *run* character changes. If we now consider the new fifteen characters string containing four different character *runs*

AAAAAABBBXXXXXT

Using the RLE encoding process, this could be compressed into four 2-bytes **RLE packets** :

6A 3B 5X 1T

This shows clearly that the **efficiency** of the method **depends** on the type of the data to be encoded. This is true to the extent that if we had a string containing fifteen different *runs*, then it would be encoded using fifteen 2-bytes **RLE packets**, thus the encoded string would be twice as big as the raw data.

The basic flow of all RLE algorithms is the same, it is shown on below in **Figure 4.13**.

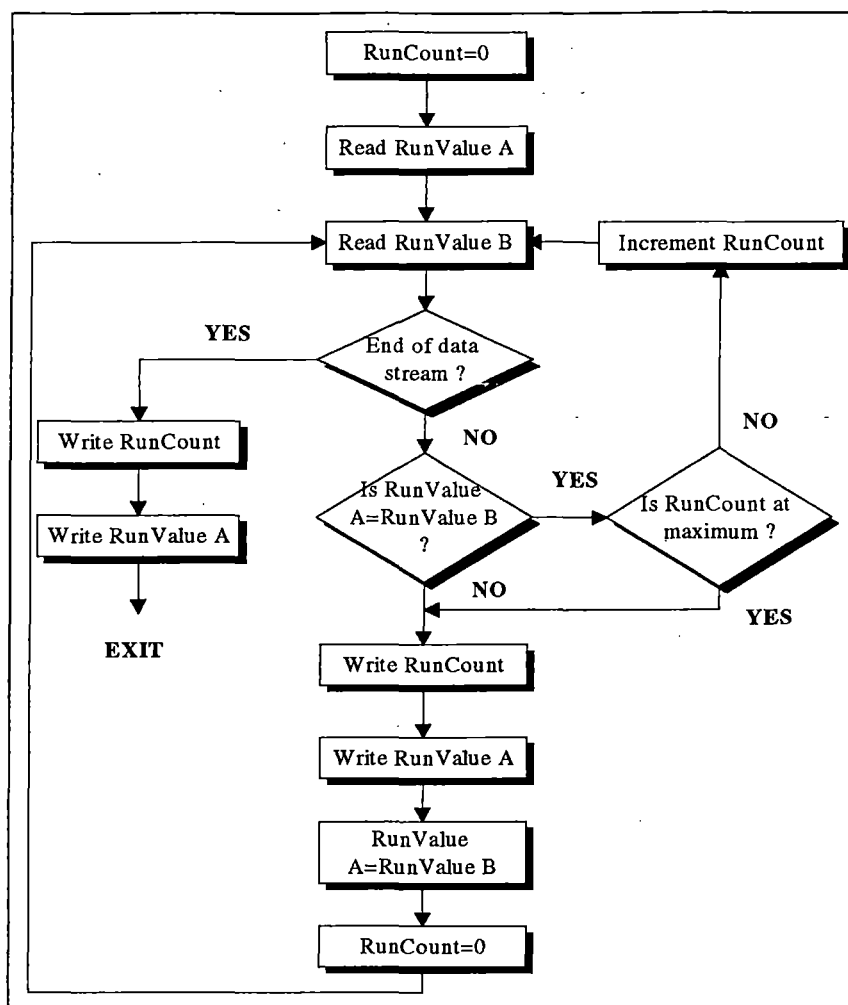


Figure 4.13 Basic run-length encoding flow^[Murray&vanRyper94]

Though data is normally run-length encoded in a **sequential process** that treats the data as a one dimensional stream rather than as a two dimensional map of data, as in our algorithm, variants of this method exist. In sequential processing, the data is encoded starting at the upper left corner and proceeding from left to right across each scan line (the X axis) to the bottom right corner of the array. This the process we have used to encode our binary arrays. It is illustrated in **Figure 4.14, a**. But alternative RLE schemes can also be written to encode data down the length of the array (the Y axis) along the columns (see **Figure 4.14, b**), to encode the array into 2-dimensional tiles (see **Figure 4.14, c**), or even to encode it on a diagonal zigzag fashion (see **Figure 4.14, d**). Nevertheless, variants such as the last one might be used in highly specialized applications, but are usually quite rare.

The last characteristic of the RLE algorithm that we have implemented in our program is its **lossless** compression. It is obvious regarding the **nature of the data** compressed that we could not afford any loss. Thus even though zeroing-out one or two least significant bits could have lead to a better compression ratio, it was not an acceptable solution for us.

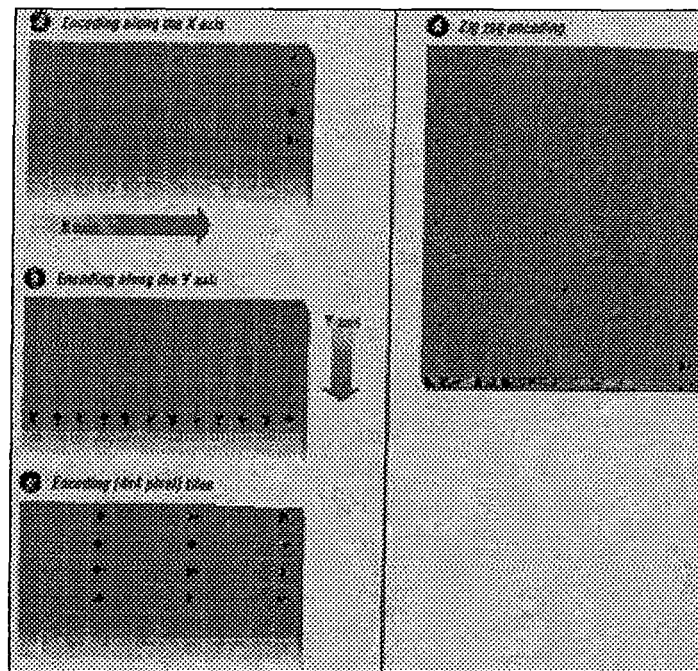


Figure 4.14 Run-length encoding variants^[Murray&vanRyper94]

3.2.3.3.3 The LRLE algorithm

In order to conclude this section, we present below in **Figure 4.15** the complete algorithm for the LRLE compression process of a binary file. The method can be divided in **three steps** :

1. The first step is the reading of a **plane in the binary file** (a plane is a set x,y,z coordinates whose z value remain the same). That plane is stored inside a bidimentional binary array.
2. The **binary array** is then coded using the Run-Length-Encoding algorithm presented above in **Figure4.13**. The only difference at this point with a traditional RLE method is that we compute the number of couples (*RunCount,RunValue*) used to code that plane. That particular value called the *PlaneValue* is used when decompressing the file to know exactly how

many couples (*RunCount,RunValue*) have to be read to recreate the original plane. These are saved in a separate file.

3. The **encoded plane** is then saved in a file whose name is the same as the binary file and whose extension is CBF (Compressed Binary File). The corresponding PlaneValue is saved in a file using the same name and PID (Planar InDex) as extension.

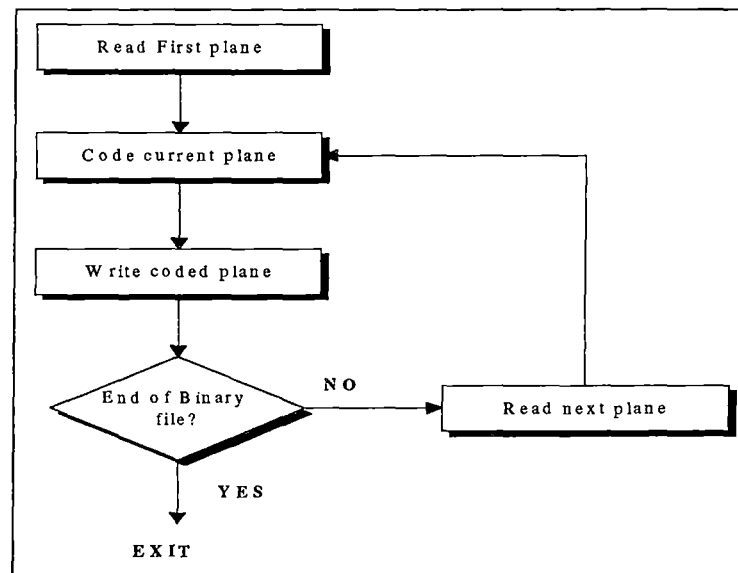


Figure 4.15 The LRLE algorithm

3.2.4 Multiple-Document Interface (MDI) Application

The **MDI** allows you to create an application that maintains **multiple forms within a single containers form**. Applications such as Windows Program Manager, Windows File Manager, Microsoft Excel and Microsoft Word for Windows have multiple-document interfaces.

An MDI application allows the user to display multiple documents at the same time, with each document displayed in its own window. Document windows are contained in a **parent-window**, which provide a workspace for all the document windows in the application. For example, Windows Program Manager allows you to create and display many different program groups. Each group is displayed in its own window and is confined to the area of the Program Manager's parent window.

In an MDI application, the **child menus** are displayed on the MDI form, rather than on the child forms themselves. When a child form has the **focus**, that child menu (if any) replaces the MDI form menu on the menu bar. If there are no children loaded, or if the child with the focus does not have a menu, the MDI form menu is displayed.

It is common for MDI applications to use **several sets of menus**. When the user opens a document, the application displays the menu associated with that type of document. Usually, a different menu is displayed when there are not any children loaded. For example, when there are no spreadsheets open, Microsoft Excel displays only the File and Help menus. When the user opens a spreadsheet, other menus are displayed (File, Edit, Formula, and so on). When a user opens a chart, a different set of menus is displayed (File, Edit, Gallery, Chart, and so on).

3.2.5 Bitmap Creation and 3D Representation

In this section we discuss the general principles of bitmaps, then say why we have chosen bitmaps and the Microsoft BMP format in particular. Then we discuss the use of gray scale color palettes and the 3D modelisation that we have made.

3.2.5.1 *Bitmap files*

Although bitmap files vary greatly in their details, all share the same **general structure** which is presented below in **Figure 4.16**.

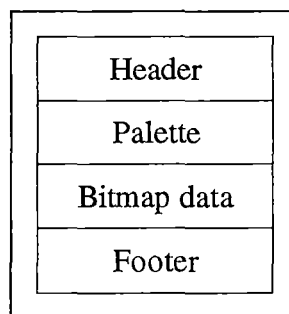


Figure 4.16 Bitmap file general structure^[Murray&vanRyper94]

« The header is a section of binary- or ASCII-format data normally found at the **beginning of the file**, containing information about the bitmap data found elsewhere in the file »^[Murray&vanRyper94]. The header contains **informations** such as a file identifier, a file version, image description informations, compression type, etc. All bitmap files have some sort of header.

The bitmap **data** usually makes up the **bulk** of a bitmap format file and is found, in general, immediately after the end of the file header. « *Bitmap data is composed of pixel values. Pixels on an output device are usually drawn in scan lines corresponding to rows spanning the width of the display surface. This fact is usually reflected in the arrangement of the data in the file* »^[Murray&vanRyper94]. Several scan lines combined form a **two-dimensional grid** of pixels data, thus we can think of each pixel in the bitmap as located at a specific logical coordinate. This particular aspect will be a key point when justifying the choice of this format.

« A **palette** which is something referred to as a color map, index map, color table or look-up table (LUT), is a **1-dimensional array of color values**. As the synonym look-up table suggests, it is the cornerstone of the method whereby colors can be referred to indirectly by specifying their positions in an array »^[Murray&vanRyper94]. This is the principle used for bimages where pixel data values are references to the array composing the color palette of file. The basic principle of this projection is shown below in **Figure 4.17**.

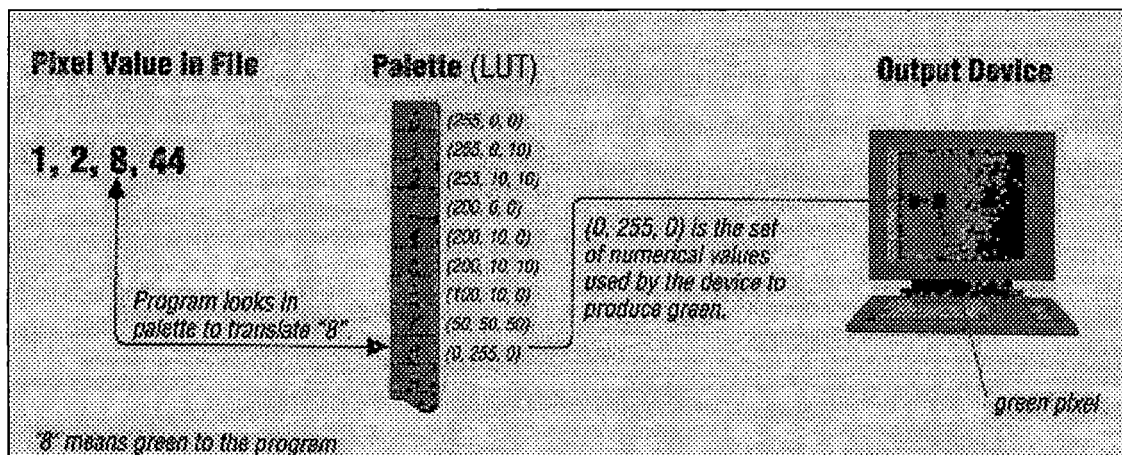


Figure 4.17 Using a palette to specify a color^[Murray&vanRyper94]

« The **footer** , sometimes called the trailer, is a data structure similar to a header and is often an addition to the original header, but appended to the end of a file »^[Murray&vanRyper94]. A footer is usually added when the file format is upgraded to accommodate new types of data and it is no more convenient to add or change informations in the header.

3.2.5.2 The Microsoft Windows Bitmap File Format

« The Microsoft Windows Bitmap (BMP) file format is one of several supported by Microsoft Windows. **Most** graphics and imaging applications operating in the Microsoft Windows environment support creation and display of BMP format files »^[Murray&vanRyper94]. The current BMP format is **hardware independent** and can accommodate images with up to 24-bit color. Its basic design, illustrated below in **Figure 4.18**, makes it a good general-purpose format that can be used for color or black-and-white image storage when file size is not a major factor. Indeed, the bitmap format which is **quick** and **easy** to read and uncompress **lacks** a superior **compression** scheme. Nevertheless, for our program this was not a key aspect.

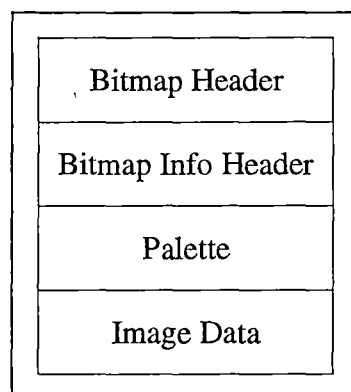


Figure 4.18 Windows Bitmap file structure

The MS-BMP format particularity is that it does contain two headers but no footer. Despite that, it is coherent with the general bitmap file format that we have discussed above.

3.2.5.3 Why the Bitmap File Format ?

The bitmap file format present a certain number of advantages, some are general purpose advantages, some are tied to our specific application, some combine both advantages.

« Bitmap files are especially suited for the storage of real-world images; complex images can be rasterized²¹ in conjunction with video, scanning, and photographic equipment and stored in a bitmap format »^[Murray&vanRyper94]. Given this, it is already much clearer that the bitmap file format is a **logical choice** for the representation of trabecular bones.

Some advantages of the bitmap file format include the following :

- *« Bitmap files may be easily created from existing pixel data stored in an array of memory »*^[Murray&vanRyper94]. This comes from the data representation used that we have explained in previously and fits perfectly to the memory arrays that we use extensively for our computations and that we have discussed in section 3.2.2.
- *« Pixel values may be modified individually or as large groups by altering a palette if present »*^[Murray&vanRyper94]. The utilization of a specific palette is also an important aspect of our program. We discuss the use we have made of the palette in the next point.
- *« Bitmap files may translate well to dot-format output devices such as printers »*^[Murray&vanRyper94]. This is important as images created with our program are susceptible to be included in reports or articles, thus it is likely that they may be printed.
- The Microsoft Windows bitmap file format is used and recognized by **almost all Microsoft Windows based applications**. Even word processor such as Microsoft Word or spread sheet such as Microsoft Excel can incorporate bitmap images in their documents. Further more, Microsoft Visual

²¹ Rasters refer to graphics data represented by color values at points, which taken together describe the display on an output device. It is a synonym for bitmap.

Basic provides functions which make display and manipulation of bitmap images much easier.

-

3.2.5.4 3D Representation

We have used two different features for a 3D representation of the trabecular bone specimen coded into binary files.

At first, we have used a **gray scale color palette** in our bitmap files. A color palette, which has been introduced above, is an array of color values. To simulate a 3D aspect, we have used a gray scale palette where colors range from white to gray and from gray to black going through variations of these colors. This 8 bits color palette is composed of **RGB triplets** (256 triplets) whose values represent the varying amounts of the red, green, and blue colors added to black to produce the desired color. Each of the three values composing an RGB triplet is the **same** in a gray scale palette (see **Figure 4.19** below).

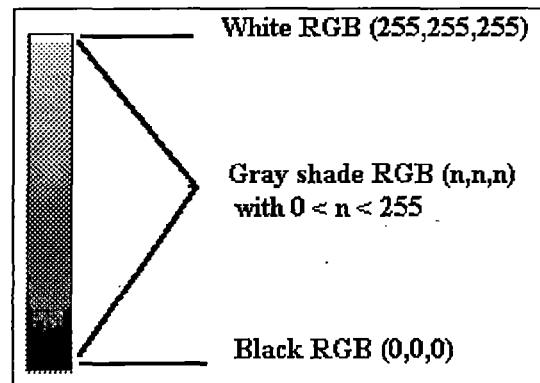


Figure 4.19 Gray scale color palette representation

Using this color palette such as first plane pixels being assigned white color, last plane pixels being assigned black color, and intermediate planes pixels being assigned a gray shade color provides a **depth feeling** that is illustrated below in **Figure 4.20**.

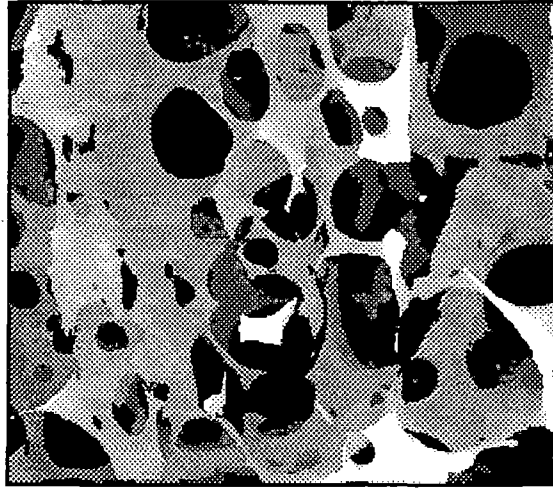


Figure 4.20 Bitmap image of a trabecular bone specimen using a gray scale color palette

Then we have decided to provide the user with a 3D representation of a **cubic trabecular bone area**. It is displayed each time the user click on a bitmap picture and it is aimed to be a **graphical feedback** whose purpose is to show to the user the area of the trabecular bone represented by the bitmap he is currently watching. We show below in **Figure 4.21** an illustration of such a representation.

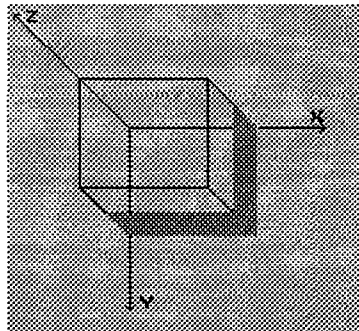


Figure 4.21 3D representation of a trabecular bone cubic area

Indeed, « *the 3D viewing process is inherently more **complex** than the 2D viewing process. In 2D, we simply specify a window on the 2D world as a viewport on the 2D view surface. Conceptually, objects in the world are clipped against the window and are then transformed into the viewport for display. The extra **complexity** of 3D viewing is caused in part by the **added dimension** and in part by the fact that **display devices** are only 2D. The solution to the mismatch between 3D objects and 2D displays is accomplished by*

introducing **projections**, which transform 3D objects onto a 2D projection plane »^[Foley&al93]. Conceptually, objects in the 3D world are **clipped** against the 3D view volume, then they are **projected**. The projection is then **mapped** into the viewport for display. This conceptual process is illustrated on the next page in **Figure 4.22**.

We do not present in details the mathematics of the projection we have used as the reader may find all needed informations about this in [Foley&al93]. Nevertheless, we can quote that we have used a **cavalier oblique parallel projection**. We have then **rotated** this projection with an angle of 45° to provide a better view to the user.

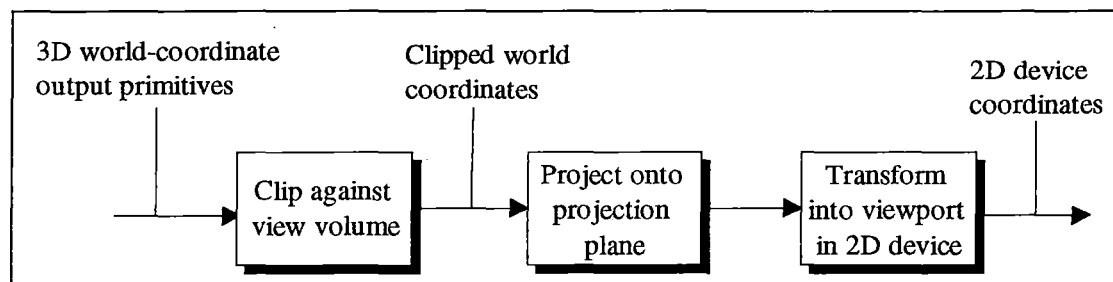


Figure 4.22 Conceptual model of the 3D viewing process^[Foley&al93]

Finally, the grayed area of the cube (as shown in **Figure 4.21**) represent the part of the image contained in a binary file that is represented by the bitmap currently displayed. If we consider **Figure 4.21**, we see that the lowest half of the cube is grayed. This means that, if the binary file contains 100 pixel planes, the bitmap displayed on the screen is a representation of the 50 first pixel planes. It also shows the orientation of the bitmap (in **Figure 4.21**, we see that orientation is along the z-axis).

3.3 *Relations of the TBMAS Program with External Applications*

In this section we describe the features we have incorporated in our program in order to facilitate its relations with other applications forming its execution environment.

A first thing to mention here is that, as already stated in chapter 3, we have kept the **Microstat ASCII format** for the files containing analysis computation results. Microstat is a statistics Windows based application which is aimed at performing statistical tests on samples and producing charts to compare different samples. This is a tool which is widely used for analyzing analysis computation results such as comparing the evolution of the Euler number for example, and it was then important to respect **compatibility** and facilitate the user's work who can switch from the **TBMAS** application to the **Microstat** application and **import** results.

Furthermore, as we said in previous section, the choice of the **Microsoft bit-map file format** was also important in this perspective. Indeed, the Microsoft BMP file format is undoubtedly the format with the **widest recognition** among Windows applications. Hence, the user can choose among a large number of applications to import image files created with the TBMAS program. For example, he could import such a file in its word processor to illustrate an article just as we did ourselves in previous section.

A last feature we have incorporated in our program is the possibility to **cut and paste charts** or **pictures** straight from our program to another application via the Windows clipboard. The **clipboard** is undoubtedly one of Windows most exciting and useful characteristics that makes it possible for different applications to exchange various data naturally [Tornsdorf&al92]. We thought that this was extremely important to implement as **data exchange** between applications is a **key aspect of modern software**.

3.4 Performances Measurements

We now present charts comparing existing program performances to the TBMAS program performances. Of course results should be tempered by the fact that measurements have **not** been made on the same computer. Existing programs performances have been tested on a PC 386 installed with a Targa 16 Raster Graphics Adapters and TBMAS performance measurements have been accomplished on a PC pentium with a Diamond Stealth 64 graphic board. All these performances measurements have been made using the same input file (Z60IM.BIN).

The figure below (**Figure 4.23**) shows a chart comparing computation time needed to perform a **MIL analysis** respectively with the existing program and with our **TBMAS** program. Indeed, the overall **gain** of time obtained by using our program may be estimated at more than **97%** (the existing program needs 5 minutes and a few seconds to perform the MIL analysis on one plane, our program only needs 10 seconds for the same task).

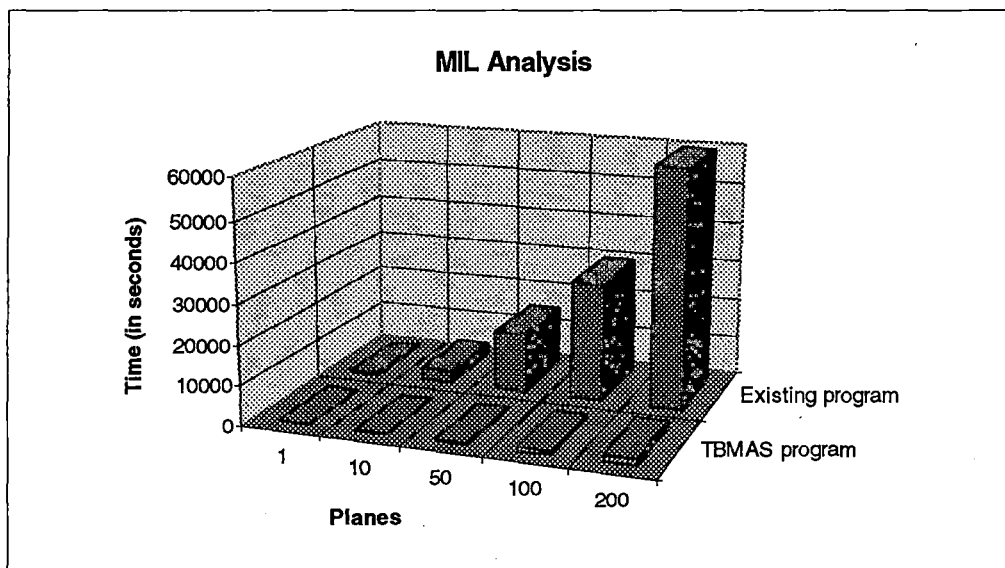


Figure 4.23 MIL analysis comparison chart

Below we show in **Figure 4.24** the comparison of computation time needed by the existing program and our program to perform the **Euler number analysis**. The overall performances increased can be estimated to more than **96%**

as the existing program needs 2 minutes 25 seconds to perform the analysis on 10 planes and our program needs only 6 seconds to perform the same task.

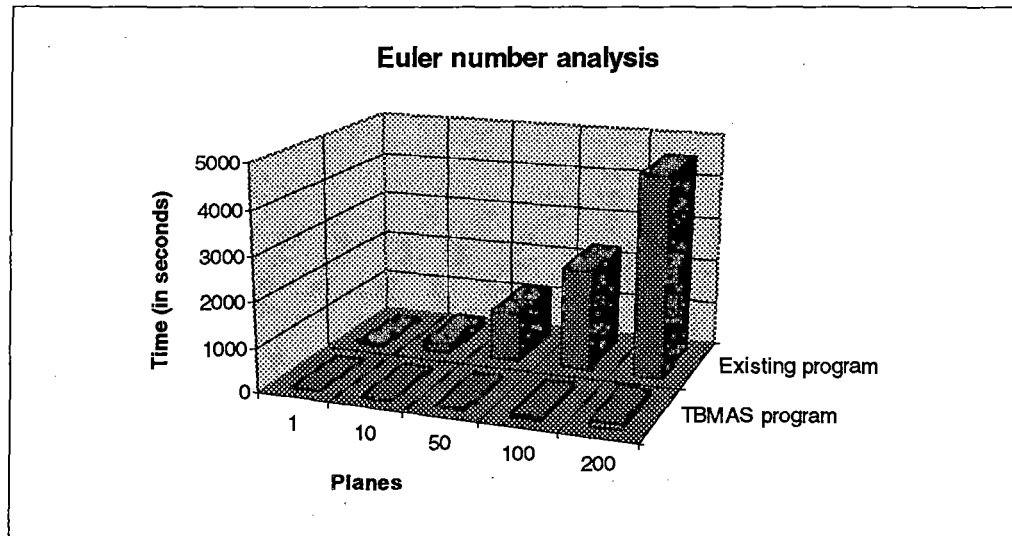


Figure 4.24 Euler number computation comparison chart

The figure below (Figure 4.25) shows a comparison of existing program performances and TBMAS performances based on time needed to perform an **osteoporosis simulation** on a complete specimen (all 200 planes). The **gain** obtained can be estimated to almost **99.5%** as the time needed to perform the osteoporosis simulation with the existing program is around 4 hours and time needed for the same task using our program is only 1 minute and 12 seconds.

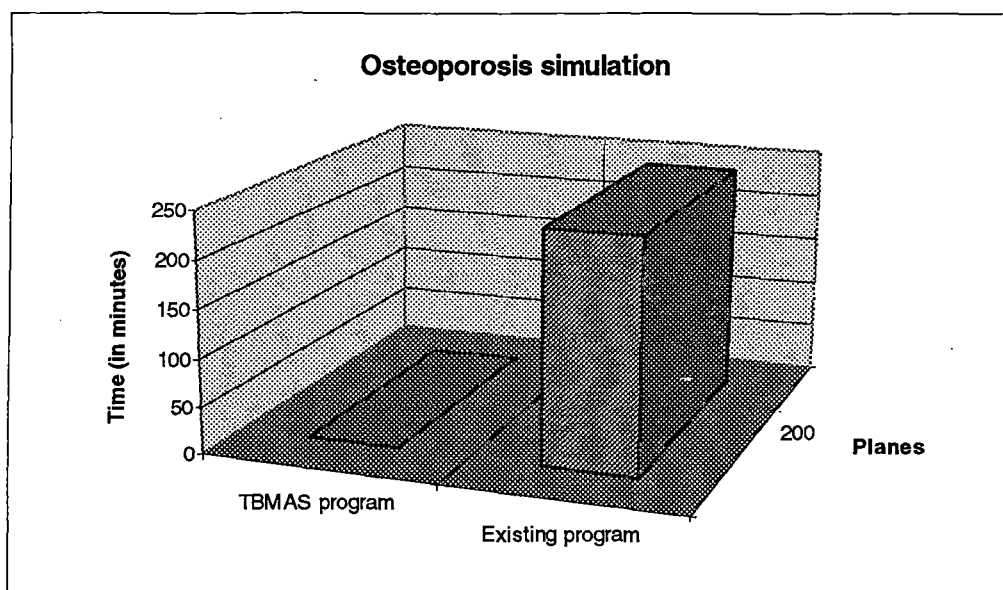


Figure 4.25 Osteoporosis simulation comparison chart

At last, we present the file size reduction obtained with different compression method in **Figure 4.26**. The gain obtained using the **PKZIP** compression utility is **65%** (original file size is 7,783,284 bytes, zipped file size is 2,725,227 bytes) which is already a good result. Nevertheless, using the unique compression method that we have discussed previously (the **LRLE** method), we obtain a gain of **89%** (LRLE compressed file size is 835,773 bytes). When combining **both** compression methods, the compression ratio obtained is truly amazing : **97%** (253,057 bytes).

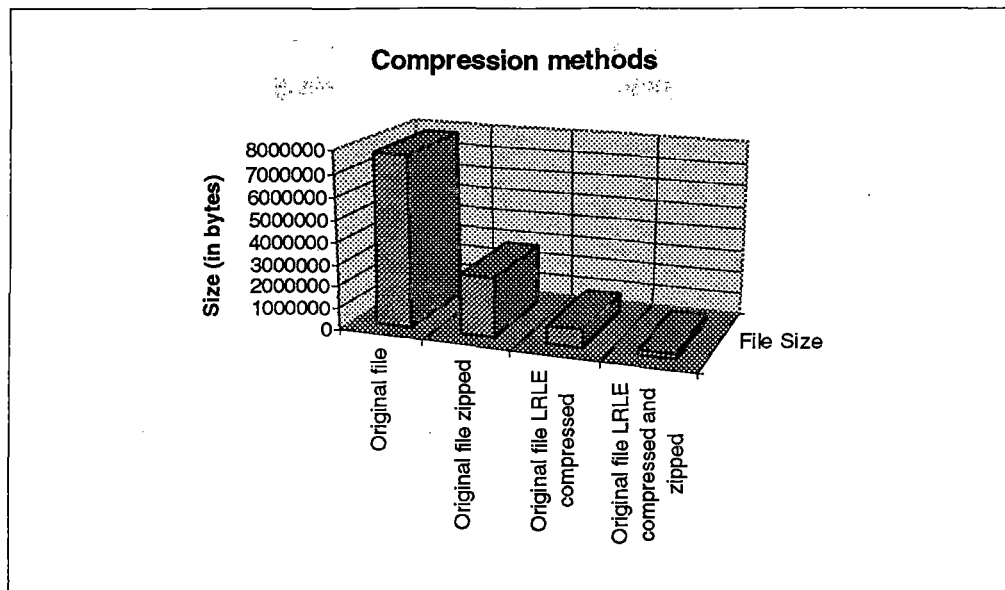


Figure 4.26 Compression methods comparison chart

3.5 An Empirical Reverse Engineering Method

In this section we discuss fundamentals of reverse engineering in the field of migrating legacy systems. We address both principles and methods based on [Brodie&Stonebraker93] and we compare with the situation to which we have been confronted and the concrete approach we have adopted.

3.5.1 Migrating Legacy Information Systems : The Problematic

Below we discuss the problematic of migrating legacy information systems and the common characteristics it has with the situation we have encountered eventhough the system on which we have worked can not be considered as an information system.

« Most large organizations are deeply mired in their information systems (IS) sins of the past. Typically, their IS are large (e.g., 10^7 lines of code), geriatric (e.g., more than 10 years old), written in COBOL, and use a legacy database service (e.g., IBM's IMS or no database management system (DBMS) at all) » [Brodie&Stonebraker93]. The system we had to work on consisted of a set of

programs²² for a total of more than 10^4 lines of code written almost 5 years ago in ANSI C along with calls to MiPS libraries²³ and poorly commented.

*« Today, legacy ISs pose one of the most serious problems for large organizations. Costs due to problems of a single legacy IS (e.g., failure, maintenance, inappropriate functionalities, lack of documentation, poor performance) can often exceed hundreds of millions of dollars per year. They are not only inordinately expensive to maintain but also **inflexible** (i.e., difficult to adapt to changing business needs), and **brittle** (i.e., easily broken when modified for any purpose) »*^[Brodie&Stonebraker93]. Eventhough the system we have been confronted with can not be considered as essential to the surviving of a business, it presents some of the mentioned above characteristics : the poor level of performances obtained, its inflexibility, its poor documentation and user interface and the fact that it is tied to a specific hardware configuration²⁴ which makes it less robust.

*« In legacy IS migration, an existing IS is evolved into a target IS by replacing the hardware and much of the software including the interfaces, applications, and databases. ... The **target environments** are intended to take maximum advantage of the benefits of **rightsized** computers, a client-server architecture, and modern software ... »*^[Brodie&Stonebraker93]. In our case, the existing set of programs has to be « migrated » into a target program, replacing the specific hardware configuration with a less specific one, replacing some of its functionalities and its interface, and taking full advantage of modern software such as MS Window, MS Visual Basic and Watcom C++.

*« A fundamental goal of legacy IS migration is that the target IS not become a legacy. ... That is, the target IS is designed to be very flexible (i.e., portable) for current and future **rightsizing** and to avoid becoming a future legacy IS »*^[Brodie&Stonebraker93]. When developing our target program, we have tried to guarantee an easy maintenance and to make platform migration easier. Hence,

²² Discussed in chapter 3

²³ See chapter 3

²⁴ See chapter 3

we have chosen to adopt a software architecture²⁵ where the interface and the functionalities are separated layers, along with an Object Oriented approach which guarantees program modularity.

