The 4th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2014)

Development of a Resident Practice Profile in a Business Intelligence Application Framework

Austin Chamney, Pilar Mata, Gary Viner, Doug Archibald, and Liam Peyton*

a Faculty of Engineering, University of Ottawa, 800 King Edward Avenue, Ottawa K1N 6N5, Canada
b Faculty of Medicine/Department of Family Medicine, University of Ottawa, 43 Bruyère Street, Ottawa K1N 5C8, Canada
c Bruyère Research Institute, 85 Primrose Avenue, Ottawa K1R 7G5, Canada

Abstract

In recent years, practice profile applications have started to appear that allow medical practitioners to log their experiences electronically to provide feedback in the form of reports. This paper presents a case study in which a Resident Practice Profile application was developed using a Business Intelligence application framework that flexibly integrates user interfaces and reporting while minimizing the labour required for application coding. The methodology used in the case was iterative action based research in which three different frameworks were used and compared in terms of their effectiveness for developing a practice profile application. As well, the Resident Practice Profile application was compared to two similar practice profile applications from the literature.

© 2014 The Authors. Published by Elsevier B.V.

Keywords: Application Framework, Business Intelligence, Development Methodology, User Interface Design, Healthcare, Practice Profiles, Performance Measurement

1. Introduction

This paper presents a case study in which a Resident Practice Profile (RPP) application was developed using a business intelligence (BI) application framework that flexibly integrates user interfaces and reporting while minimizing the labour required for application coding. The application was developed in collaboration with a team of researchers and family medicine doctors at the University of Ottawa in order to support the training of residents. Both the application and the framework used to build the application are evaluated in this paper.

* Corresponding author. Tel.: +1-613-562-5800x2122; fax: +1-613-562-5664
E-mail address: lpeyton@uottawa.ca

1877-0509 © 2014 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/).
Peer-review under responsibility of the Program Chairs of EUSPN-2014 and ICTH 2014.
doi:10.1016/j.procs.2014.08.040
Nomenclature

RPP  Resident Practice Profile
BI  Business Intelligence
EHR  Electronic Health Record
ETL  Extract, Transform and Load
J2EE  Java 2.0 Enterprise Edition
AJAX  Asynchronous JavaScript and XML

2. Background

Recent studies highlight the gap between the quantity and type of clinical problems residents see during their training and those required by program guidelines. Techniques such as self-directed reading have been added but they are usually guided by recent experiences. An Electronic Health Record is often not adequate as there is usually a disconnect between the EHR and clinical practice. Typically, the EHR logs data related to resources and billing but data related to diagnosis and treatment are absent or logged as textual descriptions. In addition, residents often work at several clinical environments each with a different electronic medical record system that is not linked to a central EHR. Practice profile applications have appeared that allow medical practitioners to log their experiences electronically to provide feedback. Just in Time Medicine (JIT) was developed by researchers at the University of Michigan to enable physicians to provide assessments of medical students on their clinical experience, while LogMD was developed in conjunction with the Association of Canadian University Departments of Anesthesia to enable physicians to track their experience against international benchmarks.

A practice profile is a type of business intelligence (BI) application used to collect data in order to monitor and measure performance. There is increasing pressure on health care organizations to deploy such applications to measure performance related to quality of care goals. Traditionally, BI applications collect data by Extracting, Transforming and Loading (ETL) data from existing operational systems (like an EHR) into a specialized reporting database optimized for reporting. Our design strategy is to use an application framework to develop browser-based applications that collect data directly into a reporting database to bypass the complexity, effort, and processing delays associated with ETL. A critical aspect of such applications is to provide seamless easy-to-use interfaces that minimize the data collection burden and maximize the value that reports provide.

An application framework is "an integrated set of software artifacts ... provide a reusable architecture for a family of related applications" so that new applications can be developed quickly. Traditional web application frameworks, such as Struts for J2EE web applications, provide a three tier architecture (browser-based user interface, application server, and database) where most development effort is directed towards code that runs on the application server in a manner that makes it difficult to directly address user interface or reporting effectiveness. AJAX is newer technology which expands the development effort to the browser-based user interface to provide more seamless and easy-to-use interfaces. However, reporting is still an issue as there is no direct relationship between data collected in these user interfaces and the effectiveness of reporting based on that data.

