A general survey of HELIX

Daniel Pisters

10 July 2000

1. The data in Helix

I will not describe the database as a such, but rather following an approach that would precede the database design.

One can distinguish two big classes of data generated and exploited in Helix:

1) the administrative data

2) the audiological data

1.1 Principal entities to the administrative level:

- **The brands and the types of hearing devices** (in term of Master To Detail relationship, a brand is the master of a number of types). These two entities are very simple, as they have each only one attribute field, that could be at the same time their key: the brand's field and the type's field. No picture representing them, nor description.

- **The hearing devices** (as instances of the preceding ones). A device refer to a brand and to a type, both selected in a list of values.

- **The remote control devices**. Very similar entity to the one of hearing device, it nevertheless must not refer to a generic type. Therefore there is no list of value mechanism, what is a gap since the remote control devices' brands are the same as the one of hearing device.

- **The maintenance contracts**. Relating evidently to a hearing device once that this one was delivered, this entity is very simple too. One can distinguish two types maintenance contracts: the white contracts of which hearing centres receive a certain provision, and those on which the names and number of customer are printed.

- **The files containing the personal data of patients** (in term of Master To Detail relationship, the file is the master of the 3 previous entities, as well as the Master of audiometries).

- **The ORL**. One should conceive a Master To Detail relationship between the ORL and their respective patients, if not a relation Many To Many insofar as a patient could consult several doctors at the same time. Currently, and it is a big gap, this relation exists only in an implicit way: the ORL name must be the same (and written exactly on the same manner) in the file of the patient as in the ORL entity.

- **The general practitioners**. The same problem exists as with ORL.

- **The mutual benefit society**. The same problem exists as with the ORL and general practitioners, otherwise that the relation is based on the number.

- **The historic** of events relative to the patient's files and to entities in relation with these (examples: free trial of a hearing device, return to the factory of a hearing device, creation of an audiometry). The historic contains principally the date of the event, a reference to the entity or object concerned (file, hearing device, remote control device) and a reference to the patient's file if there exists a relationship between this one and the object. For example, if the event is «return to the factory » of a hearing device, there is not any reference to a patient's file. On account of the return to the factory of a hearing device, the relationship with a patient's file in particular has been deleted.

- **Text, Commentaries and Courtesies**, that are used in the letter editor. These entities contain predefined texts that can be inserted, to certain places, in the different types of letters.

The gaps of the relational diagram that can be noted to the level of administrative data (not any actual relation between the ORL and the patients, but a substitution mechanism that works only if the user watched to the ORL name be the same in the two entities) are an inheritance of the old system of files FICHE, where no relation existed. A part of this inheritance could not have been eliminated directly before the elaboration of the import procedure. More of accordance to the relational diagram is obtained on a gradual manner, as go out of new versions.

1.2 Principal entities to the audiologie level:

- **The tonal audiometries** (respectively for the left and right ear). They consist each in at least two curves of which one is for the bony conduction.

- **The vocal audiometries** (with and without device). They consist in two charts containing at least three curves: right monaural, left monaural and binaural.

- **The spatial audiometries**. Opposite to the two typical others audiometries (tonal left and right and vocal, without and with device), this one is currently implemented in a rather minimalist way: it has not any chart, but only three percentage values to enter, respectively for hearing device adaptation Mono, Stereo and without hearing device adaptation.

- **The conclusions of the vocal audiometries**. Determining the prosthetic gain while comparing the results of the vocal audiometry with device to those of the vocal audiometry without device, they meet a legal requirement for the reimbursement. The audiologist chooses one of the three methods from which these conclusions can be generated.

- **The hearing device adaptation methods** to determine the insertion gain according to the results of the tonal audiometries. Although very different of one another from a purely audiologic point of view, they are until now easily reducible to a generic algorithm. These methods are based on predefined values of hearing thresholds and, to some varying near, predict a gain for each frequency in function of a ratio of the hearing loss. In the current state of Helix, these values are coded in the program in the same way as the algorithms, but one can see those methods and the domains of values to which they apply as something like a class of algorithms, of which one or more entities could be derived.

