Design and Application ofConcurrent Double KeySurvey Data Structures

C. P. E. Agbachi

Department of Mathematical Sciences, Kogi State University
Anyigba, Kogi State, Nigeria

***Abstract*—** *It is often the case in software development to find model solutions in generic data structures and algorithms. Even more common in rapid development programmes, is the use of libraries and incorporation of database engines. However, there are instances where none of these options provides a matching model to formats and methodology, for instance, in field of data collection. Neither are the workarounds perfect, the result of which is both an inadequate as well as unsatisfactory resolution. This paper examines linear data structures and adaptation to intelligent, concurrent double key models to meet requirements in Geomatic Engineering.*

Keywords—Lists, Pointers, Objects, Levelling, FieldBook, Knowledge Engineering.

I. INTRODUCTION

Survey Data Structures is a subject of data organization and arrangement in the field of Surveying and Geoinformatics. It has origins in the abstract data types in Computer Science [1]. These are fundamental and provide the foundations for building blocks that lead to designs of program modules in data acquisition and processing. A primary data type is the record, comprising of fields of information where a particular handle may have a key status, in order to serve the purpose of sorting. And while they could be many such sort fields, only one usually is processed at a time. Records are normally held in a database, storage file medium, ranging in order of hundreds and millions depending on application.

A. Lists

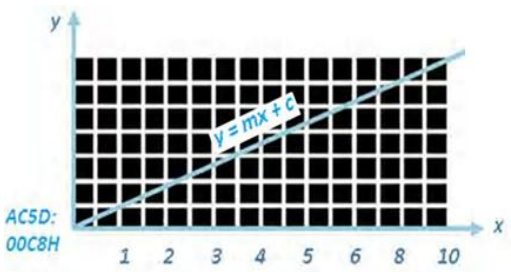

Fig. 1 Linear model of memory

A List is defined as an ordered data sequence [2]. It is family of linear data structures, where the relationship between the records in the sequence and locations in memory are linear, Fig 1. As such, a data sequence conveys the impression of contiguous locations in physical memory. In such allocations, the relationship generally is of the form $Y = MX + C$, where C is the base address, for example, AC5D:00C8H. X is the node while M, is the byte size per node.An instance is Array data type, defined in many programming environments and that can hold thousands of elements. Other examples include Queues and Stacks. In reality though, the apparent sequence of records results from modifications in the structure to address non-contiguous allocations, from the memory heap.

B. Linked Lists

A linked list is characterised by linked allocation. This is in contrast, to keeping a linear list in sequential memory locations.

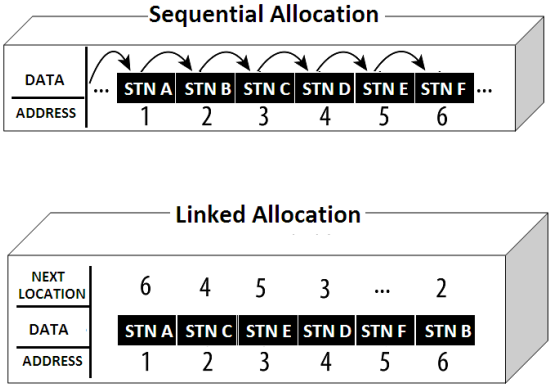

Fig. 2 Sequential and Linked Allocation

A comparison between the two models for the records, from StnA to Stn F, is shown in Fig. 2. But whileaddresses in Sequential Allocation are contiguous, in the linked model there is additional row, Next Location thatpoints to the memory address of the next item.

In a more generalized format, the linked table above is best represented as a Singly Linked List, Fig.3a, where the arrows are pointers to the next record.

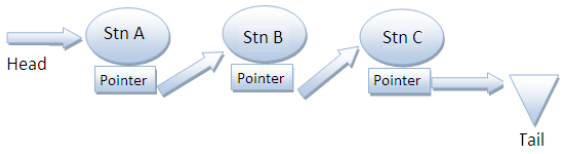

Fig. 3a Linked List

The advantage is obvious, in that the order of the sequence can be changed simply by redirecting the pointers. For instance, StnA may point to C, and C to E etc. Therefore, insertions and deletions are easily implemented by this arrangement in comparison with sequential allocation. However, there is the constraint of having to move only in the right direction. At StnC, for example, there is no information regarding the previous node. The result is that moving in the opposite direction is not possible and random insertions are fraught with difficulties. This however can be overcome with double links between the records.