3.5.2 Strategies for Migrating Legacy IS

We discuss now common strategies used to resolve the problem of migrating IS, then we describe the empirical method we have used in our case and how it can be compared to the **cold turkey** and **chicken little** strategies.

« *Cold turkey involves rewriting a legacy IS from scratch to produce the target IS using modern software techniques and hardware of the target environment* »^[Brodie&Stonebraker93]. This strategy carries substantial risk of failure according to [Brodie&Stonebraker93], we discuss below some of them.

- « *A better system must be promised* »^[Brodie&Stonebraker93]. Management will rarely budget the required major expenditure needed for such a project if the only payoff is to lower future maintenance costs.
- « *Specifications rarely exist* »^[Brodie&Stonebraker93]. Most of the time, the only documentation for legacy IS is the code itself and the original coding practice may now seem primitive. Documentation in itself is often non-existent or has been lost for a long time.
- « *Management of large projects is hard* »^[Brodie&Stonebraker93]. Most of the large projects are hard to manage and few organizations are capable of managing the development of a complex IS.
- « *Lateness is seldom tolerated* »^[Brodie&Stonebraker93]. Large projects are often late due to the above problems and some more. Thus partly or mostly completed projects are canceled due to time constraints.
- ...

« *Cold turkey involves high risk. It has been applied and has failed many times in large organizations* »^[Brodie&Stonebraker93]. We now discuss the alternative strategy referred as **chicken little**.

²⁵ See section 3.1

« *Chicken little involves migrating the legacy IS, in place, by small incremental steps until the desired long term objective is reached. Each step requires a relatively small resource allocation (e.g., a few person years), a short time, and produces a specific, small result towards the desired goal.* »

[Brodie&Stonebraker93]. This strategy presents a sharp contrast with the **cold turkey** strategy as it does not require the vast resource requirements of a complete rewrite. More, if a « *chicken little step fails, only the failed step must be repeated rather than the entire project* » [Brodie&Stonebraker93]. Hence, **chicken little** is a much safer strategy than **cold turkey**.

When implementing the **chicken little** strategy it is necessary to consider a decomposable structure for the IS architecture. In such structure, the interface, the application and the database services are considered as being different components with specific interfaces. The **Figure 4.27** illustrates a decomposable IS structure. It consists of a set of **application modules** (Mi) that interact with a **data base service**. Each application module might also have its own **user interface** (Ui) and a **system level interface** (SIi) through which it interacts with other information systems.

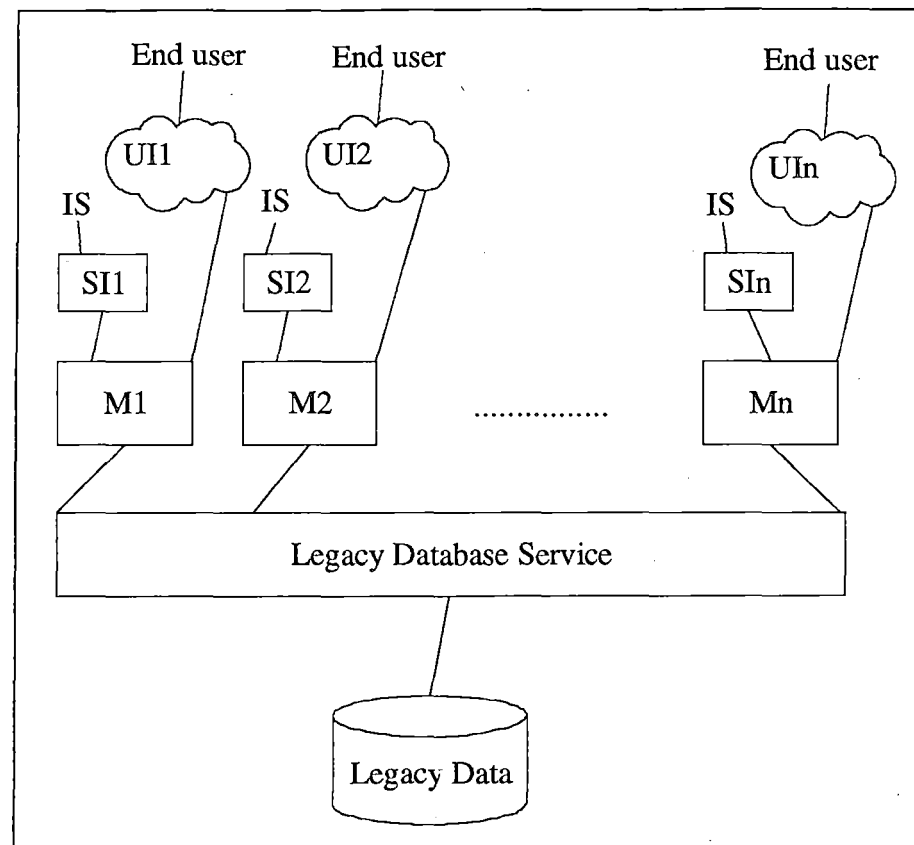


Figure 4.27 Decomposable legacy IS architecture^[Brodie&Stonebraker93]

« A *chicken little* legacy IS migration involves iteratively selecting and migrating part of the legacy IS to become new parts of the iteratively constructed target IS »^[Brodie&Stonebraker93]. Thus, during the legacy migration process, the legacy IS and the target IS form a « composite » IS. In such migration, the concept of **gateway** is essential as it affects the complexity or simplicity of the migration architecture. « By **gateway** we mean a software module introduced between operational software components to mediate between them »^[Brodie&Stonebraker93]. Its main role is to insulate some of the legacy IS components from the changes being made to others. The **gateway** « maintains the interface that the UI expects of the legacy IS even though the legacy IS is being changed *behind the scenes* »^[Brodie&Stonebraker93]. Gateways also serve to translate requests and data between the system components (i.e., a data base gateway should hide to application modules the fact that the data base application has been changed). The concept of gateway is illustrated below in **Figure 4.28** where it makes any change to the legacy IS **transparent** to the legacy

user interface (UI). This means that it maintains the interface that UI expects even though the IS has **already** been changed. Further more, as the target graphical user interface (GUI) is introduced, the gateway makes **transparent** which of the IS or the target IS are supporting particular functions.

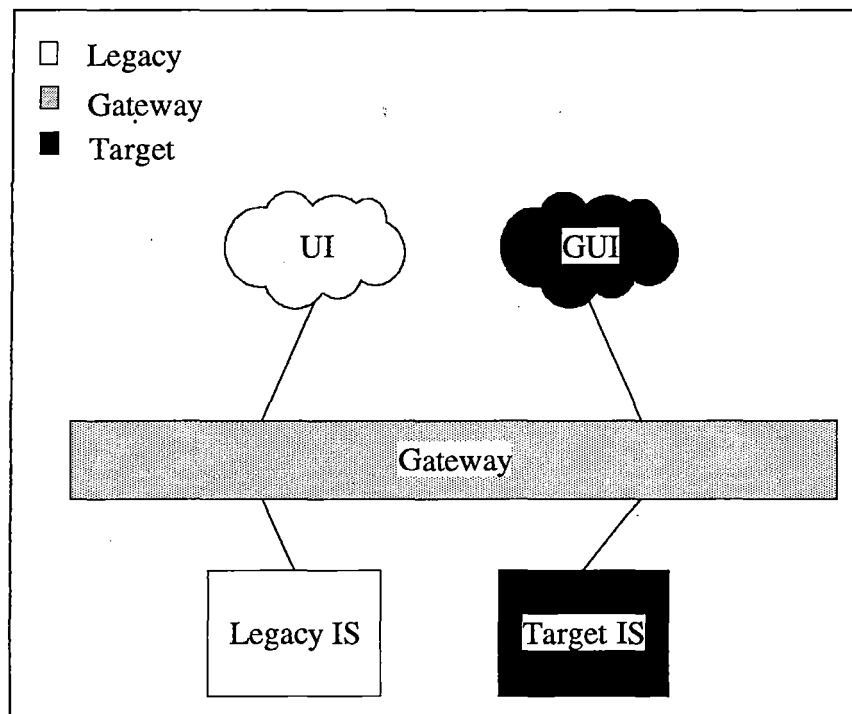


Figure 4.28 IS migration architecture [Brodie&Stonebraker93]

We now address the method we have empirically developed and used for our program. We say empirically because when confronted to the problem of reverse engineering we had to cope with, we have tried to develop our own strategy. Indeed, it can be compared to the **chicken little** strategy as it consists of incremental steps towards the desired goal. The only **restriction** to full correlation between the chicken little strategy and our strategy is that we **have not used gateways**. Indeed, the reverse engineering problem we were faced to did not required gateways as its **complexity can not be compared** to a **large organization IS migration complexity**. We discuss below the strategy we have developed and why we have done so.

Our task was not really to migrate the existing set of programs but rather to rewrite from the scratch a new program. Nevertheless we were confronted

with problems which are typical reasons for the **cold turkey** strategy to fail : we had to promise a better system, the only documentation consisted of C code poorly commented, ... We had then the idea to base our work on the only thing we had : the existing code, thus we decided to « migrate » the existing system whose architecture is represented below in **Figure 4.29**.

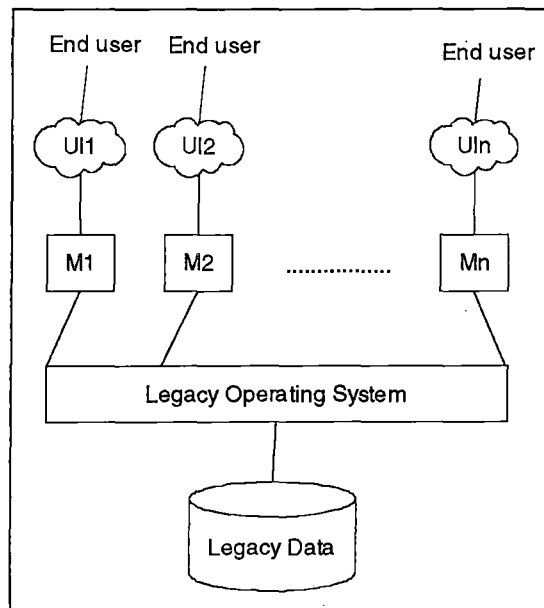


Figure 4.29 The architecture of the existing system

To undertake the **migration** of the above system, we had to perform this task in **several steps**. Indeed, this method has imposed it on us as being evident and the only solution to face the complexity of the problem. The migration process that we have realized is composed of four important steps that are described below :

1. Migration of the **application modules** to bypass hardware restrictions that have been described in chapter 3.
2. Migration of the **operating system tied modules** to provide support for the target operating system (MS Windows).
3. Migration of the **legacy data** by providing new data structures such as the compressed files that were discussed in this chapter.

4. Migration and federation of the **legacy interfaces** to provide an unified graphical interface taking full advantage of the MS Windows possibilities instead of a set of independent textual interfaces.

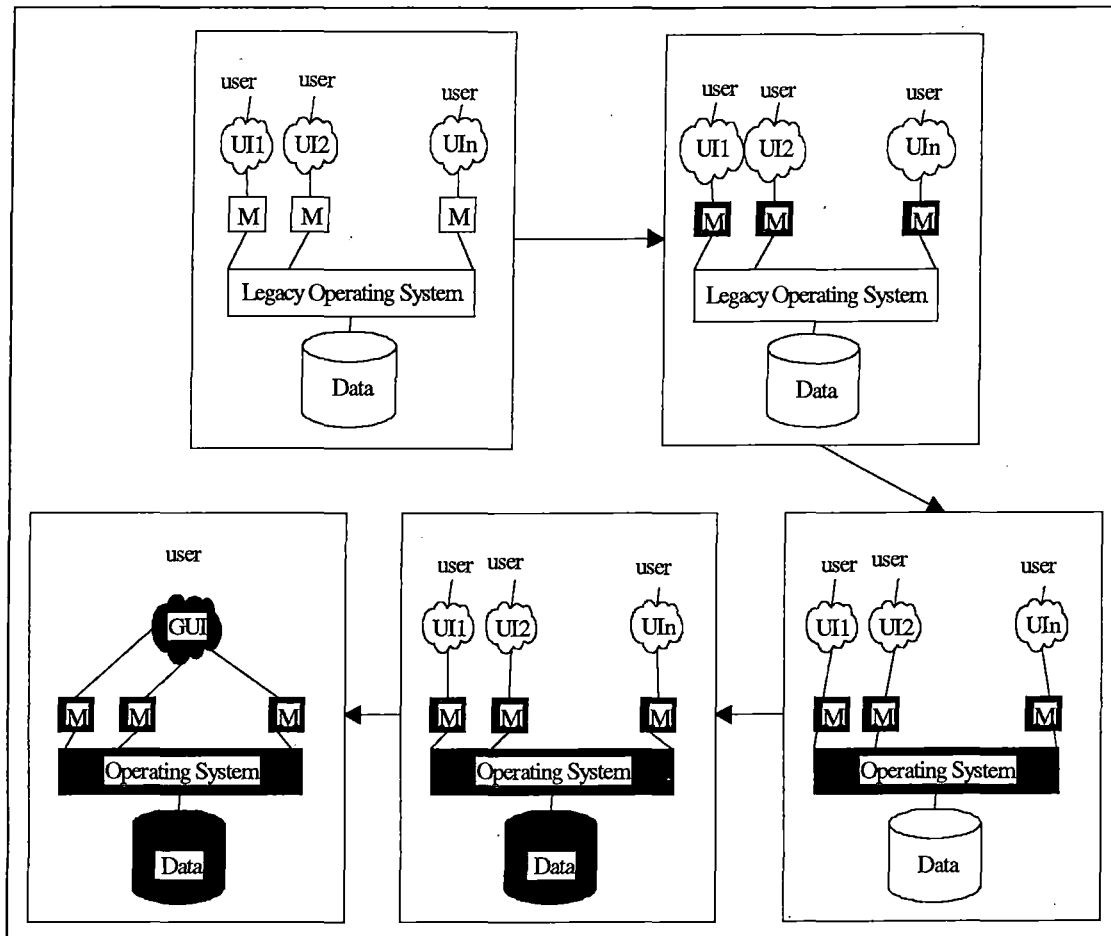


Figure 4.30 The migration process of the existing set of programs

The above figure (**Figure 4.30**) shows an illustration of the four steps method that we have detailed previously. In this figure, we use the same symbolic as in **Figure 4.28** where white modules represent legacy system modules and black modules represent target system modules.

4. Conclusion

First, we would like to recall here **requirements** that we have been asked to fulfill for the purpose of showing that, when developing our application, we have kept these in mind.

The first requirement was to free the program from a **specific configuration**. We think that we have reached this objective as the configuration needed to run the TBMAS program is a **standard MS Windows configuration** (see **Table 4.1**). In this context we have also been asked to try to make a **possible platform migration easier**. Even though it would not be true to declare our program platform independent, we think that the **software architecture** (see section 3.1) we have developed guarantees an easier migration. Indeed, interface aspect is mostly tied to the underlying platform but functionalities ; because they are written in C++, do not suffer completely from the same limitation. In conclusion, **separating** the interface and the functionalities, using an **Object Oriented** approach and C based programming language should guarantee a possible easier platform migration.

Then we have been asked to implement **functional requirements**, that is to say to provide a set of functions that would enable the user to perform **analysis**, to **examine analysis results** and to **display three-dimensional representations** of trabecular bone specimen. We think that we have fulfilled these requirements as we have implemented the **three analyses** we were asked for (MIL analysis, Euler number analysis and osteoporosis simulation), we have implemented the possibility to draw **charts** based on these analysis results and the creation of **bitmap** images that show a three-dimensional representation of a bone specimen. A last functional requirement was to equip our program with a **user-friendly interface**. Once more we think that we have reached our goal by implementing our program as an **MS Windows application** which is the de facto standard in PC user-friendly interfaces, and also by taking care to

use **ergonomic recommendations** found in [Sacre&al92], [Vanderdonckt92], [JVD95] and [JVD93].

At last, as far as requirements are concerning, we were asked to **increase** the level of **performances** obtained with existing programs. We personally think that we have **exceeded all our expectations** in this domain. Indeed, as discussed in section 3.4, we have improved MIL analysis performance with a **97%** ratio²⁶ (this means that our program uses only **3%** of the original time to perform the same task), Euler analysis performance with a **96%** ratio and osteoporosis performance with a **99.5%** ratio. Further more, we have developed and implemented a unique compression method (see section 3.2.3) which allows a **89%** storage space gain and that can be combined with traditional compressor such as PKZIP to obtain a gain up to **97%**. And finally for these requirements, we have respected the constraints to use both MS Visual Basic and Watcom C++ as programming tools.

To conclude this chapter, we want to say that the **development process** that has been described here and that we have followed can be considered as an **empirical development process originated** in the **methodological background** we have received during our studies. Thus the **layer architecture** we have developed, separating the user interface from the functionalities, and the utilization of an **Object Oriented** approach to guarantee a better modularity. Indeed, we have mainly developed our user interface using **prototypes** and it would be interesting to consider an **alternative approach** and methodology in order to analyze what might be its **contribution** regarding our methodology. Hence the next chapter where we develop the **user interface analysis** using the **TRIDENT** methodology.

²⁶ Even if, as stated in section 3.4, the computers on which both measurements have been conducted are different, we do not think this is a crucial factor as the existing program used extensively the display capacity of the Targa board and our program uses extensively memory. Hence, we do not think that performances achieved by the existing program on a powerful machine would increase dramatically, on the contrary, our program using memory extensively, can take full advantage of powerful machines.

CHAPTER 5

Another Approach for the Development Process : the Application of the TRIDENT Methodology

In this chapter, we discuss another Development Process based on the methodological framework developed in the TRIDENT project and explained in [BODART95].

<i>1. Introduction</i>	<i>2</i>
<i>2. The Methodological Framework</i>	<i>3</i>
<i>3. Graphical User Interface Specifications</i>	<i>5</i>
<i>4. Presentation Design</i>	<i>39</i>
<i>5. Derivation of the Software Architecture</i>	<i>72</i>
<i>6. Dialogue Specifications</i>	<i>93</i>
<i>7. Transformation of the TRIDENT Architecture in Visual Basic</i>	<i>107</i>
<i>8. Conclusion on the Application of the TRIDENT Methodology</i>	<i>113</i>

1. Introduction

As the reader has seen in the previous chapter, we did not use the TRIDENT Methodology to develop and design our application, but we used an empirical method originated in the methodological background we have received during our studies.

It is only in the beginning of August 1995 that Mr. Bodart, our Professor, gave us some directions concerning the state achieved by the Trident Methodology. Mr. Bodart also spoke about the advantages we could derive from using it. That is why, he asked us to complete the application of that methodology to our thesis. That is what we have tried to do in the following sections of this chapter.

2. The Methodological Framework

« The methodological framework developed on the TRIDENT project had to be informal. Task analysis, model specification and tool use are all overlapped within the framework of a methodology that supports a wide range of design situations and also provides a design space. The aim is to provide a general framework that covers various interaction styles (e.g., menu selection, form filling, command language, direct manipulation, multi-windowing) and dialogue structures (e.g., synchronous, asynchronous, multi-threaded dialogues). » ^[BODART95]

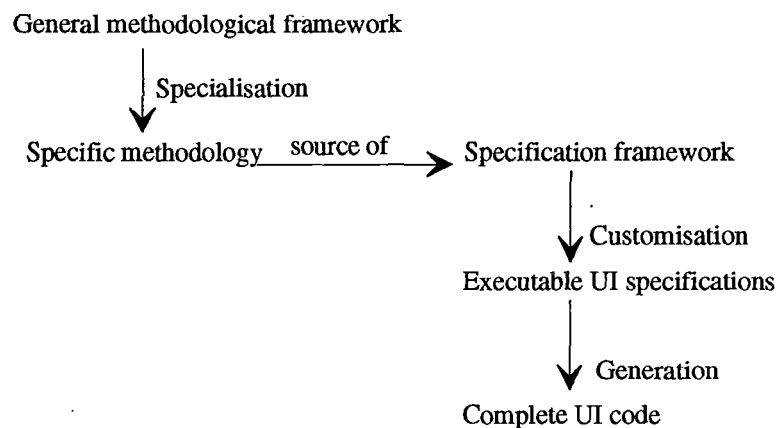


Figure 5.1 Structure of TRIDENT methodology

« This general framework can be specialized to produce a specific methodological framework. The specialization produces a design space that is specific to a given class of problems. This design space replaces the full range of different groups of user interface design options with the narrower choice of one group of option (Figure 5.1). The specific methodology can also be used to generate a specification framework. This in turn can be customized to produce executable UI specifications allowing to generate the code of the interactive application (Figure 5.1). » ^[BODART95]

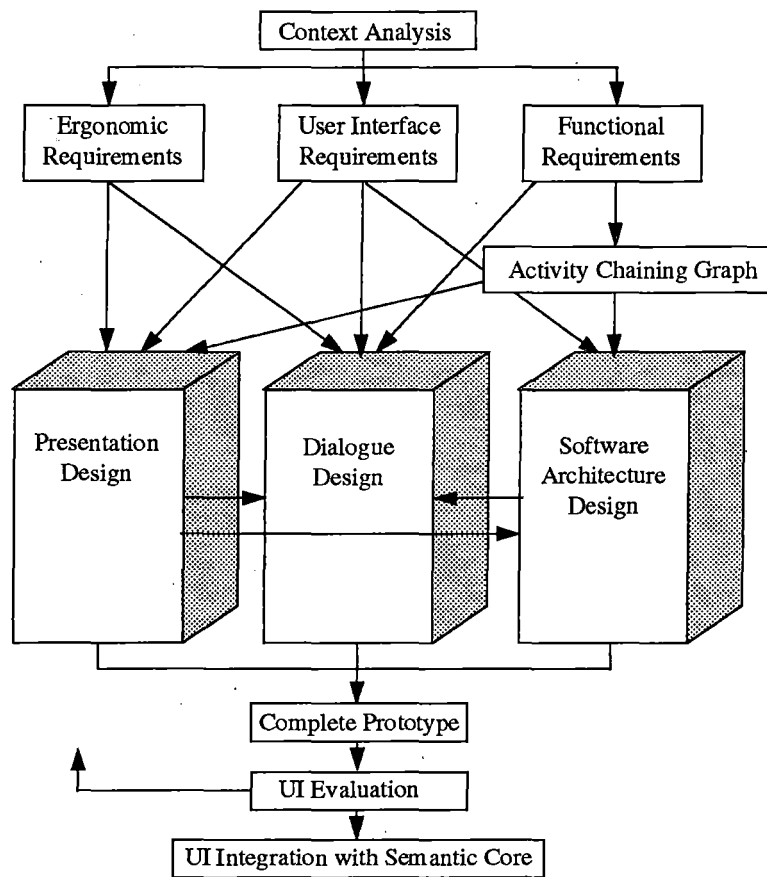


Figure 5.2 Structure of TRIDENT methodological framework.

Figure 5.2 outlines the main parts of TRIDENT methodology. The structure of TRIDENT methodological framework is due to five key activities. Four of these five activities are applied in the following sections to analyze our program:

1. forming UI specifications from the output of task analysis (Section 3) ;
2. guiding presentation design using ergonomic rules (Section 4) ;
3. deriving a software architecture from task analysis and presentation components (Section 5) ;
4. forming high level dialogue specification from the output of task analysis (Section 6) ;

3. Graphical User Interface Specifications

Starting from an extensive context analysis (task analysis, user stereotypes and workplace description), we will specify UI components from this contextual information (Figure 5.3).

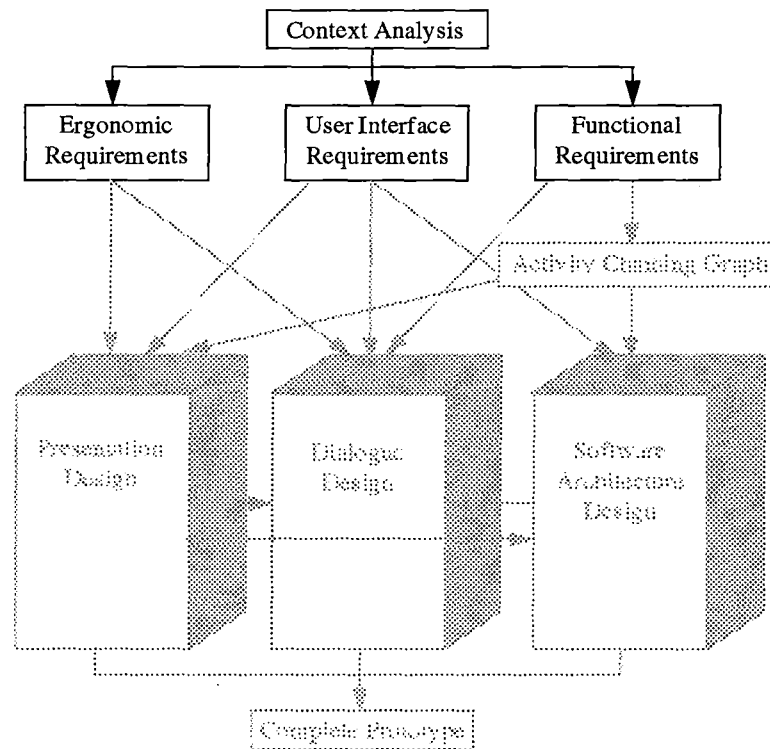


Figure 5.3 Outline of Activity 1.

3.1 The Task Analysis

« A **task** is defined as any activity carried out by an operator (often named user) that results into a significant state change in a given activity domain in a given contextual situation. »^[BODART95]

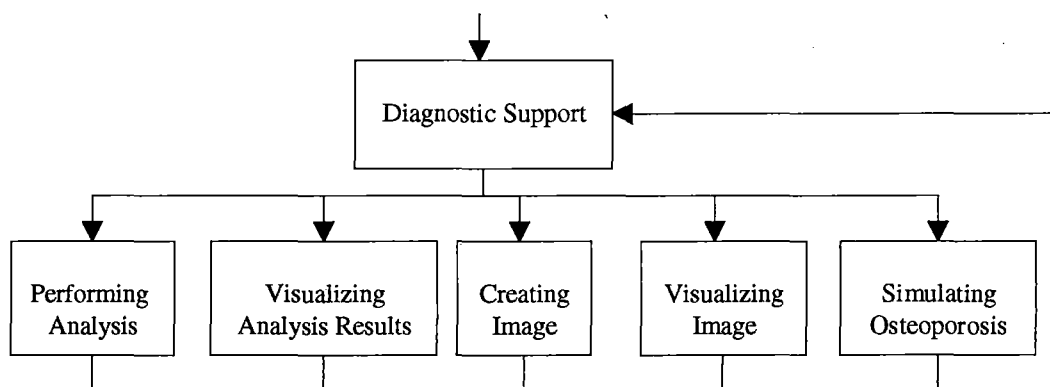
The task which is underlying to the use of the **TBMAS** software (or **Trabecular Bone and Morphological Analysis System**) is the obtaining of a **diagnostic support** as part of a morphological analysis or of a trabecular bone analysis. This diagnostic support system is presented to the user in a **toolbox** form.

This toolbox allows the user to perform analyses and simulations as well as to create image representations of bones. Those manipulations are performed on binary files which result of digitalization of bone specimen as explained in the Appendix B.

To perform the task analysis we will retain the **toolbox metaphor**. For the user of such a toolbox, it should be possible to use those tools without specified order. It is, However, obviously evident that it is impossible to drive in a nail without a hammer. So, what we mean is that it is difficult to visualize analysis results if no analyses were performed before as well as to visualize an image if it was not previously created. Nevertheless, as soon as the user has performed at least one analysis and created at least one image, he can use the tools as he wants.

The following task analysis will be a user-oriented task analysis. In other words, we will try to give the vision of the interactive sub-tasks the user will need to execute.

The following schema (**Schema 5.1**) globally describes the user's task :

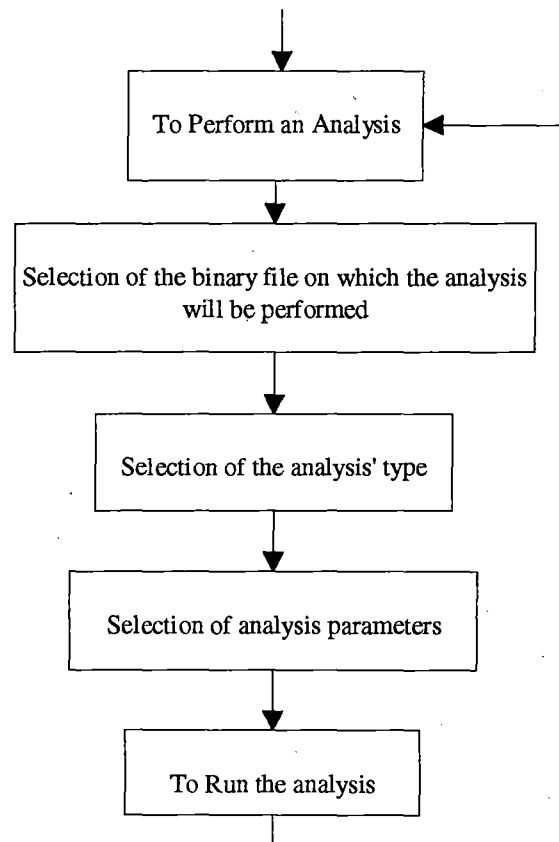


Schema 5.1 Task Analysis

We will now open one by one each box of this schema to improve it.

3.1.1 The Box Labeled : « Performing Analysis »

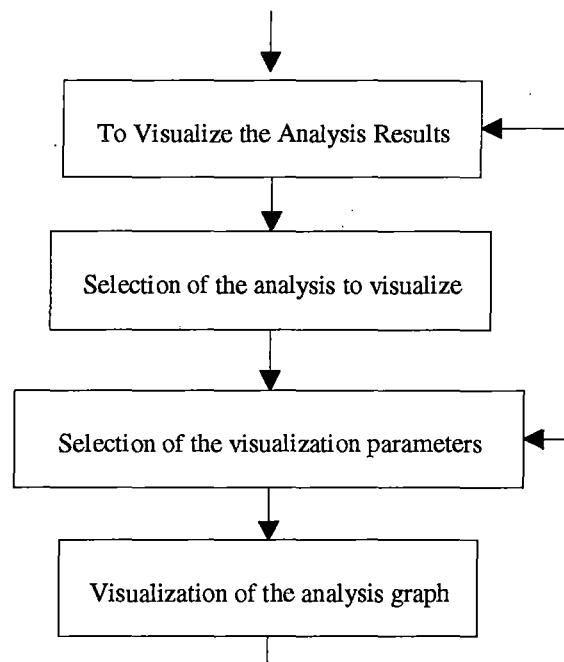
This sub-task can be globally described with the following schema (Schema 5.2) :



Schema 5.2 Analysis of the sub-task "Performing Analysis"

3.1.2 The Box Labeled : « Visualizing Analysis Results »

This sub-task can be globally described with the following schema (Schema 5.3) :



Schema 5.3 Analysis of the sub-task "Visualizing Analysis Results"

This schema can also be improved by opening the box labeled « Selection of the analysis to visualize ». Indeed, this selection process is a two-phase selection :

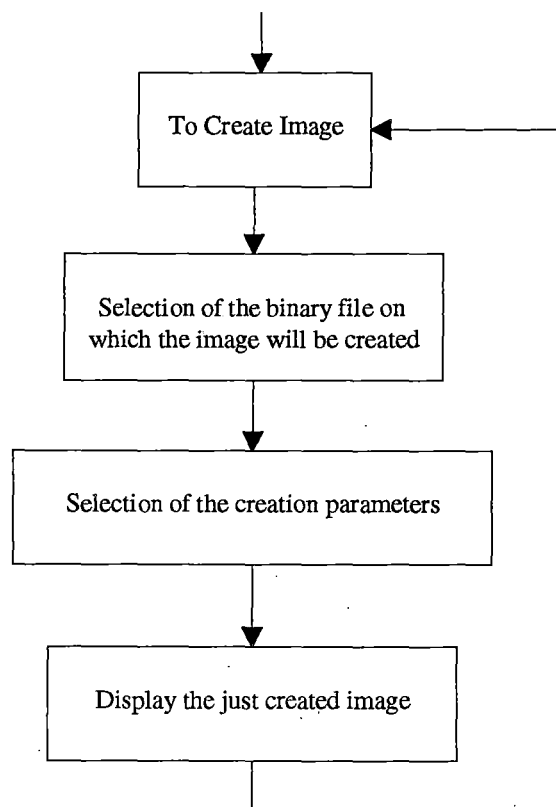
- first, the user must choose an analysis index file corresponding to a binary file and containing information about the analyses performed on this binary file,
- next, the user must choose in this index file, the concrete analysis he wants to visualize.

For example : for the following binary file : *file.bin* there could exist two kinds of analyses index file corresponding to the two kinds of analyses supported by the system : MIL analysis and the Euler Number analysis. Those two index files would be *file.mix* for the MIL analysis index file and *file.eix* for the Euler Number analysis index file. If the user wants to visualize a MIL analysis

then he should first select the MIL analysis index file and next, choose the MIL analysis file he wants to visualize among other MIL analysis files performed on the same binary file.

3.1.3 The Box Labeled : « Creating Image »

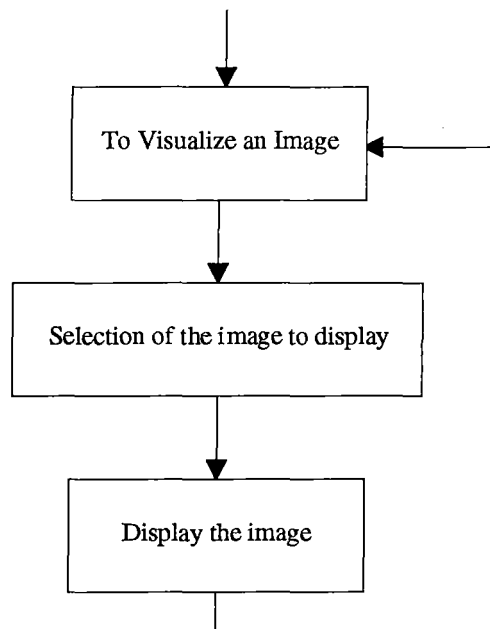
This sub-task can be globally described with the following schema (Schema 5.4) :



Schema 5.4 Analysis of the sub-task "Creating Image"

3.1.4 The Box Labeled : « Visualizing Image »

This sub-task can be globally described with the following schema (**Schema 5.5**) :



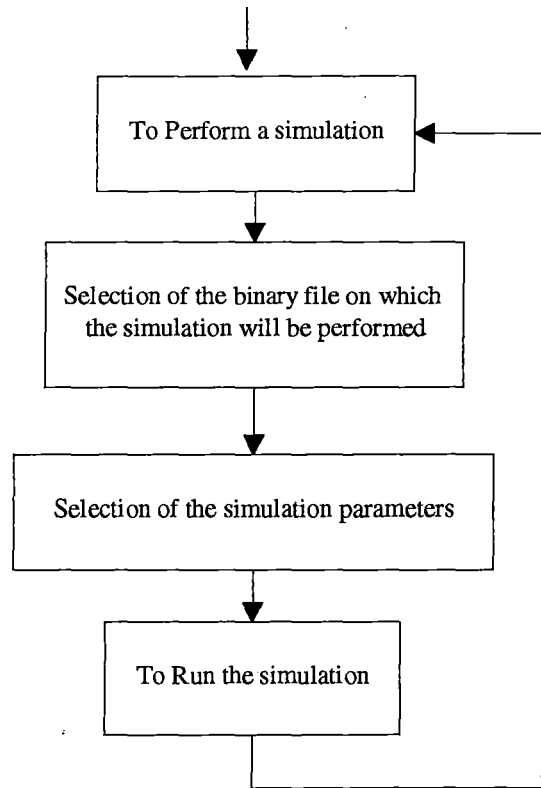
Schema 5.5 Analysis of the sub-task "Visualizing Image"

This schema can also be improved by opening the box labeled « Selection of the image to display ». Indeed, this selection process is also a two-phase selection :

- first, the user must select the Images index file corresponding to a binary file and containing information on all the images created for this binary file,
- next, the user must choose the image file he wants to display.

