MyCare Card development: the patient held electronic health record device

V O Rybynok, P A Kyriacou, J Binnersley, A Woodcock, L M Wallace

Abstract— **In the UK, in emergency situations, health professionals rely on patients to provide information about their medical history. However, in some cases patients may not remember their medication, long term illnesses or allergies, or be able to communicate this information. As a national on-line integrated patient record system has not yet been established, this paper introduces an on-going project 'MyCare Card' abbreviated as MyC^2 which aims to design and implement a patient held electronic health record device and corresponding user interface software.**

Index Terms — e-Health Systems, m-Health Systems, Telemedicine Systems

I. INTRODUCTION

In many areas of health care in the UK, particularly emergency care, health professionals rely on patients to provide information about their medical history. However, reliable information may be difficult to acquire from patients who are unwell, confused, or have communication difficulties. It has been suggested that patients taking responsibility for their records would improve safety [1].

Although previous studies have shown positive attitudes towards patient held records [2-3] and patient held electronic records [4], there is still much to learn about users` views on these. Patient held, paper based records related to maternity and child health have been used effectively for a number of years [2-3], yet this practice has not been adopted by other parts of the health service.

An electronic form of patient held records was successfully trialled in the UK, between 1989-1992 [4] where over 13,000 patients were provided with smart cards containing health information that only they and health professionals treating them were able to access. The results showed that the majority of participants were in favour of having the cards [4]. However, their use was not continued, as the technology available at the time limited its wider feasibility.

To date, few studies have identified the design requirements for patient held health records in the UK, such as the preferred form of device, methods of data entry, access rights or the information content, and compared these requirements to those of health professionals. Those emergent requirements were used in this project to produce a prototype system (records media and access software) which is being continuously evaluated and re-factored in agile development style. The system code name utilised in this project is *MyCare*. An agile development style has been used in this research in preference to the more traditional waterfall style approach [5]. As this project is aimed to design and implement a system, which will be intuitive and transparent for a non-technical user, the development had to be focused on the end-user interface and not on the internal communication and database formats. Thus, the software Graphical User Interface (GUI) has been developed first. Initially it is tied to the simple and unencrypted data format directly supported by the selected programming language and environment (see Section II). The class and module levels separation of the GUI from the data interface code allows the internal data presentation to be freely restructured while changing the GUI structure and controls types under technically sketchy and continuously changing user requirements. When firm decisions are made about the functionality of the software's user interface more sophisticated and secure internal data formats may be defined in the form of a data-base type and structure standard.

MyCare health record device development is split in two subprojects: *MyCare Card* – medical records storage media development; and *MyCare Card Browser* – GUI and database software which will allow card owners and health professionals to view and edit where appropriate information stored on the *MyCare Card*.

This paper presents the details of the conducted UK survey aiming to collect attitudes to electronic patient health records and report on the ongoing development of the *MyCare* system which includes the GUI, data base, and records storage device. Also, justification for all software development tools used for this project is discussed.

II. METHODS AND MATERIALS

A. *Establishing initial user requirements*

In order to establish the end user requirements of the system, two 'similar' questionnaire surveys were designed to collect attitudes, to patient held records and requirements for an electronic patient held record device, from the public and health professionals. The questions were based on material derived from a preliminary literature review, five focus

The research has been funded by the EPSRC grant number FP/F00323411 28/02/2007

V. O. Rybynok is with the School of Engineering and Mathematical Sciences, City University London, London, UK

P. A. Kyriacou is with the School of Engineering and Mathematical Sciences, City University London, London, UK

J. Binnersley is with the School of Art and Design, Coventry University, Coventry, UK

A. Woodcock is with the School of Art and Design, Coventry University, Coventry, UK

L. M. Wallace is with the Faculty of Health and Life Sciences, Coventry University, Coventry, UK

groups (with a total number of 25 participants) and interviews with ten health care professionals.

The first part of each questionnaire asked participants to supply demographic information to check that the sample included participants with a wide range of demographic characteristics. Participants were then asked about their experiences of using patient held records. Following this, a brief description of the proposed patient held record device was given and participants asked questions about using this.

Over 500 participants took part in the survey. Approximately half of these were members of the public and half were health care professionals. For the public survey, participants had to be sixteen years or older. Convenience sampling was used to include customers in pharmacies across the UK. These included participants from a range of ethnic backgrounds and social classes. Members of the public who had problems with their eyesight or English literacy were assisted to complete the questionnaire. The age of the respondents ranged from 17-89, with a mean age of 45 years. 43% were male and around two thirds of white European origin. Just over a quarter considered themselves to have long term health problems. 13% had used some form of patient held health records, and these were mainly maternity records.