- **The insertion gains**. They are calculated for each ear separately from the results of the tonal audiometry, according to the different hearing device adaptation methods having been implemented. Opposite to the conclusions of the vocal audiometries, the results are not get from nor saved in the database. They are calculated whenever the form is called in order to see them. The hearing device adaptation methods, not very used by the majority of the audiologists, represents the rudimentary stage of what could lead Helix to be an expert program.

2. The Helix Application

2.1 Language

Helix is written completely in BASIC VISUAL 6 Enterprise version.

2.2 Quantitative general Idea

Before attempting to describe Helix, it can be useful to give a quantitative notion of it. The executable file amounts to 4.800KB. By counting the spaces between lines, the sum of lines of code has since a long time surpassed the 100.000 (by compressing, one would obtain minimum 70.000). While traversing the VB Project Explorer, I count about 40 forms, 40 class modules (.cls) and more of 30 generic modules (.bas).

The Patient's file form, for example, amounts to 7000 lines of code. In a certain sense, there are some a lot too. That can be less explained by a complexity inherent to this form than by the multiplication of the interfaces with the other forms of the application; in fact, the Patient's File form communicates with almost all the others forms of the application. These interfaces are almost identical, so that if more advantage had been taken from their similitude in order to derived only one generic interface, where the other forms would be passed in parameter, the number of lines of code would have been significantly reduced.

The C_SpecificTreeViewManagement class module, managing the TreeView of which the icons give access to the forms individually, amounts to a little more of 4000 lines of code.

The GenericChartManagement module that contains methods generating charts, placing the points on the curves, amounts only to 1400 lines of code.

2.3 Forms, class modules, modules: towards a layered architecture

The nesting of these three modules categories (a form being a module as well as the two others) comes close to ULA (Universal Layered Architecture) but it is far from reaching the same separation level of application layers as with ULA: it is not sure, for example, that changing from ACCES to Oracle, would be as easy for Helix as for an application actually in accordance with ULA. Nevertheless, I might reasonably assert that, from this point of view, Helix is situated between an application of which the architecture is not at all ULA and another that would be it completely.

Illustrating this layered architecture of Helix from a relatively low level, the GenericChartManagement module contains graphical methods. These are wrapped by more specifics graphical methods for audiometries, in the C_SpecificChartManagement class module. The specifics methods are wrapped at their turn in all the forms showing charts for audiometrie, such as frmAudioTonaleDg, frmAudioVocaleAvecAca, frmAudioVocaleSansAca etc. but also by the functions and procedures of the letters editor that are related to the ORL letter.

This proves at least of good modularity level. In fact, the specific methods, and through them, the generic methods used to generate audiometry charts in forms and the methods used to generate charts as part of a larger picture send to the printer, are the same.

Another illustration of some layered architecture close to ULA: the audiometry data are all loaded from the database in structures maintained in memory such as collections, or in matrix implemented as flexgrid avoiding any direct reference to the database, as it is often the case through data controls. Being only temporary structures convenient to handle, certain of these flexgrids are even not visible on the screen.

2.4 The forms: organization around TreeView and navigation

2.4.1 The TreeView

The forms are organized around the TreeView, represented by icons. At the start, this arborescence is « linear », it consists only of one level: the icons are placed one underneath the other, so that the most frequently consulted (as the Patient File form) appear on the top. The Audiometry icon is preceded by a « + » sign, by clicking on which a mini-arborescence of three icons themselves preceded by «+» signs can be expanded. These three icons represent not forms that one could call directly, but rather categories, in this case, the three categories of audiometry that are defined until now (Tonal, Vocal and Spatial). By clicking on these « + » signs, one deploys the arborescence to the end of which one are located the icons giving access to the forms (charts, conclusions and hearing device adaptation methods). The forms called directly from the TreeView are independent of one another: the data contained in a form are not determined by those that are contained in another, the entity represented in a form called directly from the TreeView has no Master (even if it has as Master in the relational diagram).