3.1.5 The Box Labeled : « Simulating Osteoporosis »

This sub-task can be globally described with the following schema (Schema 5.6) :



Schema 5.6 Analysis of the sub-task "Simulating Osteoporosis"

3.2 Relations between the Sub-tasks

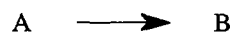
Some dependence relations exist among the sub-tasks explained above and we will try to show and explain those dependence relations in this point. Now we will explain the new notation used in the following Schema 5. 7.

- the box :

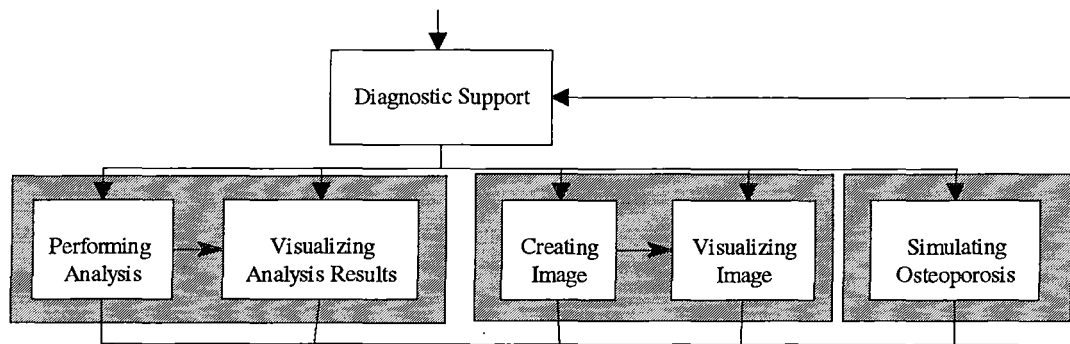


is used to show the dependence between sub-task

- the arrow :



is used to translate the fact that B can not be executed before A



Schema 5.7 Relations among Sub-tasks

The first filled-box translate the fact that the « Performing Analysis » sub-task and the « Visualizing Analysis Results » sub-task are dependent each other. Indeed, as we previously explained, it is evident that the user can not visualize the results of an analysis if none analysis was performed before (this fact is translated on the schema by the arrow). The particularity of this dependence is that it isn't always true. As a matter of fact, this dependence disappear after the user create at least one analysis and will never appear again after this because it will always be possible to visualize previous analysis' results.

What about the second filled-box ? It is the same reasoning. It is not possible to visualize an image if none was previously created. And as soon as one image is created then this dependence disappear.

The third filled-box just translate the fact that this sub-task is independent of the four other sub-tasks.

3.3 Description of the Interactive Task

In this point we will first describe each parameter of the interactive task as developed in the TRIDENT project and explained in [TRIDENT93]. And next, we will apply those parameters to the underlying task of the TBMAS software.

3.3.1 Description of the Parameters

3.3.1.1 Prerequisites of the Task

This parameter can take the *minimal*, *moderate* or *maximal* value. « *The prerequisites of the task express the amount of knowledge of the existing or future system that the user must be master of to use the system in an appropriate and effectiveness way* »^[TRIDENT93].

3.3.1.2 The Productivity of the Task

This parameter can take the *low*, *mean* or *high* value. « *The productivity of the task represent the mean execution's number of the task per unit of time. It could also include the frequency of utilization* »^[TRIDENT93].

3.3.1.3 The Objective Environment of the Task

This parameter can take the *existent* or *non-existent* value. « *The environment of the task translate the presence or the absence of specific objects which are manipulated by the user during the accomplishment of the task and this, independently of every system. Those objects can be paper-documents, files, records, and so on ...* »^[TRIDENT93].

3.3.1.4 The Reproducibility of the Environment

This parameter can take the *practicable* or *impracticable* value and is linked with the previous one. « *The reproducibility of the environment of a task is practicable or not as much as this environment exist. The reproducibility of the environment is practicable if it can be transposed in the limit of the system* »^[TRIDENT93].

3.3.1.5 The Structuration of the Task

This parameter can take the *low*, *moderate* or *high* value. « *The structuration of the task explain the degree of freedom or of constraint the user must face accomplishing the task* »^[TRIDENT93].

3.3.1.6 *The Importance of the Task*

This parameter can take the *low*, *moderate* or *high* value. « *The importance of the task inform about the fundamental, vital, crucial character of a task* »^[TRIDENT93].

3.3.1.7 *The Complexity of the Task*

This parameter can take the *low*, *moderate* or *high* value. « *The complexity of the task represent the degree of cognitive, intellectual complexity attached to the sub-tasks and actions needed to accomplish the task* »^[TRIDENT93].

3.3.2 *Application of the Parameters*

3.3.2.1 *Prerequisites of the Task*

The prerequisites of the task are moderate. Indeed, a good knowledge of the Microsoft Windows environment and of the mouse's use are the only two prerequisites to obtain a good utilization of the system.

3.3.2.2 *The Productivity of the Task*

We think that the productivity should be divided in two ideas :

- the frequency and,
- the contribution of the task in the global context of the user (that's what we call the Productivity in the strict meaning of the word).

Thus, the frequency of utilization strongly depend of the global context in which the user find himself. Indeed, if this one is a professor or a researcher, the frequency of utilization and the productivity could be higher. Else, if the user is a student, then the frequency of utilization will be probably lower quite as much as the productivity.

3.3.2.3 *The Objective Environment of the Task*

This environment does not exist under a form that the user can directly manipulate. The user is induced to use the set of tools on binary files which are the digital representation of bone specimens. But, it is impossible for him to directly manipulate those specimens.

3.3.2.4 The Reproducibility of the Environment

As we said for the previous parameter, the environment of the task does not exist under a directly manipulable form. That's why we can say that the reproducibility of the environment is impracticable.

3.3.2.5 The Structuration of the Task

The structuration of the task is moderate. Indeed, as we said above when speaking about the ToolBox metaphor, it is obviously evident that the user is unable to visualize analysis' results if no analyses were performed before as well as for the image. Nevertheless, as soon as at least one analysis is performed and as soon as at least one image is created, then the user can structure his task as he wants.

3.3.2.6 The Importance of the Task

We think that the importance of the task is low. Indeed, we think so because if the task is not correctly executed, then the user will restart it without any contrariety.

3.3.2.7 The Complexity of the Task

We think that the complexity of the task is moderate in the sense that the user must use the set of tools in an intelligent way to obtain the desired diagnostic support. It is evident that the user must have some basic-knowledge in biomedical engineering.

3.4 Description of User Stereotypes

In this point, we will first describe each parameter of the user stereotypes as developed in the TRIDENT project and explained in [TRIDENT93]. And next, we will apply those parameters to the user stereotypes extracted of our context analysis.

3.4.1 Description of the Parameters

3.4.1.1 *The Experience of the Task*

This parameter can take the *elementary, moderate* or *rich* value. « *This parameter explain the syntactic and semantic knowledge of the user relative to the task* »^[TRIDENT93].

3.4.1.2 *The Experience of Systems*

This parameter can take the *elementary, moderate* or *rich* value. « *This parameter give an idea of the user experience's level concerning the utilization of an information system and this, even if this system is dedicated to the task or not* »^[TRIDENT93].

3.4.1.3 *The Motivation*

This parameter can take the *low, moderate* or *high* value. « *The motivation translate the psychological attitude of the user facing the task to realize* »^[TRIDENT93].

3.4.1.4 *The Experience of a Complex Interaction Mean*

This parameter can take the *elementary, moderate* or *rich* value. « *This parameter translate the aptitude and the experience of a user to manipulate one or more complex interaction means such as an extended keyboard, a complex control panel, ...* »^[TRIDENT93].

3.4.2 Application of the Parameters

In this point, we will first characterize the users globally and then explain the two stereotypes extracted among the users' population.

The users :

- are made sensitive to the use of computer either by their studies or by their jobs ;
- have already used other computer-systems or software ;

- (some of them) have already used others Diagnostic Support System relative to morphological analyses ;
- are restricted to professors, researchers and students in biomedical engineering.

The two relevant stereotypes of users are :

- **beginner users**
- **advanced users**

This qualification is relative on the one hand to the system's frequency of use and on the other hand to the knowledge of the task, and thus to the **Productivity**. To obtain more information about the Productivity of the task, the user can refer to the point 3.3.1.2.

3.4.2.1 *The Beginner Users*

Those users are new researchers and students in biomedical engineering. They will use the system in an irregular and informal manner and this, in function of their labs, and so on ... They will form themselves progressively to the utilization of the system. The parameters are the following :

- **Experience of the Task** : *elementary* because they know the task thanks only to their courses and their first experiments.
- **Experience of Systems** : *moderate* because we consider that they are already initiated to the use of a computer system.
- **Motivation** : *from low to moderate* because they are in a double phase of apprenticeship during which one they must first discover the system but also discover the task.
- **Experience of a Complex Interactive Means** : *moderate* because we consider that beginner users know how to manipulate the mouse and the keyboard in a satisfactory manner.

3.4.2.2 *The Advanced Users*

Those users are professors and researchers in biomedical engineering. they know perfectly the task and they are induced to use the system for their researches. The parameters are the following :

- **Experience of the Task** : *rich* because they have acquired their own effective task with their short-cuts and habits of the system.
- **Experience of Systems** : *from moderate to rich* because their experience's level in the utilization of systems dedicated to the task or not is of a good level.
- **Motivation** : *high* because they are master of the task and the system.
- **Experience of a Complex Interaction Means** : *from moderate to rich* because we consider that they know how to manipulate the mouse and the keyboard in a very satisfactory manner.

3.5 Workplace Description

In this point, we will first describe each parameter of the workplace analysis as developed in the TRIDENT project and explained in [TRIDENT93]. And next, we will apply those parameters to the workplace of our case.

3.5.1 Description of the Parameters

3.5.1.1 Type of Treatment

This parameter can take the *mono-treatment* or *multi-treatment* value. « *This parameter specify if the user working on the considered workplace can only execute the considered task (mono-treatment) or if he can also execute other tasks on the same workplace (multi-treatment)* »

3.5.1.2 Capacity of Treatment

This parameter can take the *low*, *moderate* or *high* value. « *This parameter take into consideration the level of interruptibility, parallelism, concurrence, interpenetrability of the tasks executed on the considered workplace. The value of this attribute is closely correlated with the previous one* »^[TRIDENT93].

3.5.2 Application of the Parameters

3.5.2.1 Type of Treatment

The type of treatment is a **multi-treatment**. Indeed, the user can easily execute other tasks on the same workplace because this one is not dedicated only to the considered task. Hence, the workplace dispose of a multi-windows user interface (Ms-Windows) allowing the user to execute several task at the same time.

3.5.2.2 Capacity of Treatment

The capacity of treatment is **low to moderate**. Indeed, the task is very greedy for treatment capacity and monopolize a big amount of the workplace resources in such a way that the parallelism is almost impossible.

For example : to perform a MIL analysis on a moderate size binary file ($\pm 7M$) it takes several hours during which the workplace is monopolized. But, when visualizing the results of an analysis, the user can easily switch to his word processor or other task.

3.6 Expressing the Product of Task Analysis

In this point, we will express the product of task analysis as developed in the TRIDENT project and explained in [BODART95] by writing an **Object-Oriented Entity-Relationship Model** and identifying the **Semantic Functions of the application** before the construction of an **Activity Chaining Graph** and before the **Derivation of Dialogue Attributes**.

3.6.1 The Object-Oriented Entity-Relationship Model

As explained in the TRIDENT project : *« Task analysis not only produces task objects and actions, but also identifies relationships between objects and relationships between actions. Objects produced by task analysis and their relationships are further specified in a schema based on an object-oriented entity-relationship model. It describes not only simple entities and semantic*

n-ary relationships with attributes, but also complex entities defined as aggregates of entities and/or semantic relationships between entities. Similarly, any attribute from any entity or relationship can itself form a new entity. This schema is supposed to fill the gap between the cognitive universe of the task analyst and the conceptual world of the software engineer. This schema can be obtained as follows :

- entities and attributes come from the list of task objects with their relationships;
- relationships come from object relationships: they include inheritance, aggregation, and semantic relationships of the activity domain. »^[BODART95].

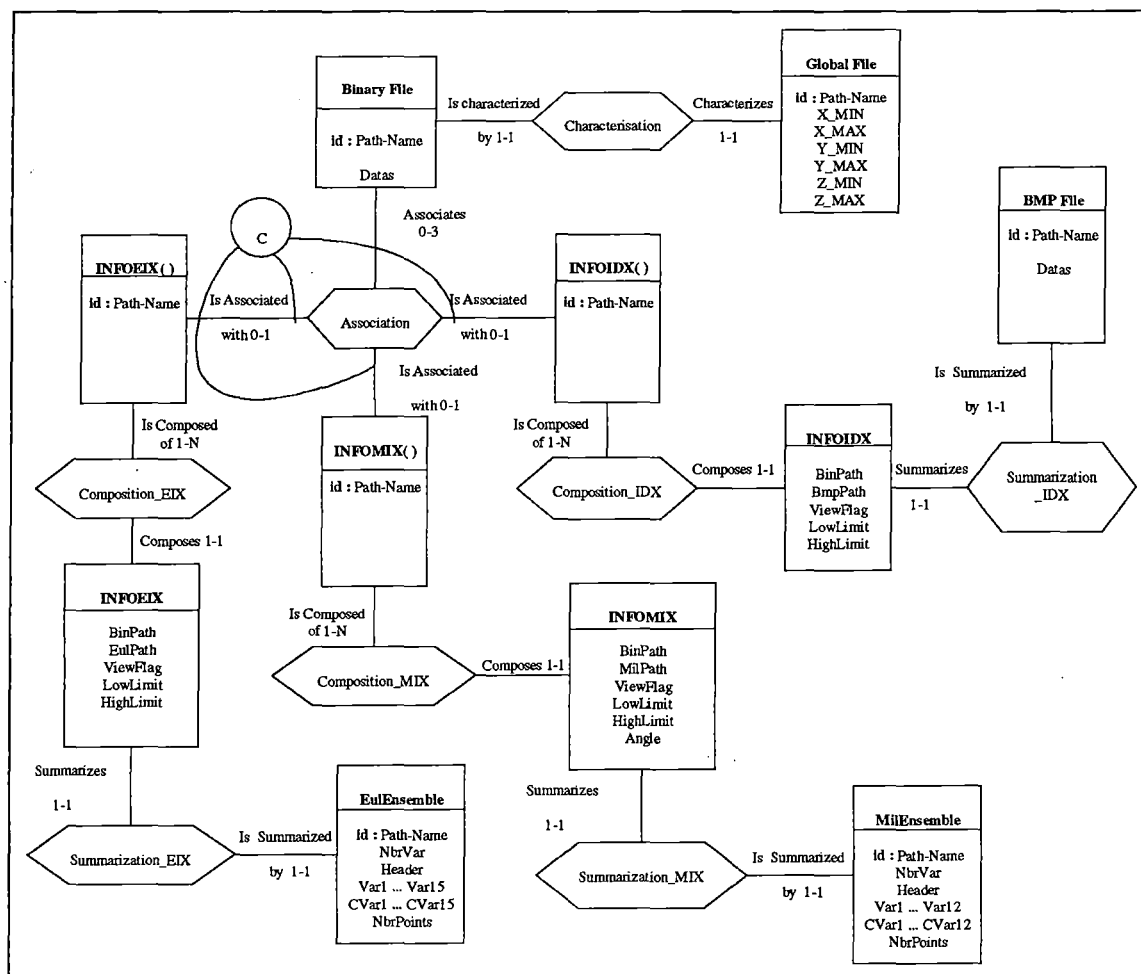


Figure 5.4 Object-Oriented Entity-Relationship Model of the Task

The figure above (Figure 5.4) represent the Object-Oriented Entity-Relationship Model of the task.

3.6.2 Identification of Semantic and Control Functions of the Application

« Task analysis produces actions that deal with task objects. These task concepts should be transformed into concepts that are relevant to software engineering. For this purpose, actions identified during task analysis are modified in accordance with software properties (e.g., precision, completeness, consistency, redundancy, generalization) in order to get functional specifications of the application.

This critical process transforms actions into functions by abstraction mechanism based on both generalization and consolidation (Figure 5.5). For example, actions that are performed on objects possessing common slots can become a more general function dealing with two entities. These functions will become methods attached to the application objects according to object-oriented programming. »^[BODART95]

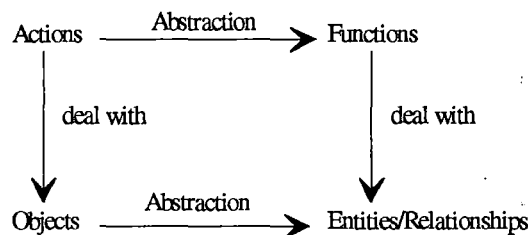


Figure 5.5 The abstraction mechanism.

To be more accurate in our analysis, we have decided to improve the concept of semantic function by establishing a distinction between « true » semantic functions (corresponding to the definition above) and « less true » semantic functions. These « less true » semantic functions are **functions that manage the information flow inside the sub-task and / or manage the information flow between the task and the user** (e.g. a function that just display information in a window to allow the user to see them). We call those functions **Control Functions**.

Instead of identifying the semantic and control functions of the global task, we have identified the semantic and control functions of each sub-task. Based on the theory described above, we have extracted the following functions :

3.6.2.1 Semantic and Control Functions for the « Performing Analysis » sub-task

For this sub-task, we have extracted the following functions :

1. Semantic Functions :

- Validate_Name_Binary_File
- Read_GBL_File
- Choose_MIL_Parameters
- Choose_EUL_Parameters
- Run_Parametrized_Analysis

2. Control Functions :

- Choose_Analysis
- Display_MIL_Parameters
- Display_EUL_Parameters

We will now give a description of those functions :

• Validate_Name_Binary_File

Type : Semantic Function

Goal : Verify that the Name_Binary_File exist on the disk

Entry : Name_Binary_File

Exit : Name_Binary_File_Validated or Name_Binary_File_Unvalidated

Action: IF DIR(Name_Binary_File) is true

Then Name_Binary_File_Validated

Else Name_Binary_File_Unvalidated

• Read_GBL_File

Type : Semantic Function

Goal : Read the contents of the Global File associated with the Name_Binary_File_Validated

Entry : Name_Binary_File_Validated

Exit : INFO-GBL

Action: - Obtain the name of the global file by replacing the « bin » extension of the Name_Binary_File_Validated file with the « gbl » extension.

- Then read the content of this file to produce INFO-GBL

• Choose_MIL_Parameters

Type : Semantic Function

Goal : Allow the user to choose the parameters of the Mil Analysis he wants to perform

Entry : MIL_Parameters

Exit : Parametrized_MIL_Analysis

Action: Record the parameters chosen by the user and construct the DLL call with those parameters

- **Choose_EUL_Parameters**

Type : Semantic Function

Goal : Allow the user to choose the parameters of the Euler Analysis he wants to perform

Entry : EUL_Parameters

Exit : Parametrized_EUL_Analysis

Action: Record the parameters chosen by the user and construct the DLL call with those parameters

- **Run_Parametrized_Analysis**

Type : Semantic Function

Goal : Perform the Parametrized_Analysis by calling the DLL

Entry : Parametrized_MIL_Analysis or Parametrized_EUL_Analysis

Exit : Analysis_Performed

Action: - If the entry is Parametrized_MIL_Analysis
 Then we execute the call to the MIL library of the DLL
 Else we execute the call to the EUL library of the DLL
 - Analysis_Performed

- **Choose_Analysis**

Type : Control Function

Goal : Allow to the user to choose between the two types of analysis performed in the task

Entry : Types_Analyses

Exit : MIL_Analysis_Chosen or EUL_Analysis_Chosen

Action: If the user select MIL_Analysis
 Then MIL_Analysis_Chosen
 Else EUL_Analysis_Chosen

- **Display_MIL_Parameters**

Type : Control Function

Goal : Display all the parameters that the user need to perform a MIL_Analysis

Entry : INFO-GBL and MIL_Analysis_Chosen

Exit : MIL_Parameters

Action: Display all the pertinent data and the INFO-GBL for the MIL analysis

- **Display_EUL_Parameters**

Type	: Control Function
Goal	: Display all the parameters needed by the user to perform a EUL_Analysis
Entry	: INFO-GBL and EUL_Analysis_Chosen
Exit	: EUL_Parameters
Action:	Display all the pertinent data and the INFO-GBL for the Euler analysis

3.6.2.2 Semantic and Control Functions for the « Visualizing Analysis' Results » sub-task

For this sub-task, we have extracted the following functions :

1. Semantic Functions :

- Validate_File_Name
- Read_MIX_File
- Read_EIX_File
- Choose_INFOMIX.MilPath
- Choose_INFOEIX.EulPath
- Choose_Graph_Parameters
- Display_Graph

2. Control Functions :

- Display_INFOMIX.MilPath
- Display_INFOEIX.EulPath
- Display_Info_on_INFOMIX.MilPath_Chosen
- Display_Info_on_INFOEIX.EulPath_Chosen
- Display_Graph_Parameters

We will now give a description of those functions :

• Validate_File_Name

Type	: Semantic Function
Goal	: Verify that the File_Name exist on the disk
Entry	: File_Name
Exit	: File_Name_Unvalidated or Name_MIX_File_Validated or Name_EIX_File_Validated
Action:	IF DIR(File_Name) is true Then Name_MIX_File_Validated or Name_EIX_File_Validated Else File_Name_Unvalidated

- **Read_MIX_File**

Type : Semantic Function

Goal : Read the contents of the MIL analyses index file
Name_MIX_File_Validated

Entry : Name_MIX_File_Validated

Exit : one-to-many INFOMIX

Action: Read the Name_MIX_File_Validated and produce one occurrence of INFOMIX for each entry in the index file

- **Read_EIX_File**

Type : Semantic Function

Goal : Read the contents of the EUL analyses index file
Name_EIX_File_Validated

Entry : Name_EIX_File_Validated

Exit : one-to-many INFOEIX

Action: Read the Name_EIX_File_Validated and produce one occurrence of INFOEIX for each entry in the index file

- **Choose_INFOMIX.MilPath**

Type : Semantic Function

Goal : Allow the user to select one of the MIL analysis file present in the index file

Entry : INFOMIX.MilPath (represent the aggregation of all the INFO-MIX.MilPath present in the index file)

Exit : INFOMIX.MilPath_Chosen (the selected one of all the INFO-MIX.MilPath of the index file)

Action: When the user click on one INFOMIX.MilPath, this one become INFOMIX.MilPath_Chosen

- **Choose_INFOEIX.EulPath**

Type : Semantic Function

Goal : Allow the user to select one of the Euler analysis file present in the index file

Entry : INFOEIX.EulPath (represent the aggregation of all the INFOEIX.EulPath present in the index file)

Exit : INFOEIX.EulPath_Chosen (the selected one of all the INFOEIX.EulPath of the index file)

Action: When the user click on one INFOEIX.EulPath, this one become INFOEIX.EulPath_Chosen

- **Choose_Graph_Parameters**

Type : Semantic Function

Goal : Allow the user to select the parameters of the graph and the variable of the analysis results he wants to visualize

Entry : Graph_Parameters

Exit : Graph_Parameters_Chosen

Action: Record the parameters chosen and give a feed-back to the user

- **Display_Graph**

Type : Semantic Function

Goal : Display a parametrized graph

Entry : Graph_Parameters_Chosen

Exit : GRAPH

Action: Display a graph thanks to the parameters given by Graph_Parameters_Chosen

- **Display_INFOMIX.MilPath**

Type : Control Function

Goal : Display all the MIL analyses path

Entry : one-to-many INFOMIX

Exit : INFOMIX.MilPath (represent the aggregation of all the INFO-MIX.MilPath present in the index file)

Action: Display INFOMIX.MilPath

- **Display_INFOEIX.EulPath**

Type : Control Function

Goal : Display all the Euler analyses path

Entry : one-to-many INFOEIX

Exit : INFOEIX.EulPath (represent the aggregation of all the INFOEIX.EulPath present in the index file)

Action: Display INFOEIX.EulPath

- **Display_Info_on_INFOMIX.MilPath_Chosen**

Type : Control Function

Goal : Display all the information about INFOMIX.MilPath_Chosen and contained in INFOMIX

Entry : INFOMIX.MilPath_Chosen

Exit : INFOMIX

Action: Display INFOMIX where
(INFOMIX.MilPath = INFOMIX.MilPath_Chosen)

- **Display_Info_on_INFOEIX.EulPath_Chosen**

Type : Control Function

Goal : Display all the information about INFOEIX.EulPath_Chosen and contained in INFOEIX

Entry : INFOEIX.EulPath_Chosen

Exit : INFOEIX

Action: Display INFOEIX where
(INFOEIX.EulPath = INFOEIX.EulPath_Chosen)

- **Display_Graph_Parameters**

Type : Control Function

Goal : Display the variables of the analysis file and other pertinent data needed by the user to parametrize the graph he wants to visualize

Entry : INFOMIX.MilPath_Chosen or INFOEIX.EulPath_Chosen

Exit : Graph_Parameters

Action: IF INFOMIX.MilPath_Chosen
THEN display all the variables of the Mil analysis file INFO-
MIX.MilPath_Chosen and other data for the graph
ELSE display all the variables of the Euler analysis file IN-
FOEIX.EulPath_Chosen and other data for the graph

3.6.2.3 Semantic and Control Functions for the « Creating Image » sub-task

For this sub-task, we have extracted the following functions :

1. Semantic Functions :

- Validate_Name_Binary_File
- Read_GBL_File
- Choose_Creation_Parameters
- Run_Creation
- Read_IDX_File
- Select_Last_INFOIDX(i).BmpPath
- Display_Last_INFOIDX(i)

2. Control Functions :

- Display_Creation_Parameters

We will now give a description of those functions :

- **Validate_Name_Binary_File**

Type : Semantic Function

Goal : Verify that the Name_Binary_File exist on the disk

Entry : Name_Binary_File

Exit : Name_Binary_File_Validated or Name_Binary_File_Invalidated

Action: IF DIR(Name_Binary_File) is true
 Then Name_Binary_File_Validated
 Else Name_Binary_File_Unvalidated

- **Read_GBL_File**

Type : Semantic Function

Goal : Read the contents of the Global File associated with the
 Name_Binary_File_Validated

Entry : Name_Binary_File_Validated

Exit : INFO-GBL

Action: - Obtain the name of the global file by replacing the « bin » ex-
 tension of the Name_Binary_File_Validated file with the « gbl »
 extension.
 - Then read the content of this file to produce INFO-GBL

- **Choose_Creation_Parameters**

Type : Semantic Function

Goal : Allow the user to select the parameters for the image creation

Entry : Creation_Parameters

Exit : one-to-three Chosen_Creation_Parameters

Action: Record one-to-three Chosen_Creation_Parameters and construct
 one-to-three call to the DLL with those parameters

- **Run_Creation**

Type : Semantic Function

Goal : Create the image

Entry : Chosen_Creation_Parameters

Exit : Performed_Creation

Action: Execute the call to the BMP library of the DLL

- **Read_IDX_File**

Type : Semantic Function

Goal : Read the contents of the Index file associated with the
 Name_Binary_File_Validated

Entry : Performed_Creation and Name_Binary_File_Validated

Exit : one-to-many INFOIDX

Action: - Obtain the name of the index file by replacing the « bin » exten-
 sion of the Name_Binary_File_Validated file with the « idx » ex-
 tension.
 - Then read the content of this file to produce one-to-many IN-
 FOIDX

- **Select_Last_INFOIDX(i).BmpPath**

Type : Semantic Function

Goal : Select the last entry in the index file

Entry : one-to-many INFOIDX

Exit : Last_INFOIDX.BmpPath

Action: Last_INFOIDX.BmpPath = last (INFOIDX.BmpPath) of the one-to-many INFOIDX

- **Display_Last_INFOIDX(i)**

Type : Semantic Function

Goal : Display the image corresponding to the Last_INFOIDX.BmpPath

Entry : Last_INFOIDX.BmpPath

Exit : Image

Action: Display the Last_INFOIDX.BmpPath image

- **Display_Creation_Parameters**

Type : Control Function

Goal : Display the global information from INFO-GBL and the other pertinent data needed by the user to create the image he wants

Entry : INFO-GBL

Exit : Creation_Parameters

Action: Display INFO-GBL and other data

3.6.2.4 Semantic and Control Functions for the « Visualizing Image » sub-task

For this sub-task, we have extracted the following functions :

1. Semantic Functions :

- Validate_Name_Index_File
- Read_IDX_File
- Choose_INFOIDX.BmpPath
- Display_INFOIDX(i)

2. Control Functions :

- Display_INFOIDX.BmpPath
- Display_Info_on_INFOIDX(i).BmpPath_Chosen

We will now give a description of those functions :

- **Validate_Name_Index_File**

Type : Semantic Function

Goal : Verify that the Name_Index_File exist on the disk

Entry : Name_Index_File

Exit : Name_Index_File_Validated or Name_Index_File_Unvalidated

Action: IF DIR(Name_Index_File) is true
 Then Name_Index_File_Validated
 Else Name_Index_File_Unvalidated

- **Read_IDX_File**

Type : Semantic Function

Goal : Read the contents of the BMP index file
 Name_Index_File_Validated

Entry : Name_Index_File_Validated

Exit : one-to-many INFOIDX

Action: Read the Name_IDX_File_Validated and produce one occurrence
 of INFOIDX for each entry in the index file

- **Choose_INFOIDX.BmpPath**

Type : Semantic Function

Goal : Allow the user to select one of the BMP image file present in the
 index file

Entry : INFOIDX.BmpPath (represent the aggregation of all the IN-
 FOIDX.BmpPath present in the index file)

Exit : one-to-many Chosen_INFOIDX(i).BmpPath

Action: For i=one to many
 -Then when the user click on INFOIDX(i).BmpPath, this one
 become Chosen_INFOIDX(i).BmpPath
 - Next i

- **Display_INFOIDX(i)**

Type : Semantic Function

Goal : Display the image corresponding to the
 Chosen_INFOIDX(i).BmpPath

Entry : Chosen_INFOIDX(i).BmpPath

Exit : Image

Action: Display the Chosen_INFOIDX(i).BmpPath image

- **Display_INFOIDX.BmpPath**

Type : Control Function

Goal : Display all the Bmp images path

Entry : one-to-many INFOIDX

Exit : INFOIDX.BmpPath (represent the aggregation of all the IN-
 FOIDX.BmpPath present in the index file)

Action: Display INFOIDX.BmpPath

- **Display_Info_on_INFOIDX(i).BmpPath_Chosen**

Type	: Control Function
Goal	: Display all the information about Chosen_INFOIDX.BmpPath and contained in INFOIDX(i)
Entry	: Chosen_INFOIDX(i).BmpPath
Exit	: INFOIDX(i)
Action:	Display INFOIDX(i) where (INFOIDX(i).BmpPath = Choosed_INFOIDX(i).BmpPath)

3.6.2.5 Semantic and Control Functions for the « Simulating Osteoporosis » sub-task

For this sub-task, we have extracted the following functions :

1. Semantic Functions :

- Validate_Name_Binary_File
- Read_GBL_File
- Choose_Simulation_Parameters
- Run_Simulation

2. Control Functions :

- Display_Simulation_Parameters

We will now give a description of those functions :

• Validate_Name_Binary_File

Type	: Semantic Function
Goal	: Verify that the Name_Binary_File exist on the disk
Entry	: Name_Binary_File
Exit	: Name_Binary_File_Validated or Name_Binary_File_Invalidated
Action:	IF DIR(Name_Binary_File) is true Then Name_Binary_File_Validated Else Name_Binary_File_Invalidated

• Read_GBL_File

Type	: Semantic Function
Goal	: Read the contents of the Global File associated with the Name_Binary_File_Validated
Entry	: Name_Binary_File_Validated
Exit	: INFO-GBL
Action:	- Obtain the name of the global file by replacing the « bin » extension of the Name_Binary_File_Validated file with the « gbl » extension.

- Then read the content of this file to produce INFO-GBL

- **Choose_Simulation_Parameters**

Type : Semantic Function

Goal : Allow the user to select the parameters for the simulation

Entry : Simulation_Parameters

Exit : Chosen_Simulation_Parameters

Action: Record Chosen_Simulation_Parameters and construct the call to the DLL with those parameters

- **Run_Simulation**

Type : Semantic Function

Goal : Perform the simulation

Entry : Chosen_Simulation_Parameters

Exit : Performed_Simulation

Action: Execute the call to the Osteoporosis library of the DLL

- **Display_Simulation_Parameters**

Type : Control Function

Goal : Display the global information from INFO-GBL and the other pertinent data needed by the user to perform the simulation he wants

Entry : INFO-GBL

Exit : Simulation_Parameters

Action: Display INFO-GBL and other data

3.6.3 Construction of Activity Chaining Graphs

«After having described static task aspects, dynamic aspects remain to be further detailed. This task behavior model can be graphically represented with an activity chaining graph (ACG). This ACG institutes a contract between the programmer who is responsible for the semantic functions of the application and the designer who is responsible of the UI.

It expresses the information flow between functions to be executed for achieving the main goal associated with an interactive task. From the graph theory viewpoint, this graph is a 1-graph without loops, that is simple. » ^[TRIDENT95].

The ACG is exemplified in (Figure 5.6). *« Each function receives input information, representing data required for the good execution of the function.*

Each function receives input information, produces output information which are either external (to the user) or internal (to another function). External information input to a function must come from an interactive dialogue. External information output from a function must go to an interactive display. Input and output information can be related using OR, AND or XOR links
 » [TRIDENT95]

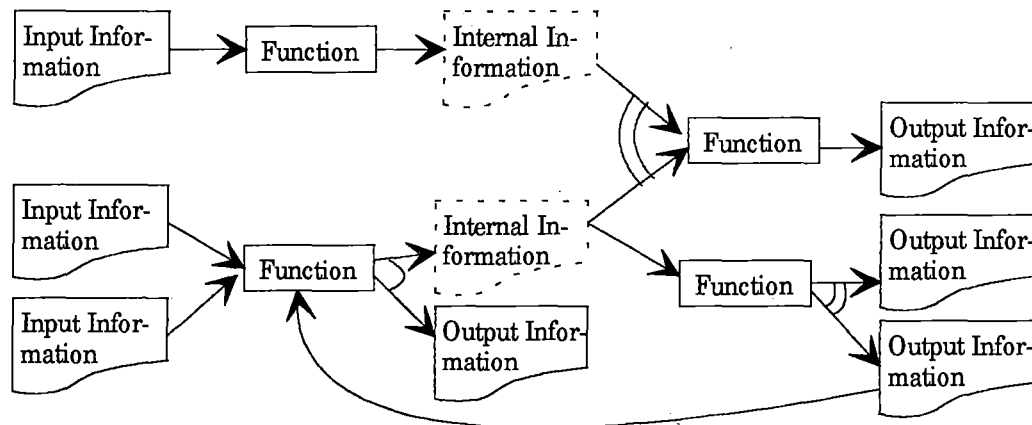


Figure 5.6 Example of Activity Chaining Graph.

As described in the TRIDENT project, an ACG is constructed as follows:

- the elementary input and output information is determined for each identified function: this information corresponds to attributes either of entities or of associated relationships; input and output information also respectively support specification of function pre- and post-conditions.
- each inter-procedure relationship is classified as AND, OR, or XOR;

As we did in the previous point with the Semantic Functions, we will construct an Activity Chaining Graph for each sub-task.

