Development of an Interactive Application for Learning Medical Procedures and Clinical Decision Making

Marcus Bloice, Klaus-Martin Simonic, Markus Kreuzthaler, and Andreas Holzinger

Institute for Medical Informatics, Medical University of Graz, Auenbruggerplatz 2, 8036, Graz, Austria

Abstract. This paper outlines the development of a Virtual Patient style tablet application for the purpose of teaching decision making to undergraduate students of medicine. In order to objectively compare some of the various technologies available, the application was written using two different languages: one as a native iPad app written in Objective-C, the other as a web-based app written in HTML5, CSS3, and JavaScript. The requirements for both applications were identical, and this paper will discuss the relative advantages and disadvantages of both technologies from both a HCI point of view and from a technological point of view. Application deployment, user-computer interaction, usability, security, and cross-platform interoperability are also discussed. The motivation for developing this application, entitled *Casebook*, was to create a platform to test the novel approach of using real patient records to teach undergraduate students. These medical records form patient cases, and these cases are navigated using the Casebook application with the goal of teaching decision making and clinical reasoning; the pretext being that real cases more closely match the context of the hospital ward and thereby increase *authentic activity*. Of course, patient cases must possess a certain level of quality to be useful. Therefore, the quality of documentation and, most importantly, quality's impact on healthcare is also discussed.

Keywords: Virtual Patients, Patient Records, Decision Making.

1 Introduction

The term *Virtual Patient* is an umbrella term for a type of interactive medical system used for teaching and learning medicine, especially clinical reasoning. According to the electronic Virtual Patients (eViP) project website a Virtual Patient can be formally defined as "an interactive computer simulation of real-life clinical scenarios for the purpose of medical training, education or assessment [8]". These systems exist in many forms, and range from physical robotic patients, entire hospital simulation systems, to online accessible interactive patient cases. The application presented in this paper takes a slightly different approach,

A. Holzinger and K.-M. Simonic (Eds.): USAB 2011, LNCS 7058, pp. 211–224, 2011.

using real electronic patient records to display patient cases to students. Using this application, users can interact with the case and make decisions based on the information contained within the case itself. Cases are presented linearly (this early version of Casebook supports linear cases only), and students assume the role of the physician examining a patient's history. At strategic points throughout the case, the student is asked what their next course of action would be based on the information known to them up to that point, thereby mimicking the context of a hospital ward. This is known as the authentic activity of learning. The full requirements of the application are described in Section 1.1 below.

Of interest to the HCI community are the experiences that were gained when developing two versions of the same application using two distinctly different technologies. Specifically, to what degree does the choice of technology affect the usability of Casebook? The application itself employs a multi-touch, gesture-based approach, and the feasibility of using HTML to implement this is also outlined. Therefore, the question of whether a web-based application can compare to a native application in terms of the user experience is the main focus of Section 2.

After the development stage of Casebook is described, the motivation and reasoning behind creating a Virtual Patient based on patient data are discussed. Section 3 addresses the motivation behind creating a Virtual Patient application using patient cases based on real medical records. First, it is argued that patient cases that consist of electronic medical records more closely match the context of the hospital ward. Second, it provides an opportunity to analyze the impact of the quality of documentation on its usefulness as teaching material.

1.1 Requirements

As stated previously, the requirements were identical for both the HTML and Objective-C versions of Casebook. Namely, the application should allow a student to view and browse patient cases, comprising of electronic health records, where at certain points stipulated by the case creator the student must answer questions as to what their course of action might be. Records can be physician notes, sonographies, radiological images, and so on. Students begin by opening a case and viewing the first patient record available (this is usually the patient presentation). Using gestures, the students can navigate through the case from left to right using a linear timeline of the patient records. Upon reaching a question, students are asked about their next course of action, and are given a choice of four answers. Upon answering, the next medical record is displayed, revealing the correct answer. At the end of the case, the student is given a summary of how they scored. Cases themselves are created manually by the teacher and bundled as a ZIP archive before being uploaded and read by the Casebook application. These ZIP files must follow a strict structure so that they can be correctly read by Casebook.

