Development of a Multi-Sensory System to Better Relay Pharmacotherapy Information

Abstract

Poor health literacy and language barriers constitute a major impediment in pharmacotherapy (the relay of medication instructions). Medical professionals, especially those involved in humanitarian-aid response, struggle to help patients who cannot communicate with them or read their medications' labeling. An alternative method to address these communication barriers in pharmacotherapy was needed.

I developed a Java software program, The Prescription Architect (TPA), to better facilitate pharmacotherapy through use of data provided and validated by an international pharmaceutical organization. Additionally, new data (translations, pictograms, and orations) can be appended to the program's databases by the user. TPA conveys medication instructions via culturally-sensitive pictograms on handouts that I designed. It can be used by a prescriber of any language and can translate, or orate, the instructions of patient prescriptions in any language.

TPA is available as a free online download and has been used by over 1000 medical professionals in over fifty countries worldwide. Other than its initial download, TPA requires no internet access to use which has allowed prescribers to bring it to remote areas and places where there have been natural disasters. I will continue development of TPA to help break down communication barriers in pharmacotherapy and help prescribers save lives.

1. Introduction

Prescribing professionals need to relay important prescription information to their patients. Unfortunately, they often rely on their own verbal and written explanations to do so (Mansoor & Dowse, 2003). In places where there are language and literacy barriers present between prescribing professionals and their patients, communicating important health information is difficult (Mansoor & Dowse, 2003). This scenario is common to professionals who work in disaster relief or remote areas where they are often asked to prescribe and relay medication instructions to foreign patient populaces. The language and literacy barriers that arise in these situations diminish the adherence of patients to proper medical procedure thereby causing potentially lethal mistakes with their medications such as improper dosage. Studies have shown that poor adherence due to these factors has contributed to over 25% higher mortality rates in patients who failed to understand the medical instructions of their prescriptions (Dray-Spira, et al. 2010).

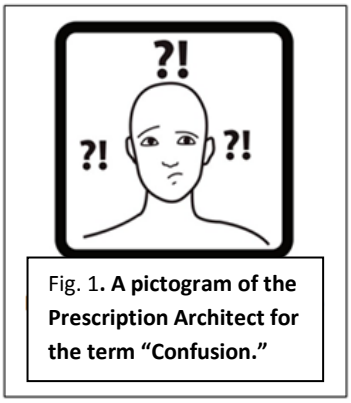


Fig. 1. **A pictogram of the Prescription Architect for the term "Confusion."**

Pictograms, which are visual representations of prescription information, have proven to be beneficial in increasing medication adherence and decreasing dosing errors (Mansoor & Dowse, 2003, Dowse & Ehlers, 2004, Yin et al., 2008). The pictogram's use as a visual medium allows instructions to transcend language barriers. Additionally, pictogram-supplemented prescriptions are perceived by the public as better and safer medications because they are easier to comprehend and easier to remember than traditional forms of relaying prescription information (Norman, 1990). This offers extra incentive for prescribing professionals to integrate pictogram use into their current medication information relay procedure (Mansoor & Dowse, 2003, Dowse & Ehlers, 2004, Yin et al., 2008).

In addition to being placed on medicinal labeling, pictograms are often placed on handouts or leaflets to be provided to the patient. Currently the integration of pictograms into prescriptions is difficult and is hampered by the reliance on inadequate software or stickers/stamping (Committee on Identifying and Preventing Medication Errors, 2007). A comprehensive, adaptable, and ergonomic (easy to use) computer program that implements sets of culturally sensitive pictograms accompanied with clear instructions in different languages was needed.

I developed The Prescription Architect (TPA), a Java software program for computers, to replace current methods of medication information relay in addition to providing extra options that prescribing professionals do not currently have. TPA allows prescribing professionals to establish their own password-protected accounts and manage their patients on an individual file basis. Prescribing professionals can easily append new medications, through a simple checkbox system, to any of their patient files and then immediately generate program outputs that I designed. These output files are created as PDF handouts and may be shared with patients as supplementary aids in addition to the verbal advice shared by prescribing professionals. In cases where the prescribing professionals do not speak the language of their patients or their patients are illiterate and/or visually impaired, TPA can generate audio instruction that directs patients how to take their medications.

TPA runs locally (using data from the system it is installed on) and without internet connectivity, which is often sparse in areas where TPA is needed most (such as in remote areas or places or natural disaster). The program stores a database of pictograms, translated terms, and audio files from which data is dynamically referenced. Because all of TPA's display terms are dynamically referenced, any change made by users to its database will instantaneously update the system. This makes the process of appending new terms, pictograms, and orations completely

automated. The process also ensures that professionals with limited computer experience can utilize the program effectively. To further simplify usability, the program also features a graphical user interface that uses buttons and text boxes.

