

Doktorandský den '04

**Ústav informatiky
Akademie věd České republiky**

Paseky nad Jizerou, 29. září – 1. říjen 2004

Obsah

Josef Špidlen:
MUDRLite - Health Record Tailored to Your Needs

1

MUDRLite - Health Record Tailored to Your Needs

doktorand:

MGR. JOSEF ŠPIDLEN

Institute of Computer Science AS CR, EuroMISE Centre,

Pod Vodárenskou věží 2, 182 07 Prague 8

spidlen@euromise.cz

školicel:

RNDR. ANTONÍN ŘÍHA CSC.

Institute of Computer Science AS CR, EuroMISE Centre,

Pod Vodárenskou věží 2, 182 07 Prague 8

riha@euromise.cz

obor studia:

Biomedical Informatics

číselné označení: 3918V

Abstract

Nowadays most hospitals use an electronic form of health records included into their hospital information systems, but these systems are more focused on the hospital management part than the clinical one. Their usage is more suitable for the hospital management than for physicians. The health record in the information system is not as much structured as necessary, it includes a lot of free-text information, e.g. discharge letters, and the set of structured collected attributes is fixed and practically impossible to be extended. Physicians gathering information for the purpose of medical studies often use various proprietary solutions based on MS Access databases or MS Excel Sheets.

The EuroMISE Centre - Cardio is developing an electronic health record (EHR) application called MUDRLite, which could easily fill the gap among existing EHRs. MUDRLite is being created within the applied research in the field of EHR design, which is based on experience gathered during cooperation in the TripleC project. MUDRLite development is an extra branch in the MUDR (MULTimedia Distributed Record) development within my postgraduate study.

MUDRLite itself is a kind of interpreter, which has to be filled in with a configuration XML file. The XML file completely describes the visual aspects and the behavior of the EHR application. It includes simple 4GL-like constructions written in the MUDRLite Language. This enables - using the event-oriented programming principles - to program various handling of range of actions, e.g. filling a form with a result of an SQL statement after clicking on a button. MUDRLite can be tailored to particular needs of a health care provider. That makes the MUDRLite application easy to use in a specific environment. In the first instance, we are testing it in the Neurovascular Department of the Central Military Hospital in Prague.

1. Introduction

The European Centre for Medical Informatics, Statistics and Epidemiology - Cardio (EuroMISE Centre - Cardio) focuses on new approaches to the electronic health record (EHR) design, including electronic medical guidelines and intelligent systems for data mining and decision support [1]. Cooperating in those research tasks within my postgraduate study I concentrate mainly on the EHR architecture and data storing principles. The participation of EuroMISE in the project I4C-TripleC [2, 3, 4] of the 4th Framework Program

of the European Commission as well as the CEN TC 251 standards and the cooperation with physicians produced much experience, which resulted into a list of 15 requirements on EHR systems [5].

To realize an EHR system, which would fulfill these requirements, EuroMISE Centre is developing an EHR application called MUDR (MULTimedia Distributed Record) [6, 7, 8, 9, 10]. MUDR have it's origin in my diploma thesis [11] which is now being extended, reevaluated, new features are being added etc. Following the requirements stated in [5], the modular structure of the system was defined. It is based on a 3-tier architecture, using a database layer, an application layer and a user interface layer, which enables the separation of physical data storage, application intelligence and the client applications.

The set of collected attributes varies in different departments, organizations and also during time. MUDR uses a dynamically extensible and modifiable structure of items based on a so-called *knowledge base* and *data file* principles as mainly described in [6, 10]. This approach allows the reorganization without change of database structure. It makes the system absolutely universal, but it brings also complications. It is quite difficult to develop universal user interfaces, which would be friendly and comfortable enough. Deploying the MUDR health record into a particular environment demands some effort; the knowledge base must be modeled and built, all the MUDR components must be installed and configured.

Currently most hospitals use an electronic form of health records included into their hospital or clinical information systems, but these systems are often more concentrated on the hospital management part than the clinical part. The usage of such systems is more suitable for the hospital management than for physicians. The health record is not structured as much as necessary, it includes a lot of free-text information, and the set of collected attributes is fixed and practically impossible to be extended. Physicians gathering information for the purpose of medical studies often use various proprietary solutions based on MS Access databases or MS Excel Sheets. MUDR usage in such cases is possible, but this solution may be too complicated and unavailing. Furthermore, the result may not be as user-friendly as a special application dedicated to particular user needs. Those were main reasons why to start another research branch called MUDRLite.