1. Doubly Linked List:

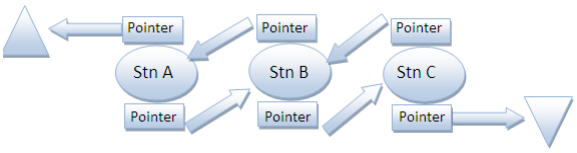

Fig. 3b Doubly Linked List

A doubly linked list, Fig 3b, comprises of two pointers, one in each direction to indicate next or preceding link in the chain. By this procedure traversing can occur in both directions. Insertions and deletions are not beset with starting from the beginning or end of the list.

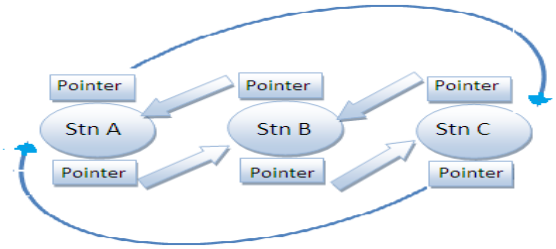

Fig. 3c Doubly Linked Circular List

It is common in some applications to be able to navigate freely within the database without the bounds of end or beginning of the dataset. In such a case, the pointers are redirected from head to tail, and vice versa, to form a doubly linked circular list, Fig 3c.

There are also other variants such as multiply linked list. However, this is mainly a case of two or more sort fields in each node. Given this feature, traversing the data set may be based at a time, on surname, or alternatively town, date of birth, etc.

C. Pointers

Pointers are unique in variable data types, in that whereas a standard variable contains the data, a pointer type holds instead, the address of the data for storage and retrieval. It has its roots, among addressing modes, in the form of indirectsystem.

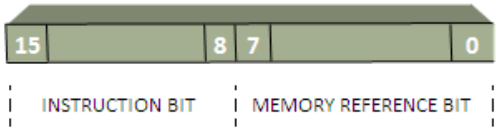

Fig. 4a Computer Instruction

Generally, a computer instruction comprises of the instruction bit and the memory reference bit, Fig. 4a. In a 16-bit architecture, each may occupy 8-bit space.Then the memory bit can address at most, up to 256 locations [0-255].That is, whereas an instruction LOAD A [255] is valid, STORE A [256] is not, as it exceeds 8-bit space.

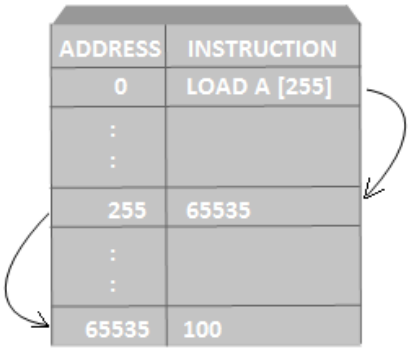

Fig. 4b Computer Program

This restriction can besurmounted by indirect addressing, Fig. 4b. As seen in the diagram, LOAD A [255] points to location 255 where contents, 65535, is the address destination. Thus, when the instruction executes, 100 is loaded into A, the register.It is observed that at address [255], the entire 16-bit space is available, and as such,the range can extend up to 65535. Therefore, the limitation in 8-bit memory reference address space is overcome [3].

1. Arrays:

Arrays are characterized by indexed addressing modes and whencombined with indirect mode add even greater flexibility and dynamism to a data structure. This leads to emergence of pointers to Arrays, and where every index is a pointer reference.

In the discussions so far, the strength of linked lists include:

- Dynamic Allocation
- Ease of Insertion and Deletion

This is in contrast to the standard array which is fixed in size, without the ability to remove or add items. Nevertheless, there are strong meritsin using Arrays[4], [5], given the following observations:

- Random access mode of operation
- A rating of O(1) performance
- Locality of reference, such that adjacent cells tend to be on the same page during virtual memory management.