2. Design

A) Program Requirements

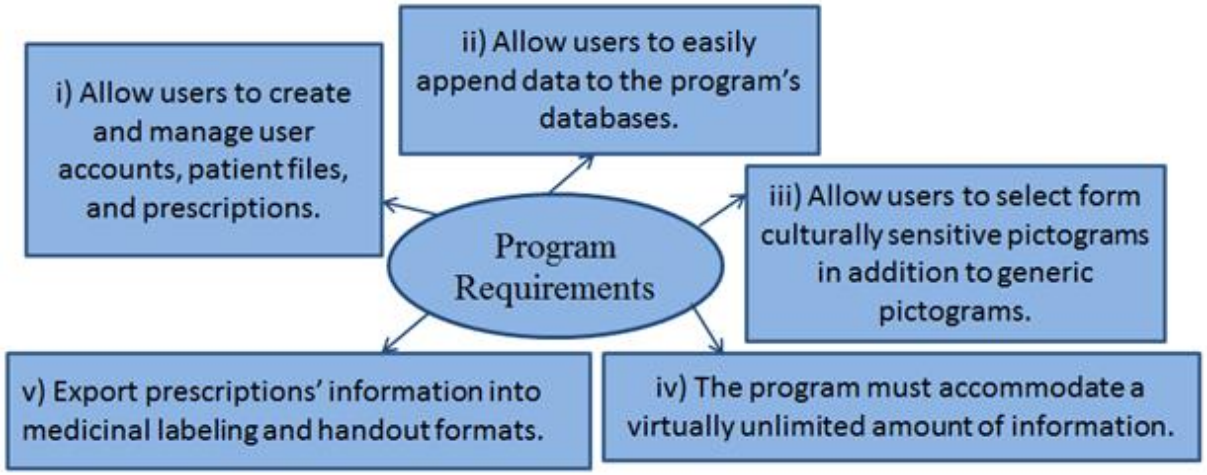


Fig. 2. **Diagramed program requirements.**

B) Design of Data Flow (Program Requirement ii)

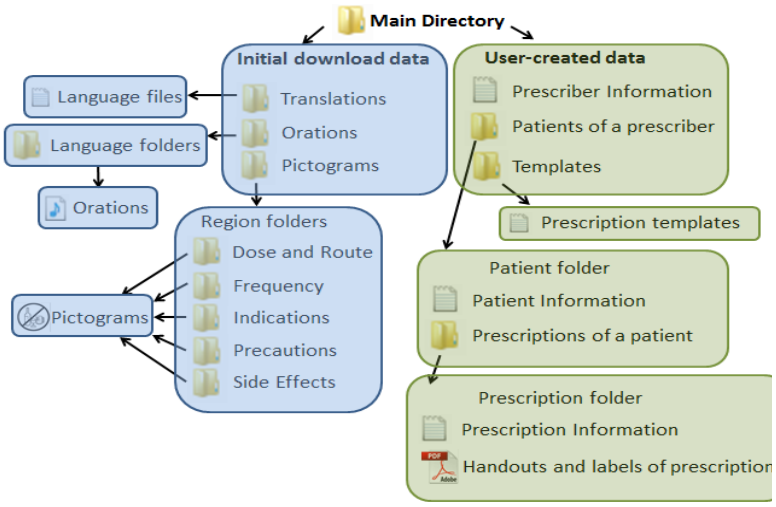


Fig. 3. **Data was primarily divided into two sections to separate user-created data from downloaded data.**

TPA was designed to not only reference data included with the initial download (such as pictograms and translations) but user-created data as well. Users could create accounts, patient files, and prescriptions through entering information to

be saved on their computers. The information referenced by the program was stored in files organized in a dynamic folder directory that grew as the user entered new data. A diagram of this directory is displayed in Fig.3. Data were primarily saved and read from text files; labels and handouts of the entered information were created as PDF's.

C) Appending User-Created Data (Program Requirement i)

Terms were saved in lists of each language's translations ("Language files" of Fig. 3). To share new terms with other prescribers around the world, users of the TPA can copy applicable folders and email them to their colleagues. Alternatively, they can send them to be evaluated for inclusion into the "initial download data" folder for all future users.

Fig. 3 depicts the graphical user interface of a section of the program that can be used to add new terms to the system database.

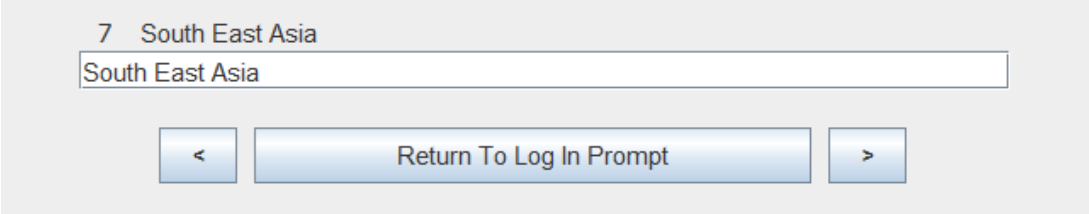


Fig. 4. **This image taken of the Prescription Architect displays the box where a translator would be prompted to translate term 7 into another language.**

The number "7" in the display of Fig. 4 (above the white box beside the header "South East Asia") references the displayed term's term number. Term numbers were assigned to every term used in the language database for reference. When a term needed to be translated, *Algorithm 1* from the Translation class' translateTerm method was used to obtain a translation.

Algorithm 1.

- Take the String (term) that needs to be translated and open the file for its language's translations.
- Locate the String in its language file and store its term number.
- Open the file for the language that the term is to be translated into.
- Locate the term's translation by using the obtained term number and return the translation.

Users may append pictograms to the program's database as well. To add a new pictogram, a user could copy the image and then paste it into the corresponding regional folder within the "Region Folders" subdirectory of Fig. 3.

D) Culturally-Sensitive Pictograms (Program Requirement iii)

All pictograms used in the program have been validated by an international pharmaceutical organization to ensure that they can be easily interpreted. However, it has been determined by various researchers that culturally-sensitive pictograms (CSP's), also known as regionally-sensitive pictograms, relay prescription information better than generic pictograms (Grenier, et al., 2011). CSP's portray instructions as they are perceived by a given region. For example, Fig. 5 depicts two different pictograms used by the program for the same term. CSP's for eleven regions along with a set of generic pictograms are included within TPA download.