2. MUDRLite

The usage of MUDRLite health record would be an easier solution. MUDRLite is also created within the applied research in the field of EHR design; MUDRLite development is an extra branch in the MUDR development and a part of my postgraduate study; it simplifies both the MUDR architecture and the MUDR data-storing principles.

2.1. MUDRLite Architecture

MUDRLite architecture is based on 2 layers. The first one is a relational database. Currently, MS SQL server versions 7 and 2000 are supported. The second layer is a MUDRLite User Interface running on a Windows based operating system.

The database schema corresponds to the particular needs and varies therefore in different environments, unlike fixed database schema in the MUDR data layer. MUDRLite universality is based on a different approach. The database schema can be designed using standard data modeling techniques, e.g. E-R Modeling. MUDRLite User Interface is able to handle various database schemas. This feature often simplifies the way of importing old data stored using different databases or files.

2.2. MUDRLite User Interface

All the visual aspects and the behavior of the MUDRLite User Interface are completely described by an XML file. The end-user sees a set of forms. A form can be defined by a form element as follows:

```
<form name="new_hosp_form" label="New Hospitalization" author="JS"
  date="27.1.2004" language="en" sizeX="420" sizeY="410"> ... </form>
```

The attributes describe the internal name of the form, the label, which will be presented to the user, who and when has created the form, which language is used in the form and the visual size of the form. The controls on the form are described using various sub-elements like `<button>`, `<combobox>`, `<groupbox>`, `<textbox>`, `<datagrid>`, `<checkbox>` etc.

A control is placed on a form using following syntax:

```
<label name="lbl_patient" label="Patient:" posX="200" posY="315"
  sizeX="80" sizeY="23" tabIndex="2" color="indigo">
```

The attributes `name`, `label`, `posX`, `posY`, `sizeX`, `sizeY`, `tabIndex` and `color` are coincidentally used in all elements describing various controls on a form. They describe the internal name of the control, the label presented to the user, position and size of the control, the tab index and the color of the control. Translating just the labels in the configuration file a new localized MUDRLite User Interface can be created very fast. A choice of over 160 color names is at your disposal. The names are defined in the GDI+ library of the .NET Framework [12]. They cover as well the usual colors as the system defined colors, e.g. the application workspace color, the window color, the window frame color etc. There are some simple elements, e.g. `<groupbox>` or `<label>`, which use just these attributes. Another attribute called `readonly` is very commonly used to determine the possibility of editing a value. More attributes are used by the other elements that will be described in more detail.

The most typical control is a text box defined by the `<textbox>` element. The additional attributes `acceptsReturn` and `acceptsTab` determine, whether the user can enter line breaks and tabulators. By the `multiline` attribute the form designer enables viewing more than one line of text in the text box. Scrollbars are placed using the `scrollbars` attribute, which can take on `none`, `horizontal`, `vertical` and `both` values.

A data grid is also a common control element in database applications. In MUDRLite the `<datagrid>` element uses the `colwidth` attribute to set the preferred column width. Other visual aspects like the columns's titles are being set using the same techniques as for setting the data grid content. They are described later.

Enumerative variables are often bound using a combo or a list box controls. A `<combobox>` element uses the `maxDropDownItems` numerical attribute to specify, how many items can be shown together. The `sorted` attribute determines whether the items should be alphabetically sorted. There are two different styles of the combo box control, a list style, where the user can just select a value from a fixed list, and an editable style, where adding new values to the list is possible. The style is being set by the attribute named `dropDownStyle`. There are two ways how to fill in the combo box with values. The first one is using the `<item>` sub-elements, which define each item by the `value` and `display` attributes. It means that the value stored in database doesn't have to be the same the user can see in the form. This helps to maintain the 3rd normal form (3NF) of the database while keeping the form user friendly. A second way to fill in the combo box is to use an SQL statement. This statement will be processed while loading the form. It should return two columns: a value member and a display member; or just one column in case the value and display members are identical. Technically this is realized using the `<items>` sub-element with the `command` attribute.