Focus groups were held for groups of people who were unable to participate in the survey. One of the groups consisted of people who did not speak English and their views were included using the services of an interpreter. Another group of disabled participants were unable to write and so would not have been able to complete a questionnaire. Groups of elderly people, teenagers, health care professionals and those with long term health problems explored the questionnaire items in more detail than was possible during the survey.

Ethical clearance was obtained to consult health care professionals. These were drawn from doctors, nurses, ambulance staff, pharmacists, physiotherapists, occupational therapists and other health professionals. Quota sampling was used in order to achieve similar numbers of individuals from each professional group. Data was collected in different areas of the UK in order to include participants from different geographical locations and working environments. 260 questionnaires were completed by health care professionals. 39% were male, 79% of white European origin and over half were under the age of forty. Data was collected from five professional groups: doctors (14%), nurses (23%), ambulance staff (23%), pharmacists (20%) and others (20%). Approximately half of this group had used some form of patient held health record, and cited the major benefit of doing so as being access to health information.

Answers from the two surveys were coded and entered into separate databases for the public and professional surveys. The analysis used the Statistical Package for the Social Sciences (SPSS). Open ended questions were thematically analyzed to establish the main themes from the responses, with content analysis used to establish the most common responses.

The results were used to derive an initial set of development requirements for the user interface software, the date which needed to be stored and the preferred storage device. When initial development requirements became available, software tool chain, programmatic libraries and development model were selected. Bug tracking and source code version control systems were also established.

B. *Open Source development model*

To reduce the cost of development and to minimize its dependency on the major computer systems manufacturers, Open Source software tools and programmatic libraries were used. As *MyCare* is a community driven type of project, the selection of Open Source development model[1] will enhance its dissemination and perhaps its wide acceptance by a large international community. Therefore, *MyCare* project may achieve the maximum development performance under the Open Source development model.

C. *Programming language, environment and GUI toolkit*

Python programming language [6-8] has been utilised in this project as a major language, run-time environment and common algorithms infrastructure. The reason why Python has been selected for this project is because it is a dynamic object-oriented programming language that can be used for many kinds of software development tasks. It offers strong support for integration with other languages and tools, and comes with extensive standard libraries [6].

wxPython was used as a primary GUI toolkit. C++ programming language was selected for security-related and low-level access algorithms implementation. *SWIG* (Simplified Wrapper and Interface Generator) is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages including Python. SWIG was chosen in this project to join low-level C++ code components with a high level Python code. Python library *py2exe* and the Windows programs installation packages builder named *Inno Setup* were utilised to make software distribution packages familiar to Windows users.

The flexibility of the Python language makes wxPython much easier to develop than its C++ counterpart, while the native C++ code of wxWidgets gives the Python GUI both the speed and native look and feel it would otherwise lack.

Thus, the *MyCare* system aims to be user-friendly, Open Source and community driven, easily extendable, cross-platform, portable, stable and secure. The tool chain described above allows the *MyCare* project to achieve its aims within reasonable timeframes.

D. *Media type selection*

According to the survey (see Section II.B) smart card type of media was preferred over other proposed devices such as USB sticks, key fobs, jewellery and devices linked to a mo-

[1] Decision on code licensing has not been made yet. The main considered option is GNU General Public License version.

bile phone.

Smart cards advantages: common and widely accepted.

Smart cards disadvantages: small memory (mainly designed to store ID and security keys) which will only allow a limited number of medical records; slow data read/write rate; requires an external card reader for every computer type, and a card reader device driver installation; requires card browser installation (Administrator access); only Windows OS can be supported with the available resources and timing (due to the different driver requirements in different OSes).

USB sticks disadvantages: less common; not widely accepted and trusted.

USB sticks advantages: large memory (4GB and above – future-proof, enough to store full MRI scans for example); high read/write speed; does not require external card reader for most computer types (including PDAs); does not require card browser installation (no Administrator access required); most of the modern OSes can be supported (all Windows versions, Mac OS and Linux, i.e. those supported by Python/wxWidgets).