3.6.3.1 Activity Chaining Graph for the « Performing Analysis » sub task

The following figure (Figure 5.7) represent the activity chaining graph of the sub-task :

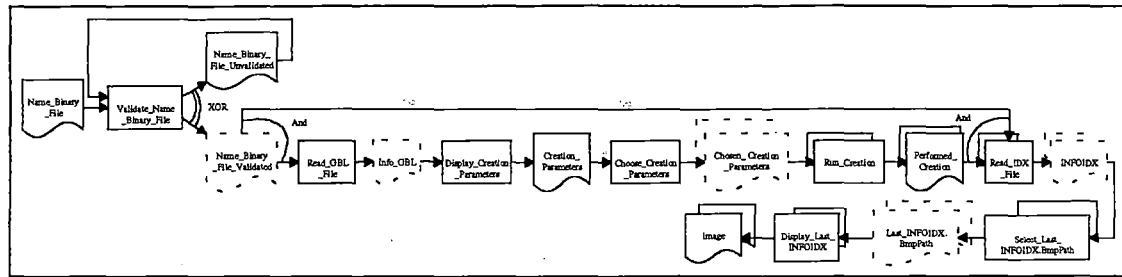


Figure 5.9 ACG for the "Creating Image" sub-task

3.6.3.4 Activity Chaining Graph for the « Visualizing Image » sub-task

The following figure (Figure 5.10) represent the activity chaining graph for the sub-task :

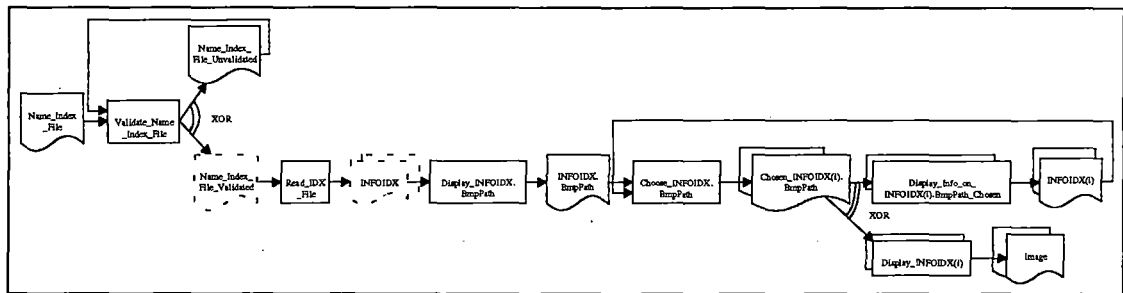


Figure 5.10 ACG for the "Visualizing Image" sub-task

3.6.3.5 Activity Chaining Graph for the « Simulating Osteoporosis » sub-task

The following figure (Figure 5.11) represent the activity chaining graph for the sub-task :

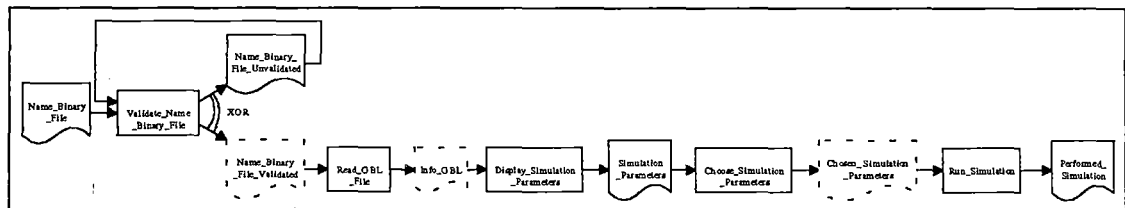


Figure 5.11 ACG for the "Simulating Osteoporosis" sub-task

3.6.4 Derivation of Dialogue Attributes

According to the TRIDENT project : « *For each interactive task, an appropriate (hybrid) dialogue style should be selected. This choice could be guided by a table that mapped task, user and workplace characteristics to a (hybrid) dialogue style.*

(Hybrid) dialogue styles are examples of design options and their principled selection is the first example of restricted choice within our methodological framework.

For each interactive task, four dialogue attributes are derived from these interaction styles: a dialogue mode (sequential, asynchronous, or mixed), a dialogue control (internal, external, or mixed), a function triggering mode (automatic, implicit or explicit manual, displayed or not) and a metaphor (conversation based, universe based or both). These four attributes are introducing four new design options to be considered. They usually have their own default value according to the chosen interaction styles.

Default dialogue attributes provide initial choices of options. They can be reviewed for specific subtasks, although the attributes for really apply at task level, where they can provide UI consistency and compatibility with the ergonomic needs of both the user and their tasks. »^[BODART95]

In this point we will the derive the dialogue attributes for the global task and we will improve this derivation for each **Presentation Unit** later in this chapter. Based on the explanation above, we have derived the following dialogue attributes :

3.6.4.1 Dialogue Mode

The task is formed by five sub-tasks : Performing Analysis, Visualizing Analysis' Results, Creating Image, Visualizing Image and Simulating Osteoporosis. The dialogue mode is **sequential** inside each of the sub-task because the execution of actions is sequential. But the dialogue mode between the sub-tasks

is **asynchronous** or at least **partly-asynchronous**. The reader can refer himself to the point 3.2 Relations between the Sub-tasks, page 11 of this chapter to have more information.

3.6.4.2 Dialogue Control

The dialogue control is **internal** for the task, that is to say, on the system's initiative. Indeed, the dialogue control of the Principal Menu of Selection which is in charge of fire the different sub-tasks, is internal because it is proposed and controlled by the application.

3.6.4.3 Function Triggering Mode

The function triggering mode is **displayed explicit manual** because functions must be triggered on the user's initiative via actions designed for this purpose. Those actions will be materialized by command button (e.g. OK button).

3.6.4.4 Metaphor

In our case, the metaphor is **mixed**, that is to say, on the one hand it is **conversation based** and on the other hand, it is **universe based**. Indeed, the most part of the interaction between the user and the system is conversation based with : choosing parameters, choosing the type of analysis, and so on ... The system proposes some choice and the user selects some of those choices. That is why we can say that it is conversation based. On the other hand, when displaying images and analysis results, we can say that the system is universe based. As we explained in sections 3.3.2.3 and 3.3.2.4, the environment of the task does not exist under a form directly manipulable by the user and the reproducibility of this environment is thus impracticable. When saying that part of the system is universe based, the reader may think that we are in contradiction with what we said above, but it is not true. As a matter of fact, the mini-world of this universe based system is non-existent in the real world, nevertheless it is a mini-world and thus we are not in contradiction because the reproducibility of this mini-world is impracticable.

3.7 Conclusion

Thanks to the parameters identified in sections 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6.4, we establish glancing through the tables of correspondence of the document « Dérivation de style(s) d'interaction » that three interaction styles can be kept (those are the three that minimize gaps with regard to reference values of the tables) :

- Menu Selection
- Forms Filling
- Natural Language

For the rest of our work, we will keep the **Menu Selection** and the **Forms Filling** interaction styles. The **Menu Selection** to select among the sub-task and the **Forms Filling** to manage the dialogue, the conversation within each sub-task.

4. Presentation Design

« The goal of this activity is to find a systematic approach for specifying UI presentation components on a pre-defined set of ergonomic rules in order :

- to meet ergonomic criteria that are relevant to the task;
- to drive this process with computer-aided active and intelligent tools (Figure 5.12). »^[BODART95]

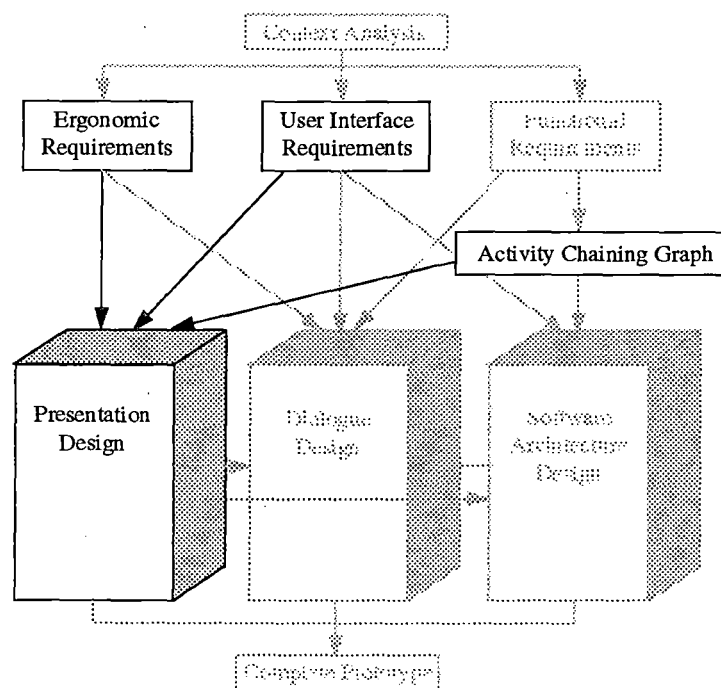


Figure 5.12 Outline of Activity 2.

Starting with the Activity Chaining Graphs, User Interface Requirements guided by ergonomic rules and using the TRIDENT methodology we will define the Presentation. But, before we need some methodological considerations.

4.1 Methodological Considerations (issued from the TRIDENT Project)

4.1.1 Presentation Content ^[BODART95]

UI presentation can be decomposed using the following four concepts which come from the TRIDENT methodology :

- *concrete interaction object* (CIO) : this is a real object belonging to the UI world that any user can manipulate such as a push button, a list box, a check box. A CIO is simple if it cannot be decomposed into smaller CIOs. A CIO is composite if it can be decomposed into smaller units;
- *abstract interaction object* (AIO) : this consists of an abstraction of all CIOs from both presentation and behavioral viewpoints that are independent of target environments;
- *window* : this is a root window either considered as a logical window for AIOs or corresponding to a physical window, a dialogue box or a panel for CIOs. Every window is itself a composite AIO at logical level or CIO at physical level, composed of other simple or composite AIOs/CIOs. All windows are geographically delimited on the user's screen;
- *presentation unit* (PU) : this comprises of an input/output facilities required for execution of specific sub-tasks. Each presentation unit can be decomposed into one or many windows which may not be all displayed on the screen simultaneously. Each PU is composed by at least one window called the *basic window* from which other windows are chained.

4.1.2 Systematic approach for specifying presentation ^[BODART95]

The approach here defines a presentation by iterative refinement starting from global objects to end with simple objects according to the structure reproduced in Figure 5.13. This approach is methodological approach of the TRIDENT project.

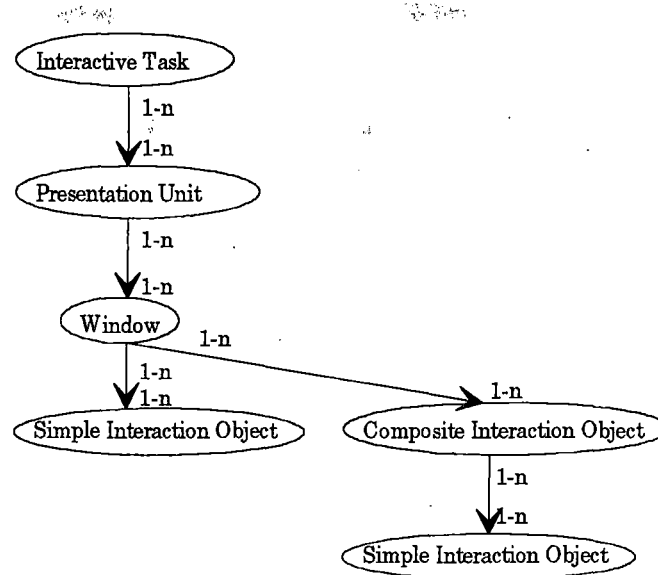


Figure 5.13 Structure of presentation.

The steps of this approach are the following :

- *Identification of Pus* : each sub-task of the interactive task is mapped onto a PU. This mapping is achieved by a series of sub-task identification criteria (for instance, a different work skill). Since the ACG graphically represents the function chaining within a particular task, a PU can correspond to a sub-graph of the ACG that can be drawn graphically distinguished on the ACG (Figure 5.14).

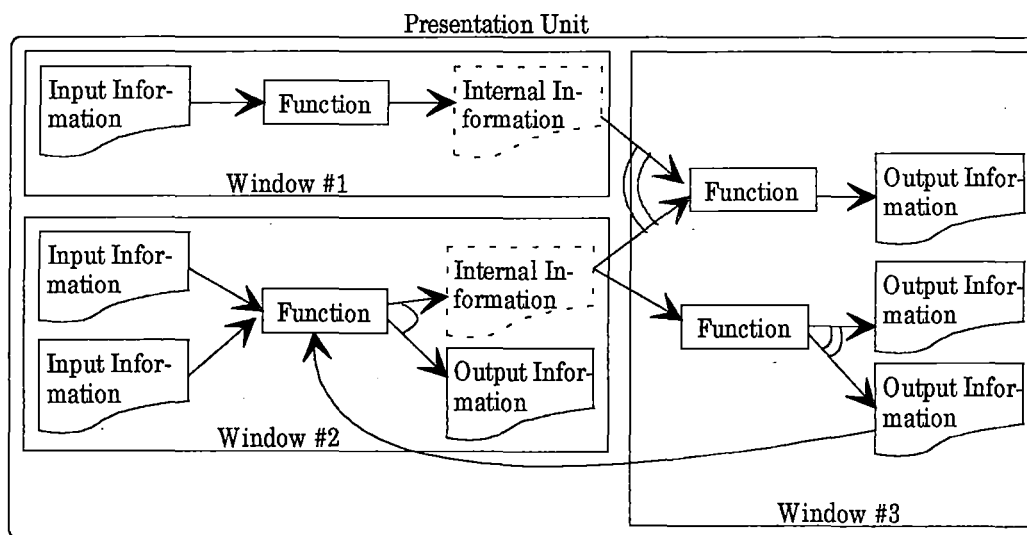


Figure 5.14 Overlapping of a PU with its three windows onto the ACG.

- *Identification of window* : for each sub-graph associated with a PU, partitioning into sub-sub-graphs identifies windows of the PU. Correct identification of these windows requires the application of precise ergonomic rules to ensure satisfaction of several ergonomic criteria (e.g., compatibility, work load, guidance). With this done, designers then know which functions will be triggered from the window, and thus which inputs and outputs must be associated with the window. Like PUs, windows correspond to sub-graphs of the ACG and can again be graphically distinguished (Figure 5.14).
- *Selection of AIOs* : each identified window can be associated with a physical window, a dialogue box or a panel ; within each window, each piece of input and output information is mapped to a simple or composite AIO (e.g., a text widget), as are functions (e.g., to push buttons or icons). Elementary information is always mapped onto a simple AIO. Composite information is mapped onto a composite AIO resulting in a hierarchy of simple AIOs. Similarly, a PU is completely defined by a hierarchy of windows.

Selection of AIOs is supported by a set of selection rules with a scope exceeding the simple physical emphasis of style guides. The rules are initially based on empirically validated cognitive principles. They are subsequently specialized in accordance with user's habits and established conventions.

Within the scope of TRIDENT project methodology, the process of selecting AIOs consists of :

- an initial proposed set of AIOs is generated automatically from the specification contained in the object-oriented entity-relationship model : for example, an edit box is selected for two digit positive integer;
- a second step extends the initially proposed AIOs on the basis of information attributes : for example, a scale is preferred when this integer is a value bounded in a specific range;

- a third step modifies the extended AIOs on the basis of user and domain preference (e.g., a thermometer is useful in medicine).
- *Transformation of AIOs into CIOs* : the complete PU hierarchy is automatically transformed into a hierarchy of CIOs depending on the particular target environment in which the designer is working. Simple and composite AIOs are mapped respectively to simple and composite CIOs.
- *Placement of CIOs* : the co-ordinates of CIOs are calculated precisely for each window by applying placement strategies. These strategies include visual design principles for three aspects: localization, scaling, and arrangement. At the end of this step, each PU hierarchy is completed with relevant positions.

4.2 Application of the TRIDENT Methodology

In this point, we will try to apply the methodology developed in the TRIDENT project to our application. The steps of this point are the following :

1. Identification of Presentation Units,
2. Identification of Windows,
3. Selection of AIOs,
4. Transformation of AIOs into CIOs,
5. Placement of CIOs, and

4.2.1 Identification of Presentation Units

In the section 3.6.2 we have decided to identify Semantic and Control functions for each sub-task. We have also decided, in the section 3.6.3, to create an Activity Chaining Graph for each sub-task of the global task. If we put those ACGs together, we obtain the ACG for the global task. With the subdivision of the global task into sub-task, and with the subdivision of the global ACG into ACG for each sub-task, as with the following methodological rule

from the TRIDENT project : « *each sub-task of the interactive task is mapped into a Presentation Unit* », we have identified the following PUs :

- **PU1** : Performing Analysis
- **PU2** : Visualizing Analysis Results
- **PU3** : Creating Image
- **PU4** : Visualizing Image
- **PU5** : Simulating Osteoporosis

We can now improve for each Presentation Unit the derivation of dialogue attributes :

- **PU1** : Performing Analysis
 - ▲ *Dialogue Control* : internal
 - ▲ *Dialogue Mode* : sequential
 - ▲ *Function Triggering Mode* : displayed explicit manual
 - ▲ *Metaphor* : conversation based
- **PU2** : Visualizing Analysis Results
 - ▲ *Dialogue Control* : internal
 - ▲ *Dialogue Mode* : sequential
 - ▲ *Function Triggering Mode* : displayed explicit manual
 - ▲ *Metaphor* : globally conversation based but universe based when displaying graphs
- **PU3** : Creating Image
 - ▲ *Dialogue Control* : internal
 - ▲ *Dialogue Mode* : sequential
 - ▲ *Function Triggering Mode* : displayed explicit manual
 - ▲ *Metaphor* : globally conversation based, but universe based when displaying the just created image

- **PU4 : Visualizing Image**
 - ▲ *Dialogue Control* : internal
 - ▲ *Dialogue Mode* : sequential
 - ▲ *Function Triggering Mode* : displayed explicit manual
 - ▲ *Metaphor* : globally conversation based, but universe based when displaying images

- **PU5 : Simulating Osteoporosis**
 - ▲ *Dialogue Control* : internal
 - ▲ *Dialogue Mode* : sequential
 - ▲ *Function Triggering Mode* : displayed explicit manual
 - ▲ *Metaphor* : conversation based

4.2.2 Identification of Windows

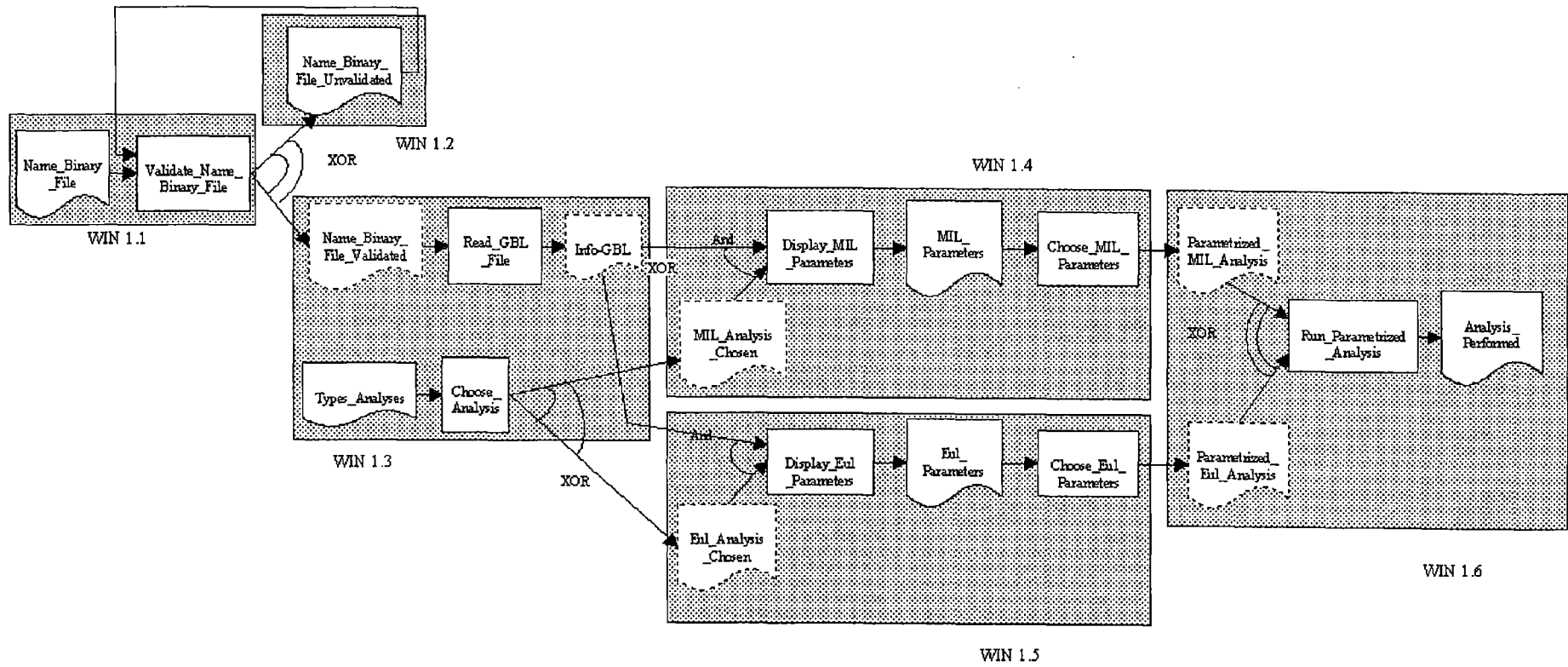
4.2.2.1 Identification of Windows for the PUI

For the PU1, we have identified the following windows (Figure 5.15) :

- ▲ WIN 1.1 : Open_Binary_File_To_Analyse
- ▲ WIN 1.2 : Binary_File_Error_Message
- ▲ WIN 1.3 : Choose_Analysis_To_Perform
- ▲ WIN 1.4 : Choose_Parameters_Of_MIL_Analysis
- ▲ WIN 1.5 : Choose_Parameters_Of_EUL_Analysis
- ▲ WIN 1.6 : Analysis_Performed_Message

Those windows have the following characteristics :

- modal,
- non-sizable,
- non-minimizable,
- non-maximizable,
- moveable.



4.2.2.2 Identification of Windows for the PU2

For the PU2, we have identified the following windows (Figure 5.16) :

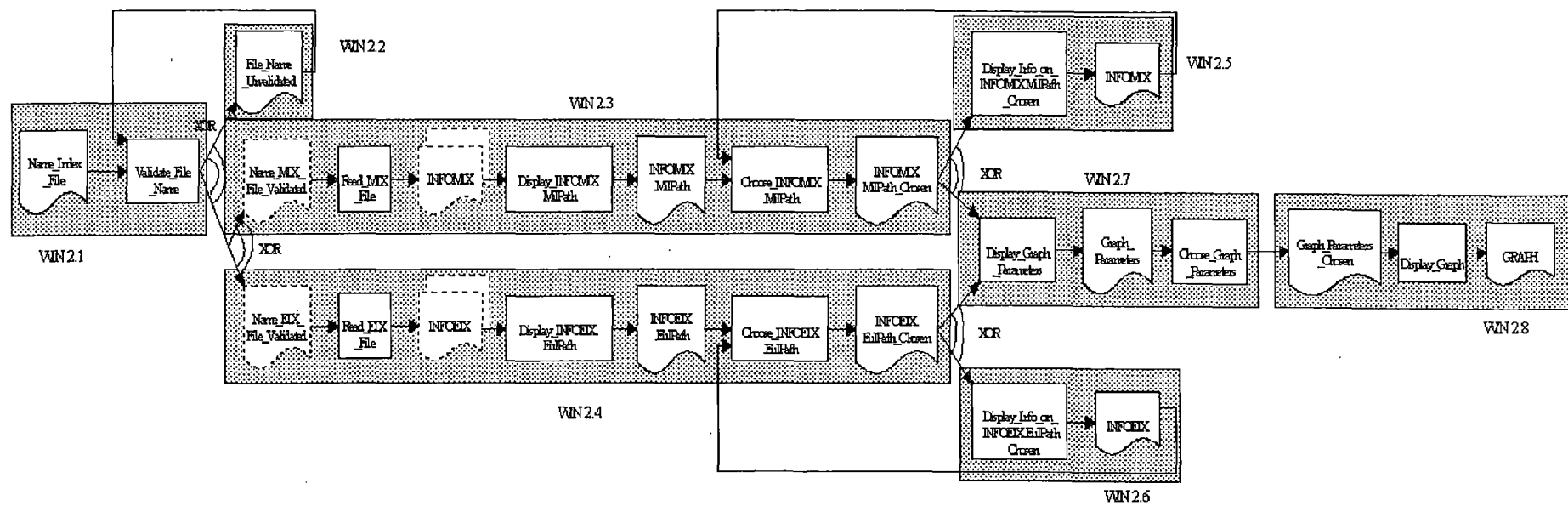
- ▲ WIN 2.1 : Open_Index_File
- ▲ WIN 2.2 : Index_File_Error_Message
- ▲ WIN 2.3 : Contents_Of_MIL_Index_File
- ▲ WIN 2.4 : Contents_Of_EUL_Index_File
- ▲ WIN 2.5 : Information_On_The_Selected_MIL_Analysis_File
- ▲ WIN 2.6 : Information_On_The_Selected_EUL_Analysis_File
- ▲ WIN 2.7 : Graph_Parameters
- ▲ WIN 2.8 : GRAPH

Those windows have the following characteristics :

- modal,
- non-sizable,
- non-minimizable,
- non-maximizable,
- moveable.

Except the window WIN 2.8 which is :

- non-modal,
- sizable,
- minimizable,
- maximizable,
- moveable.



4.2.2.3 Identification of Windows for the PU3

For the PU3, we have identified the following windows (Figure 5.17) :

- ▲ WIN 3.1 : Open_Binary_File
- ▲ WIN 3.2 : Binary_File_Error_Message
- ▲ WIN 3.3 : Choose_Image_Parameters
- ▲ WIN 3.4 : Image

Those windows have the following characteristics :

- modal,
- non-sizable,
- non-minimizable,
- non-maximizable,
- moveable.

Except the window WIN 3.4 which is :

- non-modal,
- non-sizable,
- minimizable,
- non-maximizable,
- moveable.

4.2.2.4 Identification of Windows for the PU4

For the PU4, we have identified the following windows (Figure 5.18) :

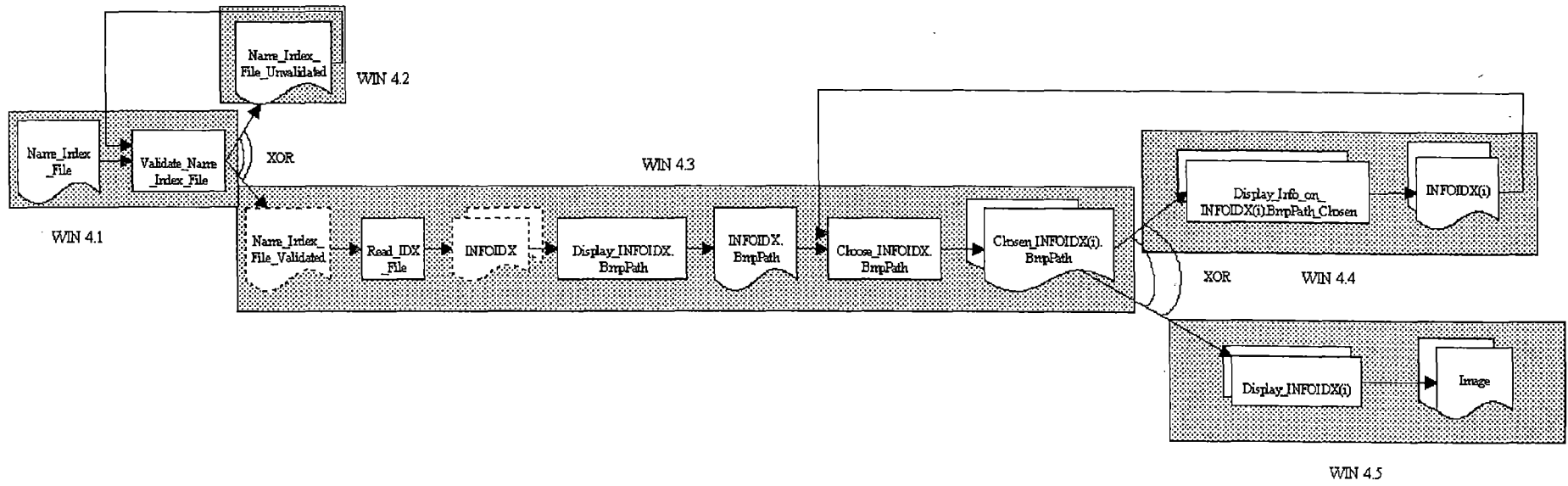
- ▲ WIN 4.1 : Open_Index_File
- ▲ WIN 4.2 : Index_File_Error_Message
- ▲ WIN 4.3 : Contents_Of_BMP_Index_File
- ▲ WIN 4.4 : Information_On_The_Selected_BMP_File
- ▲ WIN 4.5 : Image

Those windows have the following characteristics :

- modal,
- non-sizable,
- non-minimizable,
- non-maximizable,
- moveable.

Except the window WIN 4.5 which is

- non-modal,
- non-sizable,
- minimizable,
- non-maximizable,
- moveable.



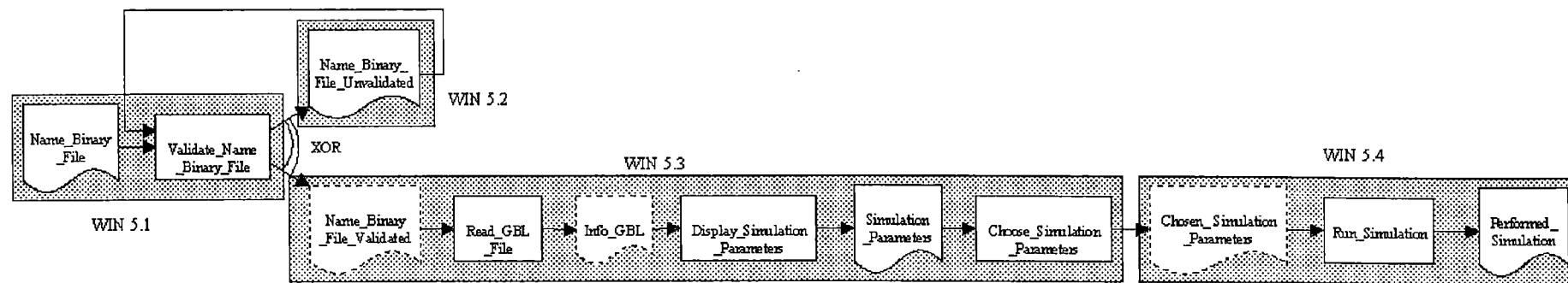
4.2.2.5 Identification of Windows for the PU5

For the PU5, we have identified the following windows (Figure 5.19) :

- ▲ WIN 5.1 : Open_Binary_File
- ▲ WIN 5.2 : Binary_File_Error_Message
- ▲ WIN 5.3 : Choose_Simulation_Parameters
- ▲ WIN 5.4 : Simulation_Performed_Message

Those windows have the following characteristics :

- modal,
- non-sizable,
- non-minimizable,
- non-maximizable,
- moveable.



4.2.3 Selection of AIOs

The windows which have been identified above, are logical windows. Thanks to the interaction styles retained for those windows (Menu selection to select among the sub-tasks and to display images and results, and Forms filling within the sub-tasks), we can infer that except the main window of the application and windows within the results of sub-tasks will be displayed, all the others windows will be Dialog Box.

For this point, we have grouped the previously identified windows into sub-group as follow :

1. Group of « Mother Window and Child Windows » which is composed of the following windows :
 - WIN 0.0 (Mother Window of the application)
 - WIN 2.8
 - WIN 3.4
 - WIN 4.5
2. Group of « Open File Dialog Box » which is composed of the following windows :
 - WIN 1.1
 - WIN 2.1
 - WIN 3.1
 - WIN 4.1
 - WIN 5.1
3. Group of « Analysis Choice Dialog Box » which is composed of the following window :
 - WIN 1.3
4. Group of « Parameters Dialog Box » which is composed of the following windows:
 - WIN 1.4
 - WIN 1.5
 - WIN 2.7

- WIN 3.3
- WIN 5.3

5. Group of « Index Dialog Box » which is composed of the following windows :

- WIN 2.3
- WIN 2.4
- WIN 4.3

6. Group of « Information and Error Windows » which is composed of the following windows :

- WIN 1.2
- WIN 1.6
- WIN 2.2
- WIN 2.5
- WIN 2.6
- WIN 3.2
- WIN 4.2
- WIN 4.4
- WIN 5.3
- WIN 5.4

4.2.3.1 Abbreviations used in the following sections

The abbreviations used in this section are the following :

Cont= continuous domain
Exp = expandable domain
Fr= data frequency in a list
Lg= item current length
Lm= maximal length of an alphanumeric item = 40 characters
Nutil= user experience level
Nvc= number of values to choose
Npo= number of possible values
Npv= number of principal values
Nsv = number of secondary values
AIO= abstract interaction object
Pref = user preference for selecting a data
Tm = maximal number of items in a list = 50 items
Va = antagonist values

4.2.3.2 AIOs Selected for the Group « Mother Window and Child Windows »

The ergonomic rules which are the basement for the selection of AIOs in this group of windows are the following :

- « *The menu design can be based on the mini-world model, that is to say reflecting the real options of the task* »^[JVD94]
- « *The menu presentation must reflect the current state of the system* »^[JVD94]
- « *Menu complexity must reflect the level of user, the functionalities must reflect the requirements of the task* »^[JVD94]
- « *The menu organization must be visible to the user* »^[JVD94]
- « *Generally, the attachment of each menu item must be appropriate to the task : either a sub-menu, either a display/perceive screen, either a secondary screen, either turn on or turn of the parameter associated to the option or neither a triggering action of the application* »^[JVD94]
- « *Each menu item attached to an action must correspond to a task unit that the user must perform* »^[JVD94]
- « *The denominations of items must correspond univocally to task units that the user must perform* »^[JVD94]
- « *The denomination « exit » must be reserved for the option consisting in abandoning the current application closing the interactive session* »

4.2.3.3 AIOs Selected for the Group « Open File Dialog Box »

The ergonomic rule which is the basement for the selection of AIOs in this group of dialog box is the following :

- « *to select a file, use a file selection list box* »^[JVD94]

Within those dialog box, you will always find :

1. a **label** which is the Dialog Box title,
2. two **command button** which are used to confirm or cancel

This last point come from the following ergonomic rule :

- « *any dialog box must have a validation button, a cancel button and if it's possible, an help button* »^[JVD94]

4.2.3.4 AIOs Selected for the Group «Analysis Choice Dialog Box»

The ergonomic rules which are the basement for the selection of AIOs in this group of one dialog box are the following :

- « Use radio button for mutually exclusive choices »^[JVD94],
- « Use command button to fire actions »^[JVD94],
- « For the selection of AIO for alphanumeric data input use the following table »^[JVD93] :

Domain	Nvc	Nsv	Exp	Npo	Lg	AIO
unknown					$\leq Lm$	single-line edit box
					$> Lm$	multiple-line edit box
mixed				[2,3]		radio-button with Npo items + single-line edit box
				[4,7]		radio button with Npo items + single-line edit box + group box
				[8,Tm]		drop-down combination box
				[Tm+1,2Tm]		scrolling combination box
				$> 2Tm$		drop-down scrolling combination box
known	> 1	$= 0$	no	[2,3]		Npo check boxes
				[4,7]		Npo check boxes + group box
				[8,Tm]		list box
				[Tm+1,2Tm]	$\leq Lm$	scrolling list box
				[Tm+1,2Tm]	$> Lm$	drop-down scrolling list box
				$> 2Tm$		drop-down scrolling list box
			yes		$\leq Lm$	combination box
					$> Lm$	drop-down combination box
	> 0				$\leq Lm$	list box
					$> Lm$	drop-down list box
$= 1$	> 0				$\leq Lm$	list box
					$> Lm$	list box
$= 0$	yes				$\leq Lm$	combination box
					$> Lm$	combination box
			no	[2,3]		radio-button with Npo items
				[4,7]		radio-button with Npo items + group box
				[8,Tm]	$\leq Lm$	list box
					$> Lm$	list box
				[Tm+1,2Tm]	$\leq Lm$	scrolling list box
					$> Lm$	scrolling list box
				$> 2Tm$		drop-down scrolling list box

Within this dialog box, you will always find :

1. a **label** which is the Dialog Box title,
2. two **command button** which are used to confirm or cancel

This last point come from the following ergonomic rule :

- « *any dialog box must have a validation button, a cancel button and if it is possible, an help button* »^[JVD94]

4.2.3.5 AIOs Selected for the Group « Parameters Dialog Box »

The ergonomic rules which are the basement for the selection of AIOs in this group of dialog box are the following :

- « *To capture a group of data of different types, use a group box surrounding the selected AIO for each of the data of the group* »
- « *Use radio button for mutually exclusive choices* »^[JVD94],
- « *Use command button to fire actions* »^[JVD94],
- « *Use check box to activate or deactivate options* »^[JVD94]
- « *Use dials, scrolling cursors or other types of AIO of the same behavior to control direction, position, amplitude, also to capture numerical data in a continuous space* »
- « *for the selection of AIO for Boolean data input, use the following table* »^[JVD93].

