

# MUDRLite – Health Record Tailored to Your Particular Needs

Josef SPIDLEN, Petr HANZLICEK, Jana ZVAROVA  
*EuroMISE Centre – Cardio, Institute of Computer Science AS CR  
Pod Vodarenskou vezi 2, 182 07 Prague 8, Czech Republic*

**Abstract.** Nowadays most hospitals use an electronic form of health records as a part of their hospital information systems. However, these systems are 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 varied 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 fulfill the gap among existing EHRs. MUDRLite is 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; it simplifies both the MUDR architecture and the MUDR data-storing principles.

MUDRLite itself is an empty body, 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 (MLL). This enables - using the event-oriented programming principles - to program various handling of range of actions, e.g. clicking on a button fills in a form with a result of an SQL statement. MUDRLite can be tailored to particular needs of any health care providers. 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]. The participation in the project I4C-TripleC [2, 3, 4] of the 4<sup>th</sup> 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]. Following the requirements stated before, 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 collectable 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. This approach allows the reorganization without change of database structure. It makes the system absolutely universal, but it raises 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 have an electronic form of health records included into their hospital or clinical information systems. But these systems are often 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 varied 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 intended for a particular user's needs.

## **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; 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 scheme corresponds to the particular needs and varies therefore in different environments, as opposed to the 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 [11]. MUDRLite User Interface is able to handle varied database schemas. This feature often simplifies the way of importing old data stored with 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 through following syntax:

```
<label name="but_del_hosp" label="Delete..." 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. A new localized MUDRLite User Interface can be created very fast, just by translating the labels in the configuration file. 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 the usual colors as well 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 only these attributes. Another attribute, called `readonly`, is very often used to determine the possibility of editing a value. The other elements use more attributes that will be described in detail.

The most typical control is the 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 the `none`, `horizontal`, `vertical` and `both` values.

The 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, as the columns' titles, can be set by the same techniques as setting of the data grid contents. These are described later.

Enumerative variables are often bound with combo or list box controls. The `<combobox>` element uses the `maxDropDownItems` numeric 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, which enables adding new values to the list. The style is being set by the `dropDownStyle` attribute. There are two ways how to fill in the combo box with values. The first one is using the `<item>` sub-elements, which define an item by the `value` and `display` attributes. It means that the value stored in database doesn't have to be the same as the one the user can see in the form. This helps to maintain the 3<sup>rd</sup> normal form [13] 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 by the `<items>` sub-element with the `command` attribute.

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

```
<form name="edit_patient" label="Edit patient details" author="JS"
language="en" date="18.2.2004" sizeX="320" sizeY="290">
  <label label="Name:" name="lbl_patient" posX="20" posY="38"
sizeX="80" sizeY="25" color="indigo"/>
  <label label="Address:" name="lbl_address" posX="20" posY="98"
sizeX="80" sizeY="25" color="indigo"/>
  <label label="Gender:" name="lbl_gender" posX="20" posY="168"
sizeX="80" sizeY="25" color="indigo"/>
```

```

<textbox name="e_nm" posX="100" posY="35" sizeX="200" sizeY="15"/>
<textbox name="e_ad" posX="100" posY="65" sizeX="200" sizeY="60"
  acceptsReturn="true" multiline="true"/>
<combobox name="c_g" posX="100" posY="135" sizeX="200" sizeY="21"
  sorted="true" maxDropDownItems="2" dropDownStyle="list">
  <item value="1" display="man"/><item value="2" display="woman"/>
</combobox>
<label label="" name="l_id" posX="0" posY="0" visible="false"/>
<groupbox name="g_p" label="Patient details" posX="10" posY="10"
  sizeX="300" sizeY="190" color="red"/>
<button label="Save" name="bs" posX="20" posY="250" sizeX="120"
  sizeY="23" tabIndex="0" color="blue"/>
<button label="Cancel" name="bc" posX="180" posY="250" sizeX="120"
  sizeY="23" tabIndex="1" color="blue"/>
</form>

```

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

The definition above creates a form as seen in the Figure 1, but there are two problems left. There is no action connected with pressing each of the two buttons, no matter how many times the user clicks, nothing happens. 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 by 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 with an event. There can be various event types. The most typical events of controls are a click and a double click. With forms, a typical event is loading a form, closing a form, etc. Starting events are bound to actions by the `invoke` attribute.

There are various action types. The action content is always described in the contents 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 identified by their names. Typically, more target controls can be 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 to the control name. The reserved 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 instead of 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 to 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`.

Another 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. For example a form editing patient details can be started by the following code:

```
<action invoke="click">
  <new_form name="edit_patient"/>
</action>
```

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 contents 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 full_name as 'Name', address as 'Address',
              sex_id as 'Gender', patient_id as 'Patient ID'
    from      patient
    where     patient_id = ':parent.patients_grid.patient ID:'"
    result   ="e_nm, e_ad, c_g, l_id"/>
</action>
```

This example shows how the “Edit patient details” form is filled with details of the patient currently selected in the patients’ grid of 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 user, which 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 specified 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 specified by the “as” SQL expression. Insert, update and delete operations are performed 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 through the `<speak>` action, which specifies the spoken text by the `text` attribute. So far only the English pronunciation can be used.