Additionally if the smart cards approach was taken, it would require the *MyCare Card* reader device, the firmware for the reader, the firmware for the card, the card reader Windows driver and the middle-ware which connects the driver and the *MyCare Card* Browser software. All of these will limit the usability and ultimately portability of the final system. With a USB stick none of the above is required apart from the Card Browser software. The media interfacing infrastructure is already supported on every computer/mobile device with the USB bus and modern OS.

Considering the advantages and disadvantages of traditional smart cards and modern USB sticks, the use of USB sticks is preferable. However, given the public's preference for a smart card, a compromise had to be found between the two types of data storage. Going back to the end-user survey it seemed that the major perceived difference between the devices was the shape and style, i.e. it had nothing to do with the storage capacity, cost, connection type, or communication protocol. Thus, as a compromise a USB card design shown in Fig. 1 was chosen which combines the advantages of both.

III. Results

A. Establishing data types specification

85% of the public said they would find an electronic patient held medical record device useful, especially if they were too ill to give information to a health professional. Positive comments from the focus group participants included, `Patient records seem to be a good way forward in both giving patients responsibility and ensuring information is passed on.` Another said, `Well, I`m allergic to penicillin and if I ever had to go into hospital and was unconscious and didn`t have anything with me, they wouldn`t know I was allergic to penicillin and they would probably give me penicillin.` Several of the participants who had problems under-

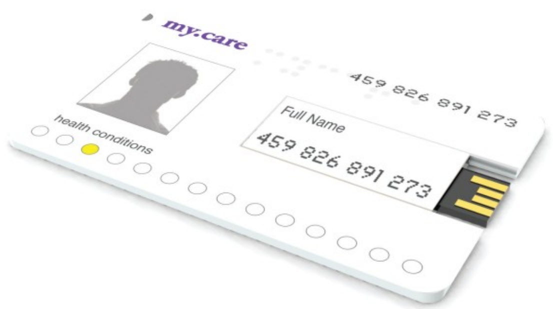


Fig. 1: Proposed *MyCare* USB card design example

standing English felt that a device would be particularly helpful for them.

The most common concern for the public was the possibility of unauthorized people gaining access to the information, which was indicated by 64% of the survey participants. Subsequent focus groups explored concerns about data security. Participants suggested features which could be incorporated into the design in order to improve data security such as the encryption of data and use of personal identification numbers. There was also support for the device having an access log that recorded who had accessed and/ or altered the information.

Approximately half of this group had used some form of patient held health record, and cited the major benefit of doing so as being access to health information. The concerns of this group related to inaccuracy of information (74%), loss of records by the patients (80%) and unauthorized access (75%).

Some 94% of the health professionals said they would find a patient held record useful and, in this case, the most common reason was to overcome communication problems.

Regarding the types of information that should be stored on the device, most of the members of the public surveyed thought that current medication, name, allergies, blood group and long term conditions should be included and that all health professionals should be able to access these items. However, over three quarters also agreed that access to other information should depend on the role of the health care professional.

For health care professionals the most important pieces of information to be held on the device related to allergies, current medication, name, long term conditions, age, major health problems in the past and next of kin. The majority of these participants thought that all health professionals should be able to access the most important pieces of information. Again, the majority of participants supported the use of a restricted access system, where the viewing of certain pieces of information was restricted to particular groups of professionals.

B. Files and modules structure

Based on preliminary data types and storage media specifications and end- user expectation, an initial test version of the *MyCare Card* Browser software has been implemented.

The source code directories tree listing of the software is shown in Lst. 1.

Lst. 1. *MyCare Card* Browser software directories tree

```
src
├─cards
├─data*
├─gui*
│ ├─containers*
│ │ ├─decl*
│ │ ├─media
│ │ └─samples
│ └─controls*
│   ├─decl*
│   └─samples
├─media
└─samples
```

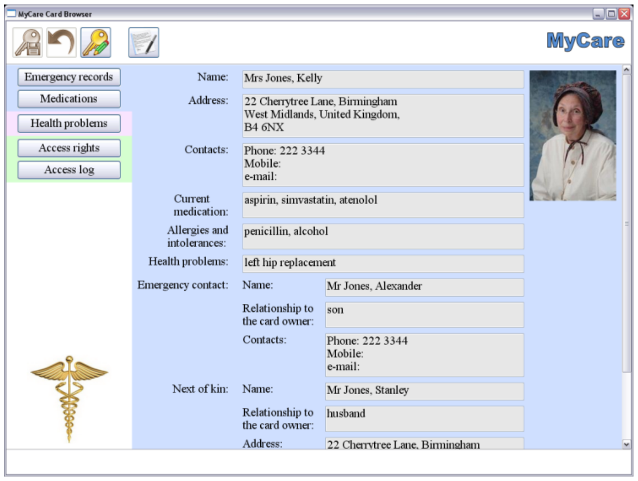
Directories marked with '*' are Python packages. Root directory `src` contains the program main executable script `mycare.py` and `mc.py` module with binder classes, which join wxWidgets standard GUI controls classes and classes contained in `gui/controls` directory with the data access code contained in `data` directory. The `src` directory also contains `security.py` module, which currently contains simple classes representing user authentication and security management functionality. In future versions of the software those classes will be converted into proxies for the more advanced analogue classes, implemented in C++.