Each case consists of a number PDF files linearly named from 1 to n (e.g. `1.pdf`, `2.pdf`, ..., n`.pdf`). Any questions that should appear between patient records for

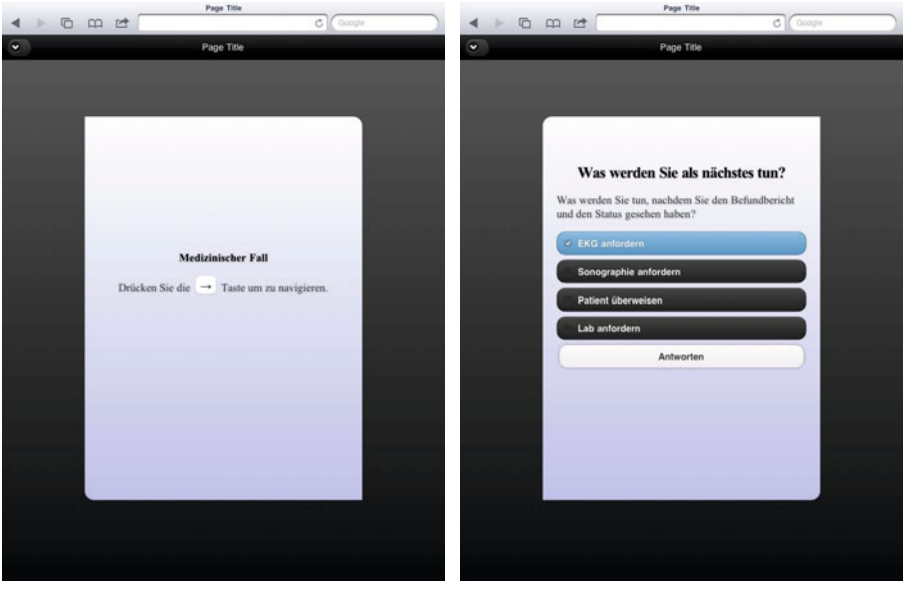
the student to answer are stored in XML files, where each question is represented by one XML file. Question XML files are named according to where they are positioned between PDF documents, with the first part of the file name denoting the previous document and a second part denoting the subsequent document, separated by a dash. For example the file `2-3.xml` denotes a question that appears between documents `2.pdf` and `3.pdf`. An example question XML file is shown just below. All PDFs must be portrait oriented and A4 in size to ensure design consistency. Each individual case is contained within a single ZIP archive.

```
<question previousFile="2.pdf" nextFile="3.pdf">
    <headerText>
        What is the your next course of action?
    </headerText>
    <text>
        After seeing the Befundbericht and Status,
        what is the next course of action?
    </text>
    <option>
        Request an ECG
    </option>
    <option correct="true">
        Request a Sonography
    </option>
    <option>
        Transfer Patient
    </option>
    <option>
        Request a Lab
    </option>
</question>
```

Concerning the usability requirements, the application was designed to be operated on tablet devices, specifically the iPad. Cases should be navigable through the use of swipe gestures, allowing for the case to be traversed both forwards and backwards. Zooming is accomplished using a standard pinch gesture. An overview of the case's timeline can also be viewed at any time (see Figures 2b and 4a). It was also a requirement that the transitions between documents be animated to emphasize the timeline-based view of the case, progressing from left to right.

2 Development

Two versions of Casebook were created in parallel, both targeting the iPad. One version was developed as a standard, native Objective-C iPad application while the other was written in HTML as a web-based solution. The HTML version



(a) The opening screen of a case. Notice that the case can be navigated using a keyboard as well as with a swipe gesture.