2.4.2 Navigation

The TreeView is made of icons representing the forms and is growing with other icons, linked up between them as the nodes of a tree, as other forms are called from the preceding ones, and that still others forms are called from the following ones, and so on.

The navigation from one form to another is based on a menu system, organized itself like a tree, on the top border of the MDI form, the parent form of all the others, the one of which the window contains the windows or icons of all other forms. This menu shows the names of the forms to which one can have access from the active form, as for example, the Hearing Device form the Patient File form.

Once a form was called from another, one can recall it either by selecting the corresponding option of the menu, or by clicking on the related icon in the TreeView. The correct place in the TreeView arborescence, that can show the same icons to different places, signals itself by the greyed colour of the form name labelling the icon. Moreover, one can go back from the called form up to the calling form by selecting the <parent> option of the menu. This option could be simpler, but less significantly, called <return>.

If, after having called the Hearing Device form from the Patient File form, we are coming back to the Patient File form, one can still call the Remote Control form, or the Maintenance Contract form without having to leave none of the forms called previously from the Patient File form. Furthermore, the Master to Detail relation existing between the entities is implemented through the forms that are representing them in an absolutely consistent way: for example, if we change the record in the Patient File form, the records corresponding to the Details (of which the new Patient File occurrence is the Master) are loaded in the other forms, and this will be verified to any extend of the Master To Detail waterfall. In other words, if, from the forms containing the Details of the current Patient File occurrence, one call other forms containing Details of the preceding ones, these will be re-actualised as well.

The navigation means leading from one form to another all to alongside the chain of the Master To Detail relations is therefore without limit.

Even the audiometry charts, and this whatever may be the number of audiometry forms open at the same time, are instantaneously re-actualised (let us say “refresh”) when the record change in Patient File form.

2.5 Interfaces between the different forms

It is evident that to succeed in such possibilities of navigations, a robust interface system between the forms as well as of communication with the structure of the TreeView is essential.

Indeed, each form contains private methods to manage the Details of the entity that it represents, while the forms representing these Details contain public methods to allow their different Master to pilot them. The private methods of the Masters use the public methods of their respective Details. It is what I call an interface system between the forms.

It is evident also that these methods allowing a form to manage the Details of the entity that it represents (let us call it once for all: *its master entity*), are independent of the technology used to access the database: little imports that this be ADO, DAO or something else. Unfortunately, I cannot say so much about the entity that the form represents directly, namely the master entity itself: if we are wanting, for example, to move to the next record in the Patient File form, it is necessary to click directly on the right arrow of the DAO data control. Moreover, a number of functionalities are bound to the triggers and events associated to this data control. Nevertheless it would not be difficult to change that, in order to be founded on another technology to access the database or even on another type of database, without overturning basically the form architecture.

2.6 Reversibility of relations through the interfaces between the forms

I have written somewhere above « different Masters » about the same Detail. There are two reasons thereto:

- The Master To Detail relationship between the different entities represented by the forms are not always the reflect of a Master To Detail in the restricted relational meaning of the term, that would presuppose the existence of this relationship, with the corresponding referential constraints, in the database.

- Moreover, and that follow directly from the preceding point, a form containing the Details of the master entity of another form normally is called from this one, but we can do the reverse as well: that is to say, to call the form containing the Master from the form containing Details.

The Patient File form can therefore be called from the Hearing Device form, on the same manner that the Device form can be called from the Patient File form, via the corresponding option of the menu. It is evident that, in accordance with the One To Many relationship that exists in the database between the Patient File and the Hearing Device entities, the Patient File form, when it is called from the Hearing Device form, will contain only one occurrence of the Patient File entity (because Master To Detail means One To Many). But if the record changes in the Hearing Device form, the record of the Patient File form will change also in order to present the Master of the new Hearing Device record.