Overall, if there are ways through which Arrays can be modified for dynamic allocation, ease of insertion and deletion, it would be a data structure of choice. In fact, in some programming languages such as BP Pascal [6], this appears to be the position regarding library options in available data structures.

D. Objects

Object oriented programming have been the norm for over a decade. Within this period many otherwise intractable problems have found solutions. Actually Objects or Class have origins in the concept of Frames in Artificial Intelligence. A frame is therefore one of the options available in models of Knowledge Representation [7]. The key characteristics and advantages are as follows:

- A frame is a data structure with typical knowledge about a particular object of concept.
- Every frame has its own name and a set of attributes or slots associated with, thus providing means of organising knowledge in slots to describe various attributes and character of the object.
- Frames are often used in production rules and provide a natural way for the structured and concise representation of knowledge.
- Inheritance is an essential feature of frame based systems, and is the process by which all characteristics of a parent frame are manifest and defined in the child.
- Frames have Methods and Demons, procedures associated with attributes and executed whenever so requested. Typically, there are two types of Methods, WHEN CHANGED and WHEN NEEDED. On the other hand, Demons have IF-THEN structure. Given the antecedent, the consequence is triggered whenever there are changes in the attribute.

In the foregoing, Object Oriented Programming (OOP) is characterised by polymorphism and data encapsulation [8]. They thus, provide foundations on which to build and develop intelligent data structures.

1. Intelligent Data Structures:

Intelligent data structures provide a model on which to modify an Array data type. A typical example is TCollection, in then RTL of Borland Pascal, and in present rebirth TList as in [9], [10].TCollection is derived from TObject with attributes, data fields, and methods as illustrated below:

```
{ TCollection types }

PItemList = ^TItemList;
TItemList = array[0..MaxCollectionSize - 1] of Pointer;

{ TCollection object }

PCollection = ^TCollection;
TCollection = object(TObject)
  Items: PItemList;
  Count: Integer;
  Limit: Integer;
  Delta: Integer;
  constructor Init(ALimit, ADelta: Integer);
  constructor Load(var S: TStream);
  destructor  Done; virtual;
  function At(Index: Integer): Pointer;
  procedure AtDelete(Index: Integer);
  procedure AtFree(Index: Integer);
  procedure AtInsert(Index: Integer; Item: Pointer);
  procedure AtPut(Index: Integer; Item: Pointer);
  procedure Delete(Item: Pointer);
  procedure DeleteAll;
  procedure  Error(Code, Info: Integer); virtual;
  function FirstThat(Test: Pointer): Pointer;
  procedure ForEach(Action: Pointer);
  procedure Free(Item: Pointer);
  procedure FreeAll;
  procedure FreeItem(Item: Pointer); virtual;
  function GetItem(var S: TStream): Pointer; virtual;
  function IndexOf(Item: Pointer): Integer; virtual;
  procedure Insert(Item: Pointer); virtual;
  function LastThat(Test: Pointer): Pointer;
  procedure Pack;
  procedure PutItem(var S: TStream; Item: Pointer); virtual;
  procedure SetLimit(ALimit: Integer); virtual;
  procedure Store(var S: TStream);
end;
```

Starting with TCollection types, of note is PItemList, a pointer to array of pointers.Then the method, Init initialises the list to ALimit, while the second variable ADelta, allows dynamic expansion of the list by this margin. With this facility, the size of the container is limited only by 16-bit architecture. Given that 4 bytes is allocated to each pointer, this translates to 16,384 items.However though, there are means to by pass this limit, conceptually by two dimensional array definitions in TItemList.

Of note are the several methods defined in the object, such as Insert, Delete and Free. The virtual

methods allow child objects to redefine inherited procedures for the purpose of custom adaptation. With this facility, objects could be taught how to adapt to meet the necessities of field data collection in Geomatic Engineering.