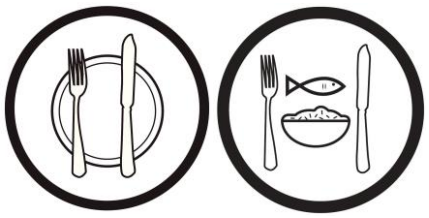


Fig. 5. **From Left to right: Eastern Mediterranean and Canadian First Nation pictograms of the term "take with food".**

E) Flexibility (Program Requirement iv)

TPA was created to be easily modifiable through the ubiquitous use of dynamic referencing. Dynamic referencing is the utilization of unknown variables rather than defined (static) values. It allowed for an unknown number of pictograms, and translations to be created by the user. In the initial display screen, where users must select the language they would like to access the program in, *Algorithm 2* from the HelperMethods' class' returnLanguages method determines what language options to display through dynamic referencing and a directory search.

Algorithm 2.

- Open the "Translations" folder of Fig. 2 and list all contained language files in an array.
- Remove the ".txt" suffix of each file name in the array and return the array.

F) Program Output Files (Program Requirement v)

The labeling and handout templates were all designed as PDF's so that they could not be easily modified if given electronically to a patient. PDF creation is not a well-supported function in Java. The library itext (iText Software Corporation) was used to facilitate the creation of the PDF files. The class PDFCreator was created to serve the purpose of utilizing itext to create the output files.

I designed the handouts, or output files, with the aid of medical professionals, to accurately portray the information of a prescription in a coherent yet concise manner. Appendix I includes an example of each handout in various languages that the program can create. Namely, the program outputs Story Boards (SB), Medication Labels (ML) and Medication Calendars (MC). The SB and ML display information concerning a single prescription. The MC provides an important visual overview that coherently summarizes all of a patient's prescriptions. The handouts were designed into columns and rows to organize the information for the patient. I used simple designs that removed peripheral information and focused on the critical aspects of the medication.

G) Programming Languages Used

I created TPA using Java because it functions efficiently on most computers including Mac and PC systems. Java is an object-oriented language which facilitated the program's design by considering each user account, patient, and prescription as its own object with its own set of data. Additionally, Java allowed for the program to be distributed as an executable jar file. This meant that the program would never need to be "installed"; it could be run solely off of a flash drive or CD. Users only need to download the executable .jar file and its directories in order to access the

program. However, my design also provided the capability to put TPA on a sky-drive. This gave prescribers a means to interact with each other's databases simultaneously.

H) Development Environments

I used the integrated development environment Eclipse for the TPA's coding because it provided an excellent user interface. Eclipse also has an available add-on named Windows Builder. Windows Builder provided an effective way to create the programs' graphical user interface through use of its WYSIWYG (What You See Is What You Get) displays.

I) Class Designs

Class/Responsibilities/Collaborators cards or CRC's summarize class' roles. CRC's were developed for all seventeen classes of TPA. Below is a sample CRC from the Patient class. A method is a function performed by the class.

Name: Patient
Methods: openPatientFile – This method calls readInPatient if the file exists.
whatsMyInfo – This method returns all information tied to a given Patient object.
writePatient – This method writes all of a Patient's information into a patient file.
readInPatient - This method reads data of a specified patient file location into a specified Patient object.
deleteMe – This method deletes a patient file and all of its prescriptions.
displayPrescriptions – This method lists all of a Patient's prescriptions.
findMe – This method determines if a patient file exists for a given Prescriber.
Collaborators: Prescriber

Fig. 6. **The Patient class's conveys the object-oriented design of TPA.**

3. Program Use

A major facet of this project was designing a system that included simple, ergonomic display screens. My solution to guide the user through the prescription generation process was the

following progression of screens which follow a user's interaction through the program. Please note all red screen numbers are superimposed.

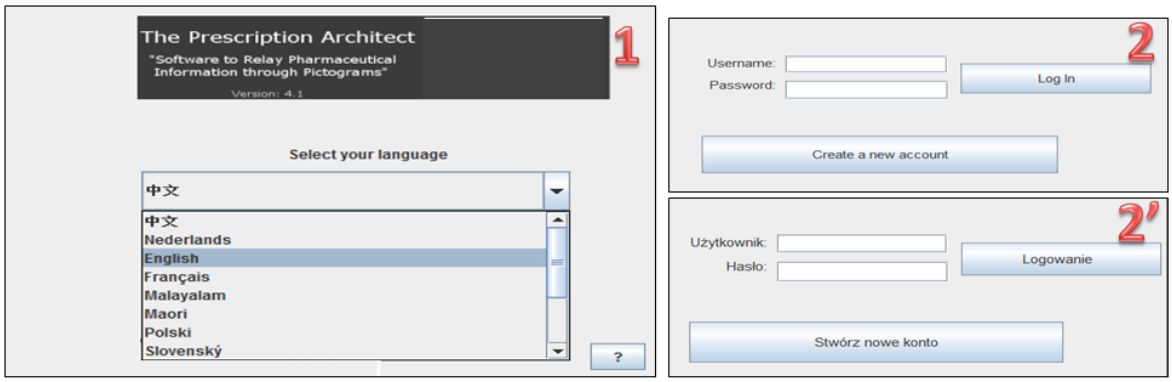


Fig. 7. **Screen 1: main display/ language selection & Screen 2: log-in prompt.**

Screen 1 provides the user with a dropdown interface through which the user chooses which language to access the program in (see *Algorithm 2*). For example, a Polish-reading prescriber would select Polski. Prescribers may create handouts of any language regardless of which language they access the program with. The "?" button provides contact information. Through uncluttered screens and simple buttons I aimed to emulate a minimalistic design that users would find visually appealing as well as easy to use. Screen 2' provides an example of what Screen 2 would look like for a prescriber who selected Polski in the Screen 1. Previous users can login to Screen 4. New users can create, and later edit, accounts through Screen 3.