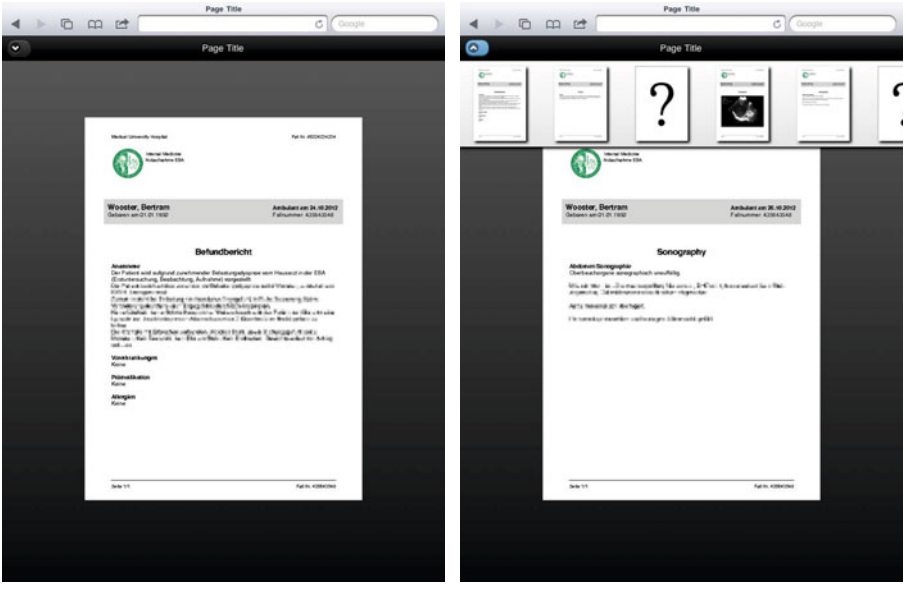
(b) Answering a question. Students are presented with questions at strategic points throughout a case, but can progress further whether they answer correctly or not.

Fig. 1. Casebook HTML application in use

aimed to mimic the look and feel of a native application, as is shown in Figure 1a and Figure 2.

In order to mimic a native application's look and feel, the jQuery Mobile framework was used. This is a "touch-optimized framework for smartphones and tablets [17]" that allows advanced interfaces to be built targeting multiple platforms. Although Casebook was developed specifically with iOS in mind (iOS being the iPad operating system), the jQuery framework also supports Android, BlackBerry, Samsung bada, Windows Phone, Palm webOS, Symbian, MeeGo, and even the Amazon Kindle. Our development focused primarily on targeting the WebKit browser engine, the rendering engine used by both iOS and the Android OS—an iPad was used for testing purposes.

The goal of developing Casebook in HTML as well as Objective-C was to be able to judge HTML's applicability in creating a touch screen, gesture-based application for viewing medical records. Optimized views of many websites exist for touch screen tablets, and several HTML applications rival native applications in terms of their usability and look & feel (see for example the Financial Times Web App: `http://app.ft.com`). Also, showcase demonstrations such as Sencha's TouchSolitaire application (see `http://www.touchsolitaire.mobi`) highlight just what is possible using HTML, CSS, and JavaScript. Having said that,



(a) Viewing a single patient record (b) Viewing the timeline of the case

Fig. 2. Casebook HTML application in use, using the timeline

our main concern before development began was performance. Due to the nature of patient records and their large file size and high resolution, it was questionable whether a browser would be able to handle large cases containing 30 to 40 patient records.

Nonetheless, there are a number of advantages inherent to web-based application development that made the idea of using it as a basis for Casebook compelling, especially considering that Casebook was intended for in-house use within the hospital campus and not for worldwide deployment. However, developers must also be aware of a number of compromises if deciding to write software for the iPad, or any other tablet, in HTML. This section will discuss the development of Casebook in both HTML and Objective-C and will emphasize the advantages and disadvantages of each. First, some general observations regarding both technologies will be made, followed by a discussion of some issues that were encountered during development that were specific to Casebook.

2.1 HTML

The following list of observations were made in favor of HTML for iPad or tablet development:

App Store. Perhaps the most obvious advantage in developing web-based applications is that the Apple App Store is avoided. This bypasses the need for paying Apple a developer fee, and also means the application is not at the mercy of the approval process.

Integrated Development Environment. When writing web applications, there is far more freedom in terms of what IDE or OS one can use for development—when writing native apps for iOS an Intel Macintosh is required, and realistically Xcode is the only IDE that can be used (although technically speaking any editor could be used in conjunction with the apple-darwin9-gcc-4.2.1 compiler).