The fact that the relations between the majority of the forms is bi-directional has an evident practical usefulness: in the case of the Patient File and Hearing Device forms, one cannot only find the hearing devices that were attributed to a patient, but also the patient to which one a hearing device was attributed.

2.7 One form = one entity

The majority of the forms in Helix represent only one entity, namely their respective master entity. If one wish to see a patient file and the hearing device(s) attributed to this patient at the same time, it is necessary to arrange the two forms on the screen so that they do not cover themselves so much as to mask their most significant fields.

This approach is not necessarily the better one, for it may require from the user to acrobatic move with the mouse pointer in order to arrange the forms correctly. That may be difficult for certain, but elementary for others having more training with mouse and multi-windowing.

I would not give a complete description of Helix without describing somewhat the type of architecture towards which one it will evolve: I am currently on the point to integrate in Helix a new architecture allowing to represent as much entities that we wants in a same form, provided that these entities be in One To Many relationship with one another. This architecture already was tested in another application; it works perfectly and manages consistently the Master To Detail relationship through nested blocs structures, and this, whatever may be the number of these nested blocs (each of which representing an entity)

It does not do any doubt this architecture to behave also correctly in Helix. Furthermore, in this architecture, the layering level is increased, eliminating the direct dependence to the triggers and events of a data control (in this case of ADO type in place of DAO).

As it is henceforth possible to represent several entities on the same form, why did I not aimed directly to adopt this architecture for the first version of Helix ?

First, the architecture **one form = one entity** and the navigation system from one form to another and therefore, from one entity to another, all to alongside the chain of Master to Detail relationships will preserve his very usefulness with an architecture allowing to represent several entities in the same form.

Secondly, in a context of emergency and of extreme planning compression, I adopted this architecture **one form = one entity** practically according to the same strategic principle that, for a battle conducted on a land with an uncertain relief, if not uneven and hilly, and therefore favourable to multiple ambushes, give the preference to small independent, movable units but strongly connected to each other by a robust and flexible communication system (the interfaces, of which I spoke, are a little, in this war metaphor that I hope not to be too ridiculous, the equivalent of the communication by radio linking up the panzers, while the heavy French tanks were still equipped with country telephones – let us imagine the easiness of the manœuvre!)

Heavy forms, containing multiple entities, would have been more difficult to manage, at least at a time when I had not yet conceived a convenient architecture.

2.8 Basic Operations on the records in a form

They are realized by means of buttons.

- Add to add a record
- Delete to delete a record
- Update to save eventual changes
- Enter Query to empty the fields before to introduce a query profile if necessary (by means of lists of values, or of values directly entered in the fields, with one or more substitution characters « % »).
- Execute Query to run the query
- Close to close the master entity recordset and then exit the form (travelling depth first alongside the Master To Details, the forms containing the Details and their related recordsets are closed before).

Nothing original otherwise that each of these operations, applying to the level of the Master, is followed or preceded by a similar operation to the level of details. Thus the query actually run by clicking on the button <Execute Query> to the Master level is followed by a query to the level of the Details, and to the level of the Details of the Details, if necessary.

On the other hand, the details are not deleted with the Master. In this case, the deletion is prevented and a message informs the user of the existence of Details, that he or she must delete before being allowed to delete the Master. This does nothing more than to repeat the very old story of all database interfaces, and I do not think there is any opportunity to break new ground in that domain.

2.9 The lists of values

It remains still a characteristic of the forms of Helix that deserve to be pointed out: all the fields contain a list of values. The source of these lists of values varies according to the mode in which the form is currently used (more precisely, it is the state of the master entity recordset).