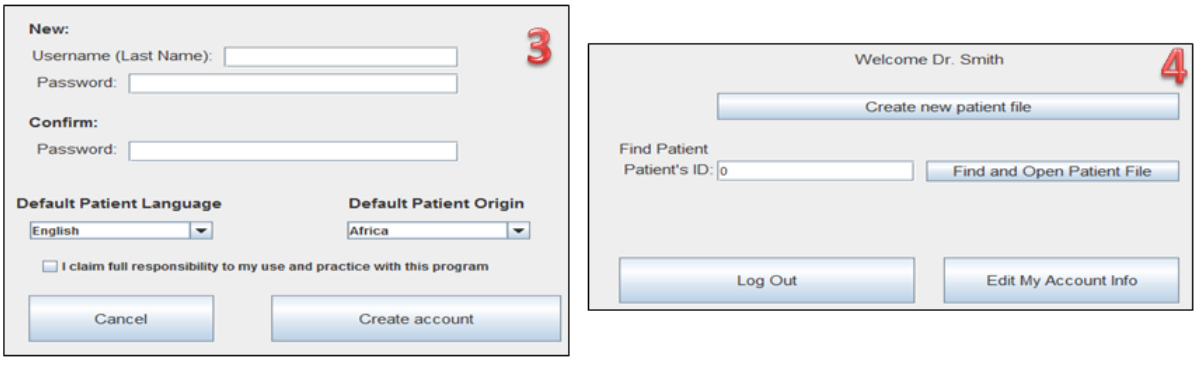


Fig. 8. **Screen 3: user (prescriber) account edit/creation & Screen 4: account options.**

Screen 4 allows users to access or "find and open [a] patient file" which will bring them to Screen 6. Patient files are saved with reference to their ID numbers which allows their data to be de-identified. Screen 5 provides an interface to create, and later edit, patient files.

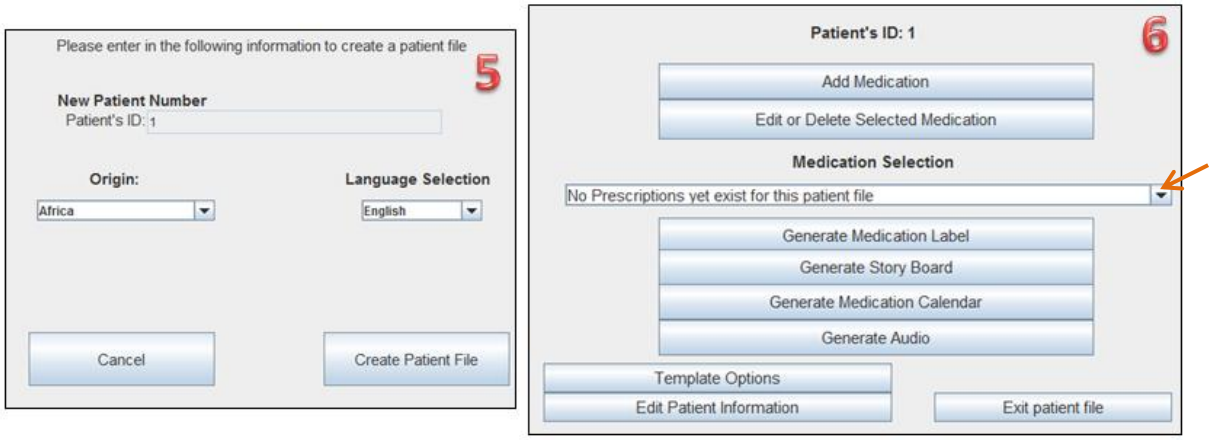


Fig. 9. **Screen 5: patient file edit/creation & Screen 6: patient file options.**

I designed Screen 5 to resemble Screen 3 so that users would find it familiar. Screen 6 is TPA's main interface and provides the user with a lot of options. "Add medication" and "Edit or delete selected medication" will both bring the user to Screen 8. The former loads Screen 8 with a new slate (no data present) and the latter will open Screen 8 with the information of whatever medication was selected in the dropdown menu that I have denoted with an orange arrow. Selecting one of the next three buttons that display "Generate…" will create the prescription selected in the box into one of three handout designs and immediately open the PDF for the user to access. Generate audio will launch an audio player that will orate the medication's information in the patient's language. "Template options" will open Screen 7.

While Screen 6's crowded display may seem divergent from TPA's other simple screen designs, the accessibility of system options that it provides far outweighs any drawbacks. Ideally, this is the display that prescribers could have in front of their patients while they explain instructions. They could click open a PDF handout and then guide the patients eyes along the

information as it is recited via TPA's "Generate Audio" feature. In other words, Screen 6 provides users with a means to relay information to a patient via three modalities simultaneously.