Domain	Antagonist values	Orientation	AIO
known	yes	vertical	vertical switch
		horizontal	horizontal switch
		circular	two-valued dial
		undefined	horizontal switch
	no		check box
unknown			check box

- « For the selection of AIO for integer data input, use the following table »^[JVD93] :

If the domain is known and the choice is simple (Nvc=1), then use the following table:

Nsv	Exp	Cont	Npo	Precision	Orientation	AIO
> 0						list box
= 0	yes					combination box
	no	no	[2,3]			radio-button with Npo items
			[4,7]			radio-button with Npo items + group box
			[8,Tm]			list box
			[Tm+1,2Tm]			scrolling list box
			> 2Tm			drop-down scrolling list box
	yes		[1,10]	low	vertical	scroll bar
					horizontal	scale
					circular	pie diagram
					undefined	scale
				high	vertical	vertical thermometer
					horizontal	horizontal thermometer
					circular	dial
					undefined	horizontal thermometer
			[11,Tm]	high		spin button
				low		scale
			> Tm	high		spin button
				low	vertical	scroll bar
					horizontal	scale
					circular	dial
					undefined	scale

- « For the selection of AIO for alphanumeric data input use the following table »^[JVD93] :

Domain	Nvc	Nsv	Exp	Npo	Lg	AIO
unknown					<= Lm	single-line edit box
					> Lm	multiple-line edit box
mixed				[2,3]		radio-button with Npo items + single-line edit box
				[4,7]		radio button with Npo items + single-line edit box + group box
				[8,Tm]		drop-down combination box
				[Tm+1,2Tm]		scrolling combination box
				> 2Tm		drop-down scrolling combination box

known	> 1	= 0	no	[2,3]		Npo check boxes
				[4,7]		Npo check boxes + group box
				[8,Tm]		list box
				[Tm+1,2Tm]	<= Lm	scrolling list box
				[Tm+1,2Tm]	> Lm ^u	drop-down scrolling list box
				> 2Tm		drop-down scrolling list box
			yes		<= Lm	combination box
					> Lm	drop-down combination box
	> 0				<= Lm	list box
					> Lm	drop-down list box
	= 1	> 0			<= Lm	list box
					> Lm	list box
		= 0	yes		<= Lm	combination box
					> Lm	combination box
			no	[2,3]		radio-button with Npo items
				[4,7]		radio-button with Npo items + group box
				[8,Tm]	<= Lm	list box
					> Lm	list box
				[Tm+1,2Tm]	<= Lm	scrolling list box
					> Lm	scrolling list box
				> 2Tm		drop-down scrolling list box

- « For the selection of AIO for elementary data input in a list, use the following table »^[JVD93]:

To input an elementary data (necessarily with unexpandable domain and simple choice in a list), then use the following table :

Type	Domain	Va	Lg	Npo	Nutil	AIO
hour					<= 5	spin button
					> 5	profiled single-line edit box
date					<= 5	spin button
					> 5	profiled single-line edit box
boolean		yes				radio-button
		no				check box
graphic						radio icon
integer	unknown					single-line edit box
	mixed					drop-down combination box
	known			[2,7]		radio-button
				[8,Tm]		drop-down list box
				> Tm		spin button
real	unknown					profiled single-line edit box

	mixed					drop-down combination box
	known					drop-down list box
alphanumeric	unknown		$\leq L_m$			single-line edit box
			$> L_m$			multiple-line edit box
	mixed					drop-down combination box
	known					drop-down list box

Within those dialog box, you will always find :

1. a **label** which is the Dialog Box title,
2. two **command button** which are used to confirm or cancel

This last point come from the following ergonomic rule :

- « any dialog box must have a validation button, a cancel button and if it is possible, an help button »^[JVD94]

4.2.3.6 AIOs Selected for the Group « Index Dialog Box »

The ergonomic rule which are the basement for the selection of AIOs in this group of dialog box are the following :

- « use command button to fire actions »^[JVD94],
- « For the selection of AIO for elementary data input in a list, use the following table »^[JVD93] :

To input an elementary data (necessarily with unexpandable domain and simple choice in a list), then use the following table :

Type	Domain	Va	Lg	Npo	Nutil	AIO
hour					≤ 5	spin button
					> 5	profiled single-line edit box
date					≤ 5	spin button
					> 5	profiled single-line edit box
boolean		yes				radio-button
		no				check box
graphic						radio icon
integer	unknown					single-line edit box
	mixed					drop-down combination box
	known			[2,7]		radio-button
				[8,Tm]		drop-down list box
				$> T_m$		spin button
real	unknown					profiled single-line edit box

	mixed					drop-down combination box
	known					drop-down list box
alphanu-	unknow		<= Lm			single-line edit box
meric	n		> Lm			multiple-line edit box
	mixed					drop-down combination box
	known					drop-down list box

Within those dialog box, you will always find :

1. a **label** which is the Dialog Box title,
2. two **command button** which are used to confirm or cancel

This last point come from the following ergonomic rule :

- « any dialog box must have a validation button, a cancel button and if it is possible, an help button »^[JVD94]

4.2.3.7 AIOs Selected for the Group « Information and Error Windows »

The ergonomic rules which are the basement for the selection of AIOs in this group of windows are issued from the section 8.2 « Messages de Guidage et d'informations » in [JVD94].

Within those messages and information windows, you will always find :

1. a **label** which is the window title
2. an **icon** which identify the type of message : error or information
3. a **label** which explain the error or give the information
4. a **command button** that the user can press to confirm his lecture of the message

4.2.4 Transformation of AIOs into CIOs

For each AIO that we have identified in the previous point, we will now take the concrete interactive object corresponding in the Microsoft Windows physical environment :

AIOs	CIOs
edition field	edit box
command button	push button
label	label
icon	icon
file selection list	file selection list box
menu	menu
menu item	menu item
group box	group box
check box	check box
radio button	radio button
spin button	spin button
selection list	list box
modal dialog box	modal dialog box

4.2.5 Placement of CIOs

The ergonomic rules which are the basement of the windows we have created are the following :

- « *mnemonic terms must be unique and, if available, must be displayed* »,
- « *every label indicating a control CIO should have a mnemonic terms* »
- « *every dialog box must have a validation button, a cancel button, and if possible, a help button* »,

- « a label must be provided for every CIO associated with a data to capture »
- « The header label of a group box must be left justified with the CIO surrounded by the group box which is placed lefter »
- « every group of command buttons in connection with the same logical set must be placed either like a line form, below the CIO in which they are placed, either in column placed at the right side of the composed CIO if the previous solution is not suitable »
- « The order of the placment of the command buttons must be as close as possible of the following : (OK), (CANCEL), (HELP) »
- « Command buttons set horizontally must be of equal vertical size »
- « Command buttons set vertically must be of equal horizontal size »

Next, we will give a description of the windows we have designed with Microsoft Visual Basic Professional Edition 3.0 .

4.2.5.1 Group of « Mother Window and Child Windows »

The following window is a sample of what we designed for this group of windows :

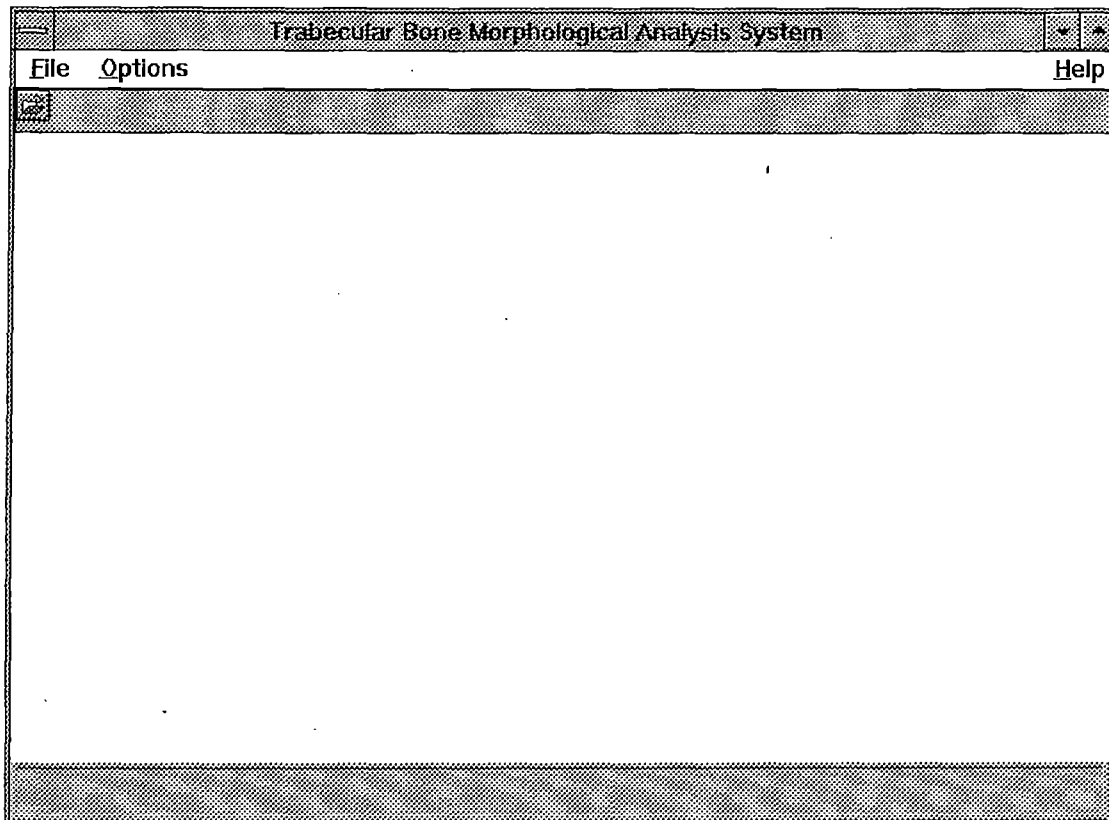


Figure 5.15 Main window of the application

The window illustrated by the Figure 5.15 is WIN 0.0 which is the main window of the application. This window allow the user to choose the sub-task he wants to perform, and this thanks to the File Menu.

The window illustrated by the Figure 5.16 represent the WIN 4.5 and WIN 3.4 windows. This window contains the image that the user wants to display (WIN 4.5) or the image that the user has just created (WIN 3.4).

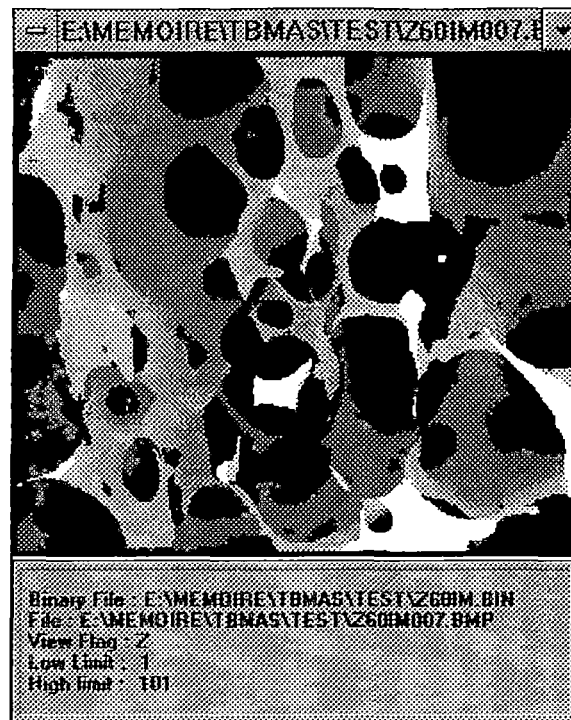


Figure 5.16 Image Window

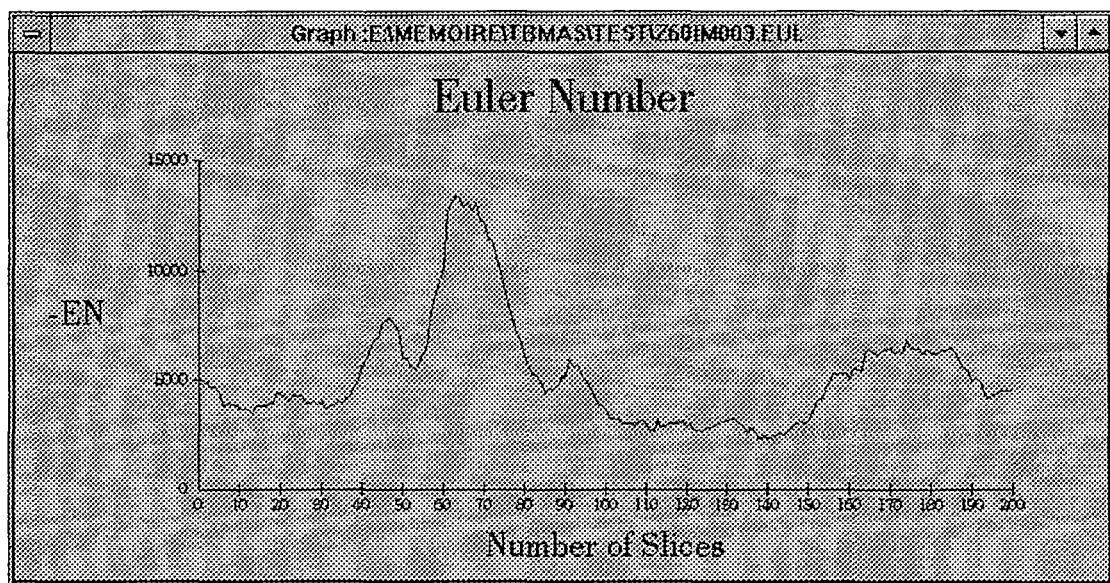


Figure 5.17 Analysis' Results Window

The window illustrated by the Figure 5.17 represent the WIN 2.8 window which show the analysis' results to the user.

4.2.5.2 Group of « Open File Dialog Box »

The following window is a sample of what we designed for this group of windows :

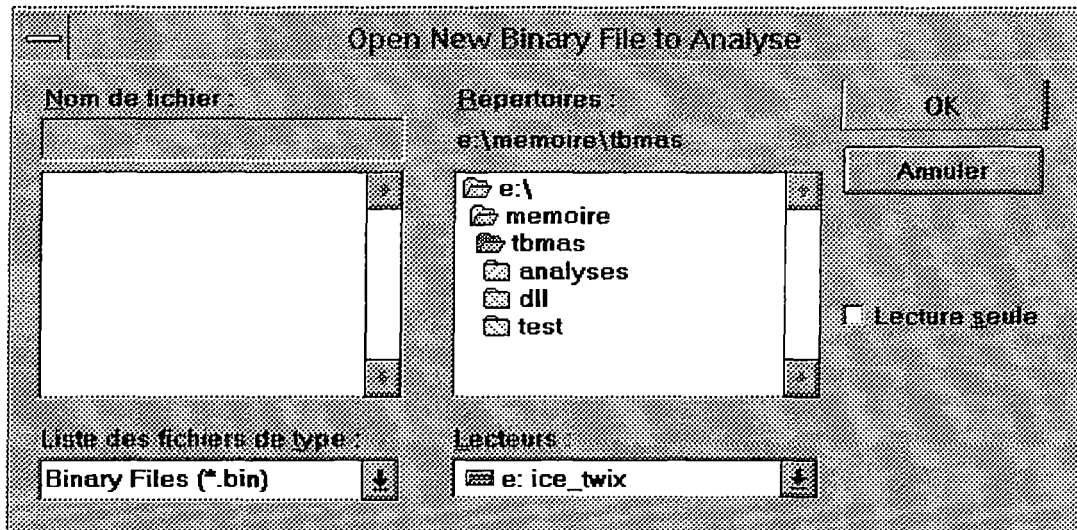


Figure 5.18 Standard Open File Window

The window illustrated by the Figure 5.18 is the Microsoft Windows Standard Open File Window that we have used to respect standardization of applications.

4.2.5.3 Group of « Analysis Choice Dialog Box »

The following window is a sample of what we designed for this group of windows :

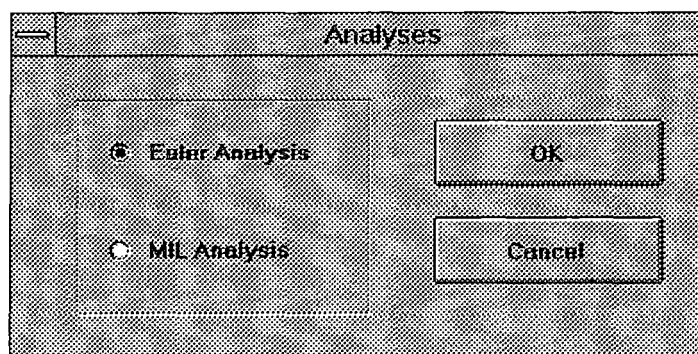


Figure 5.19 Analyses' Types Window

The window illustrated by the Figure 5.19 is the WIN 1.3 window which allows the user to choose the analysis he wants to perform.

4.2.5.4 Group of « Parameters Dialog Box »

The following windows are samples of what we designed for this group of windows :

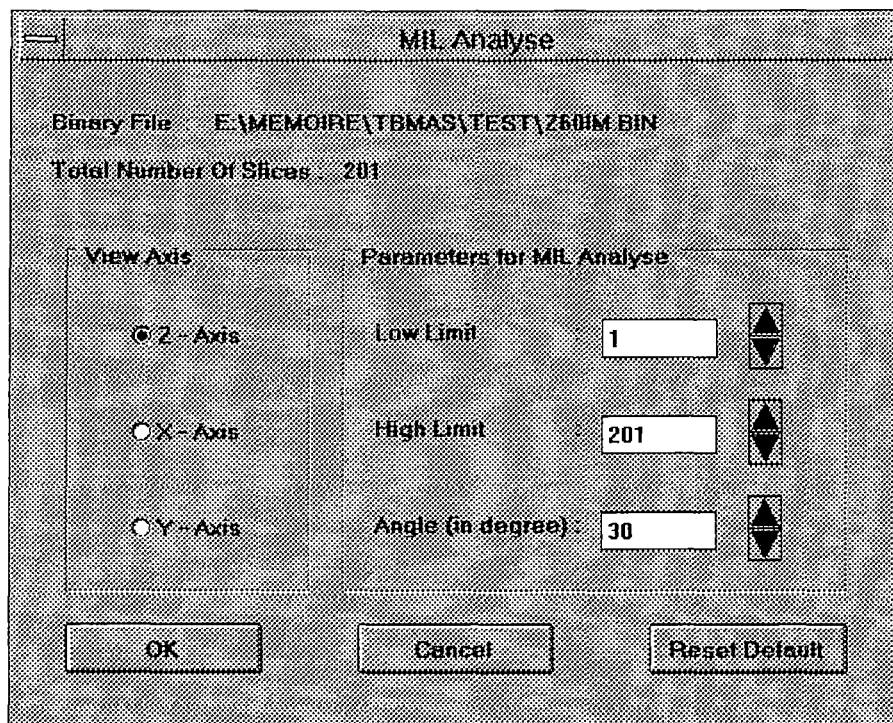


Figure 5.20 MIL Parameters Window

The window illustrated by the Figure 5.20 represent the WIN 1.4 window which allows the user to choose the parameters of the MIL analysis he wants to perform.

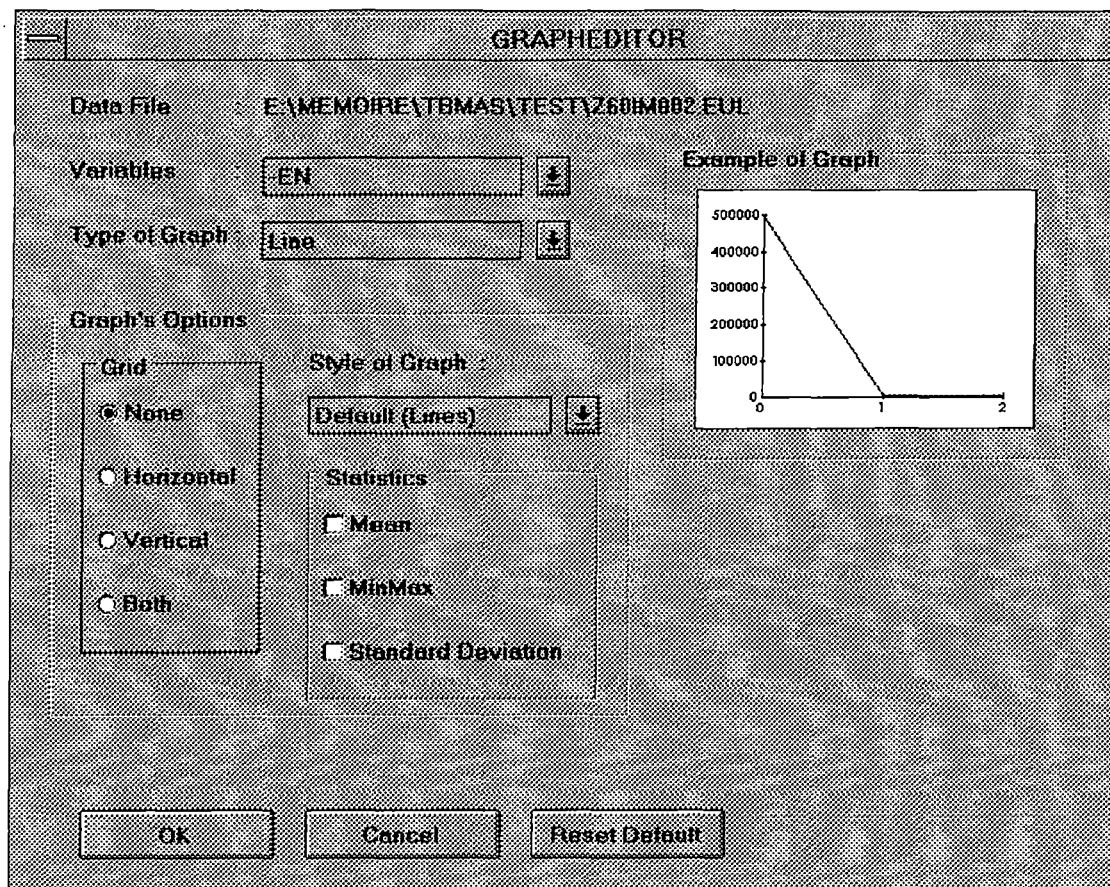


Figure 5.21 Graph Parameters Window

The window illustrated by the Figure 5.21 represent the WIN 2.7 which allows the user to select the parameters of the graph representing the analysis' results he wants to visualize.

4.2.5.5 Group of « Index Dialog Box »

The following window is a sample of what we designed for this group of windows :

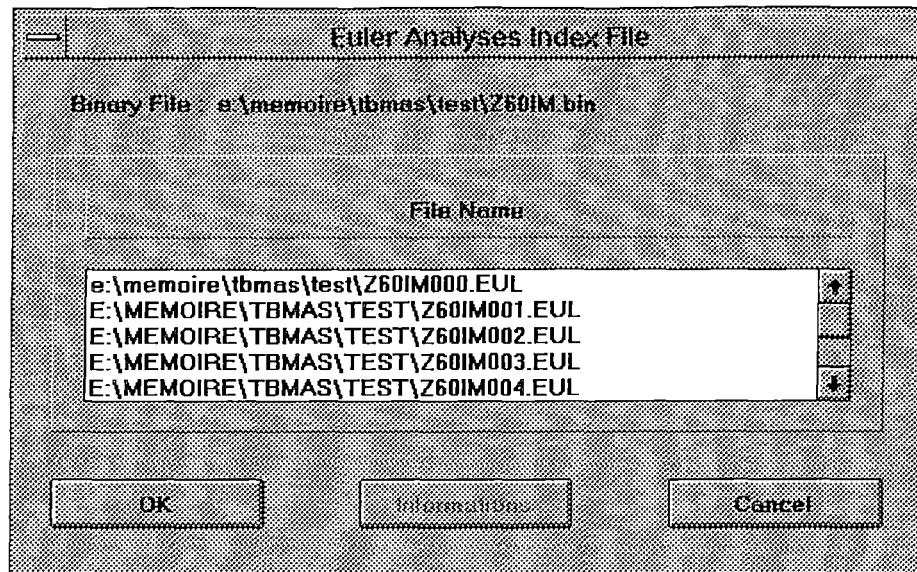


Figure 5.22 Sample of Index Window

The window illustrated by the Figure 5.22 represent the WIN 2.4 window which allows the user to visualize the contents of the index file for Euler analysis, to display information about one of the analysis file of this index, and to select the analysis file he wants to see the results.

4.2.5.6 Group of « Information and Error Windows »

The following window is a sample of what we designed for this group of windows :

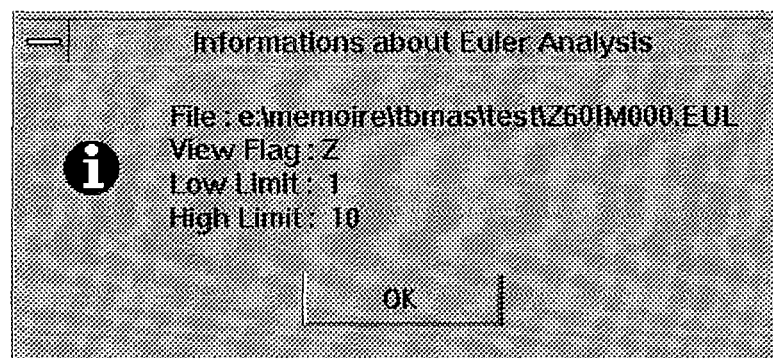


Figure 5.23 Sample of Error and Information Message

The window illustrated by the Figure 5.23 represent the WIN 2.6 window which gives information about an Euler analysis file to the user.

5. Derivation of the Software Architecture

« The aim of this activity is to systematically derive an architecture skeleton which respects architectural software quality criteria: high internal cohesion; weak coupling; independent components (Figure 5.24). »^[BODART95]

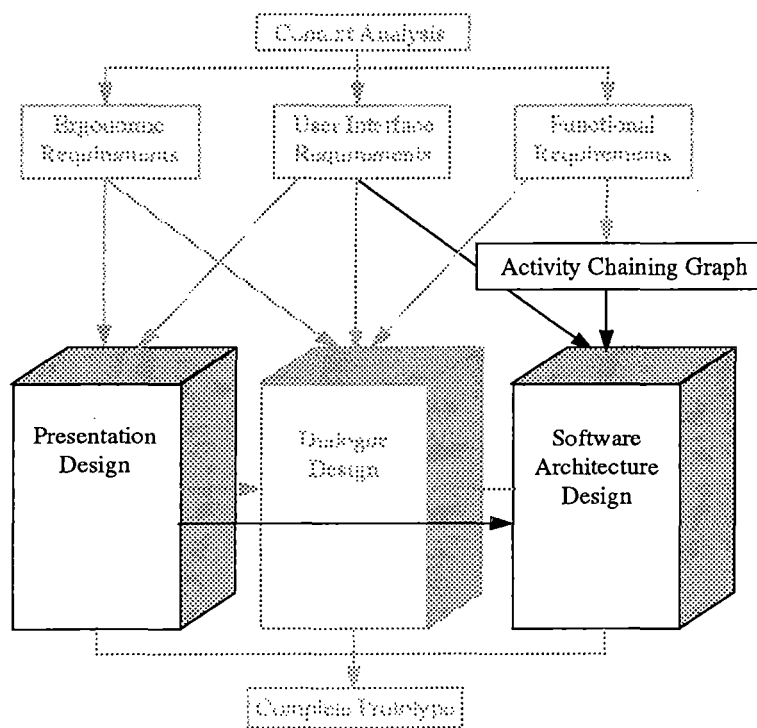


Figure 5.24 Outline of Activity 3.

5.1 Methodological Considerations^[BODART95]

The methodological considerations developed next are issued from the TRIDENT project.^[BODART95]

5.1.1 Assumptions

The proposed architecture model consists of a hierarchy that should match the following methodological assumptions :

- each hierarchy element should be derived - directly or indirectly - from task analysis;
- elements representing application and UI components should be as independent as possible. There should be further independence within UI components, i.e., between dialogue subcomponents (which realize behavior) and presentation subcomponents (which realize appearance).

The latter assumptions are possible for business oriented applications, since generally there is no semantic role for the interaction objects.

5.1.2 Content of the model

The proposed architecture model consists of a hierarchy of generic elements illustrated in Figure 5.25.

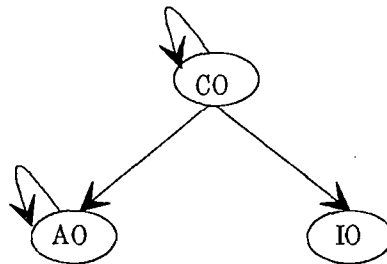


Figure 5.25 Generic scheme of the architecture model.

There are three classes of object in this architectural model. Each is distinguished, as follows :

1. The *Control Objects* (CO) class is generic with instances decomposed into COs of different types that both manage dialogue and preserve the correspondence between application data and the presentation. Each CO has a specific behavior that combines management of a portion of the dialogue and some application-presentation correspondences. A rule-based language configures CO behavior in *scripts* that have a partial graphical representation as state transition diagrams.
2. The *Application Objects* (AO) class is not generic : instances cannot be decomposed, since they represent the application functions.

3. The *Interaction Objects* (IO) class is generic and provides two types of CIOs : application-dependent CIOs that translate input and output information for functions ; and application-independent CIOs that are required for a dialogue (e.g., command buttons that trigger functions).

Identical rules for behavior and interaction apply to all these three object classes. Similarly, identical relationships link any pair of these from these three objects : each object is an *agent*.

Objects are composed in a hierarchy where parent objects "use" child objects, as follows :

- child objects send events related to signal behavior states to their parent ;
- parent objects obtain the information needed for the next interaction step by calling a child object's primitive methods.

5.1.2.1 *Application Objects (AO)*

Each function in the ACG has a single corresponding AO. Where an existing function is to be re-used, but is not implemented in an object-oriented language, it must be encapsulated in a AO, when it becomes one of the AO's methods. Otherwise, the AO should be implemented in an object-oriented language. Every AO should be the child of one (and only one) CO (i.e., the CO-Fc as we will see in next subsection), which becomes responsible for the execution of the corresponding function.

5.1.2.2 *Control Objects (CO)*

There should be a CO in the hierarchy for every composition in the presentation. The composition of the presentation encourages autonomy for windows, which are linked dynamically into PUs. These PUs realize the context for execution of an interactive task. The CO hierarchy is thus built according to our presentation structure (Figure 5.13), and thus four types of CO result :

- CO-IT : the lonely control object corresponding to the interactive task ;
- CO-PU : the control objects corresponding to the presentation units ;
- CO-W : the control objects corresponding to the windows ;
- CO-Fc : the control objects corresponding to the application functions.

The resulting structure is shown in Figure 5.26.

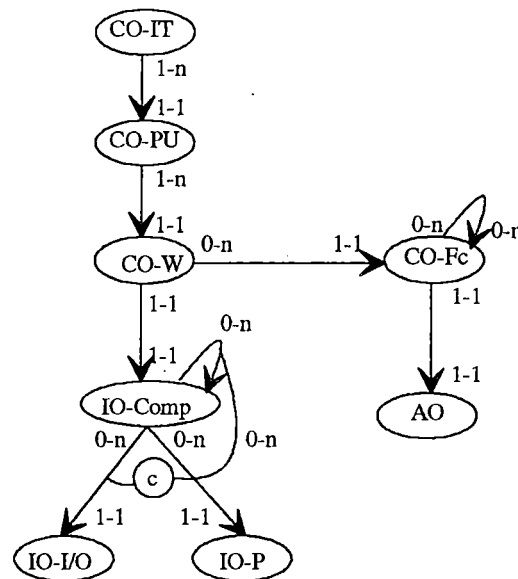


Figure 5.26 Structure of Control Objects.

Every CO corresponding to an application function (CO-Fc) is a child object of the window control object (CO-W) that holds all CIOs required to trigger the function from the functional machine. The set of AOs and corresponding CO-Fc's forms the *functional machine*, where Co-Fc's mirror the business activities represented in the ACG by the way they chain functions.

5.1.2.3 Interaction Objects (IO)

According to Figure 5.13, presentation is structured into simple and composite IOs. *Composite IOs* (IO-Comp) are all IOs corresponding to logical windows (e.g., dialogue boxes, physical windows, panels) or any grouping of simple IOs (e.g., child dialogue boxes, group boxes). IO-Comp use simple IOs for creating the UT's presentation: as *input-output interaction objects* (IO-I/O, e.g., an edit box, radio button, check box) ; or as *presentation interaction objects* (P-

IO, e.g., push buttons, icons). IOs are selected by an expert system on the basis of several parameters. They correspond to specific toolkit CIOs in physical environments like Ms-Windows, OSF/Motif. IOs are always children of CO-W control objects, to which they send events representing significant state changes. Primitive IO methods pass relevant values to parent objects. These primitive methods tend to be native to specific toolkits.

5.1.2.4 *Inter-object relations*

- In CO-IT→CO-PU interactions, the dynamic chaining of PUs is managed by the CO-IT, which loads and unloads - sequentially or concurrently - a particular PU_i according to events received from a PU_j .
- In CO-PU→CO-W interactions, the dynamic chaining of windows is managed by the CO-PU, which displays and undisplays - with tiling or overlapping - a particular window W_i according to events received from another window W_j .
- In CO-W→CO-Fc interactions, the CO-W calls the CO-Fc in order to call a semantic function, when all data required by the function to be performed are available in the child objects of the CO-W. The CO-Fc is the control object which is responsible for the final call of function contained in AO. On completion of processing, CO-Fc's send an event to their CO-W.
- In CO-Fc→AO interactions, the CO-Fc are responsible for the final triggering of functions contained in AOs. A CO-W object calls a function when all required information is input, but the final triggering will be effective only if all triggering conditions expressed in the ACG are fulfilled, that is if all termination events have been sent and internal information has been transmitted. The corresponding CO-Fc only verifies the complete precondition before executing the function.