II. ANALYSIS

In designing an intelligent system, the best approach is to start from a generic model.

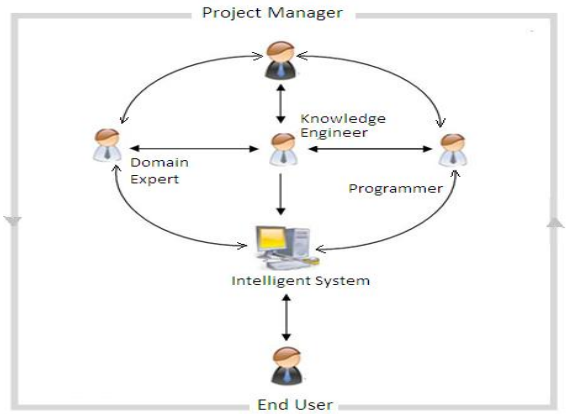

Fig. 5 Development Team

As the diagram, Fig. 5, illustrates, development comprises of a team, namely:

- Domain Expert
- Knowledge Engineer
- Programmer
- Project Manager

Domain Expert refers to and is a description of an expert in the field. In this application, a Surveying Engineer who is very knowledgeable and with considerable field experience is perfect for this task. He can then list and describe the areas requiring computer process. In other words, this wealth of experience would form the base knowledge in the mind of the machine.

Knowledge Engineering as described in [11], is the process of integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise. Thus, the Knowledge Engineer takes the skill of the Domain Expert and formulates the best model of representation for the machine.

Programming involves translating the model into a working system, and requires competence and expertise in programming languages, and familiaritywith the environment of the operating system.

The job of the Project Manager is to oversee the progress, ensuring that milestones are met and work done is satisfactory.

III.FIELD MODEL

Surveyors carry out measurements in the field to determine horizontal and vertical positions [12], [13]. The latter known as height measurements is by process of levelling. In brief, an instrument is set up and levelled between two points, A and B, to read staff measurements as shown in Fig.6.

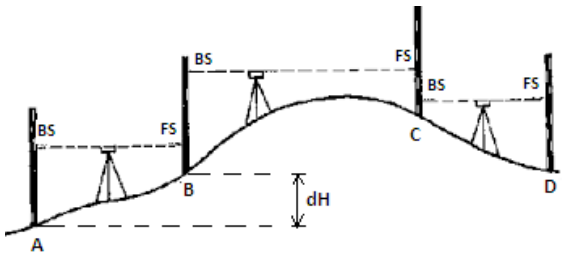

Fig. 6 Levelling Process: A → D

The height difference ΔH between the two points, with BS and FS readings, is given by $h_A - h_B$ where a rise or fall is indicated by sign of the quantity. It is a cumulative process that starts from a known position and closes at another control point. The survey can involve a network of level runs over several kilometres and comprising of a huge data set. It could also be in a dynamic environment such as construction site.

The key issues then are as follows:

1) The computability and reliability of results depend on strict adherence to the cumulative order in survey topology.
2) However, a large project often involves several parties, each of which may be following a different regime. Also, access may be restricted, such as in construction sites, meaning deviation from the planned order of survey.
3) Loss of stations and errors mean that repeats are common
4) In the event of 2) and 3), during a large survey, computation of results can often becomeburdened with at times, insurmountable difficulties.

Given the above observations, the following are recommendations:

1) A survey can start from and end at any point in the network, in any order.
2) It is vital that measurements of the network are comprehensive, fully descriptive, without any omissions.

3) Every point should have unique name identification.
4) The sorting and re-arrangement of observations into survey topology should be handled by software.