3. Resident Practice Profile Case Study

RPP is a practice profile application that enables residents in the Family Medicine program to self-assess how well their clinical experience under the supervision of a physician supervisor, provides a suitable breadth of experience across the types of patients, diagnoses and procedures that are covered in the postgraduate Family Medicine curriculum. The main requirements for RPP were that:

- Residents should be able log their clinical experience (patient visits) in less than 30 seconds on a mobile device.
- Residents should see reports that map their clinical experience in relationship to the Family Medicine curriculum.
Doctors responsible for Family Medicine program should be able to configure the system through a web browser.
The system should support 3rd party reporting tools for in-depth analysis to improve Family Medicine program.

The first step of the case study was to do a gap analysis of existing technologies. Using an EHR was considered, but it was not practical for several reasons. First, there was no EHR common to all the clinics where residents practiced. But, even if the case study was restricted to a single clinic, EHR data was inconsistent with the Family Medicine curriculum; there was no mechanism for doctors to configure the EHR to collect what was actually needed; and there was no built-in support for 3rd party reporting tools. Just In Time Medicine and LogMD were also evaluated as potential technologies, but they had a different purpose (assessment of medical students, self-assessment of physicians) and were not customizable to what was needed for Family Medicine residents. A comparison of them and the RPP application that was built for the case study is given in section 5.

The next step of the case study was to iterate through the construction of three candidate RPP applications using action research methodology, in which iterative problem solving within a community is done to evolve better approaches. The three candidate RPP applications were built using three different application frameworks: a traditional J2EE web application framework (NetBeans 8.0), and two versions of our own open source application framework that is specifically optimized for developing BI applications (QuickForms 1.0, and QuickForms 3.0). A team of software engineering and health science researchers and students worked collaboratively with doctors responsible for the Family Medicine curriculum to develop the RPP application for residents and their supervisors in the Family Medicine program. In addition to regular group evaluation sessions of the application and its design, there were three formal Think Aloud sessions conducted with the participation of four residents in addition to the research team between September 2012 and June 2013 before starting an official pilot of RPP with the four residents in July, 2013. A total of 1607 diagnoses for 1205 different visits were logged during the pilot. The final step of the case study was to evaluate the results, which is described in section 5.

4. Business Intelligence Application Framework

The main idea in an application framework is to capture common functionality in a family of applications in a reusable architecture so that very little custom code needs to be written. Creating an application should largely be a task of configuration and linking of pre-defined components. For our BI application framework, we defined an architecture that separated data collection user interfaces (forms) from the reporting database (dimensional model) in such a way that they can be configured separately but automatically linked to define a BI application. The pre-defined components in our framework were a library of mobile app controls, web page templates, and a scripted database schema. In this section, we will first describe the evolution of the architecture for our BI framework, and then give a summary of the most useful pre-defined components used in the RPP application.

4.1. Architecture

Figure 1 shows the initial J2EE architecture we started with, and the final QuickForms 3.0 architecture in terms of code complexity. In the J2EE architecture (on the left), there is potentially a great deal of complexity and code related to all the tiers that mediate and link between the user interface and the database which contains data structured around the Application Model. It can be complex to define reports against this database since it is optimized for the application and not reporting. Often there is an ETL process to move the data to a database structured around a Dimensional Model which is optimized for reporting and compatible with 3rd party reporting tools. There is no direct mapping between data elements in the User Interface, Application Server, Data Access, Application Model, or Dimensional Model. Maintaining consistency between them is a painstakingly manual process as any change is potentially repeated in all five places, and may require custom coding by five different developers. There are no pre-defined components to optimize the User Interface or Reports for a BI application.

In the QuickForms-based architecture (on the right in Figure 1), there is no Application Model, and the Application and Data Access tier are completely pre-defined. There is no customer code to write for them, as they
are designed to automatically manage the relationship between User Interface and Dimensional Model. The complexity of these two tiers for the application developer is essentially reduced to zero. A developer simply has to annotate the fields in an HTML form with the Model Names they correspond to in the Dimensional Model. Functions in the QuickForms Library automatically populate drop down choices for fields and save collected data to the Dimensional Model through communication with the QuickForms Data Access Service. The QuickForms Library also provides a powerful set of pre-defined components to optimize the User Interface (including built-in reports) for a BI application.

Fig. 1: Comparison of J2EE and QuickForms Architecture

In our case study, our initial RPP application built in J2EE has 3,154 lines of code and it became quite complex to manage all the custom code in the different tiers. Worse, it was not easy to optimize the User Interface for different types of mobile devices and form factors, and it was difficult to create anything other than simple reports. Our second RPP application, built in QuickForms 1.0, had 4,551 lines of code, almost all of which were written in JavaScript. QuickForms 1.0 provided the basic architecture shown in Figure 1 but the QuickForms Library was poorly organized with no attempt to encapsulate re-usable pre-defined components. There was a lot of custom code in the User Interface that was very complex. All the desired functionality for RPP was delivered, but it was just as complex for a developer to build as the initial RPP application. The final RPP application built in QuickForms 3.0 has only 608 lines of code. The functionality for the user is the same as the second RPP application, but the QuickForms Library and User Interface was completely refactored. There is now a systematic approach to encapsulating reusable components so that developing a BI application is now largely a matter of configuration.