- In recursive CO-Fc interactions, CO-Fc's exchange events to maintain ACG dynamics and to exchange information by calling on each others services.
- In IO (for IO-I/O or IO-P) and CO-W interactions, the parent object polls children to find out what users have input, and in turn send events to their parents that indicate their contents.

5.1.3 Systematic approach to hierarchy building

We assume that task analysis (subsection 3.1), constructing an ACG (sub-subsection 3.6.3), the development of the functional machine (sub-subsection 5.1.2.1) and the definition of presentation (sub-subsection 5.1.2.3) are all completed.

5.1.3.1 *List of hierarchy objects*

Here, we summarize the complete list of objects of the three kinds. It holds :

- a control object corresponding to the interactive task (CO-IT) ;
- a control object corresponding to each presentation unit (CO-PU₁,...,CO-PU_n) ;
- a control object corresponding to each window of each presentation unit (CO-PU₁F₁,..., CO-PU₁F_m,...,CO-PU_nF₁,...,CO-PU_nF_p) ;
- a control object for each function in the ACG (CO-Fc) ;
- an interaction object for each input/output information for all functions (IO-I/O) ;
- an interaction object for each object induced by the presentation (IO-P).

5.1.3.2 *Relationships between hierarchy objects*

"Uses" relationships linking the different objects can be systematically created as follows :

- connect any control object corresponding to a presentation unit to the control object corresponding to the interactive task from which it depends (CO-IT→CO-PU) ;

- connect any control object corresponding to a window to the control object corresponding to the presentation unit which contains this window (CO-PU \rightarrow CO-W) ;
- connect any interaction object corresponding to an input/output information to the window that uses the function with this information as parameter (CO-W \rightarrow IO-I/O) ;
- connect any interaction object corresponding to an information induced by the presentation to the window where it appears (CO-W \rightarrow IO-P) ;
- connect any object control corresponding to an application function to the control object corresponding to the window in which this function is represented (CO-W \rightarrow CO-Fc).

5.2 *Application of the TRIDENT Methodology*

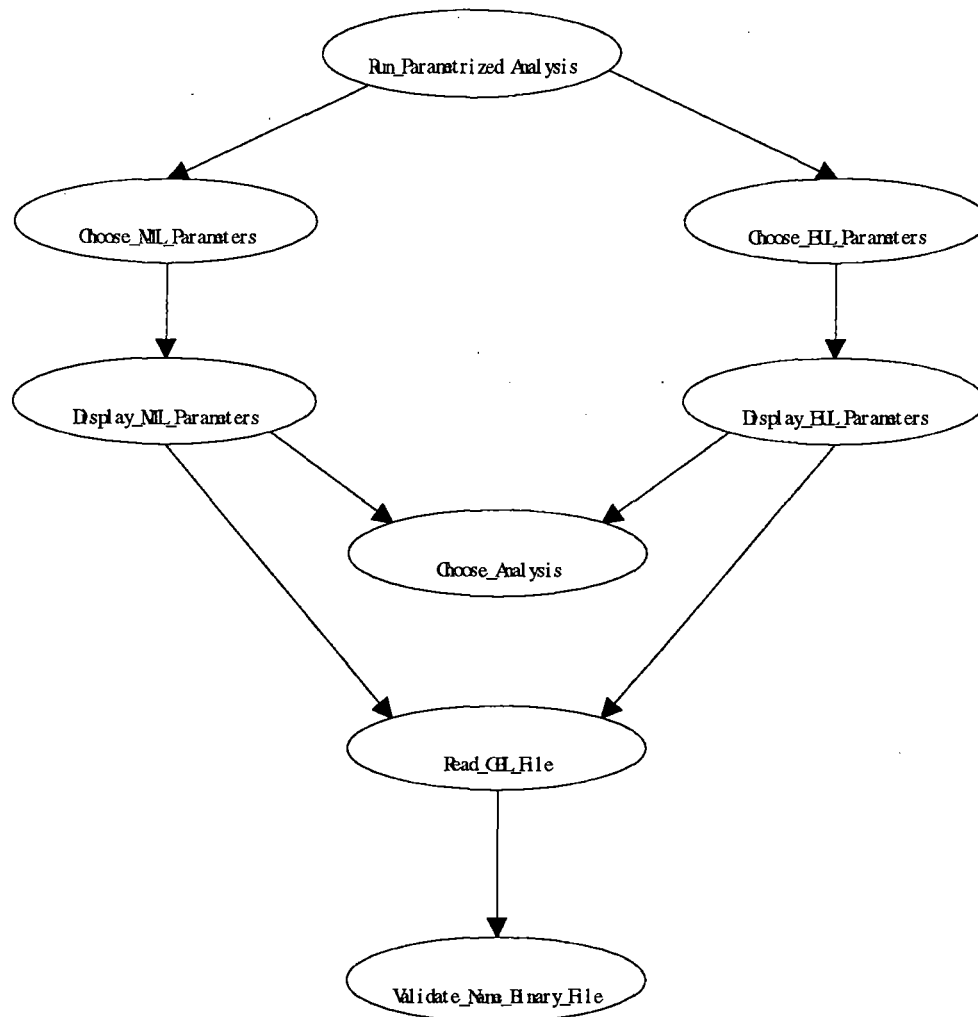
In this point, we will try to apply the methodology developed in the TRIDENT project to our application. The steps of this point are the following :

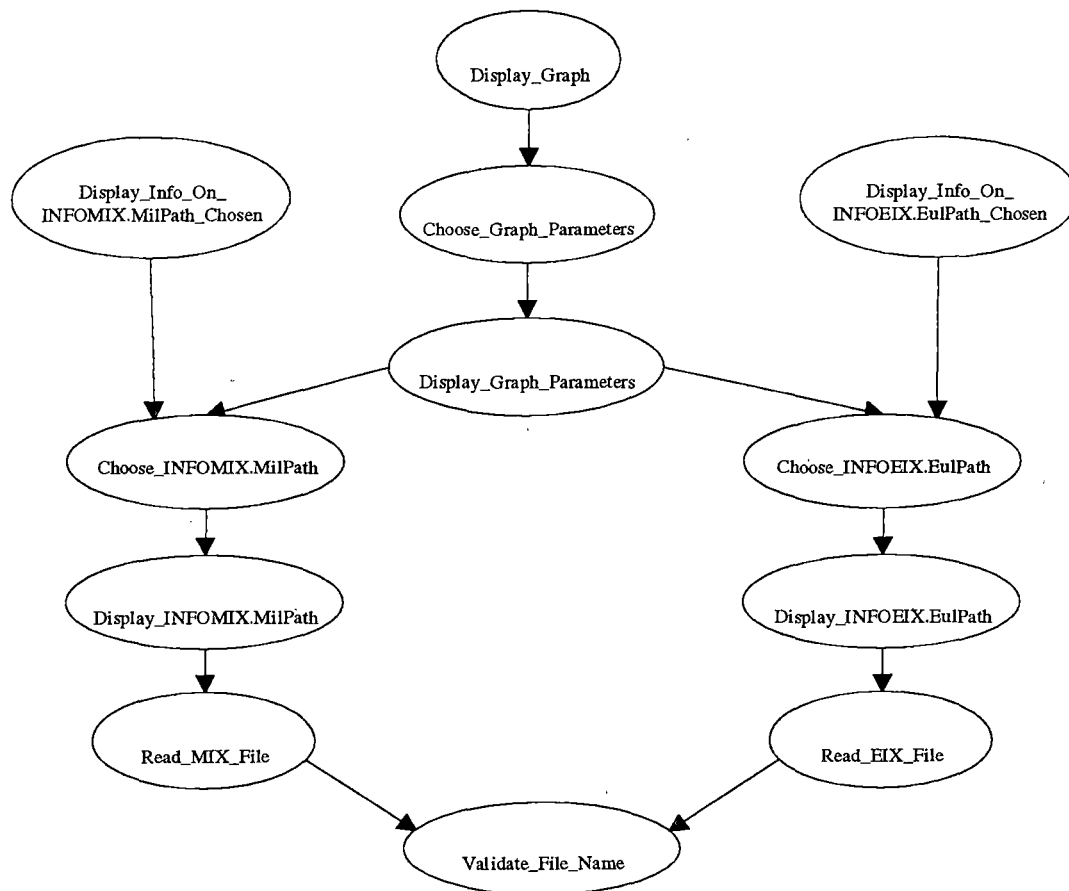
1. Construction of Functional Description's Hierarchy
2. Construction of the Control Objects of the Task's Hierarchy
3. Construction of Interactive Objects' Hierarchy

5.2.1 Construction of Functional Description's Hierarchy

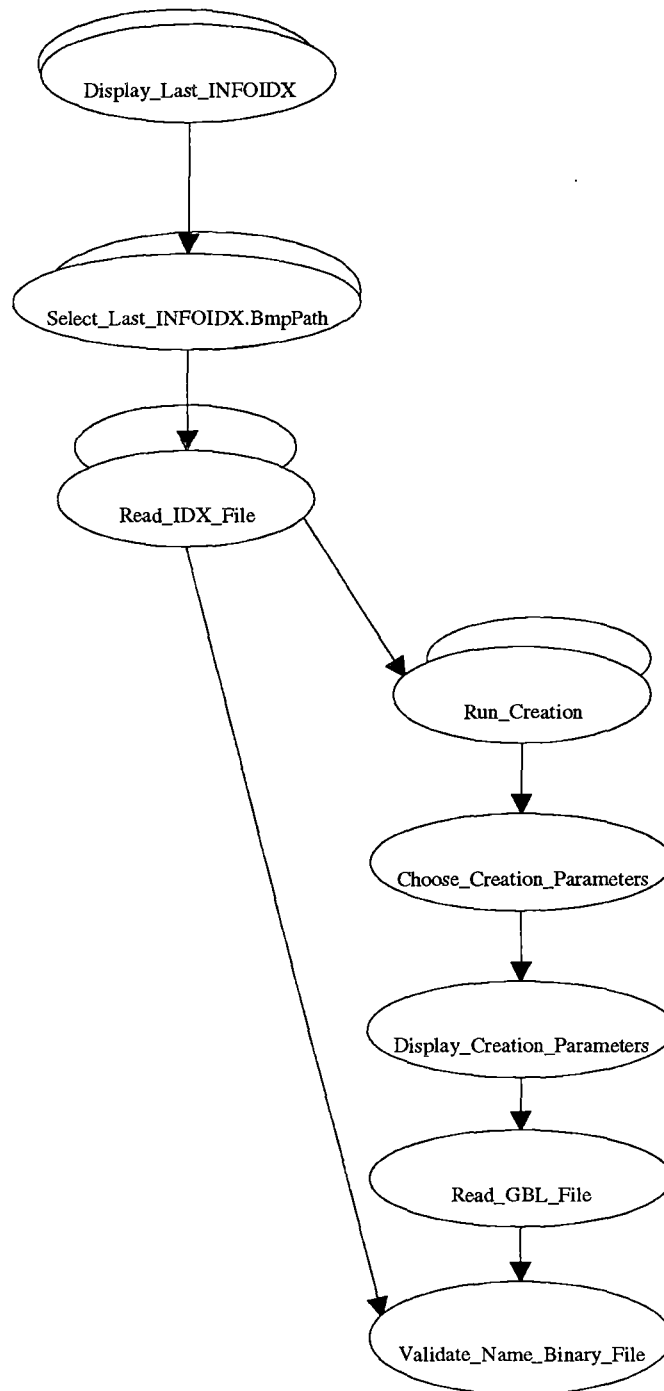
As we have already decided to distinguish each sub-task when realizing Activity Chaining Graphs, we will, in this point, distinguish each sub-task for the construction of the hierarchy.

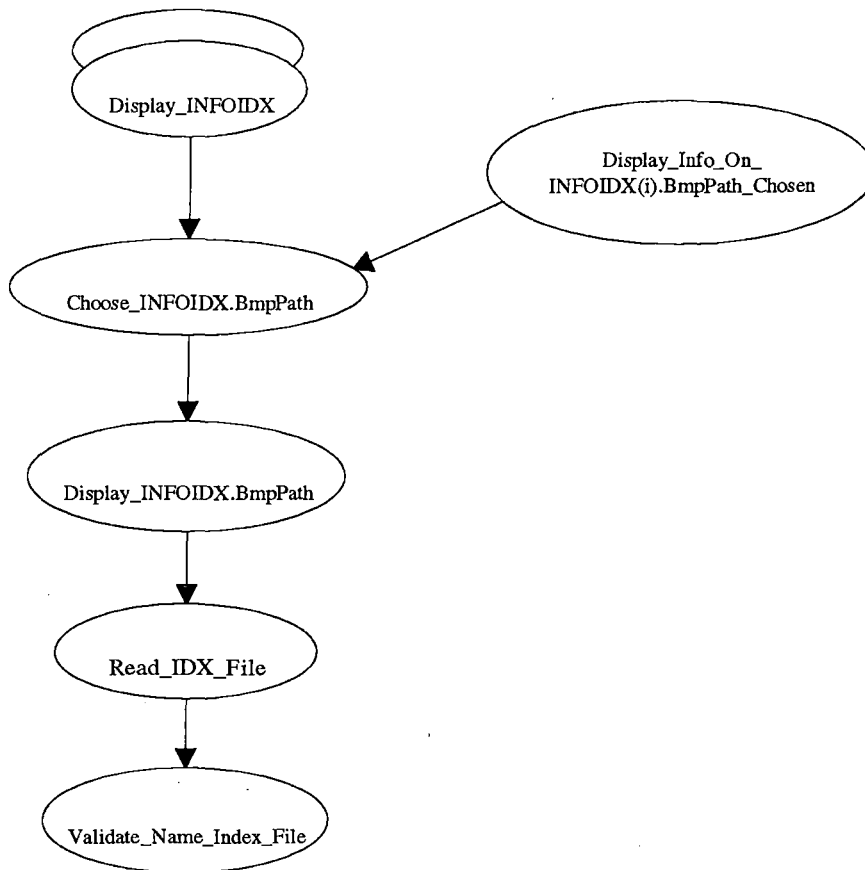
5.2.1.1 « Performing Analysis » Functional Description Hierarchy



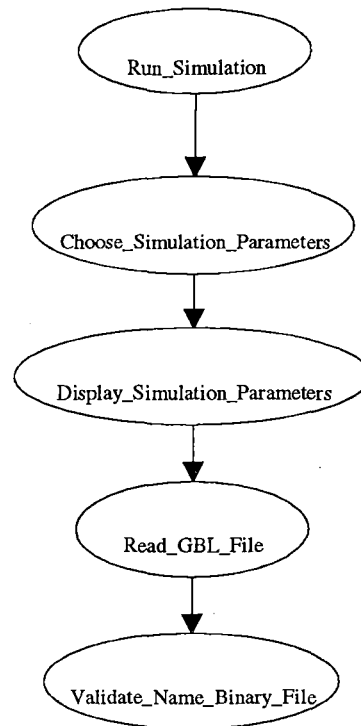
5.2.1.2 « Visualizing Analysis Results » Functional Description Hierarchy

5.2.1.3 « Creating Image » Functional Description Hierarchy

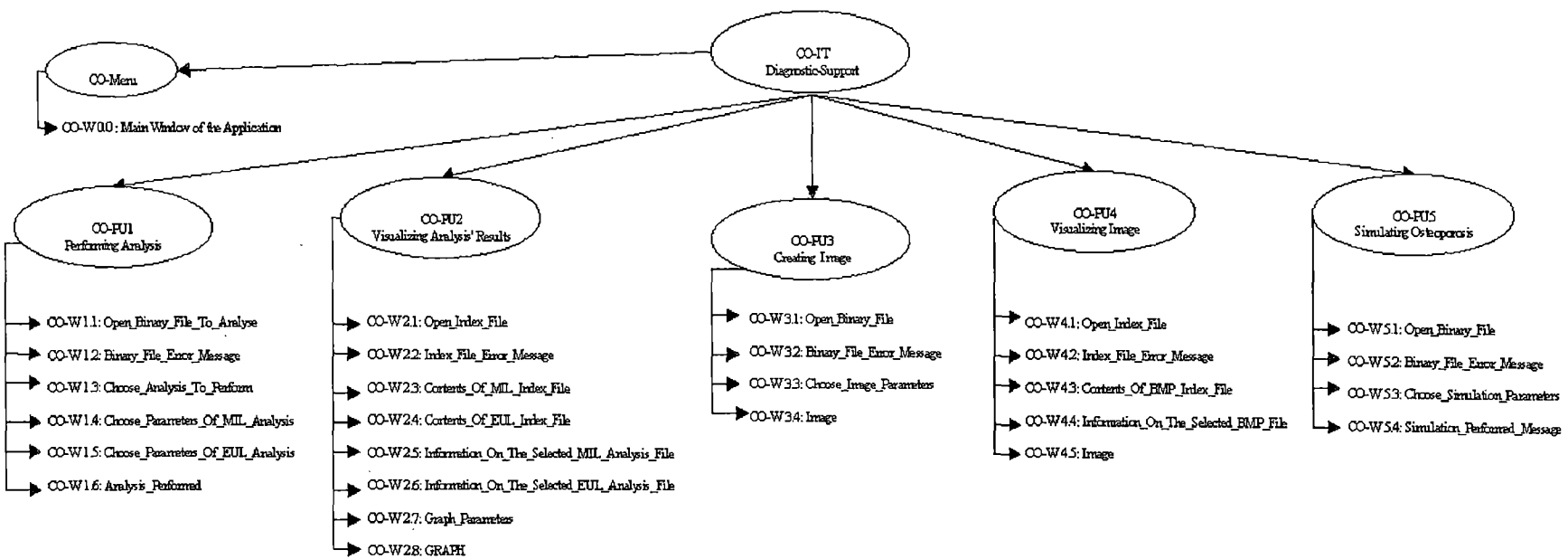


5.2.1.4 « Visualizing Image » Functional Description Hierarchy

5.2.1.5 « Simulating Osteoporosis » Functional Description Hierarchy



5.2.2 Construction of the Control Objects of the Task Hierarchy



5.2.3 Construction of Interactive Objects Hierarchy

For a question of legibility, we have chosen to describe the IOs Hierarchy for each windows of the application.

The following description is the IOs Hierarchy for the Control Object Menu (CO-W0.0) :

This window (which is the main window of the application) is composed of a Menu bar. This menu bar is composed of three menu which are the file menu, the option menu and the help menu. These menu are composed of menu items such as New Analysis, Exit, and so on ... Next to the menu bar, the window is also composed of a button bar which contains five buttons. That is what we describe next :

IO-Comp-W 0.0

Menu_Bar

Menu_File

Menu_Item_New_Analysis

Menu_Item_Open_Analysis

Sepatator

Menu_Item_Osteoporosis_Simulation

Separator

Menu_Item_Create_Image

Menu_Item_View_Image

Separator

Menu_Item_Exit

Menu_Option

Toggle_Menu_Item_ToolBar

Menu_Help

Menu_Item_About

Button_Bar

Bt_New_Analysis

Bt_Open_Analysis

Bt_Osteoporosis_Simulation

Bt_Create_Image

Bt_View_Image

The following description is the IOs hierarchy for the CO-W1.1 :

IO-Comp-W 1.1

Ed_Binary_File

File_Selection_List_Box

Drop-Down_Type_Of_File_List_Box

Directory_Selection_List_Box

Drop-Down_Drive_List_Box

Check_Box_Read_Only

Bt_OK
Bt_Cancel

The following description is the IOs hierarchy for the CO-W 1.2 :

IO-Comp-W 1.2
Lb_Error
Bt_OK

The following description is the IOs hierarchy for the CO-W 1.3 :

IO-Comp-W 1.3
Gr_Type_Of_Analysis
Rb_Euler
Rb_MIL
Bt_OK
Bt_Cancel

The following description is the IOs hierarchy for the CO-W 1.4 :

This window is composed of a label indicating the path of the binary file, another label indicating the total number of slices in the file, a group box for the view axis which is composed of three radio-button (one for each axis), another group box for the parameters which is composed of three label and three spin button (respectively one for the low limit, one for the high limit and the last for the angle). Then the window also contains three push button, one for OK, one for Cancel and the last one for Reset the default values of the radio-button and the spin buttons. That is what we describe next :

IO-Comp-W 1.4
Lb_Binary_File
Lb_Total_Number_Of_Slices
Gr_View_Axis
Rb_Z_Axis
Rb_X_Axis
Rb_Y_Axis
Gr_Parameters
Lb_Low_Limit
Spin_Bt_Low_Limit
Lb_High_Limit
Spin_Bt_High_Limit
Lb_Angle
Spin_Bt_Angle
Bt_OK
Bt_Cancel
Bt_Reset_Default

The following description is the IOs hierarchy for the CO-W 1.5 :

IO-Comp-W 1.5
 Lb_Binary_File
 Lb_Total_Number_Of_Slices
 Gr_View_Axis
 Rb_Z_Axis
 Rb_X_Axis
 Rb_Y_Axis
 Gr_Parameters
 Lb_Low_Limit
 Spin_Bt_Low_Limit
 Lb_High_Limit
 Spin_Bt_High_Limit
 Bt_OK
 Bt_Cancel
 Bt_Reset_Default

The following description is the IOs hierarchy for the CO-W 1.6 :

IO-Comp-W 1.6
 Lb_Analysis_Performed
 Bt_OK

The following description is the IOs hierarchy for the CO-W 2.1 :

IO-Comp-W 2.1
 cfr IO-Comp-W 1.1

The following description is the IOs hierarchy for the CO-W 2.2 :

IO-Comp-W 2.2
 Lb_Error
 Bt_OK

The following description is the IOs hierarchy for the CO-W 2.3 :

IO-Comp-W 2.3
 Lb_Binary_File
 Gr_File_Index
 Lb_File_Name
 List_Box_MIL_Analysis_File
 Bt_OK
 Bt_Information
 Bt_Cancel

The following description is the IOs hierarchy for the CO-W 2.4 :

IO-Comp-W 2.4
 Lb_Binary_File

- Gr_File_Index
 - Lb_File_Name
 - List_Box_EUL_Analysis_File
- Bt_OK
- Bt_Information
- Bt_Cancel

The following description is the IOs hierarchy for the CO-W 2.5 :

- IO-Comp-W 2.5
 - Lb_Information
 - Bt_OK

The following description is the IOs hierarchy for the CO-W 2.6 :

- IO-Comp-W 2.6
 - Lb_Information
 - Bt_OK

The following description is the IOs hierarchy for the CO-W 2.7 :

- IO-Comp-W 2.7
 - Lb_Data_File
 - Drop-Down_List_Box_Variables
 - Drop-Down_List_Box_Type_Of_Graph
 - Gr_Graph_Options
 - Gr_GRID
 - Rb_None
 - Rb_Horizontal
 - Rb_Vertical
 - Rb_Both
 - Drop-Down_List_Box_Style_Of_Graph
 - Gr_Statistics
 - Check_Box_Mean
 - Check_Box_MinMax
 - Check_Box_Standard_Deviation
 - Gr_Example_Of_Graph
 - GRAPH
 - Bt_OK
 - Bt_Cancel
 - Bt_Reset_Default

The following description is the IOs hierarchy for the CO-W 2.8 :

As we used the MDI property (see chapter 4), this window which is a child window, replace the menu bar of the parent window by its own menu bar which is different. That is the reason why we describe the new menu bar here.

IO-Comp-W 2.8

Menu_Bar

Menu_File

Menu_Item_New_Analysis

Menu_Item_Open_Analysis

Sepatator

Menu_Item_Osteoporosis_Simulation

Separator

Menu_Item_Create_Image

Menu_Item_View_Image

Separator

Menu_Item_Close

Separator

Menu_Item_Exit

Menu_Edit

Menu_Item_Copy

Menu_Graph

Menu_Item_Other_Variable

Cascading_Menu_Statistics

Toggle_Menu_Item_Mean

Toggle_Menu_Item_MinMax

Toggle_Menu_Item_StdDev

Separator

Menu_Item_New_Data

Menu_Option

Toggle_Menu_Item_ToolBar

Menu_Window

Menu_Item_Cascade

Menu_Item_Tile_Horizontal

Menu_Item_Tile_Vertical

Menu_Item_Arrange_Icons

Separator

Menu_Item_Close

Menu_Help

Menu_Item_About

Button_Bar

Bt_New_Analysis

Bt_Open_Analysis

Bt_Osteoporosis_Simulation

Bt_Create_Image

Bt_View_Image

GRAPH

The following description is the IOs hierarchy for the CO-W 3.1 :

IO-Comp-W 3.1

cfr. IO-Comp-W 1.1

The following description is the IOs hierarchy for the CO-W 3.2 :

IO-Comp-W 3.2

Lb-Error

Bt_OK

The following description is the IOs hierarchy for the CO-W 3.3 :

IO-Comp-W 3.3

Lb_Binary_File

Lb_Total_Number_Of_Slices

Gr_View_Flag

Check_Box_X_Axis

Check_Box_Y_Axis

Check_Box_Z_Axis

Gr_Parameters

Lb_Low_X

Lb_High_X

Spin_Bt_Low_X

Spin_Bt_High_X

Lb_Low_Y

Lb_High_Y

Spin_Bt_Low_Y

Spin_Bt_High_Y

Lb_Low_Z

Lb_High_Z

Spin_Bt_Low_Z

Spin_Bt_High_Z

Bt_OK

Bt_Cancel

Bt_Reset_Default

The following description is the IOs hierarchy for the CO-W 3.4 :

As already explained above, this window is also a child window, and thus replace the menu bar of the parent window by its own menu bar which is different. That is the reason why we describe the new menu bar here.

IO-Comp-W 3.4

Menu_Bar

Menu_File

Menu_Item_New_Analysis

Menu_Item_Open_Analysis

Sepatator

Menu_Item_Osteoporosis_Simulation

Separator

Menu_Item_Create_Image

Menu_Item_View_Image

Separator

Menu_Item_Close

```

        Separator
        Menu_Item_Exit
    Menu_Edit
        Menu_Item_Copy
    Menu_Option
        Toggle_Menu_Item_ToolBar
    Menu_Window
        Menu_Item_Cascade
        Menu_Item_Tile_Horizontal
        Menu_Item_Tile_Vertical
        Menu_Item_Arrange_Icons
        Separator
        Menu_Item_Close
    Menu_Help
        Menu_Item_About
    Button_Bar
        Bt_New_Analysis
        Bt_Open_Analysis
        Bt_Osteoporosis_Simulation
        Bt_Create_Image
        Bt_View_Image
    Win_IMAGE
        Image
        Gr_Info
            Lb_Binary_File
            Lb_BMP_File
            Lb_ViewFlag
            Lb_LowLimit
            Lb_HighLimit
    
```

The following description is the IOs hierarchy for the CO-W 4.1 :

```

IO-Comp-W 4.1
    cfr IO-Comp-W 1.1
    
```

The following description is the IOs hierarchy for the CO-W 4.2 :

```

IO-Comp-W 4.2
    Lb_Error
    Bt-OK
    
```

The following description is the IOs hierarchy for the CO-W 4.3 :

```

IO-Comp-W 4.3
    Lb_Binary_File
    Gr_File_Index
        Lb_File_Name
        List_Box_BMP_File
    Bt_OK
    Bt_Information
    Bt_Cancel
    
```

The following description is the IOs hierarchy for the CO-W 4.4 :

- IO-Comp-W 4.4
 - Bt_OK

The following description is the IOs hierarchy for the CO-W 4.5 :

- IO-Comp-W 4.5
 - cfr IO-Comp-W 3.4

The following description is the IOs hierarchy for the CO-W 5.1 :

- IO-Comp-W 5.1
 - cfr IO-Comp-W 1.1

The following description is the IOs hierarchy for the CO-W 5.2 :

- IO-Comp-W 5.2
 - Bt_OK

The following description is the IOs hierarchy for the CO-W 5.3 :

- IO-Comp-W 5.3
 - Lb_Binary_File
 - Lb_Total_Number_Of_Slices
 - Gr_Parameters
 - Lb_LowLimit
 - Spin_Bt_LowLimit
 - Lb_HighLimit
 - Spin_Bt_HighLimit
 - Lb_Indice
 - Spin_Bt_Indice
 - Bt_OK
 - Bt_Cancel
 - Bt_Reset_Default

The following description is the IOs hierarchy for the CO-W 5.4 :

- IO-Comp-W 5.4
 - Lb_Simulation_Performed
 - Bt_OK

6. Dialogue Specifications

« Normally, each dialogue layer should be specified in a notation that task analysts can understand, but can also be executed by some software tools. Multiple formalisms could be avoided by using ACG constructs. Most current approaches do not use common specifications : task analysts tend to use natural language descriptions for dialogue or task-related notations ; construction tools that execute (or generate) dialogues use formal languages (e.g., Petri nets, rule languages, attributed grammars), although they are unfamiliar to task analysts.

In this section, we will investigate some perspectives for modeling the dialogue in order to fill this important gap (Figure 5.27) »^[BODART95].

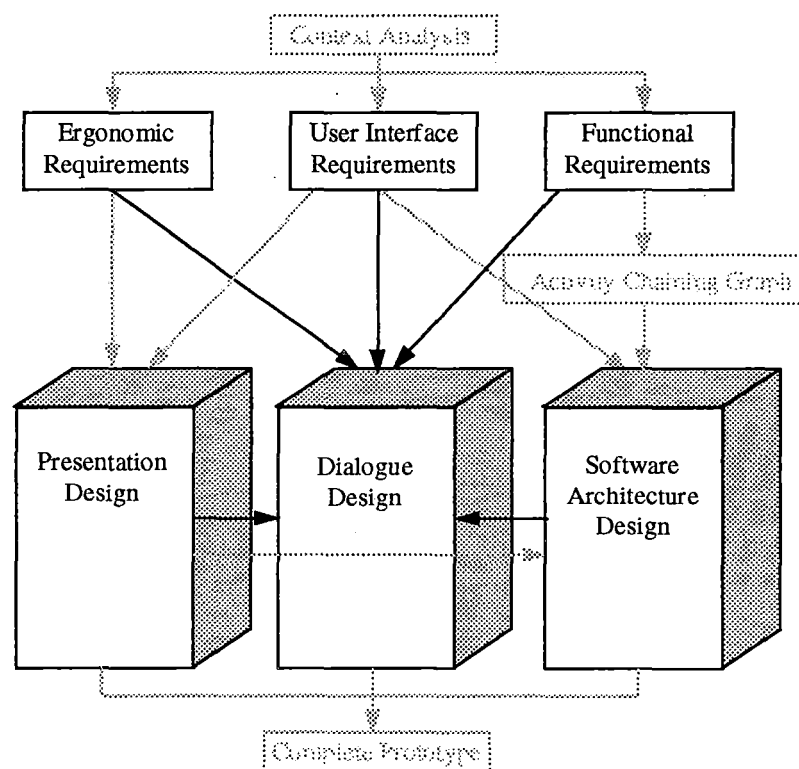


Figure 5.27 Outline of Activity 4.

« We retain the need for independence between the dialogue and the application functions. Such independence is easy to achieve for the presentation components, since control objects guarantee correspondence between these and the application functions. Control objects thus can guarantee the separation of the presentation and application objects. However, separation between dialogue is more critical : the functional logic of the ACG should not govern the progress of the dialogue, but the dialogue should be compatible with it, and clearly the ACG cannot be achieved »^[BODART95].

6.1 Methodological Considerations

The methodological considerations developed next are issued from the TRIDENT project.^[BODART95]

6.1.1 Dialogue content

Dialogue is related to four types of element (i.e., PU, window, CIO, and function) and can be described at three levels of abstraction (Figure 5.28) :

1. *inter-PU level* : the triggering of sub-tasks represented by PUs at the interactive task level ;
2. *intra-PU level* (or inter-window level) : the chaining of windows at the PU level ;
3. *intra-window level* : the behavioral dependencies among CIOs in the same window.

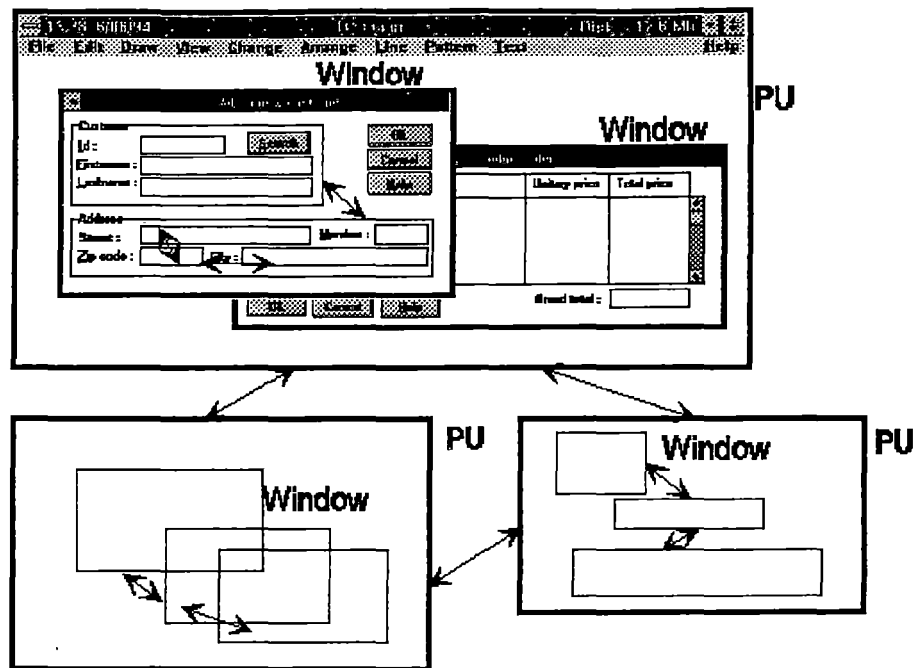


Figure 5.28 The dialogue levels.

CIOs linked by a bi-directional link (arrow) in Figure 5.28 become involved in the same dialogue at the inter-window level ; similarly, windows linked by a bi-directional arrow are involved in intra-PU dialogue, just as PUs involved at the inter-PU level are linked by such arrows.

6.1.2 Systematic approach for specifying a dialogue

Initially, the approach should be driven by task analysis:

- a list of dialogue states (e.g., default, display, history, object, initial, final, related/alternative) can be derived from the decomposition of a task into goals and sub-goals ;
- a list of dialogue transitions (e.g., triggers, actions, constraints) can be derived from the relationships between procedures implied by associated (sub-) task goals.

The approach iteratively refines dialogues, starting with high-level dialogues and ending with low-level ones. The following steps will achieve this refinement :

- *specifying inter-PU dialogues*, by using *presentation unit chaining graphs* that specify the conditions under which PUs are initialized, activated, and terminated. This graph should be expressed in terms of activation/deactivation of PU, serialization/parallelism, foreground/background, suspend/restore, ...

The graph can be derived from the PU's definition and from the decomposition of the interactive task into subtasks. Further support comes from dialogue ergonomic rules and from constructs for sequence and concurrency.

- *specifying intra-PU dialogues*, by using *window chaining graphs* that specify window manipulation operations. This graph should be expressed in terms of multi-windowing operations since multi-windowing is widely recognized as an effective support for multiple activities : creation/deletion, activation/deactivation, minimizing/maximizing, iconifying/restoring, ... Also, one can decide the window configuration and screen layout : partial or total tiling, partial or total overlapping, cascading.

Specifying this dialogue level should be driven by cognitive psychology principles related to multiple activities and ergonomic rules on multi-windowing. A tool that would be able to suggest the designer appropriate window chaining would be a significant contribution if this tool could intrinsically take care of ergonomic criteria (e.g., work load).

- *specifying intra-window dialogues*, by using interaction object graphs that should be expressed in terms of the many primitives provided by CIOs in a window : (un)display, (de)activation, (de)highlighting, inter-CIO influences, dynamic or function update, ...