IV. FRAMEREPRESENTATION

In translating the grasp of field model into a computer compatible format, Knowledge Base, the recommendations are critical in assessing the result. By taking these into account, the following scene emerges:

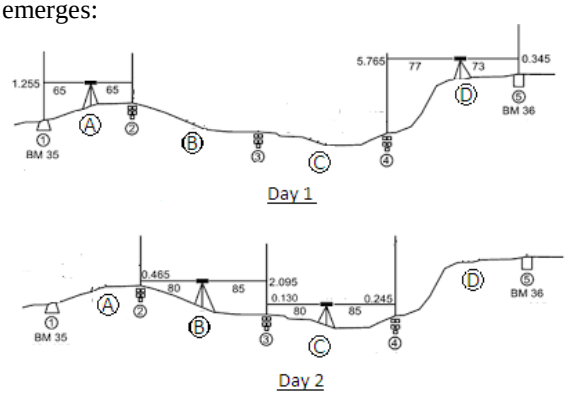

Fig.7a

In day 1, as in Fig. 7a, only two setups, A and D, were observed in the course of the survey. Possibly due to access, no measurements are recorded for setups at B and C. On the second day, the survey is completed.

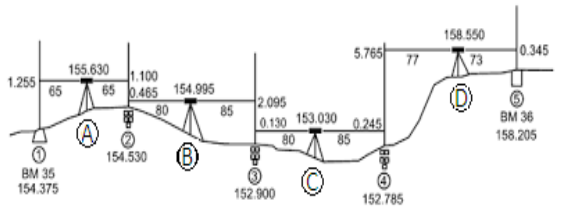

Fig.7b Survey Topology

The reconstruction that takes place after the survey sees the data set reflecting survey topology, Fig. 7b. Additional level runs emanating or joining the network are equally sorted and appended. Furthermore, repeats in the survey automatically supersedeany previous or erroneous observations in the database. In order to achieve this,the appropriate data structures and objects have to be formed.

A. Frames

There are three models to consider, namely the work at each setup, the database and the interface representing the traditional recording of data in a field book.

1. Setup Record:

```
type
  PSet_UpRecord = ^TSet_UpRecord;
  TSet_UpRecord = record
  BackSightStn: array [0..5] of char;
  BackSightReading: array [0..10] of char;
  BackSightDistance: array [0..10] of char;
  BackSightCircle: array [0..10] of char;
  ForeSightStn: array [0..5] of char;
  ForeSightReading: array [0..10] of char;
  ForeSightDistance: array [0..10] of char;
  ForeSightCircle: array [0..10] of char;
  Diff: array [0..10] of char;
  SumDiff: array [0..10] of char;
  SumDistance: array [0..10] of char;
  Remarks: array[0..250] of Char;
  Counter: array[0..15] of Char;
  Link : Boolean;
end;
```

The Setup record is by definition such that all the information about the setup is represented. Critical for the description are the sight station names, BS and FS, and corresponding readings. The attributes are as described above.

```
PSet_UpObj = ^TSet_UpObj;
  TSet_UpObj = object(TObject)
  Set_UpRecord: TSet_UpRecord;
  constructor Load(var S: TStream);
  procedure Store(var S: TStream);
end;
```

Fig.8a Set_UpObj

Next, the record structure is consolidated into an object data type, Fig. 8a, as a slot inside the object. With this arrangement, the observations at each setup constitute an object that can be dropped into a container database, Fig 8b.

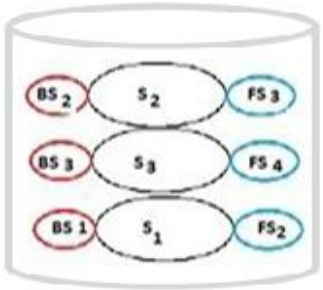

Fig. 8b Container of Observations

2. Container:

```
type
PSet_UpCollection = ^TSet_UpCollection;
TSet_UpCollection = object(TCollection)
  function Found(Item: Pointer): Boolean;  virtual;
  functionSearchForPosition(Item: Pointer):
        Integer;virtual;
  procedure Filter(Item: Pointer); virtual;
  procedure Insert(Item:  Pointer); virtual;
  procedure Error(Code, Info: Integer); virtual;
end;
```

Container, in this instance, is a database and an object derived from TSet_UpCollection. As such it

inherits TCollection data type, the base object, providing the foundation for modifications to match required descriptions.

There are five methods, such as Found,Filter, and Insert, but the function, SearchForPosition stands out as it is responsible for determining where to slot the observation in line with survey topology, Fig 8c.