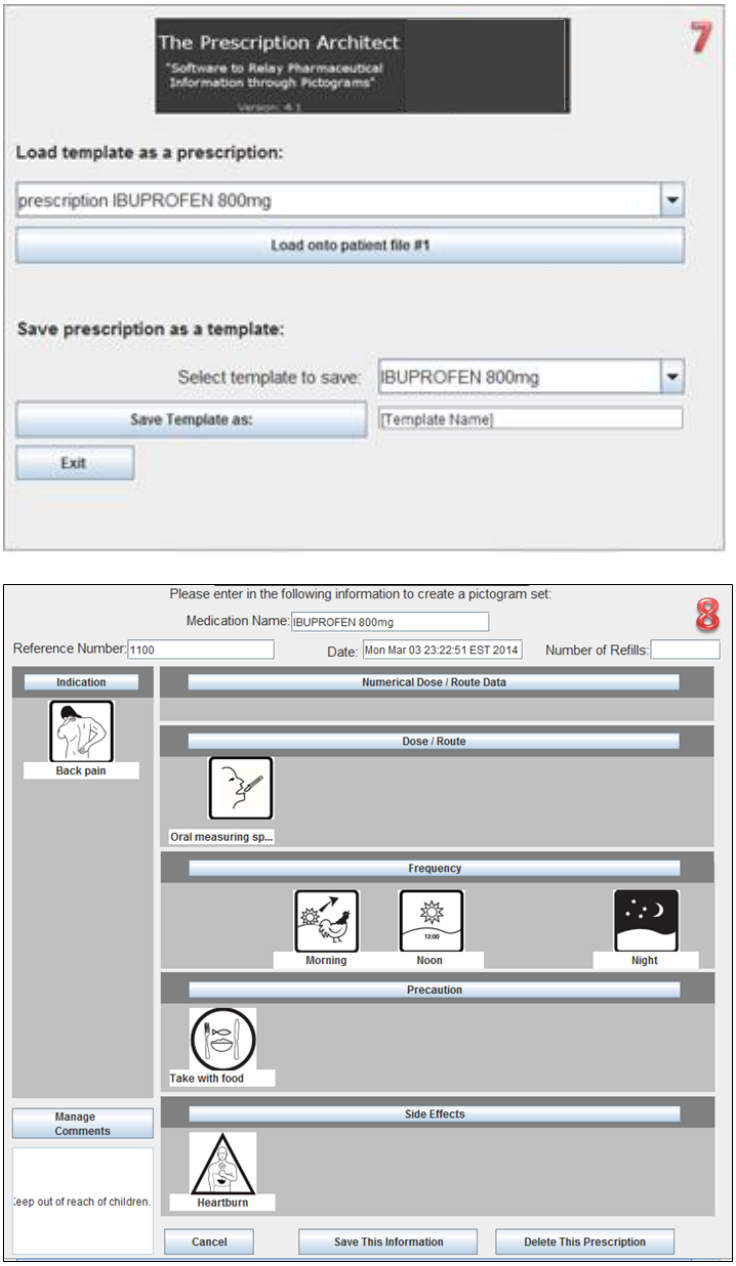


Fig. 10. **Screen 7: templates & Screen 8: prescription creation.**

Screen 7 allows users to select from any templates stored locally on their computer and load them onto their patient files. Conversely, users can take prescriptions that they have made on their patient files and save them as templates to provide to other prescribers or use by multiple patients.

Screen 8 is the interface through which prescribers create, or edit, prescriptions. I sought to make the process easy through compartmentalizing prescriptions into pieces. Sections can be independently edited by clicking on their respective buttons such as "Indication" or "Side Effects". In designing this screen, I realized that with the exception of "numerical dose / route data" and "manage comments" (which bring the user to Screen 10 or Screen 11 respectively) information could be simplified to pictogram selection.

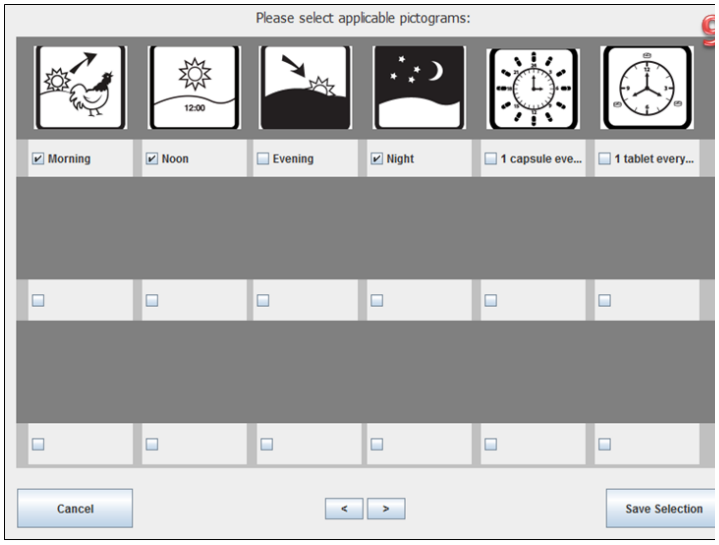


Fig. 11. **Screen 9: pictogram selection.**

Because each section included pictograms, I determined that I could route *all* of those section buttons to bring the user to Screen 9. The only difference was a dynamic reference that changed what kind of pictograms to display based on what section had been selected. In this display of Screen 9 frequency options are displayed so this user must have pressed "Frequency" on Screen 8. Selected pictograms will show up when the user returns to Screen 8 to complete other facets of the prescription.