This can be the creation mode (new record state) or the query mode when the record is empty and all fields ready to be used as part of a filter or query profile (after having clicked on <Enter Query>). In the first case, the lists of values related to fields containing foreign keys, present the different values of the primary key in the referred entity. In the second case, they present only the values existing already in the different records of the master entity of the form. By using the words primary and foreign keys I do not speak of the actual fields because they are hidden numeric, but rather of significant fields from the user point of view of what the relationships may be.

2.10 The Historic

See the description that was already given in the paragraph concerning the administrative data type.

The menu of the form Historic modifies itself according to the object to which one the current record (represented by a single row in a flexgrid) is related. If this object is a hearing device, one will see the option Hearing Device from which one the Hearing Device form can be called. The Hearing Device form contains only the record of the entity Hearing Device for which a predefined type of event was recorded at some point: these predefined types of events for a hearing device are in chronological order “in stock”, “free trial”, “delivery”, or “return to factory”. One will see also the option Patient File, giving access to the form that will be empty if the hearing device was returned to the factory. Some subtle mechanism deserve here also to be pointed out: called in relation to the event “return to the factory”, the Patient File form will remain empty even if, meanwhile, the concerned hearing device has been attributed to someone else (a mind with more subtle criticism could object that in this case it would have been better not to show an option inducing the user to believe that there is something to see in the form). If the record is related to the creation of an audiometry, one will see the option Audiometry, with sub-options giving access to the different types of audiometries. The option Patient File will appear also with, in this case, no questionable pertinence.

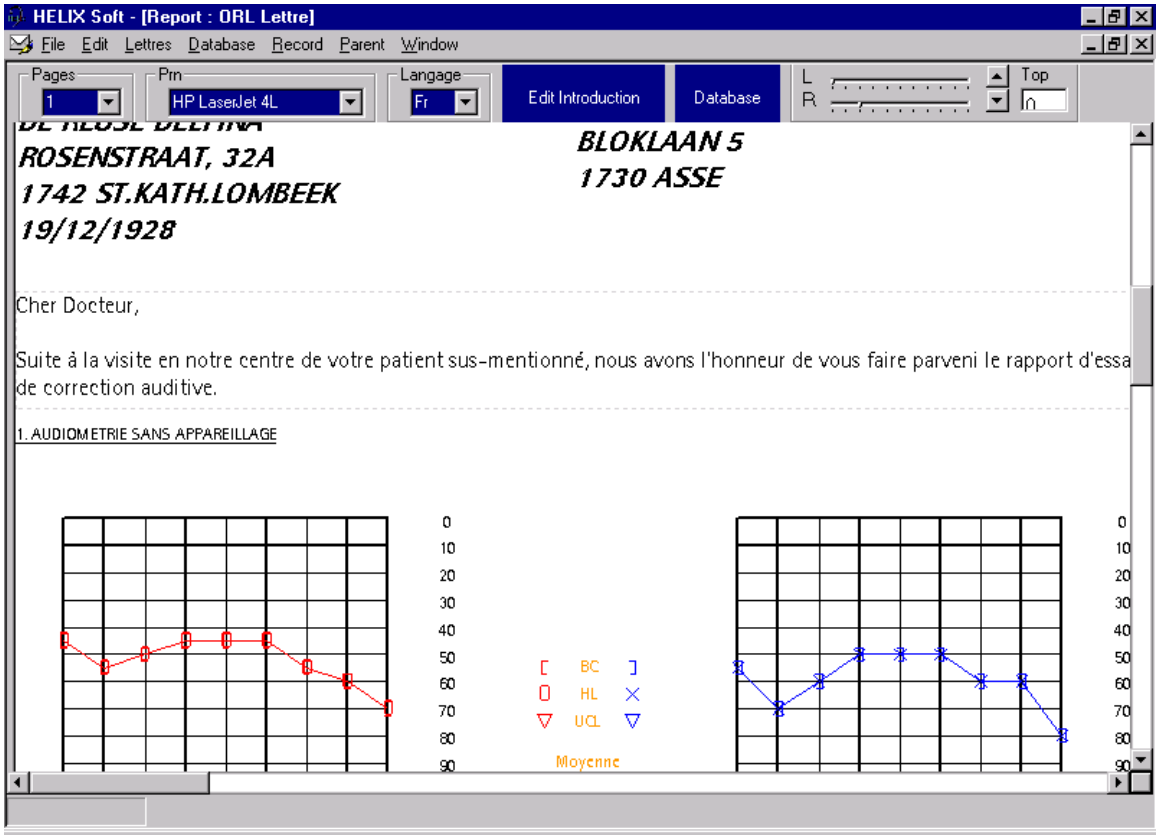
The records are sorted opposite to the chronological order, so that the most recent event appears on the top row.