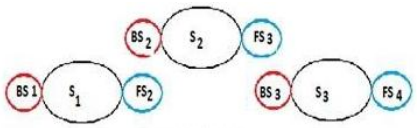

Fig. 8c Sorted Data Set

3. *Interface:*

```
PFieldBook= ^TFieldBook;
TFieldBook= object(TDlgWindow)
   Remarks, Counter: PEdit; Data: Array[0..6,0..3] of PEdit;
   ref_Id:integer;
   NewDat,SaveDat,XDat,DelDat,ISight:PButton;
   FbkScroller:PScrollBar;
    EditBrush,RefBrush,StatBrush:HBrush;
   constructor Init(AParent: PWindowsObject; AName:PChar);
   procedure SetUpWindow; virtual;
   procedure WMControlColor(varMsg: TMessage);
       virtualwm_First + wm_CtlColor;
   constructor Load(var S: TStream);
   procedure Store(var S: TStream); virtual;
   procedure LoadNaSet_Ups;
   procedure AddDeltaH;
   procedure TransferInfo(Direction:Byte);
   procedure NewSet_Up(varMsg:TMessage);
      virtualid_First + 151;
   procedure Save_Input(varMsg:TMessage);
      virtual  id_First + 152;
   procedure Cancel_Input(varMsg:TMessage);
      virtual  id_First + 153;
   procedure DelSet_Up(varMsg:TMessage);
      virtual  id_First + 154;
   procedure I_SightCollection(varMsg:TMessage);
      virtual  id_First + 155;
  procedure  Response(varMsg: TMessage);
    virtual  id_First + 156;
  procedure ShowSet_UpHeader(NaSet_UpRecNum: Integer);
   procedure Restore;
   procedure InitializeSet_Ups;
   destructor Done; virtual;
   function GetClassName: PChar; virtual;
   procedure  GetWindowClass(varAWndClass:  TWndClass);
virtual;
end;
```

Interfaceis thefield book, derived from TDlgWindow object, and providing means of communicating with the container, database. Normally, data collection is automated computer process and this interface enables the surveyor/engineer to review data in traditional environment. Input can be manual, with editing options. Besides the editing forum, there is also the option of list review and printing of pages from the field book.

V. PROGRAMMING

The key challenge here is to implement the methods of TSet-UpCollection. A starting point is an examination of the input object, TSet_UpObj. Looking at Fig 8a, it is obvious that two keys BS and FS have to be considered in order to locate where to insert an observation. Furthermore, it isclear the evaluations would have to be in tandem. Hence, the required outline of program steps.

A. *Algorithm*

```
procedureTSet_UpCollection.Insert(Item: Pointer);
begin
Filter(PSet_UpObj(Item));
   If not Set_UpDuplicate then
AtInsert(SearchForPosition(PSet_UpObj(Item)),
PSet_UpObj(Item));
end;
```

The key issue is insertion and sorting of survey objects, and the heart of it lies in a review of the above method, TSet_UpCollection.Insert. Hence, the first call in the procedure is Filter. And if there is no duplication of observation, an insertion is carried out at the desired location. These routines would now be examined in further detail.

1. *Filter:*

Filter performs a number of checks on the input object. These are:

- Validation of data, to ensure that key fields, BS and FS are valid entries.
- Determine if the record already exits in the database and overwrite if input is in file mode.Checking for duplication requires concatenating the two fields, such that Search_Key = BSName + FSName
- It is also important to check for instances of double levelling in the network and reduce as such.

After filter routine has executed, the statement, If_not_Set_UpDuplicate, returns a Boolean answer before the next process can commence.

2. *Searching:*

The searching process for position that a new objet occupies is best illustrated with reference to Fig 7a. A follow up is the equivalent location in the TSet_UpCollection, container database, Fig 9.