Device Independence. Using a framework such as the aforementioned jQuery Mobile, or Sencha Touch, multiple devices can be targeted. This potentially means a higher customer base and there is certainly more flexibility in terms of what hardware can be purchased.

Updates. Updates made on the server side are instantly propagated by subsequent client requests. Fragmentation of client versions is avoided.

In-house Distribution. By controlling access to within your LAN or WAN, the distribution of web-based applications can be tightly managed. iOS applications can also be distributed in-house, but an enterprise developer account is required to do so.

Development Platform. A dual core Intel-based Apple Mac computer is required for iOS development. Almost any computer can be used to develop web-based applications.

Device Specific. HTML applications can be designed to accept keyboard input and therefore can also function on standard PCs. Google Chrome and Apple Safari are both based on the WebKit rendering engine, the same engine used by the iPad and Android browsers.

2.2 Objective-C

A number of advantages that exist when developing in HTML that may make it convenient for certain types of deployment, especially for in-house applications and applications that must run on a range of devices. At the same time, web-based applications are not ideal for all situations. During the development of Casebook, the following list of general observations were made in favor of writing programs natively (i.e., in Objective-C):

Monetization. Apple has paid out over $2 billion to developers since opening the App Store, allowing publishers to charge users anything from between $0.99 and $999.99 for their applications. Web-based applications, on the other hand, must rely on advertising schemes such as Google's Adsense for revenue. It is worth noting that reliable information concerning the click through rate (CTR) for mobile advertisements is difficult to find, and would warrant further study. It is certainly conceivable to suggest that the CTR is lower for mobile devices such as the iPhone than for desktop machines, especially when one considers the interruption incurred when a new browser window is opened in iOS.

Hardware Access. Native applications have access to the complete array of sensors on the iPhone or iPad, including the GPS device, magnetometer, accelerometer, and gyroscope. Conversely, HTML applications have no access

to a device's hardware. There are efforts to help reduce this deficit, such as the HTML5 `geolocation` API which can resolve the user's location based on their IP address, although with a far lower accuracy than that of GPS.

iOS API. The iOS API is large and mature, with extensive documentation available from directly within the Xcode IDE. While the HTML5 specification has been finalized, implementation of this specification is incomplete and differs from engine to engine. Documentation is therefore fragmented and sparely distributed. JavaScript also suffers from the lack of a centralized point of access for documentation and this is compounded by the fact that JavaScript code is interpreted differently from browser to browser. By all accounts, using a single, officially supported API has definite benefits for the developer.

Xcode. The Xcode IDE itself is a powerful platform on which to work, complete with a debugger, code completion, as well as the aforementioned inline help, automatic error detection, and an interface builder. Developing HTML applications requires more effort on the part of the programmer, with many editors offering no more than syntax highlighting. Writing complex HTML and JavaScript without the aid of a debugger is an error prone and difficult process. That said, IDEs designed specifically for HTML5 development, such as Aptana, are becoming more common, and Ext Designer even includes an interface builder.

Security. By deploying apps to your users via the App Store, you mitigate the risk of a malicious attack on your own servers or hardware. HTML applications must be hosted, at both your expense and your risk. Web servers are potential targets for attacks, including denial-of-service exploits and outright theft of confidential data.

Notifications. Native applications can make use of Apple's notification framework to send messages to users, even when the app is closed. As of yet, there is no way in which a HTML application can do this, although there is a draft specification that aims to address this (WebKit's `webkitNotifications` is one implementation of this draft, for example).

Multi-touch. Multi-touch is an inherent part of the Objective-C framework for developing mobile applications. As described previously, advanced touch-enabled applications can be built using frameworks such as Sencha Touch or jQuery Mobile. Sencha, for example, comes with a gesture library [11]. However, these are third party libraries that could disappear at any time, or change their terms of service or license agreements to be incompatible with your project.

If your application requires specific iOS features such as the ability to send notifications, or requires access to the hardware of the device on which it is running, there is no choice but to develop natively. If not, however, there is little reason to dismiss HTML. There are also some caveats that must be considered, such as the reliance on third party frameworks, the lack of a definitive and comprehensive IDE and debugger, and somewhat fragmented help. But this is

certainly changing; more robust and mature IDEs are appearing, and there are even projects being developed that convert HTML applications into their native equivalents (for example PhoneGap, see `http://www.phonegap.com`)[20].