4.2. Pre-Defined Components

4.2.1. Templates and Summaries

Figure 2 shows an overview of the RPP application. On the left is the home page which uses one of the built-in home page templates with header, footer, application tabs, and a standard login/logout. The titles, tabs etc. are customizable in an application configuration file. There is a sophisticated, built-in grid control for displaying visits that supports filtering, searching, sorting by clicking on columns etc. The columns used both in the grid and in the associated filters are all easily configured. On the right is the overview summary for a particular visit. The user can see at a glance a summary of what has been logged for Tracking, Demographics and Assessment without opening the forms for those. The form summaries are a built-in control that can be flexibly configured to determine what fields are summarized. The idea for form summaries came from one of the Think Aloud sessions when residents were having difficulty seeing the complete overview of what they had filled in for a patient visit. As well, the Tracking summary greatly simplified data entry. Typically, a resident would see many patients in a row at a given clinic. When the Tracking information was entered once for a visit, it was automatically repeated and pre-filled for subsequent visits and the resident could see at a glance that no data entry was needed for tracking.
4.2.2. Simple Forms and Reports with Lookup Table Management

The critical element of the QuickForms architecture is, of course, the ability to separate and yet automatically link forms with the Reporting Database. In Figure 3, we see the Demographics Form on the left and its associated Age / Gender report on the right. The developer links a field in the form to the dimensional table that supplies the drop down values for that field by simply adding the Model Name of the table to the form. These same values are used in the report associated with the form. The Age/Gender Report uses the same values for Age and Gender that appear in the selection controls for Age and Gender. In this case, the resident has seen a disproportionate number of females to male patients in the 21-35 year range, but this is normal for family doctors who see maternity patients. There is also a similar Special Populations / Gender report corresponding to the Care of Special Populations drop down in the form, but this is not shown here. In addition, there is a special Design button on the form on the left that is available for users with Administrator privileges. The design button brings up a dialog where administrators can edit the values in the lookup tables for any of the form fields, or download and upload files of the values for a particular field.
4.2.3. Multi-Level Selection Control and Drillable Reporting

The form for Assessment is more complex. In Figure 4 on the left, we see a tabbed multi-level selection control provided by QuickForms that a resident can use to select any and all diagnoses and procedures that apply to a patient. There are roughly 500 diagnoses and procedures covered by the Family Medicine curriculum and they are organized in the multi-selection control in a similar, though simplified, fashion to their classification in the curriculum. At the top level, there are six tabs corresponding to the major curriculum domains: Care of Adults, Care of Children, Care of Elderly, Maternity, End of Life, and Procedures. Within each tab, there is another list of groupings, and within each grouping there are the selections to be made. In this case, the resident is selecting “Removal of Foreign Body” from the list of Eye, Ear and Nose Procedures under the Procedures tab. The organization of the tabs and groupings and selections is automatic based on the lookup table for the assessment field in the database. All the developer has to do is specify the column name in the fact table that links to the lookup table. The organization is driven by the classification columns (one for tabs, one for groupings), and the sequence number column to specify the order of values. As well the Design Button facilitates maintenance of diagnoses and procedures by launching a dialog that allows separate files to be uploaded and downloaded for each tab.

The drillable report on the right in Figure 4 leverages the same classification columns from the assessment lookup table to organize the report of how many assessments and procedures a resident has seen. One can see at a glance that, in addition to Eye, Ear and Nose procedures, the resident has done very few Gastrointestinal, Injection, or Resuscitation procedures. In addition, the vast majority of their diagnoses have been for the Care of Adults.

5. Results

We compare our experience building RPP using three different frameworks in Table 1, while in Table 2 we compare our final RPP version with two other practice profile applications. In Table 1, we can see that QuickForms 1.0 achieved our main user requirements of optimizing and linking user experience and reporting, while QuickForms 3.0 greatly reduced the complexity of the code through better encapsulation and packaging of pre-defined components. This improved the ability to configure the application and greatly reduced development effort.