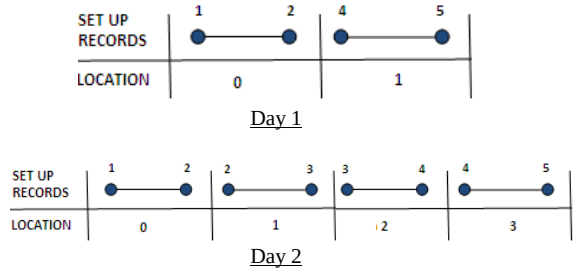

Fig. 9 Profile of Database

In Day 1, work comprises of twodistinct level runs, [1-2] and [4-5]. However, in Day 2, with insertions of the day's work, the database assumes a profile representation of survey topology.

The required program steps may be formulated as follows:

1) Given a new input of observations, X-Y, where X is the BS and Y is the FS, take the pair:

   a) Locate a previous set of observations where the FS, and BS in the input object are one and the same station, X.
   b) Locate a previous set of observations where the BS, and FS in the input object are one and the same station, Y.

2) Case Found of:
   Either
   1a: True: Insert the new object X-Y after Found location, n+1.
   Or
   1b: True: Insert the new object X-Y at Found location, n.
   Else
   Append new object to last count

3. Implementation:

Implementation process is through the method, SearchForPosition, outlined below. Note the two variables, Key1 and Key2. The first two lines ofinstructions fetch and assign the BS and FS, respectively.

The functions MatchKey1 and MatchKey2, as defined in the method, perform Program Step 1), through iterating process, LastThat and FisrtThat, returning pointers F1 and F2, respectively.

```
functionTSet_UpCollection.SearchForPosition(Item: Pointer): Integer;
var I : integer ; KeyName2, KeyName1: string; F1, F2: PSet_UpObj;  ANode: Boolean;

function MatchKey1 (Readings: PSet_UpObj): Boolean; far;
var A: string;
begin
   A :=StrPas(Readings^.Set_UpRecord.ForeSightStn);
   MatchKey1 := (A = Key1) ;
end;

  function MatchKey2 (Readings: PSet_UpObj): Boolean; far;
var B: string;
begin
     B :=StrPas(Readings^.Set_UpRecord.BackSightStn);
     MatchKey2:= (B = Key2) ;
end;

begin
   Key1 := StrPas(PSet_UpObj(Item)^.Set_UpRecord.
           BackSightStn);
    Key2 := StrPas(PSet_UpObj(Item)^.Set_UpRecord.
            ForeSightStn);

     F1:=LastThat(@MatchKey1);
     F2:= FirstThat(@MatchKey2);

     if (F1 <> nil) then
        SearchForPosition :=IndexOf(F1)+1
     else if (F2 <> nil) then
        SearchForPosition :=IndexOf(F2)
     else
       SearchForPosition := Count ;
end;
```

Program Step 2) is then executed in an IF-THEN-ELSE block to make a decision.

The routine, SearchForPosition, returns an integer, Index, to the calling program. Hence an insertion is carried out by the last call in the Insert method, outlined above, such as AtInsert(Index, NewObject).

VI. APPLICATION

The object, TSet_UpCollection, is the foundation of the data collection module in LMS, Level Monitoring System [14]. Evolving into SMS, Survey Management System, it was further acclaimed for its innovation and use in industry [15].

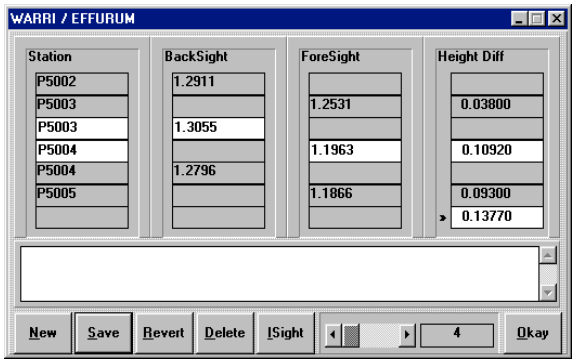

Fig. 10 Field Book Interface

A typical field book, interface to database, is shown in Fig. 10. Object input is emphasised as well as orientation to network modality. There is also

adequate support and provision for intermediate sight observations. These are database objects defined inside each setup, and accessible through the ISight button.