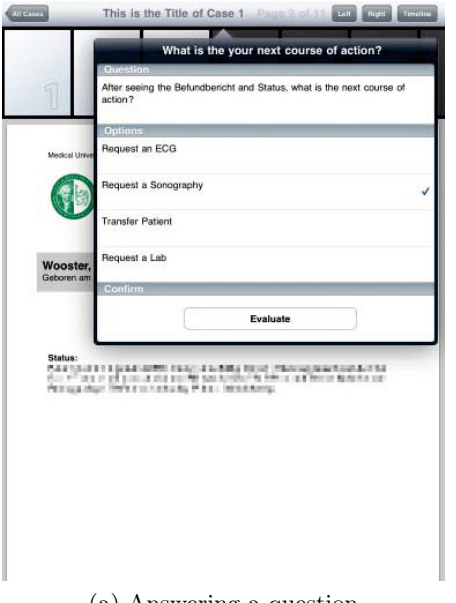
2.3 Lessons Learned during Development

Once two working prototypes were developed and preliminarily tested, it was ultimately decided to continue further development of the Objective-C application only. The main reason for this was performance; while it was possible to entirely duplicate the functionality of the Objective-C application, the performance of the HTML application deteriorated rapidly when displaying cases consisting of more than about 5 or 6 patient records. Part of the requirements for the project was that the cases themselves consist of individual A4-sized PDF files in portrait orientation. This was to ensure that no unrecognized filetypes would be encountered and that all patient records were the same size and orientation on the screen. Text within PDFs is vector based, which allows for close up zooming. Patient records that consist solely of scans, such as sonographic images, are wrapped in PDF files, as seen in Figure 4b. Unfortunately, WebKit cannot natively display PDFs meaning that all PDFs had to be rasterized as PNG images for the web-based Casebook application. This meant that when zooming into text the PNG images would become blocky and unreadable and the zoom functionality was eventually deactivated.

This led to the next problem that was encountered: because all the patient records had to be rasterized as PNG images, the performance of the transition animation between documents would become unacceptably slow as more PNG files load into the browser's memory. To compensate for this, files in the patient cases are dynamically loaded only when needed. In other words, after a question is answered the next group of records is loaded into memory. This method worked well, but as cases progress increasingly larger amounts of memory is required. Because there is no way to deallocate memory once it has been loaded, long cases would result in the user experiencing progressively slower response times to their swipe gestures, to the point where the application would be become all but unusable.

Furthermore, it was observed that the development time for the native application was shorter than that of the HTML application. The main reason for this being the iOS API, which has been designed from the ground up to make it easy to create multi-touch and gesture-based applications. While Sencha Touch and jQuery are impressive in their own right, they are based on technology that was arguably never intended for such application development. The result is that workarounds must often be used to emulate trivial features of the iOS API. From the point of view of the Casebook project, all subsequent development time will concentrate on the native application, and further work on the HTML application will cease.

Fig. 3. Casebook Objective-C application, viewing a single patient record



(a) Answering a question

(b) Viewing a sonograph. Images are wrapped in a PDF template to ensure design consistency.

Fig. 4. Objective-C Casebook Application

3 Motivation and Theory

Upon reaching the point of graduation, students of medicine must embark on a new challenge when they start working as doctors during their internships. This is the point where students must now apply what they have learned during their studies to real world situations that require quick thinking in time critical situations. However, recent work has outlined that as many as 40% of students do not feel they are prepared for their first medical positions, and this perceived lack of preparedness is attributed to a number of factors such as students' level of knowledge of communications skills, paperwork, and ward work [12][14]. Crucially however, according to Illing, et al., student exposure to clinical processes and practice is directly correlated to their preparedness or perceived preparedness. Illing, et al. relate their work to the ideas outlined in Lave and Wenger's book Situated Learning, where preparedness increases with authentic activity [16]. The theory of authentic activity ties in closely to that of context, which stipulates that information learned in the context of where it will be used increases information retention, facilitates learning, and improves the transfer of knowledge [1]. Currently Virtual Patient applications and Case-Based Learning tools attempt to increase this authentic activity by simulating the environment and situation in which medical students will eventually work. Many Virtual Patients are case based—they consist of cases that can be interacted with and the student learns clinical reasoning and decision making by examining and working with these cases. Nonetheless, studies indicate that rule-based diagnoses are frequently made due to skills that students gain from both Case-Based Learning and Problem Based Learning, thus confirming their legitimacy as a teaching method[7][4][22].