There are some more controls being prepared like tree views, tab panels, image boxes, scroll bars, progress bars, status bars, tool bars and tool tips, but they have not yet been completely implemented. Their full description will be included into the MUDRLite Designer's Manual after they have been finished. A small example of a user defined form can be seen in the Figure 1. It is defined as follows:

```
<form name="lab_details" label="Laboratory details" author="JS"
  language="en" date="5.7.2004" sizeX="250" sizeY="260">
```

```

<label label="LDL:" posX="20" posY="38" sizeX="80" sizeY="25"/>
<label label="HDL:" posX="20" posY="68" sizeX="80" sizeY="25"/>
<label label="Total:" posX="20" posY="98" sizeX="80" sizeY="25"/>

<textbox name="t_ldl" posX="100" posY="35" sizeX="130" sizeY="15"/>
<textbox name="t_hdl" posX="100" posY="65" sizeX="130" sizeY="15"/>
<textbox name="t_tot" posX="100" posY="95" sizeX="130" sizeY="15"/>

<groupbox label="Cholesterol (mmol/l)" posX="10" posY="10"
  sizeX="230" sizeY="120" color="red"/>

<label label="Blood Sugar" posX="20" posY="158"
  sizeX="80" sizeY="25"/>
<label label="mmol/l" posX="197" posY="158" sizeX="40" sizeY="25"/>
<label label="Uric Acid" posX="20" posY="188" sizeX="80" sizeY="25"/>
<label label="umol/l" posX="197" posY="188" sizeX="40" sizeY="25"/>

<textbox posX="100" posY="185" sizeX="90" sizeY="15"/>

<button label="Save" name="but_save" posX="20" posY="220"
  sizeX="80" sizeY="23" color="blue"/>
<button label="Cancel" name="but_exit" posX="150" posY="220"
  sizeX="80" sizeY="23" color="blue"/>
</form>

```

The screenshot shows a window titled "Laboratory details" with a "Database" section. It contains a form with the following fields and values:

| Cholesterol (mmol/l) | |
|----------------------|---------------|
| LDL: | 2.70 |
| HDL: | 1.50 |
| Total: | 4.20 |
| Blood Sugar | 9.45 mmol/l |
| Uric Acid | 345.00 umol/l |

At the bottom of the form are two buttons: "Save" and "Cancel".

Figure 1: Example of a simple user-defined form.

This definition creates a form as seen in the Figure 1, but there are two problems left. There is no action connected with pressing the two buttons. The other problem is that even though the form should load the details of selected patient after having started, it always opens empty. Both the problems can be solved using the MUDRLite Language.

2.3. MUDRLite Language

MUDRLite Language (MLL) is a simple event based language used to describe the behavior of the MUDRLite User Interface. It contains 4GL-like constructions, which allow processing of database operations.

The MLL constructions are included in the XML configuration file using the `<action>` element. This element can be placed as a sub-element under a form or under a control element. This determines whether the action should be bound to the entire form or just to a control element in the form. Each action starts on an event. There can be various event types. On controls, the most typical events are a click and a double click. On forms, a typical event is loading a form, closing a form, etc. Events are bound to actions by the `invoke` attribute.

There are various action types. The action content is always described in the content of the `<action>` element. Simple actions work with controls on a form. You can for example clear, hide or show a control using the `<clear>`, `<hide>` and `<show>` sub-elements. The target controls are specified by the `result` attribute referring by their names. Typically, more target controls are specified simply separating their names by commas. Sometimes a reference to a control placed on another form is needed. In this case a dot separated full name is used. It is constructed by the form name connected by a dot with a control name. A dedicated name `parent` is used to reference the parent form; that is the form, which had been active before the current form opened. Therefore no form can be named "parent". The dot separated full name can be more complicated in case you need to address a special part of a control, not the whole control. A typical example is addressing a column of a data grid. In this case a dot is used to connect the control name with the control part name, e.g. the column name of a data grid. In this case the full name can look like this: `parent.patients_grid.patient_id`.

Other action types are working with whole forms. A form can be closed by the `<exit_form>` action element. A new form can be opened by the `<new_form>` element with the `name` attribute, which specifies the form to be opened.

Most powerful actions are using the MLL Language to communicate with the database and to set/get values into/from controls. An extended form of SQL is used in the content of these actions in the `command` attribute. The control names in these commands are bounded with a pair of colon marks. The result controls are specified by their names in the `result` attribute. Again, more result controls can be addressed separating their names or dot separated full names with commas. A select action can be executed by this code:

```
<action invoke="load">
  <select
    command = "select t.tot, t.ldl, t.hdl, t.bs, t.uacd
              from laboratories t
              where t.patient = ':parent.patient_id:' and
                    t.id = ':parent.exam_grid.lab_id:'"
    result = "t_tot, t_ldl, t_hdl, t_bs, t_ua" />
</action>
```

This example shows how the "Laboratory details" form is filled with details of the laboratory examination currently selected in a grid on the parent form. It also demonstrates a simple trick. The patient's identifier is stored in an invisible label, which has the role of an internal temporary variable. When the user presses the "Save" button, this variable is used to specify the patient who should be updated with an MLL update action.