The `gui` directory contains evt.py module with wxWidgets events declarations and `helper.py` module with common algorithms, such as warning or error messages pop-ups. `gui/containers` directory contains modules with classes for components acting as containers for the components displaying and editing medical data. Classes for displaying and editing components are contained in `gui/controls` directory. Apart from the container and control apparent roles separation there is another significant difference between these directories: all modules contained in the `controls` directory only depend on Python and wxPython libraries' modules; while modules in the `containers` directory also depend on `mc.py` module in the source tree root.

In the current version, the `data` directory contains the `model.py` module with the data structure model definition, and modules with classes which provide data save, load and individual fields access functionality through the common class interface. In further development the same interface will be used to access the proxy class for the C++ code which will provide access to the data stored on the *MyCare* Card.

C. GUI model

`gui/containers/decl` and `gui/controls/decl` directories contain modules, automatically generated from interface declaration XML files with GUI components presentation and layout descriptions. Interface declaration XML files are currently handled by the wxGlade GUI designer and stored in its native wxg file format. In further development, wxWidgets native xrc resources file format will be used together with dynamic loading (parsing) instead of the current-



Name:	Mrs Jones, Kelly	
Address:	22 Cherrytree Lane, Birmingham West Midlands, United Kingdom, B4 6NX	
Contacts:	Phone: 222 3344 Mobile: e-mail:	
Current medication:	aspirin, simvastatin, atenolol	
Allergies and intolerances:	penicillin, alcohol	
Health problems:	left hip replacement	
Emergency contact:	Name:	Mr Jones, Alexander
	Relationship to the card owner:	son
	Contacts:	Phone: 222 3344 Mobile: e-mail:
Next of kin:	Name:	Mr Jones, Stanley
	Relationship to the card owner:	husband
	Address:	22 Cherrytree Lane, Birmingham

Fig. 2: *MyCare Card* Browser GUI screen example.

ly utilized static code generation.

Interface declaration classes in `decl` directories are extended with corresponding GUI logic and parametrically controlled functionality in modules contained in `gui/containers` and `gui/controls` directories. This extension is done by inheritance, constructors overloading and defining new event handlers. Such approach allows us to separate GUI presentation, GUI logic and GUI content code parts and already proved its efficiency. Example of a typical *MyCare* Card Browser screen in its current version is shown in Fig. 2.

D. Data-flow model

Module `mc.py` which defines binder classes can be considered as the *MyCare* Card Browser engine. GUI and data classes binding is based on the idea of multiple inheritance and Run Time Type Information (RTTI), natively supported by Python. The key methods code of the core classes in the `mc.py` is shown in Lst. 2.

Lst. 2. Key methods of the Card Browser engine core classes.

```python
class DataControl:
    def __init__(self):
        self._dataPath = None
        self._accessLevelPath = None
        self._logRec = ""
    def UpdateControlFromData(self):
        raise NotImplementedError()
    def SetDataPath(self, value):
        self._dataPath = \
            self.__FormatPath(value)
    def SetAccessLevelPath(self, value):
        self._accessLevelPath = \
            self.__FormatPath(value)
    def SetLogRec(self, value):

class DataViewControl(DataControl):
    def __init__(self):
        DataControl.__init__(self)
        self.__dataFormatter = FormatNone
    def SetDataFormatter(self, value):
        self.__dataFormatter = value

class DataEditControl(DataControl):
    def __init__(self):
        DataControl.__init__(self)
    def UpdateDataFromControl(self):
        raise NotImplementedError()
```

```
class DaPanel(wx.Panel):
    def __init__(self, *args, **kwds):
        wx.Panel.__init__(self, *args, **kwds)
        # Update all Data Aware controls
        # when their panel is shown
        self.Bind(wx.EVT_SHOW, self.OnShow)
    def UpdateControlsFromData(self):
        for ctrl in self.GetChildren():
            if isinstance(ctrl, DataControl):
                ctrl.UpdateControlFromData()
    def OnShow(self, event):
        if not event.GetShow(): return
        evtsrc = event.GetEventObject()
        if isinstance(evtsrc, DaPanel):
            evtsrc.UpdateControlsFromData()
        event.Skip()
```