However, Case-Based Learning, in its current form at least, suffers from a number of detrimental issues that have recently been identified. Research has shown that Case-Based Learning can actually encourage adverse student performance, such as the phenomenon of premature closure—a situation where a diagnosis is made before all alternative diagnostic paths have been explored [2][3]. As well as this, developing cases for Virtual Patients can be an expensive and time consuming process. In 2007, Huang et al. reviewed over 100 virtual patient applications and found that in 34% of the cases, development costs were over $50,000 and a staggering 85% of the cases cost over $10,000 to develop (this was due, mostly, to the very nature of virtual patients that are built from scratch—they are generally rich in media and incur extensive production costs)[13].

It is the opinion of the authors that a new approach could address many of these issues at once. This novel approach uses real medical data as the basis for the cases within the Virtual Patient. This has several advantages over Virtual Patient cases that are based on fictitious data and must be manufactured. First, we believe that the trend towards more and more interactive and media rich cases actually decreases the authentic activity perceived by the students. On the other hand using real medical records will increase this perception of authentic activity and more closely mimic the context of the ward. This idea was touched upon by Dammers et al., who used real patients to teach problem solving [6].

Second, cases based on real patient records can be collected from modern hospital information systems with ease—this mitigates premature closure by ensuring there is a large pool of cases on which students can base their learning and diagnostic reasoning [18]. Hospital information systems contain millions of patient records and potentially thousands of suitable cases. Third, the cost and time effort involved in producing a case is reduced dramatically as cases do not need to be produced or manufactured from scratch. As mentioned previously, cases can be extremely costly to produce, limiting their suitability for small institutions that do not have the time or monetary resources required for the development of case-based Virtual Patients. Such costs also inhibit the feasibility of introducing case-based Virtual Patients in to institutions based in less developed countries.

3.1 Teaching Using Medical Records

Using real patient data makes possible a unique way of teaching the procedure of documentation to students of medicine. Analyzing cases that were written by medical professionals provides important insight into how this crucial task is undertaken. Not only can students learn good documentation practice, but they can also experience examples of badly written documentation or poorly documented cases. This has multiple benefits: first, students learn how to document well by example and second, students learn first hand the impact of good documentation on the understandability of a patient's history. The more students realize the importance of good documentation, the better they understand the impact of good documentation on the quality of care. Therefore, Casebook is an attempt to promote the need for good documentation standards and methods, teach documentation skills, prep students for real life documentation work, and emphasize the importance of thorough and well written patient history taking. This will also allow students to learn the importance of documentation in the wider clinical context—this is becoming more important as patient records are being used in an ever increasingly varied number of situations. This has been observed by Ganslandt et al. where they found uses of electronic patient records in areas such as clinical research, clinical management, and quality management [10]. Also, physicians already spend a disproportionate amount of time writing medical reports, and making the most use of these records after they have been documented is surely logical. According to preliminary work by Oxentenko, et al., 67.9% of physicians spend up to 4 hours per day writing documentation, while only 38.9% spend this amount on time in direct patient contact [19]. Therefore, because of the increasing variety of use of medical records, and the effort that must be invested in to writing patient reports in the first place, it is essential that students realize that good documentation is beneficial for many areas of medicine.

As mentioned previously, premature closure is one source of diagnostic error that is known to cause adverse conditions in patient outcomes. Other sources of diagnostic error include aggregate bias, anchoring, ascertainment bias, base-rate neglect, confirmation bias, diagnostic momentum, overconfidence bias, representative error, and search satisfying [5]. By using real patient data, cases where such errors are known to have occurred can be shown to the students and the

point at which the diagnostic error manifested can be discussed. Students can learn a great deal about the pitfalls of diagnostic error by examining previous examples of when they happened in real world situations.