Typically the count of values returning by the select command should correspond to the count of controls specifying in the `result` attribute. An exception is using a data grid as a result control. In this case the column names of the data grid are being specified by the "as" SQL expression.

Insert, update and delete operations are performing by the <insert>, <update> and <delete> elements using the same principles as the <select> element.

There is an additional action included in the first MUDRLite version. A text can be read by the computer using the <speak> action specifying the spoken text by the `text` attribute. So far just the English pronunciation can be used.

Of course, many different actions can be put one after another in a sequence. Figure 2 shows a more complex example – a form designed for the Neurovascular Department of the Central Military Hospital in Prague (English translation).

The screenshot shows the MUDRLite - Neurovascular application window. The window title is "MUDRLite - Neurovascular". The main content area is divided into two sections: "Patients" and "Hospitalizations".

Patients Section:

- Search bar: Name: Alena
- Buttons: Search..., Edit..., New...
- Table with columns: Name, Birth number, Address

| Name | Birth number | Address |
|------------------|--------------|--------------------------|
| Jindráková Alena | <anonymized> | Nikoly Vapcarova 26, P-4 |
| Kalinova Alena | <anonymized> | Teplicka 7, Krupka |
| Kotoučová Alena | <anonymized> | Polská 58, P-2 |
| Mocová Alena | <anonymized> | Kněžmost 225, MB |
| Mrackova Alena | <anonymized> | Tovarni 260, Dubi |
| Nováková Alena | <anonymized> | Jiráškova 4211, Libeň |
| Řípová Alena | <anonymized> | Rytřova 6, P-4 |
| Sedláčková Alena | <anonymized> | Kosmonautů 1550, Turnov |

Hospitalizations Section:

- Buttons: New..., Edit..., Delete...
- Table with columns: Number, Year, Age, Code, Risk group, Out - 1 m, Out - 1 ye, Out - final

| Number | Year | Age | Code | Risk group | Out - 1 m | Out - 1 ye | Out - final |
|--------|------|-----|------|------------|-----------|------------|-------------|
| 5 | 1992 | 58 | 1 | 2 | 1 | 3 | 3 |
| 6 | 1996 | 62 | 2 | 2 | 2 | 1 | 1 |
| 7 | 2001 | 67 | 2 | 3 | 2 | 2 | 1 |

Hospitalization Details:

- Input Finding: Aneurism, rep.
- Risk Factors: Smoking 20/day, diabet.
- Summary: Surg. op.
- Checkboxes: Aneurism, AVM, Carotids
- Exit button

Figure 2: MUDRLite form designed for the Neurovascular Department of the Central Military Hospital in Prague.

3. MUDRLite Deployment

MUDRLite deployment to a particular environment needs preparation phases. First of all, the physicians must specify what information should be stored. This must be done precisely. A model including all collected attributes must be built. It must include the attribute types, the relationships among them, units of measurement, specification of numerical attributes' precisions etc. Together with data engineers an entity-relationship model (E-R Model) is built.

Two more phases are following in parallel. One of them is the data migration from an existing system. MUDRLite is seldom being deployed to an environment, where no data has been collected yet. This is being done using various techniques and SQL Server Data Transformation Services. The result of the second phase is the definition of MUDRLite user-defined forms and MUDRLite application behavior using the MLL Language. Various XML editors can serve to help creating the XML configuration file. A special application called "MUDRLite Forms Designer" is being prepared to simplify this phase. After the XML configuration file is finalized, MUDRLite should be tested. Then the application is being fine-tuned according the physicians' comments.

4. Results and Future Work

MUDRLite testing confirmed that this health record is flexible enough to allow dynamical changes of the database structure demanding no or just small changes in the configuration XML file. The XML file can be constructed using various XML editors, but we are preparing a MUDRLite Forms Designer to make this process easier. The 2-tier architecture separates the user interface from the data storing part. This enables a remote access to the health record. To make the remote access more flexible we plan to develop a Pocket PC version of the MUDRLite User Interface, which should run on various portable devices.

We have also verified the functionality and simplicity of the MLL Language. It is useful and sufficient for many applications, mostly thanks to the power of the SQL. But we still would like to increase the power of the MLL Language. We plan to do this by including arithmetical expressions and elements of logical conditions into the language. We are aware of the fact that we also have to keep it as simple as possible.