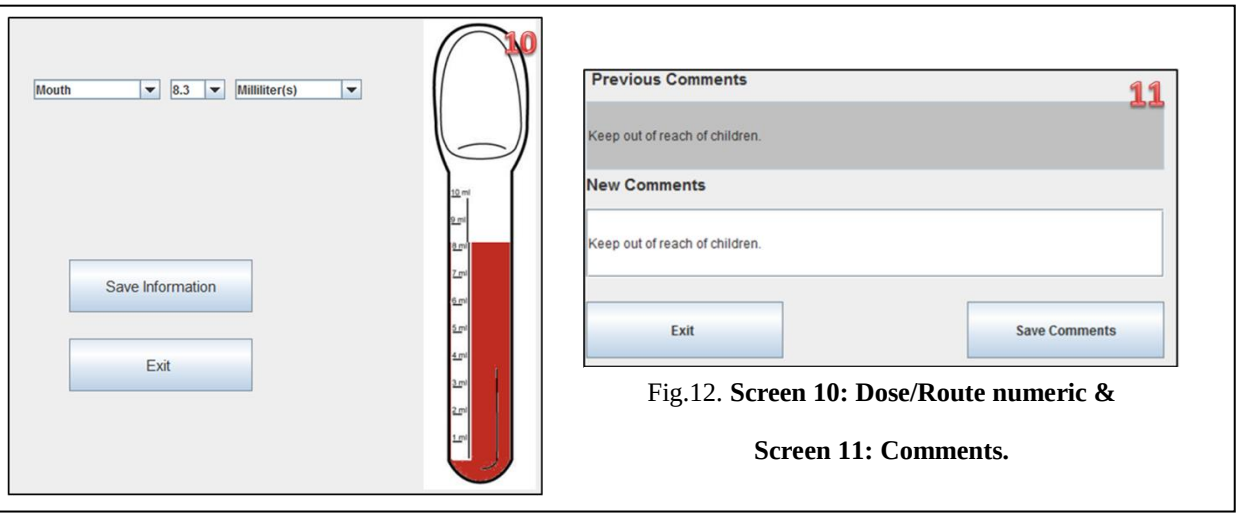


Fig.12. **Screen 10: Dose/Route numeric & Screen 11: Comments.**

Screen 10 allows for numerical Dose / Route data to be entered and displays the medication in the selected medium to help teach patients how to measure their medication by graduated means if applicable. Screen 11 allows for comments to be appended to the prescription. I included these screens as supplementary means to represent information. Perhaps a prescriber doesn't have a pictogram but wants to write out an instruction, or a prescriber wants

to include his or her name with a contact phone number in the information. By providing this completely optional comment section, TPA accommodates people of many scenarios. Through dividing the program between many screens, I was able to specialize most of TPA's displays. The use of white-space helped in simplifying their designs as well; and resulted in providing users with a straightforward, ergonomic system.

With a user account and patient file already created, a prescriber can generate medication handouts for a patient in five steps (depicted by the purple lines in Fig. 13) which typically take less than twenty seconds. If prescribers need to create or edit a prescription before doing so, Fig. 13 outlines their options.

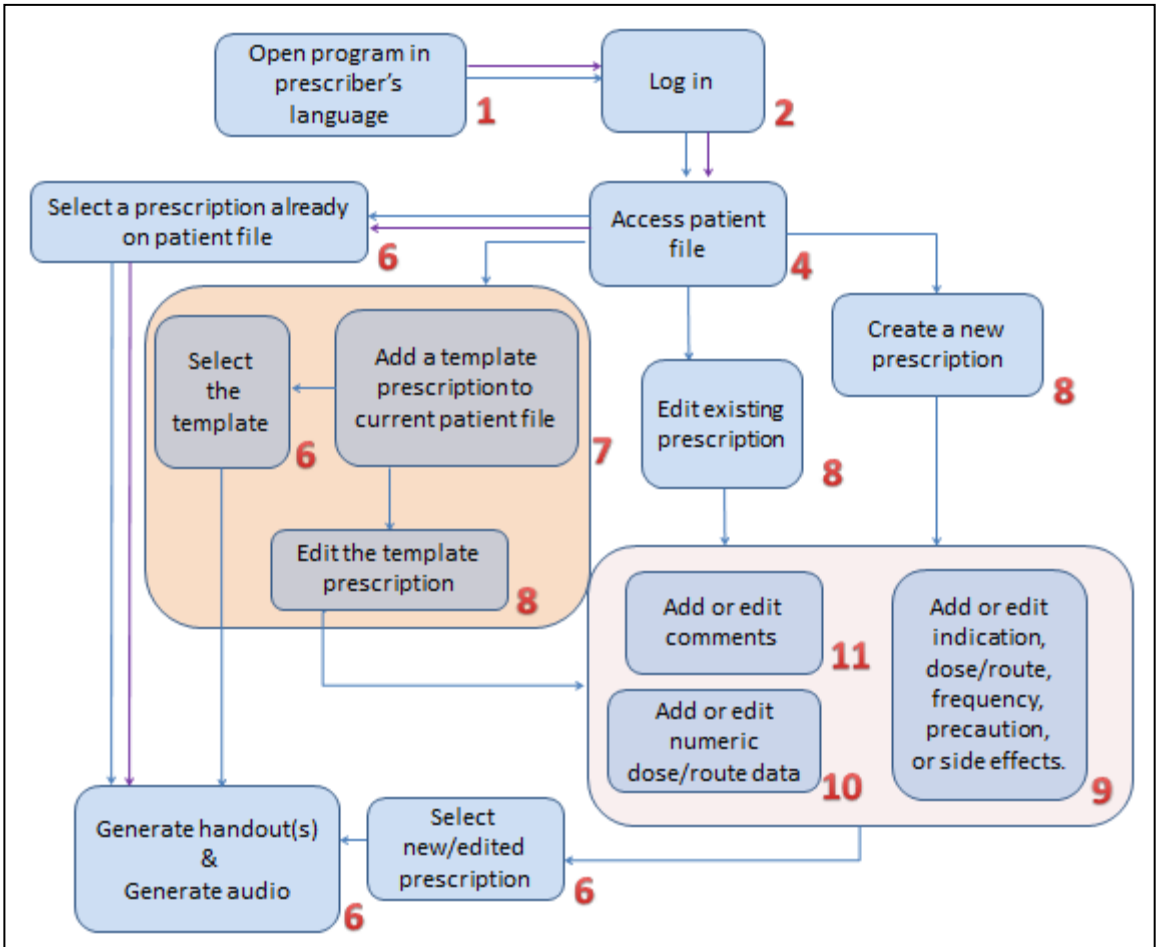


Fig. 13. **Outline of a user's steps to generate handouts or audio once a user account and patient file have been created. The orange shading represents interactions with templates and the red shading represents changing/adding information of a prescription.**

Implementation

Before designs were made for a graphic user interface (GUI), code was developed for a text-based software program to effectively manage input and output data. This early stage allowed for a stable connection between the software and its local directories to be developed. The next stage required development of a way to view and select pictograms. It was at this point that the creation of code for a GUI began. Utilizing Windows Builder for Eclipse, the aforementioned screens were developed to guide the user through the account creation process and eventually the pictogram selection process.