2.11 The letters editor

A very big and qualitatively important part of the application Helix is the letter editor. Yet from a quantitative point of view, significant even if it is always a little silly, this one amounts to 9000 lines of codes without counting the C_SpecificMultiTextEditor class module that is used to manage several editable and sizable fields, namely text boxes, situated at fixed places in the whole document.

I stressed above (in the paragraph Forms, class modules and modules: towards a layered architecture) the fact that the graphical methods used in all the forms presenting audiometry charts are used also in the functions and procedures of the letters editor that are related to the ORL letter.

A screen copy will give a better idea of this editor than any description lacking its colours and other layout features, all of which, even looking like finery, are functional:



As soon as the mouse pointer run across it, doted lines are surrounding the editable field that contains the introduction formula « Dear Doctor, … » while, on the upper part of the window, two blue rectangles appear. The first one <Edit introduction> indicates the type of text that can be enter in the editable field while the second blue rectangle <Database> means that standard introduction texts can be retrieved from the database.

The audiometry chart underneath can not be modified. The data (among which the points on the curves) come from the database, from which they are retrieved with the same procedures and handle by means of the same memory structures as those used in the various audiometry forms.

The charts and the other pictures cannot be modified, but as one can change de size of the textboxes (by moving their edges with the mouse pointer), the other part of the letter, editable or not, situated underneath but on the same page, are shifted down- or upward accordingly.

The contain of certain textboxes, as the comment about the (hearing device) adaptation type, on the second page of the letter, is generated automatically, but can be refined manually.

There exist 3 categories of information in a letter:

- The header with the names and address, respectively of the patient and of the ORL (or of the general practitioner, or of the mutual benefit society)

- The textboxes

- The pictures

For each of these 3 categories of information, the user can choose a character police that will be memorised and recalled each time he or she will enter into letter editor.

The user can also modify the margins and the space between the header and the top of the page; these features are also saved in the database and recalled each time the user enter into the letters editor.

All the labels and the texts automatically generated in the letter can be generated in Dutch or in French. Note in passing that the whole Helix application is completely bilingual: all the labels of the forms, the captions of the fields, the options of menus are in the language specified to the level of the user profile.

Nevertheless, it is still necessary to be able to choose the language used for the letter for a Dutch speaking audiologist could well address a French speaking ORL and vice versa.

The letter editor is always called from the Patient File form. All information in the letter will always refer to only one patient whose the last audiometry is show by default.

The menu option <record> gives access to the classical sub-options <First>, <Previous>, <Next>, <Last> allowing to change the audiometry record when the patient has several. One can thus generated an ORL letter on the different audiometries linked to same the Patient File occurrence without having to leave the editor.

The number of pages in a letter is virtually infinite, as well as the number of textboxes by page, but the number of pages and the number and places of the textboxes, are set up programmatically in the canvas of the letter, of which the type is defined according to the addressee: letter to the ORL, letter to the G.P., letter to the mutual, letter to the patient.

The conception of this editor is particular if not strange: the textboxes are inserted between non-updatable pictures, as charts. One could have imagined the opposite approach as with a WORD document of which the totality could be edited freely and in which one the pictures would be inserted automatically according to a template. This is probably in this direction that the letters will evolve in the later versions.