Derivation of these dialogues should be driven by the four dialogue attributes for PUs, by interaction styles and by the functional logic underlying the ACG. Initially, the derivation only produces a skeleton of the interaction graph, leaving the designer with many further decisions on syntactic and

lexical level details. Decisions can be organized by formalisms such as the state transition diagram and the interaction object graph. UAN can also specify this detail.

One problem is the diversity of formalisms used across the different levels of dialogue specification.

6.2 Application of the TRIDENT Methodology

In this point, we will try to apply the methodology developed in the TRIDENT project to our application. The steps of this point are the following :

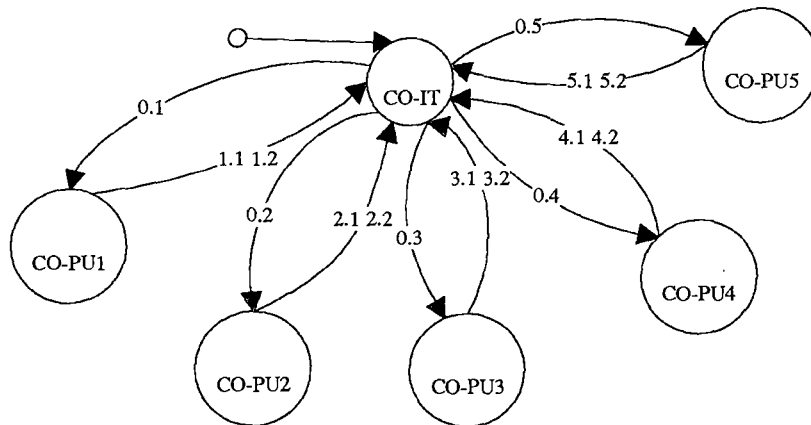
1. Inter-PU Dialogues Specifications
2. Intra-PU Dialogues Specifications
3. Intra-Window Dialogues Specifications

6.2.1 Inter-PU Dialogues Specifications

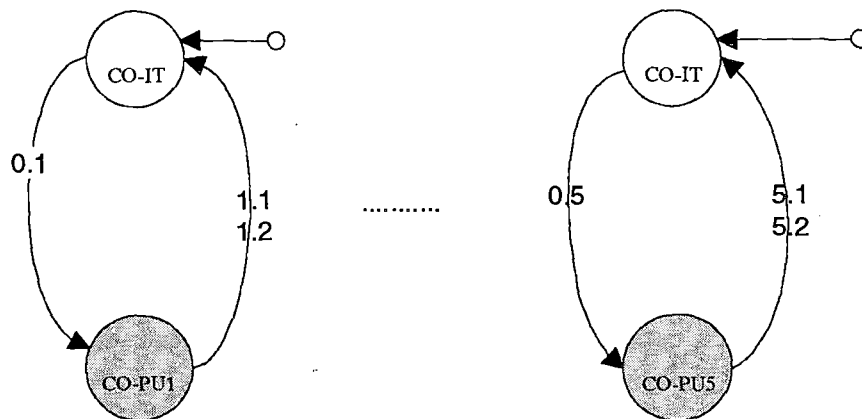
Script of Diagnostic Support (CO-IT)

Objects Used	
CO-Menu	
CO-PU1	
CO-PU2	
CO-PU 3	
CO-PU4	
CO-PU5	
Events Received	
0.1 Evt-New-Analysis	from CO-Menu
0.2 Evt-Open-Analysis	from CO-Menu
0.3 Evt-Create-Image	from CO-Menu
0.4 Evt-View-Image	from CO-Menu
0.5 Evt-Simulation	from CO-Menu
1.1 Evt-OK-New-Analysis	from CO-PU1
1.2 Evt-Cancel-New-Analysis	from CO-PU1
2.1 Evt-OK-Graph	from CO-PU2
2.2 Evt-Cancel-Graph	from CO-PU2
3.1 Evt-OK-Creation	from CO-PU3
3.2 Evt-Cancel-Creation	from CO-PU3
4.1 Evt-OK-View	from CO-PU4
4.2 Evt-Cancel-View	from CO-PU4
5.1 Evt-OK-Simulation	from CO-PU5
5.2 Evt-Cancel-Simulation	from CO-PU5

State-Transition Diagram



Before starting with the script itself, it is important to make some remarks about the above mentioned state-transition diagram (STD). Indeed, sometimes you can face very complex STD and it is useful to be able to decompose such a complex STD into less-complex STD. The previous STD is not a complex diagram, each couple of relations between the CO-IT and each of the CO-PU are independent each other, thus we could decompose it as follows :



The following symbole :



represent the fact that CO-PU can also be represented by a STD which is not showed here for a question of legibility. The complete STD for the CO-PU1 is represented at page 103 when we describe the intra-PU1 script.

Initial State

Begin

Activate Script-Menu
State=Diagnostic-Support

End

State-Transition

IF State=Diagnostic-Support **and** Evt-New-Analysis

Begin

Activate Script-Performing-New-Analysis
State=New-Analysis

End

IF State=Diagnostic-Support **and** Evt-Open-Analysis

Begin

Activate Script-Open-Analysis
State=Open-Analysis

End

IF State=Diagnostic-Support **and** Evt-Create-Image

Begin

Activate Script-Create-Image
State=Create-Image

End

IF State=Diagnostic-Support **and** Evt-View-Image

Begin

Activate Script-View-Image
State=View-Image

End

IF State=Diagnostic-Support **and** Evt-Simulation

Begin

Activate Script-Simulation
State=Simulation

End

IF State= New-Analysis **and** Evt-OK-New-Analysis

Begin

Deactivate Script-Performing-New-Analysis
State= Diagnostic-Support

End

IF State=New-Analysis **and** Evt-Cancel-New-Analysis

Begin

Deactivate Script-Performing-New-Analysis
State= Diagnostic-Support

End

IF State=Open-Analysis **and** Evt-OK-Open-Analysis

Begin

Deactivate Script-Open-Analysis

State= Diagnostic-Support

End

IF State=Open-Analysis **and** Evt-Cancel-Open-Analysis

Begin

Deactivate Script-Open-Analysis

State= Diagnostic-Support

End

IF State=Create-Image **and** Evt-OK-Create-Image

Begin

Deactivate Script-Create-Image

State= Diagnostic-Support

End

IF State=Create-Image **and** Evt-Cancel-Create-Image

Begin

Deactivate Script-Create-Image

State= Diagnostic-Support

End

IF State=View-Image **and** Evt-OK-View-Image

Begin

Deactivate Script-View-Image

State= Diagnostic-Support

End

IF State=View-Image **and** Evt-Cancel-View-Image

Begin

Deactivate Script-View-Image

State= Diagnostic-Support

End

IF State=Simulation **and** Evt-OK-Simulation

Begin

Deactivate Script-Simulation

State= Diagnostic-Support

End

IF State=Simulation **and** Evt-Cancel-Simulation

Begin

Deactivate Script-Simulation

State= Diagnostic-Support

End

6.2.2 Intra-PU Dialogues Specifications

In this point, we have only developed one intra-PU script in full details because we thought it was not relevant to perform the same task with all PU.

Script of Performing-New-Analysis (CO-PU1)

Objects Used

CO-Win 1.1
CO-Win 1.2
CO-Win 1.3
CO-Win 1.4
CO-Win 1.5
CO-Win 1.6

Used by CO-IT

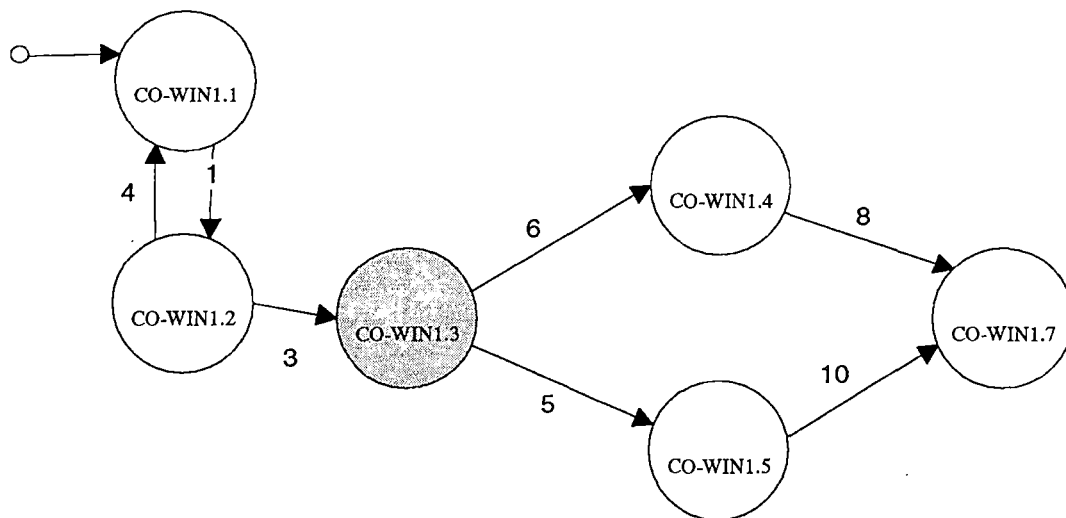
Events Received

1 Evt-OK-Binary-File	from CO-Win 1.1
2 Evt-Cancel-Binary-File	from CO-Win 1.1
3 Evt-Binary-File-Validated	from CO-Win 1.1
4 Evt-OK-Binary-File-Error-Msg	from CO-Win 1.2
5 Evt-OK-EUL-Analysis	from CO-Win 1.3
6 Evt-OK-MIL-Analysis	from CO-Win 1.3
7 Evt-Cancel-Choose-Analysis	from CO-Win 1.3
8 Evt-OK-MIL-Param	from CO-Win 1.4
9 Evt-Cancel-MIL-Param	from CO-Win 1.4
10 Evt-OK-EUL-Param	from CO-Win 1.5
11 Evt-Cancel-EUL-Param	from CO-Win 1.5
12 Evt-OK-Analysis-Performed-Msg	from CO-Win 1.6

Events Generated

Evt-OK-New-Analysis	to CO-IT
Evt-Cancel-New-Analysis	to CO-IT

State-Transition Diagram



Remark : the fact that the CO-Win 1.3 is in a grey color mean that this state is decomposed at page 105.

Initial State

Begin

Activate Script-Open-Binary-File
State=Open-Binary-File

End

State-Transition

IF State=Open-Binary-File **and** Evt-OK-Binary-File

Begin

Deactivate Script-Open-Binary-File
Activate Script-Validate-Binary-File
State=Validate-Binary-File

End

IF State=Open-Binary-File **and** Evt-Cancel-Binary-File

Begin

Deactivate Script-Open-Binary-File
Generate Evt-Cancel-New-Analysis
Deactivate Script-Performing-New-Analysis

End

IF State=Validate-Binary-File **and** Evt-OK-Binary-File-Error-Msg

Begin

Deactivate Script-Validate-Binary-File
Activate Script-Open-Binary-File
State=Open-Binary-File

End

IF State=Validate-Binary-File and Evt-Binary-File-Validated

Begin

Deactivate Script-Validate-Binary-File

Activate Script-Choose-Analysis

State=Choose-Analysis

End

IF State=Choose-Analysis and Evt-OK-EUL-Analysis

Begin

Deactivate Script-Choose-Analysis

Activate Script-EUL-Analysis

State=EUL-Analysis

End

IF State=Choose-Analysis and Evt-OK-MIL-Analysis

Begin

Deactivate Script-Choose-Analysis

Activate Script-MIL-Analysis

State=MIL-Analysis

End

IF State=Choose-Analysis and Evt-Cancel-Choose-Analysis

Begin

Deactivate Script-Choose-Analysis

Generate Evt-Cancel-New-Analysis

Deactivate Script-Performing-New-Analysis

End

IF State=MIL-Analysis and Evt-OK-MIL-Param

Begin

Deactivate Script-MIL-Analysis

Activate Script-MIL-Analysis-Performed

State=MIL-Analysis-Performed

End

IF State=MIL-Analysis and Evt-Cancel-MIL-Param

Begin

Deactivate Script-MIL-Analysis

Generate Evt-Cancel-New-Analysis

Deactivate Script-Performing-New-Analysis

End

IF State=EUL-Analysis and Evt-OK-EUL-Param

Begin

Deactivate Script-EUL-Analysis

Activate Script-EUL-Analysis-Performed

State=EUL-Analysis-Performed

End

```

IF State=EUL-Analysis and Evt-Cancel-EUL-Param
    Begin
        Deactivate Script-EUL-Analysis
        Generate Evt-Cancel-New-Analysis
        Deactivate Script-Performing-New-Analysis
    End

IF State=EUL-Analysis-Performed and Evt-OK-Analysis-Performed-
Msg
    Begin
        Deactivate Script-EUL-Analysis-Performed
        Generate Evt-OK-New-Analysis
        Deactivate Script-Performing-New-Analysis
    End

IF State=MIL-Analysis-Performed and Evt-OK-Analysis-Performed-
Msg
    Begin
        Deactivate Script-MIL-Analysis-Performed
        Generate Evt-OK-New-Analysis
        Deactivate New-Analysis
    End

```

6.2.3 Intra-Window Dialogues Specifications

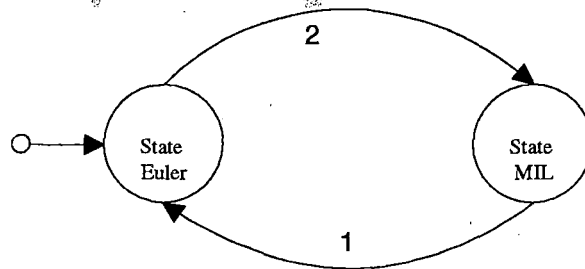
In this point, we have only developed one intra-window script in full details because we thought it was not relevant to perform the same task with all windows.

Script of Choose-Analysis-To-Perform (CO-WIN 1.3)

Objects Used	
Gr-Type-Of-Analysis	
Rb-Euler	
Rb-Mil	
Bt-OK	
Bt-Cancel	
Used by CO-PU1	
Events Received	
1 Evt-Rb-Euler-Selected	from Rb-Euler
2 Evt-Rb-Mil-Selected	from Rb-Mil
3 Evt-Bt-OK-Selected	from Bt-OK
4 Evt-Bt-Cancel-Selected	from Bt-Cancel
Events Generated	
Evt-OK-EUL-Analysis	to CO-PU1

Evt-OK-MIL-Analysis	to CO-PU1
Evt-Cancel-Choose-Analysis	to CO-PU1

State-Transition Diagram



Initial State

Begin

Display WIN 1.3
 PR-BR-Selected-Item (Rb-Euler)
 Bt-OK Active
 Bt-Cancel Active
 State=EULER

End

State-Transition

IF State=EULER And Evt-Rb-Euler-Selected

Begin

PR-BR-Selected-Item (Rb-Euler)
 State=EULER

End

IF State=EULER And Evt-Rb-Mil-Selected

Begin

PR-BR-Selected-Item (Rb-Mil)
 State=MIL

End

IF State=EULER And Evt-Bt-OK-Selected

Begin

Generate Evt-OK-EUL-Analysis
 Deactivate Script-Choose-Analysis-To-Perform

End

IF State=EULER And Evt-Bt-Cancel-Selected

Begin

Generate Evt-OK-Cancel-Choose-Analysis
 Deactivate Script-Choose-Analysis-To-Perform

End

IF State=MIL And Evt-Rb-Euler-Selected

Begin

PR-BR-Selected-Item (Rb-Euler)

```
                State=EULER
            End
    IF State=MIL And Evt-Rb-Mil-Selected
        Begin
            PR-BR-Selected-Item (Rb-Mil)
            State=MIL
        End
    IF State=MIL And Evt-Bt-OK-Selected
        Begin
            Generate Evt-OK-MIL-Analysis
            Deactivate Script-Choose-Analysis-To-Perform
        End
    IF State=MIL And Evt-Bt-Cancel-Selected
        Begin
            Generate Evt-OK-Cancel-Choose-Analysis
            Deactivate Script-Choose-Analysis-To-Perform
        End
```

7. Transformation of the TRIDENT Architecture in Visual Basic

In this point, we will first explain the method used in the TRIDENT project to transform the Trident architecture of an application into Visual Basic. The methode is explained in [JML95]. Next, we will compare it with the method we have used to program in Visual Basic.

7.1 The Visual Basic Architecture

Visual Basic is composed of the following objects :

▲ Concrete Interaction Objects (CIO) :

These objects contain their primitives (**CIO-Primitives**) and events they are able to generate (**CIO-Evt**) ;

▲ Forms :

The forms are contained in « **.frm** » files. A form correspond roughly with a logical window. We can have as much form as we want, a « **.frm** » file is created for each form and contains :

- the internal declarations of the form ;
- the description and property of the CIO belonging to the form ;
- the procedures associated to significant events generated by CIO : each procedure of that kind is attached to one and only one event of one and only one CIO. We can have as much procedures in the form as we want.

▲ Modules :

Modules are contained in « **.bas** » files. A module correspond roughly with a set of Basic procedures. we can have as much modules as we want and these modules contain :

- the internal declarations of the module ;
- the procedures that other forms or modules can use.

7.2 Transformation of the TRIDENT Architecture in Visual Basic

After this recall of the Visual Basic architecture, we can say that a form mix **Presentation** aspects (part 1 of the .frm file) and **Dialogue** aspects (part 2 and 3 of the .frm file). Observing the third part of a .frm file, we discover that we can consider it as a realization of an intra-window script. (**Script-Fen**). If we make correspond a Visual Basic form to each Trident logical window, the cut imposed by Visual Basic is perfectly consistent with the cut in CO and with the State-Transitions diagram of the script's specification.

Indeed :

1. the CO-Fct fit to a special module called **Fonctional Module** :
 - the functional primitives of the CO-Fct correspond to procedures (Proc-Fct) which are all contained in a Visual Basic module called **Functional Module** for the circumstances.
2. each CO-WIN and its AIO correspond to a Visual Basic Form :
 - CO-WIN declarations correspond to internal declarations of the form ;
 - AIO attributes correspond to form property ;
 - AIO primitives (AIO-Primitives) correspond to Visual Basic CIO primitives (CIO-Primitives) ;
 - events generated by AIO (AIO-Evt) correspond to concret events generated by Visual Basic CIO (CIO-Evt) ;
 - primitives offered by the CO-WIN (Win-Primitives) correspond to procedures written in the form procedure (Win-Prim-Proc) ;
 - the CO-WIN script corresponds to a set of procedures written in the form procedures (Script-Win-Proc) with a procedure for each **IF**... in the CO-WIN script ; these procedures are of the following form :
 SUB CIO-Evt (* Reaction to an event coming from a CIO*)
 ...
 ...
 call (Script-PU-Proc) (*call to a procedure of superior level in order to ,
 simulate the event generation*)
 ...
 ...

```
call (Proc-Fct) (*call to a procedure of the functional module to realize
a call to a primitive function*)
```

```
...
```

```
END SUB
```

- the form procedures can also contain other procedures (e.g. for internal management).

3. each CO-PU corresponds to a Visual Basic module called **PU-Model** :

- CO-PU declarations correspond to internal declarations of the PU-Module ;
- the primitives offered by the CU-PU (PU-Primitives) correspond to procedures written in the procedures of the PU-Module module (PU-Prim-Proc) ;
- the CO-PU script corresponds to a set of procedures written in the procedures of the PU-Module module (Script-PU-Proc) with a procedure for each **IF...** of the script of the CO-PU ; these procedures are of the following form :

```
IF state (*reaction to a call from a Script-Win-Proc procedure of the associated
VB form*)
```

```
...
```

```
...
```

```
call (Win-Prim-Proc) (*call to a lower level procedure to realize the
primitive call*)
```

```
...
```

```
...
```

```
call (Script-IT-Proc) (*call to a higher level procedure in order to si
mulate the events generation*)
```

```
...
```

- the procedures of the module PU-Module can contain other procedures.

4. the CO-PU (which is unique) corresponds to a unique Visual Basic module called **IT-Module** :

- the declarations of the CO-IT correspond to internal declarations of the IT-Module module ;

- the CO-IT script corresponds to a set of procedures written in the IT-Module module (Script-IT-Proc) with a procedure for each IF... of the CO-IT script ; these procedures are of the following form :

IF state (*reaction to a call from Script-IT-Proc procedure of the PU-Module module*)

...

call (PU-Prim-Proc) (*call to a lower level procedure to realize the primitive call*)

...

- the procedures of the IT-Module module can contain other procedures.

To summarize the situation, we obtain the following property :

- if we associate a form to each CO-WIN, its script (which is the script of the Intra-Window dialogue) corresponds to the Script-Win-Proc procedures of the form. In each of these procedures, the generation of an CO-Child event (here an CO-WIN) destined to the CO-Parent (here the CO-PU) can be replaced by a call to a procedure, specific to the module corresponding to the CO-Parent, in other words a Script-PU-Proc of the PU-Module module.
- if we associate a module to each CO-PU, its script (which is the script of the Intra-PU dialogue) corresponds to the Script-PU-Proc procedures of the PU-Module module. In each of these procedures, the generation of an CO-Child event (here an CO-PU) destined to the CO-Parent (here the CO-IT) can be replaced by a call to a procedure, specific to the module corresponding to the CO-Parent, in other words a Script-IT-Proc of the IT-Module module.
- if we associate a module to the CO-IT, its script (which is the script of the inter-PU dialogue) corresponds to the Script-IT-Proc procedures of the IT-Module module.
- the procedures described in these modules and which would not correspond to the reception of an event generated by a CO-Child, are primitives which can be used by CO-Parents. Thus, we realize the hypothesis that a CO-Parent can not receive events from its children, but the CO-Parent can use primitives belonging to its children.

7.3 Comparison with the Method we have used

The method we have used to develop the TBMAS User Interface is based on the traditional Visual Basic architecture as described in the section 7.1. We have as many « .frm » files as we have windows in the application. These « .frm » files contain :

- the internal declarations of the variables which are local to the form ;
- the description of the form and the properties of the CIO contained in the form ; this part of the file is produced automatically when designing the form with the Integrated Development Editor ;
- the procedures associated to the CIO Events.

We also have some « .bas » files which are modules containing procedures that can be used by other modules or by the forms of the application.

The things that are common to the two methods are the following :

- the separation of the Graphical User Interface and the application ;
- the internal declarations of the variables which are local to the form. The declarations are both contained in « .frm » files ;
- the description of the form and the CIO properties are also contained in the « .frm » files in both methods ;

The differences between the two methods lie in the place where the procedures associated to the CIO events are written. As a matter of fact, in the way we use Visual Basic, we write the basic code responding to particular events in their associated templates offered by Visual Basic. Hence, those basic code can manipulate any other form components. In other words, any CIO can control and manipulate any other one. That means that if you want to modify a CIO behavior, you do not know where to search because its behavior can be scattered in all the « .frm » files.

But, with the method developed in the Trident project :

A. The dialogue is **structured into three levels** which are the following :

- ▲ **Inter-PU level** which corresponds to the Interactive Task dialogue level,
- ▲ **Intra-PU level** which corresponds to the sub-task dialogue level,
- ▲ **Intra-Window level** which allows to realize the dialogue inside the form, or what we can call the procedural or functional dialogue.
- The dialogue of level 1 (Inter-PU) is realized in a basic file called the IT-Module.bas,
- The dialogue of level 2 (Intra-PU) is specified in as much « .bas » files as existing Presentation Units,
- The dialogues of level 3 (Intra-Window) are concentrated in the « .frm » files with a « .frm » file for each form.

B. The inter-level communication, which is realized at the script level by calls to primitives (belonging to a CIO-Child) by a CIO-Parent, has been realized in Visual Basic by a CALL. The communication from a Child-level to a Parent-level (which is realized in the script by the generation of an event) is translated in Visual Basic by the call (by the Child) of a CIO-Parent procedure. Now, this CIO-Parent procedure must realize what was specified in its script associated to the concerned message reception.

Thus, the programmed set is not only architected, structured respecting the separation of the application aspect from the interface aspect, but the dialogue programming is also structured and respects the three dialogue levels of the Trident methodology.

8. Conclusion on the Application of the TRIDENT Methodology

In this section, we have tried to realize a Costs-Benefits analysis about the application of the TRIDENT Methodology. The first step is to give ourselves some evaluation criterions. The following is the short list :

- Maintainability,
- Reusability,
- Simplicity.

These criterions are about the code of the application but we can also add some criterions concerning project management :

- The clarity and legibility of specifications,
- The speed of Development,
- The presence or absence of feed-back from users and actors,
- The improvement of inter-disciplinarity.

Concerning the code generated with the method developed by Jean-Marie Leheureux in [JML95] for the TRIDENT Project as explained previously, we can say that :

- The code is **maintainable** because the method uses an architectural approach which organizes the code in a recognizable manner. Indeed, the code is divided into dialogue level as explained in the previous section,
- The code is **reusable**. Indeed, object-oriented code is developed for reuse.
- The code is **more simple** because modular design removes the use of « spaghetti code ».

In the clarity and legibility of specifications point of view, we can say that each key activity of the TRIDENT Methodology has, in its results, graphical representation of the work done, such as the Entity-Relationship Model, the Activity Chaining Graph, the software architecture, the State-Transition dia-

gram, and son on ... So, we think that having such a graphical support for communication between development team and customer and user is easier and better to improve inter-disciplinarity and feed-back exchange between actors of the developemnt process because a graphical representation is easier to understand and is more natural.

We know that this type of representation for the functional specifications is not as totally correct and complete as the predicate logic can be, but we think that ACG, STD and even the scripts are better when you want to have as much feed-back as possible from development process actors. Indeed, it is not so difficult to understand the meaning of an ACG or of an STD or even of a script as it is for a page written in predicate logic. Of course, it isn't as much correct and complete.

In the speed of development point of view, we can say that it is always the same question : **is using a method improve the development speed ?**

You know, when programming you always think that it is loosing time that making a task analysis...

As we explained in the introduction of this chapter, we didn't use the TRIDENT Methodology to develop our application, thus we have some difficulties answering this question. Generally, we can say that a method improves the development speed only if it isn't too severe, too strict. And we think that the TRIDENT Methodology is easy to apply and is natural to apply. We don't have to much experience using it but we were master of it in a short time which mean that the method is not very difficult to understand. Hence, as the method is structured, architected, it should normally improve the development speed. As we said above, we don't have a lot experience using this method and the TRIDENT team don't have more of it because as we explained in the introduction of this chapter, the method, in the state within it is now, is only two months old. Anyway, we think it should be faster using this method than using an empirical development process as we did.

In the inter-disciplinarity point of view, we can say that :

- Forming UI specifications from the output of the task analysis promotes a better communicability of the work done by task analyst towards the software engineer,
- Guiding presentation design by ergonomic rules helps the graphical designer of the UI,
- Deriving an architecture from task analysis and presentation components is particularly well suited for the software engineer. The proposed systematic approach provides guidance grounded on the work of task analyst without forgetting specific goals of the information system,
- Specifying a high level conversation from task analysis should offer to the dialogue designer the faculty of conversation prototyping more rapidly and easily than with empirical manual methods. And that means to have feedback from user more rapidly,

It is visible that the TRIDENT Methodology promotes the improvement of the inter-disciplinarity.

In the feed-back from users and actors point of view, we can say that the approach of the problem offered by the TRIDENT Methodology, using a set of key activities allows to the development team to receive feed-back from users but also from any actor of the development process after each key activity. Indeed, after each key activity, the development team can meet the customer, the user to discuss about the activity results and this easier with the graphical results such as STD, ACG, Entity-Relationship Model. So, this allows the user to give a feed-back at each step of the development process as for the task analysis as well for the conversation prototype.

CHAPTER 6

Conclusion

The object of this thesis was to create a **MS Windows** application which consists in a **toolbox** for both **students** educational and research training. We have detailed throughout all the thesis the different **requirements** we were asked to fulfill and how we did so. We would like to recall here that we have respected **all** requirements. That is to say, we have fulfilled **functional requirements** as our application provides the user with the desired functionalities (MIL and Euler analysis, osteoporosis simulation, three-dimensional representation of trabecular bone, ...) and a user-friendly interface. Further more, we have also tried and succeeded in fulfilling **non functional requirements** such as performances increase (i.e., all analysis take less than 5% of the time needed by the existing programs to complete). Finally, we have succeeded preventing our application from being tied to a **specific configuration** as it only requires a standard Windows configuration, and, due to methodological choices and our software architecture, it could also be **migrated to an other platform more easily** than the existing programs.

When developing that thesis, we have been confronted with problems and requirements that allowed us **to extend our knowledge in multiple domains**. For example, we have gained knowledge in the biomedical field and in particular concerning the osteoporosis problematic, in reverse engineering methodology, in management of more complex projects, in team work management, ... We have also learned mastering powerful programming languages such as the C++ and the Visual Basic, along with a better understanding of MS Windows.

If we consider the **development process** we have followed, we can say that it presents some **strong points** but also some **weaknesses**. We think that **separating** user interface and functionality development is a good solution because it brings in a better **modularity** and it facilitates team work. Furthermore, the utilization of an **Object Oriented approach** is also very important for the same reasons and it should make the **maintenance** work easier. Nevertheless, our interface development has lacked a **methodology**. This would have been particularly critical if our user interface would have been more complex. Hence we think that adopting the **TRIDENT Methodology** for analyzing and

designing the interface as much as the functionalities is very important. Furthermore, it would provide a more **uniform** software development, from the user interface to the functionalities, as it is also based on the **Object Oriented model**.

We would like to insist on the advantages we have derived from using the TRIDENT Methodology. Indeed, if tomorrow, we are face designing and developing a very complex interactive task, we will use it without any doubts. Its architectural approach promotes team work, feed-back, but also help the development team to be master of its « subject ». Applying that methodology after we had complete the design and development of our application make us think to possible changes we could do and changes we did.

Finally, we would like to introduce in the last chapter **further developments** that seem interesting to consider. These take into consideration next generation operating system such as **MS Windows NT** which provide full 32-bit support and which look promising to provide even better performances. We also present some other improvements susceptible to make **interaction** with other applications easier for the user, and aimed at providing him with a better **three-dimensional representations** of digitized trabecular bone specimen.

CHAPTER 7

Further Developments

In this chapter, we discuss what further developments of the TBMAS program might be interesting to consider in the future. These developments should increase performances obtained and provide a better user feedback.

<i>1. Windows NT and 32-bit Programming</i>	2
<i>2. Providing OLE Support</i>	3
<i>3. Texture Mapping</i>	5

1. Windows NT and 32-bit Programming

In this section we discuss the advantages that could result from migrating our program to a 32-bit Windows NT application. We have based this section on [Myers&Hamer93].

Windows NT applications benefit from the following **advantages** :

- true **multitask** operating system
- full **32-bit** processing
- **linear memory** addressing
- extended **graphical** functionalities
- ...

Using **32-bit** processing can lead to a spectacular **improvements** in **memory access** and **computations** speed. These are aspects that have been largely developed in our program. Indeed, migrating our application to the Microsoft Windows NT operating system could provide us with even **better global performances**. However we already exceeded our expectations in increasing performance level dramatically (see chapter 6 for actual performance evaluation).

Further more, Windows NT provides a **better management** of **virtual memory** along with the possibility to address up to 4 gigabytes of virtual memory.

Finally, Windows NT brings in substantial **ameliorations** to **DLL management**. For example, thanks to linear memory addressing, there are no more problems caused by DLL routines addressing the user stack. Further more, it is now possible to use all C execution routines inside a DLL and its entry and exit point mechanisms have been much simplified.

2. Providing OLE Support

In this section we show what advantages could result from the utilization of the OLE (Object Linking and Embedding) technique in our program.

OLE is a set of **protocols** and procedures introduced in 1988 by Aldus corporation and since then introduced by Microsoft in its operating system for creation and maintenance of **composite files** ([Myers&Hamer93]). External data blocks inside a composite file are called **objects**, the application which receives data objects and creates a composite file is called an **OLE client**, the application which exports data objects to other applications is called an **OLE server**. The interactions between OLE client and server are illustrated below in **Figure 8.1**.

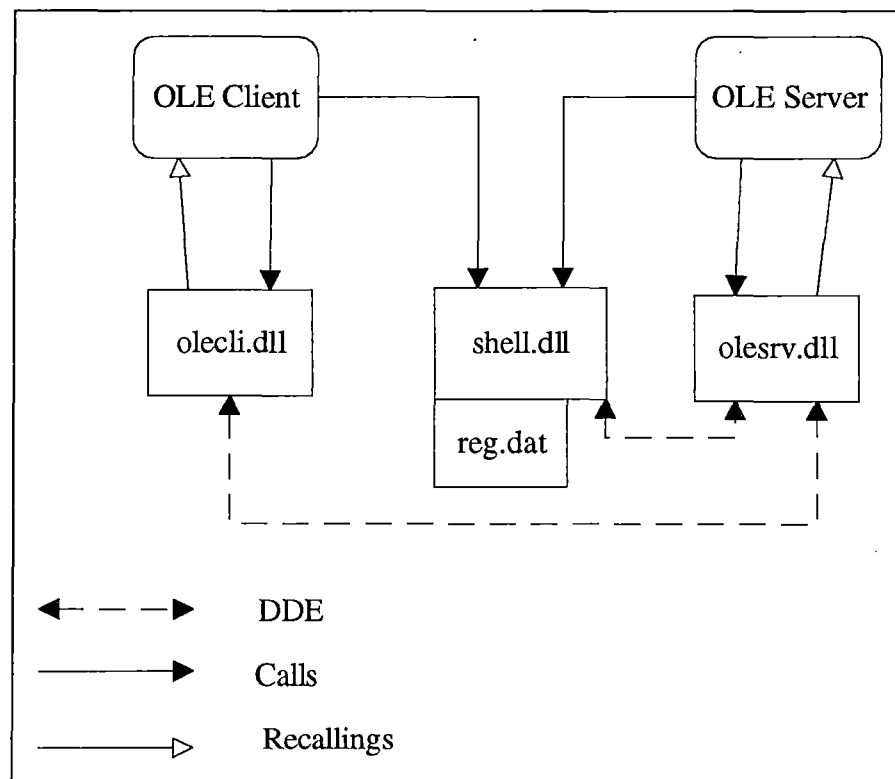


Figure 8.1 OLE client and server interactions^[Myers&Hamer93]

When the user starts an OLE operation in an OLE client application, calls are sent to the **OLESRV** library through the **OLECLI** library. **OLESRV** will then send data to the appropriate OLE server application. When that operation is finished, updated data are sent back to the client through calls between **OLESRV** and **OLECLI** libraries.

The first mechanism provided by OLE to transfer data from a server to a client is called **object embedding**. In this case, a data **object** is fully and independently **incorporated** to the composite file. But the object knows where he comes from and if the user double-click on it, the server application will be started and the object may then be edited. When the server is closed, the updated data object is incorporated to the composite file.

The second mechanism provided by OLE is called **object linking**. In this case, the client application does not include a copy of the data object but a **link** to the appropriate server application. If the data object is modified in the server application, then the modifications will be reported to the client as the composite file contains links to data objects.

We have discussed in chapter 6 our program relations with other applications, using **OLE** protocols instead or as a complement of the cut and paste featured in our program might provide more **user comfort**. For example, he might perform a MIL analysis, create a chart based on the results and insert the chart into its word processor. Then he might want to change chart aspect. By simply double clicking on the object he would then be able to make the desired changes.

3. Texture Mapping

We discuss in this section the texture mapping principles. Texture mapping might be used to give the user a better feedback since it would enabled him to have a three-dimensional visualization of the trabecular bone specimen.

« Texture mapping was one of the first developments towards making images of three-dimensional objects more interesting and apparently more complex. The original motivation for texture mapping was to diminish the shiny plastic effects produced by the simple Phong¹ reflection model and to enable different objects to exhibit different surface properties (apart from the trivial distinction of color) »^[Watt93]. Most texture mapping methods do not give the impression of actual surface perturbation but modulate the color of a surface using repeating motifs or a frame-grabbed image.