This process of incremental design allowed for my program to reach the people who needed it most as soon as it could. All later versions of the program were backwards compatible so that users could install via big bang adoption, simply transferring all old files to the new system (rather than adopting the new version in parallel).

To allow for dynamic referencing (as discussed in Section E: Flexibility (Program Requirement iv)), several methods/algorithms were created to handle unknown values. These include directory lists such as *Algorithm 2* as well as a recursive delete directory method. In other cases, such as when displaying the pictograms of Screen 9 or the translations of Screens 2-11, more complicated algorithms needed to be created.

An example of dynamic referencing in TPA is present in Screen 9. Its ability to convey pictograms of different section types has already been explained, but how does it determine what pictograms to display? Because an unknown amount of translations or pictograms could be appended to the program (limited only by the accessing computer's memory space), a single screen could not display them all at once. Instead, Screen 9 only displays some of the pictograms based off of the current display's slide number (where the initial slide number is one and

pressing the  arrow brings the user to slide 2, then 3, and so on). Eighteen slots were created to display eighteen pictograms at once and algorithm of what image to display in each slot was:

(slot number + ((slide number - 1)*18))

Therefore, when a user scrolled to the second slide to view more pictograms, slide 2's slot number 1 would display the 19th pictogram (1 + ((2-1)* 18) = 19). This principle of dynamic referencing is present in many aspects of TPA and is what allows it to be so versatile.

4. Testing

Testing was conducted on several different computers to ensure that TPA was compatible with various accessing systems. It was determined, through use on a laptop, that the display of Screen 8 was too large for smaller screen sizes. A scrolling pane was developed to accommodate smaller displays.

Another issue uncovered by testing was that characters not included in Java's default fonts, such as Arabic and Chinese characters, were not compatible with the system. To solve this problem, I integrated UTF (Unicode Transformation Format) encoding to allow for the software program to utilize those characters. Algorithm 3, from the Translation Class' setTerm method, utilizes UTF to read in a language file that may contain characters of a non-default font.

Algorithm 3.

```
File languageTermsInput = new File("Input" + "/" + "Translations" + "/" + language + ".txt");
BufferedReader fileReader = null;
try
  {
        FileInputStream inputStream = new FileInputStream(languageTermsInput);
        DataInputStream streamToAnalyze = new DataInputStream(inputStream);
        fileReader = new BufferedReader(new InputStreamReader(streamToAnalyze,"UTF-8"));
```

In laymen's terms, this algorithm first acquires a file to inspect. The file's location is denoted within the pink box. Note that the language is not a complete String, but instead yet another dynamic reference to the variable "language" (this allows *Algorithm 3* to be used on any

language). Within the green box, a FileInputStream and then a DataInputStream are created using the file so that its information can be read into the computer. The final line of code creates a buffered reader to interpret this data (which is delivered via the variable "streamToAnalyze") into a new format, "UTF-8". This process is used to read in all text files so that users are not limited by the characters they choose to use.

5. Conclusion

The objective of this project was to better facilitate the relay of medication information to patients. My program, The Prescription Architect (TPA), accomplished this objective by providing a means to relay medication information via various modalities. The program incorporated culturally-sensitive pictograms as digital images and provided several PDF handout designs upon which medication information could be saved, printed, or emailed to the patient. An audio component provides medication instructions in a patient's native language which could be used as an alternative modality to aid illiterate and visually impaired patients. Extensive dynamic referencing, as well as the translation system I engineered, allowed the program to accommodate patients and prescribers of any language. Many other additional features such as the template system (to share prescriptions between pharmacotherapy networks or between patients) and the Dose/Route numeric display (to help teach patients how to measure their medications y graduated means), supplement the program with resources for prescribers to use in helping their patients.

I introduced TPA at an international pharmaceutical conference. TPA was then published on the website of an international pharmaceutical organization. It has since been downloaded by over one thousand individuals around the world in over 50 countries. A usability survey was created by an international pharmaceutical organization and de-identified data is being collected

to evaluate ways in which the program can be improved. Suggestions that have been implemented as a result of previous testing include the template system, more initial download language options, and optimized medication handout designs.

Through multiple modalities, TPA provides its users with a means to overcome language barriers and poor health literacy which have previously impeded pharmacotherapy in remote areas as well as locations experiencing natural disasters.

6. Future Work

While TPA is being used around the world, I continue work on the program. New versions are released on a monthly basis and quick download response indicates a strong user community. The next version of the software is scheduled for release by mid-October of 2014 and will include a database with several new languages, including Arabic, in the initial download folder. With the help of several pharmaceutical organizations I continue to distribute the program to new users. It is my hope that, through providing my program to prescribers worldwide, TPA can continue being used to help medical professionals save lives.

References

[Special issue]. (2008). A picture worth a thousand words: The use of pictograms for medication labeling, In Practice, 21(1).