Further applications of the data structure, finds a place in derivates that lead to the construction of Traverse data types. There are also edge definitions, lists and sort in graphs, to provide survey network descriptions. It is thus an important innovation.

VII. CONCLUSION

On reflection, the accomplishment in this project is attributable to the development team, comprising of the Domain Expert, Knowledge Engineer and Programmer. This concept is very important in the success of any such venture.

A number of hands may be desirable, but the size of the personnel is less important than the degree of overlap, communication and understanding among the team. Indeed, ideally the team should operate as a one-man team. That is without the need for bridges or interpreters in order tobe in touch. This is because, often milestones are not met, and projects fail due to the fact that just when it is thought that developers have understood what is supposed to be done the opposite, more often than not,turns out to be the case. As a result, the need for a common language among development team cannot be overemphasised.

No less and very important in application development is methodology, hence as in this instance, the concept of concurrent double key data structures. As pointed out in [16], whatonce were intractable problems tend to find solutions in new concepts such as AI [17], [18] and techniques of Knowledge Processing [19], [20].

REFERENCES

[1] Jean-Paul Tremblay, Paul G. Sorenson, “An Introduction to Data Structures With Applications”, McGraw Hill Computer Science Series 2nd Edition

[2] Donald E. Knuth, “The Art of Computer Programming, Vol. 1 (3rd Edition): Fundamental Algorithms”, Addison Wesley Longman Publishing Co.

[3] P. C. Pittman, “Design of Digital Systems: Book 6 Computer Architecture”, Cambridge Learning Enterprises, 1974.

[4] Niklaus Wirth, “Algorithms and Data Structures”, © N. Wirth 1985.

[5] Julian Bucknall, “The Tomes of Delphi: Algorithms and Data Structures”, © 2001, Wordware Publishing, Inc.

[6] Borland, “Borland Pascal with Objects User’s Guide”, Copyright 1983, 1992 Borland International.

[7] Olubusayo Dare, “Artificial Intelligence: Concepts, Methodology and Application”, Bachelor’s Thesis, Kogi State University, Anyigba, 2012

[8] Tom Swan, ”Borland Pascal 7.0 Programming for Windows”, Borland Bantam, 1993

[9] Tom Swan, ”Foundations of Delphi Development for Windows 95”, IDG Books Worldwide, Inc 1995

[10] Lazarus/Free Pascal Compiler, http://www.lazarus-ide.org/

[11] *Edward* A. *Feigenbaum, Pamela* McCorduck, “The Fifth Generation 1st Edition

[12] R. E. Davis, F.S. Foote, J. M. Anderson, and E. M. Mikhail, “ Surveying Theory and Practice”, McGraw-Hill, 1968

[13] W. Schofield, ”Engineering Surveying, Vol. 2.”, Butterworth and Co. (Publisher) Ltd., 1974

[14] C. P. E. Agbachi, “Design and Implementation of a Level Monitoring System, APC Presentation, Royal Institution of Chartered Surveyors (RICS) 1995

[15] Leica Report(UK), http://www.pecaconsult.com/Leica_ report.pdf

[16] C. P. E. Agbachi, “Surveying Software”, Chartered Institution of Civil Engineering Surveyors ICES, October 2011, http://mag.digitalpc.co.uk/fvx/ces/1110/?pn=44

[17] George F. Luger, “Artificial Intelligence, Structures and Strategies for Complex Problem Solving”, 4th Edition 2001

[18] Negnevitsky, Michael, Artificial Intelligence: A Guide to Intelligent Systems”, Pearson Education 2002

[19] Rudi Studer, V. Richard Benjamins, Dieter Fensel, “Knowledge Engineering: Principles and Methods”, https://www.researchgate.net/profile/V_Richard_Benjamins/publication/222305044_Knowledge_engineering_principles_and_methods._Data_Knowl_Eng_25%281-2%29161-197/links/0fcfd50c3673c0368e000000.pdf

[20] Franz J Kufess, “Knowledge Processing”, Computer Science Department, California Polytechnic State University, San Luis Obispo, CA