« By far and away the most common form of texture mapping involves the use of a two-dimensional texture domain $T(u,v)$ »^[Watt93]. We could create the **two-dimensional texture domain** of a trabecular bone specimen using the same **technique** as for **gray scaled** bitmaps (see chapter 6). Further more, the bone specimen can be considered as 4 cube or box. Hence it would be necessary to compute a texture for each box sides. All textures would then be unified to form the texture domain $T(u,v)$.

Once the texture domain is known, the **texture process** can be described as a **two transformations** process as shown on the next page in

Figure 8.2. *« The first transformation, sometimes known as surface parametrization, takes the two-dimensional texture pattern and glues it on the object. The second transformation is the standard object to screen space mapping »^[Watt93].* Hence the object we want to map to is a **cube**, the first transformation would not be difficult as cubic surfaces parametrizations are

¹ The Phong reflection model is the de facto reflection model used in the computer graphics community

readily available. The second transformation implies a pixel-by-pixel ordering of each polygon for Z-buffer hidden surface removal and other algorithms. The easiest way to do this is by **inverse mapping** but we do not deepen the subject as it not the purpose of this section to present a complete texture mapping theory.

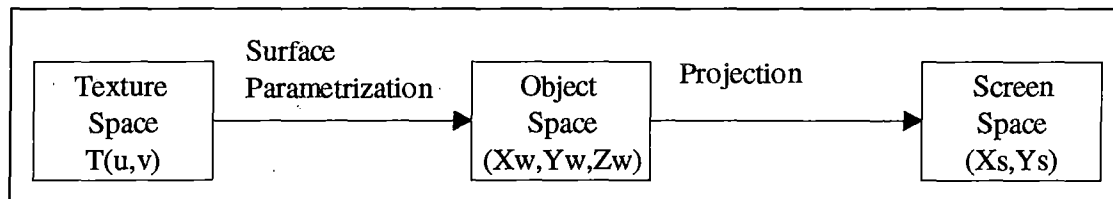


Figure 8.2 Overall texture mapping process^[Watt93]

We present below in **Figure 8.3** an illustration of what could be a trabecular bone three-dimensional representation using texture mapping.

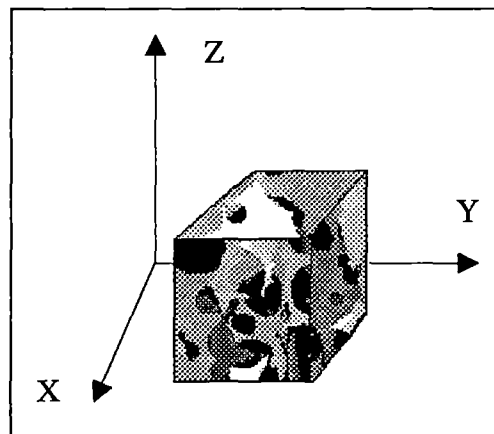


Figure 8.3 Trabecular bone representation with texture mapping

CHAPTER 8

References

<i>1. Books</i>	2
<i>2. Articles</i>	4

1. Books

- [Booch94]

G. Booch, *Analyse & conception orientées objets*, Addison-Wesley, Paris, France, 1994

- [Entsminger92]

Gary Entsminger, *Secrets of the Visual Basic™ for Windows™ Masters*, SAMS Publishing, A division of Prentice Hall Computer Publishing, 11711 Noth College, Carmel, Indiana 46302 USA, 1994

- [Foley&al93]

J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes., *Computer Graphics principles and practice second edition*, Addison-Wesley, Reading, Massachusetts, USA, 1993

- [Hansson-Keller, 1995]

T. Hansson and T. S. Keller, *The Lumbar Spine, Second Edition*, Ed. Sam Wiesel, Jame Weinstein et al., W.B. Saunder Company, 1995

- [JVD95]

J. Vanderdonckt, *Guide Ergonomique des Interfaces Homme-Machine*, Presses Universitaires de Namur, Namur, Belgique, 1994

- [Murray&vanRyper94]

J.D. Murray and W. vanRyper, *Encyclopedia of graphics file formats*, O'Reilley & associates, Sebastopol, Californie, USA, 1994

- [Myers&Hamer93]

B. Myers and E. Hamer, *Windows NT programmation*, Sybex, Paris, France, 1993

- [Norton&Yao92]

P. Norton and P. Yao, *Programmation Windows 3.1*, Sybex, Paris, France, 1992

- [Tornsdorf&al92]

H. Tornsdorf, M. Tornsdorf, I. Bauder, J. Bär, H. Frater and M. Schüller , *Le grand livre de Microsoft Windows 3.1*, Micro Application, Paris, France, 1992

- [Watt93]

A. Watt, *3D computer graphics second edition*, Addison-Wesley, Wokingham, England, 1993

2. Articles

- [Bodart89]

F. Bodart , *Cours d' IHM*, Facultés Universitaires de Namur, 1989 .

- [BODART95]

F. Bodart, A-M. Hennebert, J-M. Leheureux, I. Provot, J. Vanderdonckt, G. Zucchinietti, « Key Activities for a Development Methodology of Interactive Applications », *Critical Issues in User Interface Systems Engineering*, Chapter 7, D. Benyon and Ph. Palanque (Eds.), Springer-Verlag, Berlin, 1995

- [Brodie&Stonebraker93]

M. Brodie and M. Stonebraker, « DARWIN : On the Incremental Migration of Legacy Information Systems », *GTE Laboratories Technical Report*, March 1993.

- [Chevalier et al, 1992]

F. Chevalier, A. M. Laval-Jeantet, M. Laval-Jeantet, C. Bergot, « CT Image Analysis of the Vertebral Trabecular Network in vivo », *Calcified Tissue International*, Springer-Verlag, New-York Inc, USA, © 1992.

- [Dubois93]

E. Dubois, *Cours de MDL*, Facultés Universitaires de Namur, 1993.

- [Feldkamp et al, 1989]

Lee A. Feldkamp, Steven A. Goldstein, A. Michael Parfitt, Gerald Jesion, Michael Kleerekoper, « The Direct Examination of Three-Dimensional Bone Architecture In Vitro by Computed Tomography », *Journal of bone and Mineral Research*, Volume 4, Number 1, Mary Ann Liebert, Inc., Publishers, 1989

- [Frost1993]

H. M. Frost, « Suggested fundamental concepts in skeletal physiology », *Calc Tiss Int*, volume 52, pages 1-4, 1993

- [JML1995]

J-M. Leheureux, « Transformation de l'architecture TRIDENT d'une application en Visual Basic », Diffusion interne au projet TRIDENT, FUNDP, Institut d'Informatique, 20 avril 1995

- [JVD93]

J. Vanderdonckt, « A Corpus of Selection Rules for Choosing Interaction Objects », TRIDENT Project, FUNDP, Institut d'Informatique, Technical Report, August 1993

- [Keller et al, 1989a]

Tony S. Keller, T. H. Hansson, A. C. Abram, D. M. Spengler, M. M. Panjabi, « Regional variations in the compressive properties of lumbar vertebral trabeculae : Effects of disc degeneration », *Spine*, Volume 14, pages 1012-1019, 1989a

- [Keller et al, 1989b]

Tony S. Keller, D. M. Spengler, « Regulation of bone stress and strain in the immature and mature rat femur », *Journal of Biomechanics*, volume 22, pages 1115-1128, 1989b

- [Keller et al, 1992a]

Tony S. Keller, Eko Moeljanto, J. A. Main, D. M. Spengler, « Distribution and Orientation of Bone in the Human Lumbar Vertebral Centrum », *Journal of Spinal Disorders*, Volume 5, Number 1, pages 60-74, Raven Press, Ltd., New-York, USA, © 1992

- [Keller et al, 1992b]

Tony S. Keller, M. Zhu, M. H. Pope, « Fracture risk associated with changes in vertebral architecture : Implications for the aging spine », *International Society for the Study of the Lumbar Spine*, Chicago, IL, pages 114-115, 1992b

- [Keller et al, 1993]

Tony S. Keller, I. Ziev, E. Moeljanto, D. M. Spengler, « Interdependence of Lumbar Disc and Subdiscal Bone Properties : A Report of the Normal and Degenerated Spine », *Journal of Spinal Disorders*, Volume 6, Number 2, pages 106-113, Raven Press, Ltd., New-York, USA, © 1993

- [Keller, 1994]

Tony S. Keller, « Predicting the Compressive Mechanical Behavior of Bone », *Journal of Biomechanics*, Volume 27, Number 9, pages 1159-1168, Copyright © 1994 Elsevier Science Ltd., Printed in Great Britain, 1994

- [Keller, 1995]

Tony S. Keller, « Increased Risk of Fracture Associated with Changes in Vertebral Structure in the Aging Spine », *ASME Bioengineering Conference*, Breckinridge, CO, June 28-July 2, 1995

- [Kuo-Carter, 1991]

Arthur D. Kuo and Dennis R. Carter, « Computational Methods for Analyzing the Structure of Cancellous Bone in Planar Sections », *Journal of Orthopaedic Research*, 9:918-931, Raven Press, Ltd., New-York, USA, © 1991 Orthopaedic Research Society

- [MDNN94]

Rick Raddatz and Ken Bergmann, « Programming with Microsoft Visual Basic : An architectural approach for Visual Basic Information Systems », *Microsoft Developer Network News, Technical Resources for Developers*, November 1994, Volume 3, Number 6, page 6.

- [Meyer85]

B. Meyer, « On formalism in specifications », *IEEE Software*, January 1985, pages 6-26.

- [Moeljanto91]

E. Moeljanto, *user guide to custom imaging software*, Department of Orthopaedics & Rehabilitation, Vanderbilt University, Nashville, 1991.

- [Mylopoulos95]

J. Mylopoulos, « Conceptual Modelling for Information System Engineering », *Lecture Series presented at the University of Namure*, April 24 - May 4 1995.

- [Parnas72]

Parnas, « On the criteria to be used in decomposing system into modules », *communication of the ACM*, 1972 vol.15, page 12.

- [Sacre&al92]

B. Sacre, I. Provot-Scare and J. Vanderdonckt , *Une description orientée objet des objets interactifs abstraits utilisés en interface homme-machine*, Facultés Universitaires de Namur, Octobre 1992.
- [TRIDENT93]

J. Vanderdonckt, « Révision du document concernant la dérivation de style(s) d'interaction », Diffusion interne du projet TRIDENT, FUNDP, Institut d'Informatique, 3 novembre 1993
- [Vanderdonckt92]

J. Vanderdonckt, *Corpus ergonomique minimal des applications de gestion*, Facultés Universitaires de Namur, Octobre 1992.
- [Wolff1892]

J. Wolff, « Das Destez der Transformation der Knochen », *Hirschwald*, Berlin, 1892
- [Zetterberg et al, 1990]

C. Zetterberg, S. Mannius, D. Mellström, et al, « Osteoporosis and back pain in the elderly. A controlled epidemiologic and radiographic study », *Spine*, volume 15, pages 783-786, 1990
- [Zetterberg et al, 1994]

C. Zetterberg, Ä. Sjöstedt, L. Ziden, et al, « Epidemiology of hip fractures in Göteborg, Sweden 1940-1991 », *Scandinavian Orthopaedic Association*, Proceeding of the 47th Assembly, Reykjavik, Iceland, June 8-11. *Acta Orthop Scand* 1994 ; 65(suppl260)30.
- [Zhu et al, 1994]

Mengke Zhu, Tony S. Keller, Eko Moeljanto, Dann M. Spengler, « Multiplanar Variations in the Structural Characteristics of Cancellous Bone », *Bone*, Volume 15, Number 3, pages 251-259, 1994 Copyright © 1994 Elsevier Science Ltd., Printed in the USA

APPENDIX A

Conventional Stereology

In this chapter we describe some Conventional Stereology principles which are explained in [Kuo-Carter, 1991].

<i>1. Introduction</i>	2
<i>2. The Directed Secant Method</i>	3
<i>3. Methods</i>	5
3.1 Phase Distribution Method	7
3.2 Primary Direction Method	9
<i>4. Practical Considerations</i>	13

1. Introduction

Stereologic studies of trabecular bone have in the past been based primarily on polar plots of the mean intercept length (MIL), as described below. When characterizing the orientation of a material, the plot of the MIL is fit to a single ellipse. Because the principal axes of an ellipse are orthogonal, this technique presupposes that the material being studied is orthotropic. This condition must be verified when applied to trabecular bone, in light of the objections to the trajectorial theory.

We will describe below computational methods which can be used in stereologic analysis of a two-dimensional section of a substance to test for orthotropy within the plane, to give a measure of the isotropy or departure of isotropy that is superior to conventional measures, to provide quantitative evidence of the number and degree of orientations, and to test models of the material architecture directly - all of which are readily applicable to developing a better understanding of the relationship between mechanical loading and trabecular bone architecture.

2. The Directed Secant Method

Stereology uses statistical measures gathered from two-dimensional sections of a substance to infer information concerning the structure of the material. The stereologic technique most often used for analysis of trabecular bone morphology is the *directed secant method*.

This method calls for laying a grid of parallel lines, at an angle Θ about an arbitrary axis, across the section, as in Figure 1A, and counting the intersections between the test lines and the boundaries of the « trabeculae ». This process is repeated with the test lines arranged at a series of angles Θ from 0 to 180°.

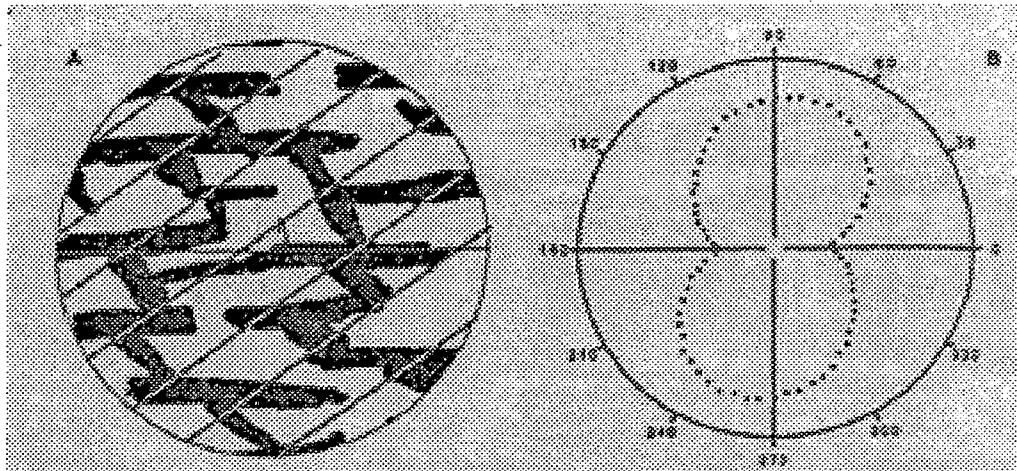


Figure 1 A : Sample pattern exhibiting obvious nonorthotropic characteristics. Test lines at angle Θ shown. **B :** Rose of intercepts for sample pattern

The total number of intersections per unit test line length $[I_L(\Theta)]$ is recorded for each angle, and its polar plot is often referred to as the « rose of intercepts » as shown in Figure 1B. Alternatively, the inverse of $I_L(\Theta)$, the mean intercept length $MIL(\Theta)$ is plotted. Typically, a single ellipse is fit to the polar plot of $MIL(\Theta)$. However, because an ellipse has orthogonal principal axes, using these axes to characterize the material orientation may be

misleading if the substance is not orthotropic, even if the ellipse provides a good fit.

The degree of orientation is another important stereological measure. It is easily determined based on the stereological measure of the boundary length per unit area B_A (also referred to as perimeter length density) :

$$B_A = \frac{\pi}{22} \overline{I_L(\Theta)} \quad (1)$$

B_A is used to find the degree of orientation

$$\% \text{ orientation} = \frac{100 [I_L(\max) - I_L(\min)]}{B_A} \quad (2)$$

This measure is most applicable for materials possessing a single direction of orientation ; for materials with multiple directions of orientation, the results suggest little about the actual morphology of the test substance.

3. Methods

It has been noted that the directed secant method, when applied to a sample pattern of straight lines with orientation Φ with respect to the horizontal, will produce a rose of intercepts such as shown in Figure 2.

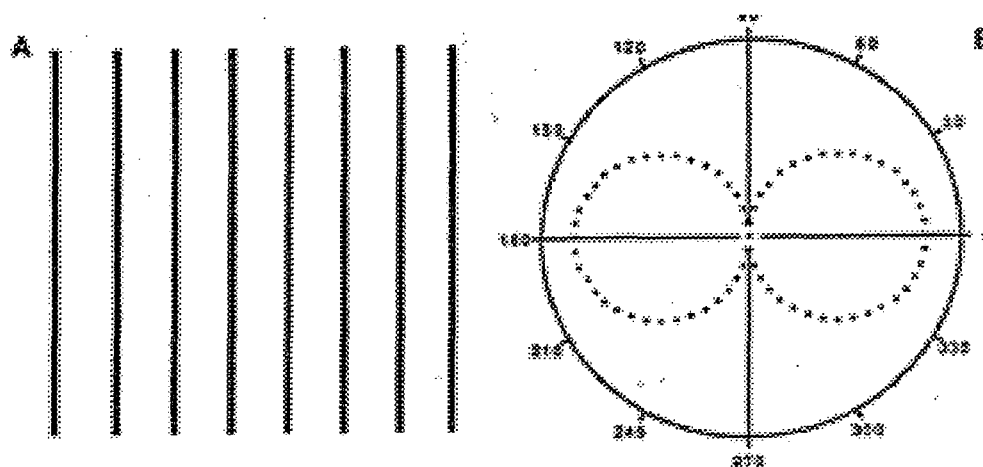


Figure 2 A : Sample pattern of straight lines with orientation $\Phi=90^\circ$. **B :** Corresponding rose of intercepts

Mathematically, the rose of intercepts for this sample pattern can be expressed as :

$$I_L(\Theta) = c_0 |\sin(\Theta - \Phi)| \quad (3)$$

where c_0 is an indicator of the total line length in the direction Φ .

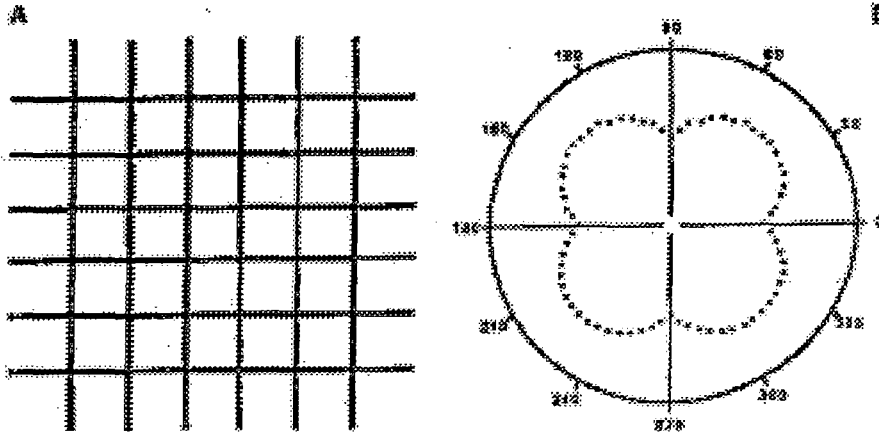


Figure 3 A : Sample pattern of straight lines in two perpendicular directions. **B :** Corresponding rose of intercepts

Figure 3 shows that the rose of intercepts for a sample pattern consisting of an array of straight lines in two different directions Φ_1 and Φ_2 is of the form :

$$I_L(\Theta) = c_1 |\sin(\Theta - \Phi_1)| + c_2 |\sin(\Theta - \Phi_2)| \quad (4)$$

and c_1 and c_2 describe the relative total line lengths in the two principal directions. The corresponding $MIL(\Theta)$ plot is a combination of two ellipses.

Trabecular bone is often considered to be made up struts and plates arranged in a variety of positions and directions. If we assume that a two-dimensional slice of trabecular bone exposes these structural elements as a series of lines of various lengths arranged at various positions and orientations, we can find the estimated rose of intercepts $\hat{I}_L(\Theta)$ with :

$$\hat{I}_L(\Theta) = \sum_{j=1}^n c_j |\sin(\Theta - \Phi_j)| \quad (5)$$

where the bone pattern is modeled as a series of lines pointing in n directions, with a weighting (or degree of orientation) c_j corresponding to each direction Φ_j . We likened this relation to a convolution between the sine function and c expressed in the continuous domain.

3.1 Phase Distribution Method

From Equation 5, we see that the rose of intercepts is simply a sum of rectified sine waves of identical frequencies but varying phases - the phases being the angles of orientation. Just as a frequency distribution plot produced from a Fourier transform provides the magnitude of sinusoidal signals over a range of frequencies, the phase distribution plot provides the magnitudes of sine waves over a range of phases.

Plots similar to the phase distribution have proved useful in describing the preferred orientations of trabecular bone, but they typically are vaguely qualitative in nature. The rose of intercepts, however, can be used to produce a more quantitative phase distribution computationally. The technique is very flexible and is compatible with many conventional automated stereology systems.

The continuous function c is solved by deconvolving the continuous version of Equation 5. Unfortunately, this technique does not constrain the magnitudes to be positive, resulting in a different interpretation of c from that given above. We can solve the continuous problem for c (which was denoted the distribution density $f(\Theta)$) constrained positive, with a Fourier series approximation. The results are problematic for complex materials, because the harmonics of the Fourier series confuse the results. For the purposes of studying trabecular bone, we prefer discrete « bins » for the phase distribution.

To produce the phase distribution from the rose of intercepts, the angles Φ_j ($j=1, 2, \dots, n$) are typically selected over a range $0^\circ \leq \Phi_j < 180^\circ$, distributed uniformly. Least-squares parameter estimation facilitates the calculation of the unknown magnitudes c_j ($j=1, 2, \dots, n$). If the rose of intercepts is given at m angles Θ_i ($i=1, 2, \dots, m$), the number of phases that can be solved for is $n \leq m$. Use of $n \geq m$ results in overfitting of the data.

We wish to minimize the difference J between the measured rose of intercepts and that obtained from the model. Using a least-squares measure of this difference, the objective can be written as :

$$\begin{aligned} \text{minimize } J &= \sum_{i=1}^m \left(I_L(\Theta_i) - \hat{I}_L(\Theta_i) \right)^2 \\ &= \sum_{i=1}^m \left(I_L(\Theta_i) - \sum_{j=1}^n c_j \left| \sin(\Theta_i - \Phi_j) \right| \right)^2 \quad (6) \end{aligned}$$

subject to $c_j \geq 0$ for $j = 1, 2, \dots, n$. This can be expanded after substituting :

$$\Phi_{ij} = \left| \sin(\Theta_i - \Phi_j) \right| \quad (7)$$

so that the objective is to minimize :

$$J = \sum_{i=1}^m \left(I_L(\Theta_i)^2 + \sum_{j=1}^n (c_j \Phi_{ij})^2 + 2 \sum_{j=1}^n \sum_{k=1}^n c_j c_k \Phi_{ij} \Phi_{ik} - 2 I_L(\Theta_i) \sum_{j=1}^n c_j \Phi_{ij} \right) \quad (8)$$

subject to $c_j \geq 0$ for $j = 1, \dots, n$. $I_L(\Theta)^2$ is not a function of c_j and therefore may be dropped from the objective function. By also making the following substitutions into the elements of Q , an n by n matrix, and b and x , both n by 1 vectors :

$$Q_{jk} = \sum_{i=1}^m \Phi_{ij} \Phi_{ik} \quad (9)$$

$$b_j = - \sum_{i=1}^m I_L(\Theta_i) \Phi_{ij} \quad (10)$$

$$x_j = c_j \quad (11)$$

for $j = 1, \dots, n$ and $k = 1, \dots, n$, the optimization problem may be written as :

$$\text{minimize } \bar{J} = \frac{1}{2} x^T Q x + b^T x \quad (12)$$

subject to $x \geq 0$.

Parameters describing the accuracy of the least-squares fit can be obtained using the following equations :

- error sum of squares : $SSE = \sum_{i=1}^m (I_L(\Theta_i) - \hat{I}_L(\Theta_i))^2$ (13a)

- total sum of squares : $SST = \sum_{i=1}^m (I_L(\Theta_i) - \overline{I_L(\Theta_i)})^2$ (13b)

- estimated variance : $s^2 = \frac{SSE}{n-2}$ (13c)

- coefficient of determination : $r^2 = 1 - \frac{SSE}{SST}$ (13d)

3.2 Primary Direction Method

The trajectorial theory as expressed by Wolff argues that in a two-dimensional slice there are only two directions, intersecting at 90° , that the trabecular architecture will follow. In studying this theory, it may be desirable to compare stresses calculated with a finite element model to the two (or more) primary directions of the observed trabeculae. Once an assumption has been made concerning the number of trabecular trajectories, one can use the primary direction method to calculate the primary directions, and the degree of isotropy (the lack of orientation). While no additional information is supplied, the results are packaged in a form suitable for comparison with such models.

Suppose a test substance is known to be oriented along two directions, i.e., the substance can be modeled as straight lines at two angles Φ_j ($j=1,2$). These angles and the corresponding magnitudes c_j can be solved directly. Then the objective can be written as :

$$\text{minimize } J = \sum_{i=1}^m \left(I_L(\Theta_i) - c_1 \left| \sin(\Theta_i - \Phi_1) \right| - c_2 \left| \sin(\Theta_i - \Phi_2) \right| \right)^2 \quad (14)$$

with respect to $0^\circ \leq \Phi_j < 180^\circ$, $c_j \geq 0$ for $j = 1, 2$, which will give the best fit for two directions of orientation. Note that the difference with the phase distribution method is that Φ_j are here considered to be unknowns.

The primary direction method described can be extended to accommodate isotropy. The rose of intercepts for a completely isotropic substance is given by :

$$I_L(\Theta_i) = c_0 \quad (i = 1, 2, \dots, m) \quad (15)$$

Thus a substance with two directions of orientation and a degree of isotropy can be modeled with an estimated rose of intercepts :

$$\hat{I}_L(\Theta) = c_0 + c_1 \left| \sin(\Theta - \Phi_1) \right| + c_2 \left| \sin(\Theta - \Phi_2) \right| \quad (16)$$

and the objective function is :

$$\text{minimize } J = \sum_{i=1}^m \left(I_L(\Theta_i) - c_0 - c_1 \left| \sin(\Theta_i - \Phi_1) \right| - c_2 \left| \sin(\Theta_i - \Phi_2) \right| \right)^2 \quad (17)$$

with respect to $0^\circ \leq \Phi_1 < 180^\circ$, $0^\circ \leq \Phi_2 < 180^\circ$, $c_0 \geq 0$, $c_1 \geq 0$, $c_2 \geq 0$.

Because this objective function is not smoothly differentiated with respect to the independent variables Φ_j , the solution is difficult to obtain using standard nonlinear programming techniques. However, if the number of primary directions is small, search methods for Φ_j are adequate.

A crude but simple search method is simply to use the quadratic programming technique to minimize :

$$J(\Phi_1, \Phi_2) = \sum_{i=1}^m (I_L(\Theta_i) - c_0 - c_1\Phi_{i1} - c_2\Phi_{i2})^2 \quad (18)$$

with respect to $c_0 \geq 0$, $c_1 \geq 0$, $c_2 \geq 0$ for a list of values of Φ_1 and Φ_2 . From the table of values of $J(\Phi_1, \Phi_2)$, one can choose the minimum, which is the best fit for the parameters $c_0, c_1, c_2, \Phi_1, \Phi_2$.

To solve Equation 18, the same technique as presented for the phase distribution method is used, except with :

$$Q = \begin{bmatrix} 1 & \sum_{i=1}^m \Phi_{i1} & \sum_{i=1}^m \Phi_{i2} \\ \sum_{i=1}^m \Phi_{i1} & \sum_{i=1}^m \Phi_{i1}^2 & \sum_{i=1}^m \Phi_{i1}\Phi_{i2} \\ \sum_{i=1}^m \Phi_{i2} & \sum_{i=1}^m \Phi_{i1}\Phi_{i2} & \sum_{i=1}^m \Phi_{i2}^2 \end{bmatrix} \quad (19a)$$

and :

$$b = \left[-\sum_{i=1}^m I_L(\Theta_i) - \sum_{i=1}^m I_L(\Theta_i)\Phi_{i1} - \sum_{i=1}^m I_L(\Theta_i)\Phi_{i2} \right]^T \quad (19b)$$

Equation 13 can be used to calculate the coefficient of determination for the least-squares fit.

Equation 2 is adapted so that the relative degrees of orientation and isotropy can be expressed as the relative contribution each direction makes to the total rose plot :

- degree of isotropy : $d_0 = \frac{\int_0^\pi c_0 d\Theta}{A} = \frac{c_0 \pi}{A}$ (20a)

- degree of orientation direction Φ_1 :

$$d(\Phi_1) = \frac{\int_0^\pi c_1 |\sin(\Theta - \Phi_1)| d\Theta}{A} = \frac{2c_1}{A} \quad (20b)$$

- degree of orientation direction Φ_2 :

$$d(\Phi_2) = \frac{\int_0^\pi c_2 |\sin(\Theta - \Phi_2)| d\Theta}{A} = \frac{2c_2}{A} \quad (20c)$$

where :

$$A = \int_0^\pi c_0 + c_1 |\sin(\Theta - \Phi_1)| + c_2 |\sin(\Theta - \Phi_2)| d\Theta \quad (20d)$$

$$= \pi c_0 + 2c_1 + 2c_2$$

4. Practical Considerations

The methods described here are highly dependent on the accuracy and resolution of the imaging systems used to digitize the image and produce the rose of intercepts. Many conventional techniques for performing automated stereology have inherent inaccuracy associated with the digitization of the sample image. This is true of computer analysis systems that either rotate test lines at varying angles across a stationary sample or lay a constant set of test lines across a sample that is rotated within the computer memory.

APPENDIX

B

Computed Tomography Imaging

In this appendix, we describe the creation process of trabecular bones pictures which is used by Professor Tony S. Keller and described in [Zhu et al, 1994]. A quantitative serial sectioning technique and a video-imaging procedure are used to obtain precise (20- μ m voxels, in a 5mm x 6mm x 7mm test volume) digital images of lumbar vertebral trabecular bone specimens. We first describe the preparation of the specimen, then we describe the imaging process.

<i>1. Specimen Preparation</i>	<i>2</i>
<i>2. Imaging Process</i>	<i>4</i>

1. Specimen Preparation

Human lumbar spines are harvested during routine autopsies. Some 9mm x 9mm x 9mm cubic cancellous bone specimens are prepared from the vertebral centrum of the spines using an IsometTM low-speed diamond saw. Selection of the samples is regionally random. However, the vertebral regions that have apparent defects in the continuity of trabeculae due to blood vessels and bone diseases are avoided. Bone specimens are irrigated with 0.9% saline during machining and following mechanical testing, and are stored frozen at -30°C.

The bone specimens are thawed at room temperature for 2 hours before mechanical testing. Using an MTS 858 BionixTM test system, each specimen is nondestructively loaded in compression ($\epsilon_{\max} = 1\%$) along three orthogonal axes corresponding to the superior-inferior (SI), anterior-posterior (AP), and medial-lateral (ML) axes. The surfaces of stainless-steel load platens are polished to a surface flatness of $2 \mu\text{m cm}^{-1}$ and lubricated prior to testing of each specimen. Load and displacement are recorded at 1kHz using a NicoletTM 430 digital oscilloscope. Displacement are measured by means of crosshead movement and are corrected for the test machine compliance. A stress-strain analysis program was developed to correct the recorded displacement, and to determine the elastic modulus, E . Elastic modulus is computed from the slope of the stress-strain curves using a strain range of 0.1 - 0.8%.

After mechanical testing, the bone marrow is removed from the specimens using a high-pressure water jet and defatted with several acetone washes and rinses. The marrow-free samples are then dried in a furnace at 100°C for 1 hour, and weighed on a Mettler AE 163 (Hightstown, NJ) analytical balance. Apparent dry density of the specimens is calculated as the ratio of the dry weight to the cube volume, the latter measured using a caliper ($\pm 0.025\text{mm}$). Specimens are then bleached using 3% hydrogen peroxide, embedded in black-colored polyester resin and centrifuged at 1000 rpm. The centrifuge process

facilitate infiltration of the polyester resin into the pores of the cancellous bone samples.

2. Imaging Process

In order to obtain a detailed understanding of the bulk variations in bone structure, bone specimens are serially sliced along the superior-inferior axis every 20 μm using a Reichert-Jung^R polycut E microtome. At an image resolution of 20 $\mu\text{m}/\text{pixel}$, 16-bit color video images of each sectioned surface are recorded using an image acquisition and analysis system. This image system (represented in Figure 1) consist of a CCD camera and a PC-80386 computer installed with a TARGATM 16 graphic board (Truevision, Inc., Indianapolis, IN) and a MIPS (Map and Image Processing Software) program (MicroImages, Inc., Lincoln, NE). The TARGA board is able to convert the image of the CCD camera into a digital screen display of 510 x 480 pixels with 32,768-color resolution. A total of 250 planar digital images spanning 5 mm in depth for each specimen are obtained (one sample has only 200 images), producing a 3D 16-bit image array comprised of approximately 50 million 20- μm voxels or volume pixels. For the remaining cancellous bone specimens, only the orthogonal surfaces are imaged. The 16-bit images are thresholded into white (bone) and black (marrow) binary images. The binary image arrays are then processed on a pc-80386 computer using C-language programs to determine planar structural properties (see Chapter 3).

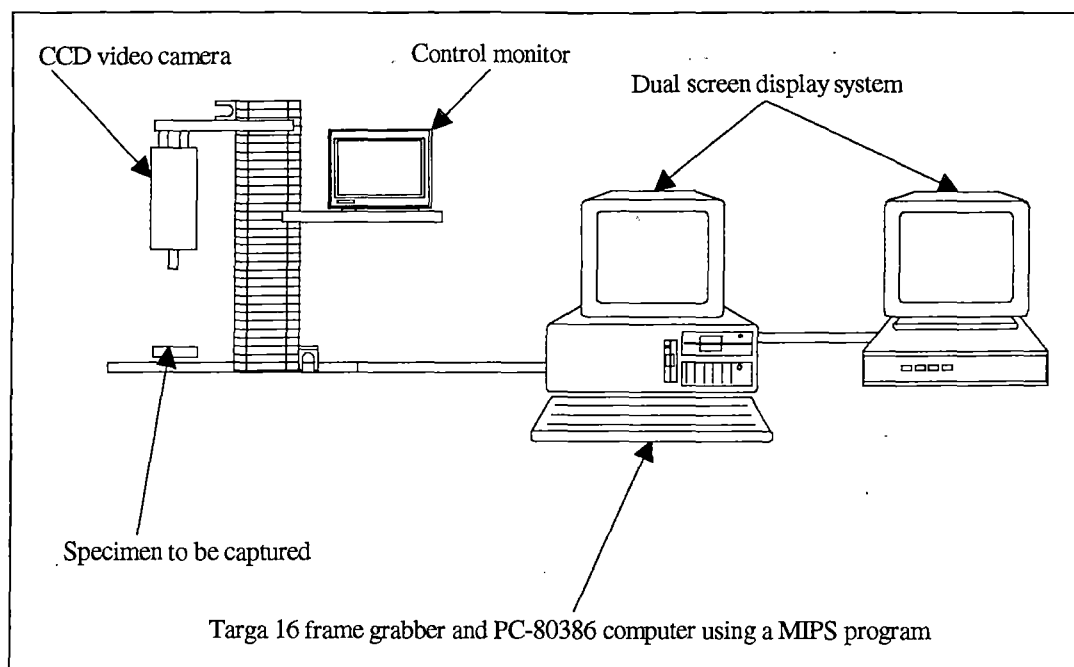


Figure 1 Image acquisition system