This leads on to a final point regarding the annotation of patient records by physicians. Work by Eva, K.W. (2005), stated that properly conveying knowledge and reasoning strategies to novice diagnosticians is fundamental to a student's understanding of the procedure of a case [9]. Eva's work recognized that this task is difficult as the clinical teacher must understand the strategies that expert clinicians use to make their diagnostic decisions. Interestingly, a model formulation described by Johnson, et al. outlines a system where a physician's thought processes can be documented along with the standard documentation and history taking [15]. As future work, we propose a similar system, whereby diagnostic reasoning notes could be added to patient records at the time of documentation in the form of meta-data annotations specifically with the aim of using these records for teaching purposes. That way, physicians and medical professionals could document their reasons for making certain decisions at the point at which they are made. This information would be saved as supplemental meta-information not normally visible on the patient record itself, but would be used by Casebook when the patient records are viewed by students who are observing the case. Students viewing the cases themselves can earn credits for comments, questions, and discussion regarding the case. The case creator's task would be to act as moderator for the discussion and offer help, pointers, and feedback regarding any questions or issues that may arise.

4 Future Work

Due to time constraints, Casebook has so far only been tested on a small amount of users. The winter semester of 2011 will provide the opportunity to test the application on a large amount of students who will use Casebook as part of their seminar on Decision Support. Their feedback will be used to decide on further development directions and serve as a constant testing environment for the application. Future work will also entail annotating a number of cases with meta-information to provide the students insight into decision making criteria. In other words, cases will be supplemented with information outlining the reasoning and criteria physicians had for making certain diagnostic decisions. The question being, when cases are supplemented with information on the clinical reasoning of physicians, do students gain more useful insight into the mind's of doctors and how they think and make diagnostic decisions? Do they then learn more about reasoning strategies? These are extremely pertinent questions in the field of medical education, documentation, and health information systems. Once a collection of annotated cases has been assembled, we will also be able to ascertain the granularity of annotation required.

User feedback aside, Casebook will nonetheless be further developed to support non-linear cases. The static and linear approach currently employed has its limitations when several branches occur within one case at the same time.

Currently, it is possible to order patient records logically in a linear fashion, but this is not optimal from our point of view.

In the long term, a case repository will be built to collect cases developed by teachers. Cases themselves will have to be tagged with keywords from the National Library of Medicine's controlled vocabularies to ensure that they can more easily be found and that similar cases can be logically grouped together[21].

5 Conclusion

This paper outlined the development of a Virtual Patient application that uses real patient data to teach students. The application itself was developed using two different technologies in parallel; one as a standard iPad application running on iOS, the other as a web-based HTML application designed to be accessed using tablet browsers. Tablets were chosen as the platform on which to develop Casebook for a number of reasons. The multi-touch metaphor utilized by tablets emphasizes the perception of navigating through a timeline, revealing more of the patient's case as the records are traversed from left to right. In the environment of a classroom or seminar, tablets make ideal devices to view patient cases. They can be readily passed around a classroom, and encourage team work and discussion within groups of students in a seminar session. Not only this, but tablets are increasingly being used as replacements for workstations in hospital wards, thus further increasing the authentic activity and learning context. Ultimately, performance issues led to the abandonment of the web-based application, due mainly to the fact that PDFs must be rasterized before being displayed within a browser window.

The motivation for using real patient data as the basis for a Virtual Patient was also described in this paper, and several advantages for both the learner and the teaching institution were outlined. First, hospital information systems contain huge pools of medical cases allowing for collections of case-based Virtual Patients to be compiled. This can help to avoid premature closure as mentioned throughout the paper, but is also very cost effective. Second, due to the real world nature of the cases, the perception of authentic activity can be increased better preparing students for their internships upon graduation. Third, good documentation practice can be learned and its importance appreciated, thus increasing the quality of the documentation that the students will eventually write as junior doctors. With so many students expressing concerns about their preparedness for clinical work, it was felt that their exposure to clinical practice could be increased by allowing them to analyze actual patient cases and learn from the real world work of clinicians.