In Table 2, RPP, Just In Time Medicine (JIT) and LogMD are compared. JIT and LogMD were reviewed by the entire team of software engineering and health science researchers, students and doctors but not the residents.
(because JIT and LogMD had a different target audience and purpose). The main difference between them and RPP is the fact that the QuickForms application framework allows RPP to be completely configured and customized, whereas JIT and LogMD have been optimized for a particular task.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>J2EE</th>
<th>QuickForms 1.0</th>
<th>QuickForms 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Factors</td>
<td>PC browser optimized</td>
<td>All form factors</td>
<td>All form factors</td>
</tr>
<tr>
<td>Form / Report Linkage</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3rd Party Reporting</td>
<td>No. Hard-coded reports</td>
<td>Yes. Dimensional model</td>
<td>Yes. Dimensional model</td>
</tr>
<tr>
<td>Application Configuration</td>
<td>None</td>
<td>Data Tables</td>
<td>Data Tables. UI Templates.</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>No. Mixed layers</td>
<td>No. Complex JavaScript</td>
<td>Yes. QuickForms library</td>
</tr>
<tr>
<td>Lines of Code</td>
<td>3154</td>
<td>4551</td>
<td>608</td>
</tr>
</tbody>
</table>

RPP, JIT and LogMD are, at a high level, similar BI applications. All three are focused on reporting and ease-of-use for data entry. JIT has a rigid longer data entry to ensure the same steps are followed for all students to ensure consistency in how students were assessed for a single clinical experience. RPP was designed for faster data entry, because it was important for a resident to log all patient encounters so they could self-assess their entire clinical experience. RPP had the most sophisticated support for capturing diagnoses, because the full spectrum was needed to capture the clinical experience. The QuickForms application framework had built in support for BI applications that made the linking between forms and reports more direct and in general improved reporting and configuration.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>JIT</th>
<th>LogMD</th>
<th>RPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Student assessment</td>
<td>Physician self-assessment</td>
<td>Resident self-assessment</td>
</tr>
<tr>
<td>Form Factors</td>
<td>All form factors</td>
<td>All form factors</td>
<td>All form factors</td>
</tr>
<tr>
<td>Data Entry</td>
<td>2-5 minutes. Rigid.</td>
<td>&lt;2 minutes</td>
<td>&lt;30 seconds</td>
</tr>
<tr>
<td>Diagnosis Selection</td>
<td>One level</td>
<td>Two levels (procedures)</td>
<td>Two levels (all)</td>
</tr>
<tr>
<td>Form / Report Linkage</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Templated Summaries</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Drill-through Reporting</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>3rd Party Reporting</td>
<td>No. Hard-coded reports</td>
<td>No. Hard-coded reports</td>
<td>Yes. Dimensional model</td>
</tr>
<tr>
<td>Application Configuration</td>
<td>None</td>
<td>None</td>
<td>Data Tables. UI Controls.</td>
</tr>
</tbody>
</table>

6. Conclusions and Future Work

The QuickForms 3.0 framework has greatly reduced the effort in building a BI application like RPP, and it has greatly improved the configurability and user experience for both data collection and reporting compared to what our research team was able to achieve with a traditional web application framework or QuickForms 1.0. More work is needed, of course, before we can make any definitive claims about how well the framework would work for other teams and other applications. It should be noted that there is a number of simplifying assumption in our work:

- No rich content or content management (Voice, Pictures, Attachments)
- Multi locale issues are ignored.
- Concurrency issues in which two or more users might edit the same form or edit the same lookup table at the same time are ignored. The nature of our applications so far, allow us to assume only one user at a time
Communication, service, and/or application integration issues. Applications are strictly standalone.

Security and fault tolerance issues except for obvious issues are ignored. The simplicity and benign nature of our application in a research setting has allowed this so far.

We believe our approach should be useful for most form-based monitoring and logging applications where data is collected for analysis and/or performance management. In addition to RPP, there are several other applications that graduate students from the University of Ottawa are in the process of building for ongoing case studies.

- Cardiac Care Patient Flow Tracking System (hospital)
- Palliative Care Patient Tracking System (community)
- Software Engineering Course Preferences Tracking System
- Blogging Application (Enter and report on blog entries for any blog)

Acknowledgements

This work was supported by an AIME grant from the School of Medicine at the University of Ottawa, the Mitacs Accelerate program, IBM and an NSERC Discovery Grant. We would like to thank the residents who participated in this case study, especially Alexandre Labelle. We would also like to thank Dr. Susan Humphrey-Murto and Dr. Eric Wooltorton for their participation and extremely useful feedback throughout this research.

References