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

**Patients**

Name:

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>	Jiráskova 4211, Libeň
Řípová Alena	<anonymized>	Rytířova 6, P-4
Sedláčková Alena	<anonymized>	Kosmonautů 1550, Turnov

**Hospitalizations**

Number	Year	Age	Code	Risk grou	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:

Risk Factors:

Summary:

Aneurism  AVM  Carotids

**Figure 2.** MUDRLite form - Neurovascular Department of the Central Military Hospital in Prague.

### **3. MUDRLite Deployment**

MUDRLite deployment to a particular environment expects 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 as well types as the relationships among them, units of measurement, specification of numerical attributes' precisions etc. Together with data engineers, an entity-relationship model (E-R Model [11]) is built.

Two more phases are following parallelly. 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. 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. To simplify this phase, a special application called "MUDRLite Forms Designer" is being prepared. After the XML configuration file is finalized, MUDRLite should be tested. Then the application will be fine-tuned according the physicians' comments.

### **4. Results**

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 much simpler. 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; it is mostly thanks to the power of the SQL. But we still would like to increase the power of the MLL Language more. 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.

### **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 way the quality of healthcare could be increased, which brings benefit to 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. In the first instance, we are deploying MUDRLite to the Neurovascular Department of the Central Military Hospital in Prague. We also started cooperation with the Dental Medicine Department of the Charles University, 1<sup>st</sup> Faculty of Medicine, which should result in spreading MUDRLite among

Czech stomatology ambulances. We hope this will introduce our solutions into the real practical use, which should bring advantages for the health care in the Czech Republic.

## **Acknowledgment**

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

## **References**

- [1] Rauch J.: Mining for Statistical Association Rules. In: Fong J., Ng M.K. (eds.): The Fifth Pacific/Asia Conference on Knowledge Discovery and Data Mining, University of Hong Kong (2001) 149-158
- [2] Iakovidis I.: Towards Personal Health Record: Current Situation, Obstacles and Trends in Implementation of Electronic Healthcare Record in Europe. *Int J Med Inform* 52, Nos. 1-3 (1998) 105-115
- [3] Pierik F. H., Ginneken A. M., Timmers T., Stam H., Weber R. F.: Restructuring Routinely Collected Patient Data: ORCA Applied to Andrology. In: Bommel J. H., McCray A. T. (eds.): Yearbook of Medical Informatics 98, Health Informatics and the Internet, Schattauer, Stuttgart (1998) 257-263
- [4] Ginneken A. M. The Computerized Patient Record: Balancing Efort and Benefit, *Int J Med Inform* 65, (2002) 97-119
- [5] Hanzlicek P.: Development of Universal Electronic Health Record in Cardiology. In: Surján G., Engelbrecht R., McNair P. (eds.): Health Data in the Information Society. Amsterdam, IOS Press (2002) 356-360
- [6] Zvarova J., Hanzlicek P., Spidlen J.: Electronic Health Record in Cardiology: Pilot Application in the Czech Republic. In: MIST2002 Proceedings, Taiwan, John Wiley & Sons Pte Ltd. (2002) 10-13
- [7] Spidlen J., Adaskova J., Heroutova H., Zvarova J., Mazura I.: Statistical Processing of Genetic Information in the MUDR Health Record. In: Vignerova J., Riedlova J., Blaha P. (eds.): Anthropology and Society, Charles University, Prague (2003) 190
- [8] Hanzlicek P., Spidlen J., Heroutova H.: User interface of MUDR Electronic Health Record, *Medical Informatics Europe Proceedings CD* (2003)
- [9] Hanzlicek P., Spidlen J., Zvarova J.: The Internet in Connecting Electronic Health Record Mobile Clients, *Technology and Health Care*, Vol. 10, Num 6, IOS Press (2002) 502-503
- [10] Spidlen J., Hanzlicek P.: Implementace formalizovaných lékařských doporučení v elektronickém zdravotním záznamu MUDR. In: Svatek V. (ed.): Znalosti 2003 proceedings of the 2<sup>nd</sup> annual conference, VŠB-TU Ostrava (2003) 386-391
- [11] Reingruber M. C., William W. G.: The Data Modeling Handbook : A Best-Practice Approach to Building Quality Data Models, John Wiley & Sons, Inc. (1994), ISBN: 0-471-05290-6
- [12] Microsoft Corporation: Microsoft® .NET Framework 1.1 Class Library Reference Volume 7: System.Windows.Forms, System.Drawing, and System.ComponentModel, Microsoft Press® (2003), ISBN: 0-7356-1818-6
- [13] Piattini M., Diaz O.: Advanced Database Technology and Design, Artech House, Inc. (2000), ISBN: 0-89006-395-8

## **Address for correspondence**

Josef Spidlen, EuroMISE Centre – Cardio, Institute of Computer Science AS CR,  
Pod Vodarenskou vezi 2, 182 07 Prague 8, Czech Republic  
Email: spidlen@euromise.cz