References

1. Bergman, E.M., Van Der Vleuten, C.P.M., Scherpbier, A.J.J.A.: Why don't they know enough about anatomy? A narrative review. Medical Teacher 33(5), 403–409 (2011)
2. Bowen, J.L.: Educational strategies to promote clinical diagnostic reasoning. New England Journal of Medicine 355(21), 2217–2225 (2006)

3. Colliver, J.: Effectiveness of problem-based learning curricula: research and theory. Academic Medicine 75(3), 259–266 (2000)
4. Cook, D., Erwin, P., Triola, M.: Computerized virtual patients in health professions education: A systematic review and meta-analysis. Academic Medicine 85(10), 1589–1602 (2010)
5. Croskerry, P.: The importance of cognitive errors in diagnosis and strategies to minimize them. Academic Medicine 78(8), 775–780 (2003)
6. Dammers, J., Spencer, J., Thomas, M.: Using real patients in problem-based learning: students' comments on the value of using real, as opposed to paper cases, in a problem-based learning module in general practice. Medical Education 35(1), 27–34 (2001)
7. Dolmans, D., Schmidt, H.: The advantages of problem-based curricula. Postgraduate Medical Journal 72(851), 535–538 (1996)
8. eViP Electronic Virtual Patients: About Virtual Patients, `http://www.virtualpatients.eu/about/about-virtual-patients/` (accessed July 2011)
9. Eva, K.W.: What every teacher needs to know about clinical reasoning. Medical Education 39(1), 98–106 (2005)
10. Ganslandt, T., Krieglstein, C., Mueller, M., Senninger, N., Prokosch, H.: Electronic documentation in medicine; flexible concepts versus isolated solutions. Zentralblatt fuer Gynaekologie 122(8), 445–451 (2000)
11. Garcia, J., De Moss, A.: Sencha Touch in Action, 1st edn. Manning Publications (2011)
12. Goldacre, M., Lambert, I., Evans, J., Turner, G.: PRHOs' views on whether their experience at medical school prepared them well for heir jobs: national questionnaire survey. BMJ 326, 1011–1101 (2003)
13. Huang, G., Reynolds, R., Candler, C.: Virtual patient simulation at US and Canadian medical schools. Academic Medicine 82(5), 446 (2007)
14. Illing, J., Morrow, G., Kergon, C., Burford, B., Davies, C., Baldauf, B., Morrison, G., Allen, M., Spencer, J., Peile, E., Johnson, N.: Do medical graduates need more on-the-job experience? A prospective qualitative study comparing three diverse UK medical schools. Medical Education 43, 39 (2009)
15. Johnson, S., Bakken, S., Dine, D., Hyun, S., Mendonça, E., Morrison, F., Bright, T., Van Vleck, T., Wrenn, J., Stetson, P.: An electronic health record based on structured narrative. Journal of the American Medical Informatics Association 15(1), 54–64 (2008)
16. Lave, J., Wenger, E.: Situated Learning. Cambridge University Press (1991)
17. jQuery Mobile Framework: jQuery Mobile, `http://jquerymobile.com/` (accessed July 2011)
18. Norman, G.: Research in clinical reasoning: past history and current trends. Medical Education 39(4), 418–427 (2005)
19. Oxentenko, A.S., West, C.P., Popkave, C., Weinberger, S.E., Kolars, J.C.: Time spent on clinical documentation, a survey of internal medicine residents and program directors. Archives of Internal Medicine 170(4), 377–380 (2010)
20. Stark, J.: Building iPhone Apps with HTML, CSS, and JavaScript, Making App Store Apps Without Objective-C or Cocoa, 1st edn. O'Reilly (2010)
21. Stead, W., Searle, J., Fessler, H., Smith, J., Shortliffe, E.: Biomedical informatics: changing what physicians need to know and how they learn. Academic Medicine 86(4), 429–434 (2011)
22. Williams, B.: Case based learning—a review of the literature: is there scope for this educational paradigm in prehospital education? Emergency Medicine Journal 22(8), 577–581 (2005)