Agbaje, A. (2012, April 17). Best way to export data from Java to MS Excel [Online forum post]. Retrieved from http://stackoverflow.com/questions/10187218/best-way-to-export-data-from-java-to-ms-excel

Committee on Identifying and Preventing Medication Errors, Board on Health Care Services, Institute of Medicine. (2007). Preventing medication errors: Quality chasm series. Washington, DC: National Academy of Sciences, 196-200.

Chakraborty, A. (2007, May 24). How To Read / Write Excel Spreadsheet From Java. Retrieved February14, 2013, from http://tech.gaeatimes.com/index.php/archive/how-to-read-write-excel-spreadsheet-from-java/

Dowse, R., & Ehlers, M. (2004). Medicine labels incorporating pictograms: do they influence understanding and adherence? *Patient Education and Counseling, 58*, 63-70.

Dowse, R., & Ehlers, M. S. (2001). The evaluation of pharmaceutical pictograms in a low-literate South African population. *Patient Education and Counseling, 45*, 87-98.

Dray-Spira, R., Gary-Webb, T. L., & Brancati, F. L. (2010). Educational Disparities in Mortality *Among Adults With Diabetes in the U.S. Diabetes Care, 33*(6), 1201-1205.

Grenier, S. G., Vaillancourt, R. V., Pynn, D. P., Cloutier, M. C. C., Wade, J. W., Turpin, P. M. T., Pascuet, E. P., & Preston, C. P. (2011). Design and development of culture-specific pictograms for the labeling of medication for first nation communities. *Journal of Communication in Healthcare*, *4*(4), 238-245.

Hameen-Anttila, K., Kemppainen, K., Enlund, H., Patricia, J. B., & Marja, A. (2003). Do pictograms improve children's understanding of medicine leaflet information? Patient Education and Counseling 55, 371-377.

iText Software Corporation: (2013). itext [Software].

Mansoor, L. E., & Dowse, R. (2003). Effect of Pictograms on Readability of Patient Information Materials. *The Annals of Pharmacotherapy, 3*7, 1003-1009.

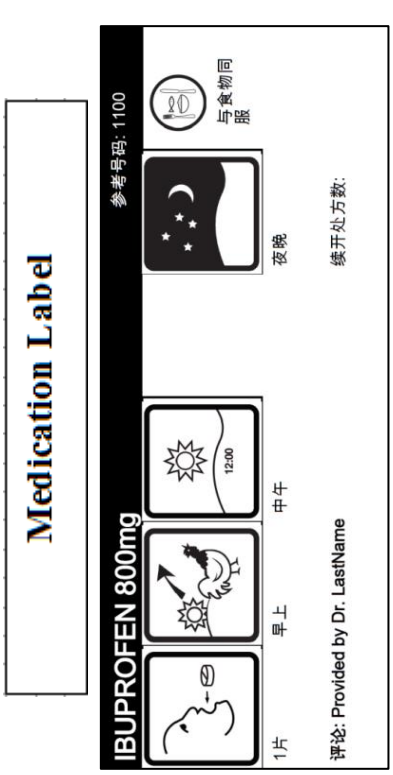
Norman, D. A. (1990). *The Design of Everyday Things*. New York: Doubleday

Saitta, K. (2002, February 24). Sending email using SMTP and Java [Online forum post]. Retrieved from http://www.developerfusion.com/code/1975/sending-email-using-smtp-and-java/

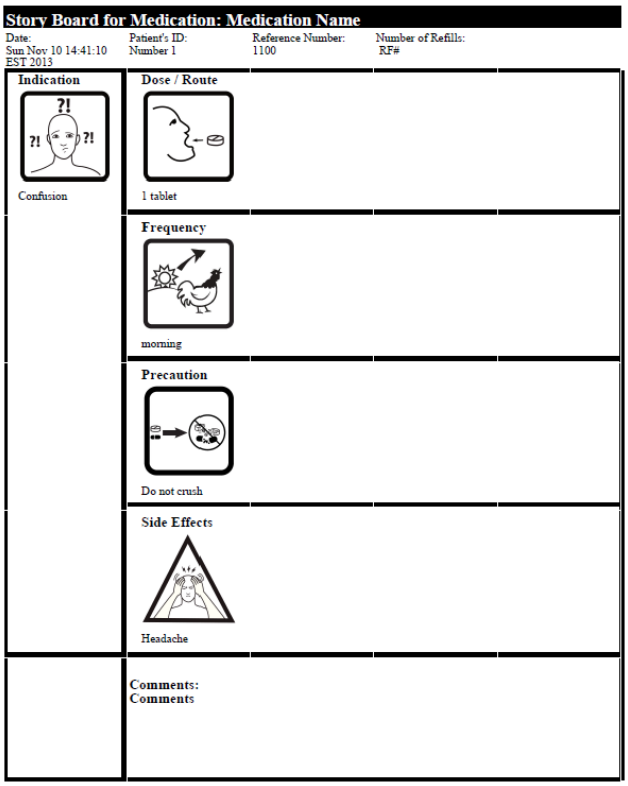
Sorfleet, C., Vaillancourt, R., Groves, S., & Dawson, J. (2009). Design, development and evaluation of pictographic instructions for medications used during humanitarian missions. C P J / R P C, 142(2), 82-87.

Yin, H. S., Dreyer, B. P., Schaick, L. V., Foltin, G. L., Dinglas, C., & Mendelsohn, A. L. (2008). Randomized Controlled Trial of a Pictogram-Based Intervention to Reduce Liquid Medication Dosing Errors and Improve Adherence Among Caregivers of Young Children. *Arch Pediatr Adolesc Med, 162*(9), 814-822.

Appendix 1: Medication Handout Designs



Medication Story Board



Medication Calendar