When GUI control has to be "data aware" and to view or to edit certain fields of the data structure, it simply inherits from the "normal" wxWdgets control and from either `DataViewControl` or `DataEditControl`. Such inherited class must only overload `UpdateControlFromData()` function if it is a view control, and in case if it is an edit control, it must also overlord `UpdateDataFromControl()`.

`DataControl::_dataPath` instance variable defines path to the data represented by the control. `DataControl::_accessLevelPath` defines path to the data record, which represents access level to the data represented by the control. `DataControl::_logRec` instance variable defines data access log message which identifies accessed data field in a human readable format. When a new instance of the "data aware" control is constructed, all the above mentioned variables have to be assigned using `SetDataPath()`, `SetAccessLevelPath()` and `SetLogRec()` member functions of the `DataControl` class.

All "data aware" controls have to have "data aware" containers (i.e. parent windows). In the current version of the software only `DaPanel` container type is supported. It is inherited from the `wx.Panel`.

There are two major advantages of the described approach: firstly the association between data path and "data aware" controls can be defined during controls classes instances construction, instead of maintaining a separate associations file; and secondly the container classes automatically "know" from RTTI request (`isinstance()` called from `DaPanel::OnShow()`) which contained controls must be updated when a show event occurs. The first advantage also allows defining data path and controls associations from some wxWidgets GUI designers, which support extra properties setting or defining new design-time components. The latter significantly simplifies development by joining GUI layout definition with data path allocations.

IV. Conclusion and Further Work

The initial survey results have been used to draw up a list of user requirements that was used to produce 3D and physical design models of *MyCare* Cards and Card Browser software. In terms of GUI and data structure design, initial development cycles have led to a working software prototype which has been further evaluated in terms of its aesthetics, ease of use and the ease with which different levels of access can be associated with different types of information.

The research highlights that the initial barriers to the use of electronic health record devices will be the security of information. In the proposed system, data can only be accessed using a personal identification number, and will only include information that the individual is willing to enter and share with others. A similar level of authentication will be required by health professionals to read it and, therefore, such fears appear to be exaggerated from a computing perspective. Additionally the device will be independent of centrally held records, so will not provide a route in to more sensitive information, The developed *MyCare Card Browser* classes interfaces allow the embedding of the user authentication and data encryption algorithms minimizing the potential for fraudulent use of devices that have been reported as lost or stolen.

The current version of the *MyCare Card Browser* source, its corresponding tool chain, revision control and issue tracking web-systems represent a flexible and extendable software infrastructure which will be used as a platform in the final stage of the *MyCare* project development.

Future stages of the research will entail the development of more detailed usage scenarios, *MyCare Card* prototype development, testing of the device and evaluation of the usability of the *MyCare Card Browser* interface. The final stage of the research will use the experiences of the project as a starting point for a series of dissemination activities across the UK, which will broaden discussion of the personal and shared responsibilities for health care.

References

More information on *MyCare* project and *Card Browser* source code is available at http://mycare.webfactional.com

[1] J. Hall, "Workshop discussion paper: what roles for the patient in patient safety research?," in Patient Safety Research Conference, Porto, 2007.

[2] A. J. Hampshire, M. E. Blair, N. S. Crown *et al.*, "Variation in how mothers, health visitors and general practitioners use the personal child health record," *Child: Care, Health and Development,* vol. 30, no. 4, pp. 307-316, 2004.

[3] H. Phipps, "Carrying their own medical records: the perspective of pregnant women.," *Australian and New Zealand Journal of Obstetrics and Gynaecology,* vol. 41, no. 4, pp. 398-400, 2001.

[4] NHS Management Executive, "The care card: evaluation of the Exmouth project," *London: HMSO*, 1990.

[5] J. Younker, *Foundations of Agile Python Development*: APRESS, 2008.

[6] "Pyhon homepage," http://www.python.org/.

[7] D. M. Beazley, *Python Essential Reference*: Sams, 2006.

[8] A. Martelli, *Python in a Nutshell (In a Nutshell*: O'Reilly Media, Inc., 2006.