On the other hand we found out that the bottleneck that eliminates usage of the system as a clinical information system lies in the absence of interfaces to other EHR systems. That is why we have started the research how to support such interfaces. Because of the varying database layer we can not implement such support directly into the MUDRLite application but we are trying to suggest special communication modules which could be connected to MUDRLite. Each module should support one EHR data standard, e.g. the Czech data standard "DASTA" or in Western Europe commonly used "HL7". We suppose there will always be another additional configuration file needed to define mapping between values stored in the database and attributes transferred via the selected standard.

5. Conclusion

Our interest is to increase the quality of EHR systems, to simplify data sharing and data migration among various EHR systems and to help in overcoming the classical free-text based health record. This is the way, which would increase the quality of healthcare, which brings benefit for the patient first of all. Most healthcare providers use a kind of an EHR system. But often the health record is not structured as much as necessary. Physicians gathering information for the purpose of medical studies often use varied proprietary methods.

We present the MUDRLite universal solution. It is an easy way to build electronic health record tailored exactly to your needs. We recognize just one problem that can emerge in case the user needs more security and finer rights policy than the database server can provide. Better policy can be hard to gain because we are mainly using the database server authentication and authorization (A/A) rules that seem to be sufficient in most environments. In the first instance, we are deploying MUDRLite to the Neurovascular Department

of the Central Military Hospital in Prague.

Acknowledgment

The work was partially supported by the grant number LN00B107 of the Ministry of Education of the Czech Republic.

References

- [1] J. Rauch, “Mining for Statistical Association Rules”, In: *J. Fong, M.K. Ng (eds.): The Fifth Pacific/Asia Conference on Knowledge Discovery and Data Mining*, University of Hong Kong, pp. 149–158, 2001.
- [2] I. Iakovidis, “Towards Personal Health Record: Current Situation, Obstacles and Trends in Implementation of Electronic Healthcare Record in Europe”, In: *Int J Med Inform*, Vol. 52, pp. 105–115, 1998.
- [3] F.H. Pierik, A.M. Ginneken, T. Timmers, H. Stam, R.F. Weber, “Restructuring Routinely Collected Patient Data: ORCA Applied to Andrology”, In: *J.H. Bemmell, A.T. McCray (eds.): Yearbook of Medical Informatics 98, Health Informatics and the Internet*, Schattauer, Stuttgart, pp. 257–263, 1998.
- [4] A.M. Ginneken, “The Computerized Patient Record: Balancing Efort and Benefit”, In: *Int J Med Inform*, Vol. 65, pp. 97–119, 2002.
- [5] P. Hanzlíček, “Development of Universal Electronic Health Record in Cardiology”, In: *G. Surján, R. Engelbrecht, P. McNair (eds.): Health Data in the Information Society*, IOS Press, Amsterdam, pp. 356–360, 2002.
- [6] J. Špidlen, A. Říha, “Electronics Health Record and Telemedicine”, In: *F. Hakl (ed.): Institute of Computer Science AS CR - PhD Conference Proceedings*, Matfyzpress, Paseky nad Jizerou, pp. 133–143, 2003 (in Czech).
- [7] J. Zvárová, P. Hanzlíček, J. Špidlen, “Electronic Health Record in Cardiology: Pilot Application in the Czech Republic”, In: *MIST2002 Proceedings*, John Wiley & Sons Pte Ltd., Taiwan, pp. 10–13, 2002.
- [8] J. Špidlen, J. Adášková, H. Heroutová, J. Zvárová, I. Mazura, “Statistical Processing of Genetic Information in the MUDR Health Record”, In: *J. Vignerová, J. Riedlová, P. Bláha (eds.): Anthropology and Society*, Charles University, Prague, p. 190, 2003.
- [9] P. Hanzlíček, J. Špidlen, J. Zvárová, “The Internet in Connecting Electronic Health Record Mobile Clients”, In: *Technology and Health Care*, Vol. 10, Num 6, IOS Press, pp. 502–503, 2002.
- [10] J. Špidlen, P. Hanzlíček, “The Implementation of Formalized Medical Guidelines in the MUDR Electronic Health Record”, In: *V. Svátek (ed.): Znalosti 2003 Proceedings of the 2nd Annual Conference*, VŠB-TU Ostrava, Ostrava, pp. 386–391, 2003 (in Czech).
- [11] J. Špidlen, “Database Representation of Medical Information and Guidelines”, *Diploma Thesis*, Charles University in Prague, Faculty of Mathematics and Physics, Prague, 2002 (in Czech).
- [12] Microsoft Corporation, “Microsoft .NET Framework 1.1 Class Library Reference Volume 7: System.Windows.Forms, System.Drawing, and System.ComponentModel”, Microsoft Press, ISBN: 0-7356-1818-6, 2003.