Design Idea of an Automative Application to Complete Repetitive Task

Qi He
1002509831
University of Toronto
heqi4@cs.toronto.edu

Meng Zhang
1004063292
University of Toronto
mzhan63@cs.toronto.edu

Wei Zheng
999986261
University of Toronto
zhengw14@cs.toronto.edu

ABSTRACT
Performing repetitive tasks on the computer are both boring and error-prone. We designed an application to identify and automate the repetitive tasks to help computer users to complete these tasks. This report shows our design justifications, introduces functionality of the application and provides a sample walk-through from users' perspective to demonstrate how this application will be used.

AUTHOR KEYWORDS
Repetitive computer tasks; automation; interactive operation; interaction elements.

ACM CLASSIFICATION KEYWORDS
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
The graphical user interface (GUI) allows users to interact with electronic devices through graphical icons and visual indicators. It makes many users who do not have specific knowledge in the computer able to complete their tasks using a computer [1]. However, for the repeated actions, it cannot be performed effectively or efficiently. Very often we must do some simple repetitive actions on the computer frequently, such as mouse movements, keystrokes, and text input. This work is monotonous and extremely boring. Besides, human factor causes mistakes even when we are doing repetitive work. Thus, optimization and automation of computer work are very important and helpful.

In our work, we designed a system to support the recognition and automation of the repetitive operations. Our system uses concise interface which enables the user to have little learning cost. The user only needs to click "Start Record" to active the application. Then, the system will monitor and record user actions in four aspects, including mouse click, keyboard input, file selection and windows changes [3]. By analyzing these actions after user repeatedly operates twice, our system could intelligently identify the loop of the repetitive operation. Afterward, when the user tends to perform the loop for the third time, a small prompt window appears. The window informs the user that repeated actions are detected and ask whether the user wants those actions to be conducted automatically. If the user confirms the request, multiple files can be chosen to be processed automatically following the repetitive actions learned by our application. Finally, the system would notify user tasks have been completed successfully.

The goal of our work is to realize repetitive actions be dynamically identified and automatically performed. We are committed to improving the user experience in handling repetitive tasks effectively. Apart from completing the function successfully, we also focus on enhancing the interaction between human and computer [8] in dealing with repetitive tasks and reducing the cost of learning for the user.

SKETCHES OF THE INTERFACE DESIGN
We sketched [9] how our user interface should look like at the early stage of our solution designing. Before sketching, we agreed to follow the principles of user interface design including but not limited to "Interfaces exist to enable interaction" [5, 7].

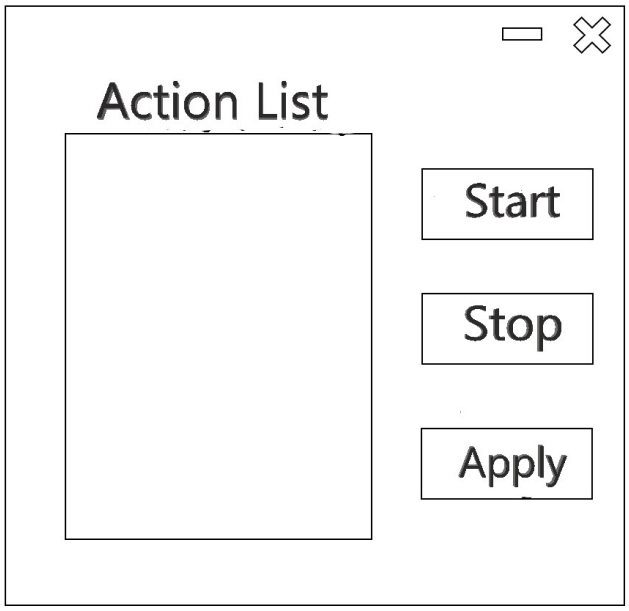


Figure 1. First Sketch of Application Interface

First Sketch
Our first sketch is shown in Figure 1. The interface includes three buttons on the right side aligned vertically: "Start" button, the user clicks this button to inform the application to record his/her actions; "Stop" button, the user clicks this button to inform the application his/her repetitive action is completed; "Apply" button, the user clicks this button to command the

application to perform repetitive tasks on selected files. There is an action list on the left side to show users what actions have been detected as repetitive and these actions are what the "Apply" function will execute.

The advantages of this sketch are: first, it informed the user what actions will be repeated; second, the interface is very simple with only four elements.

But we also discovered disadvantages of this sketch shortly by imaging interacting with the interface [4].

1. The action list is probably useless because users are aware of their actions. They may not need to review the action list.
2. We found the "Stop" button can be removed because the application knows when the repetitive actions start. It can then identify cycles in a series of actions since the start point as a repetitive task. In this way, the user can complete actions more naturally rather than making users worried about when they should hit the "Stop" button.
3. The text on buttons is too vague. This may confuse users about the functions behind.

Overall, we realized the need to simplify and optimize the interface for easier use.

Improved Sketch

Based on the analysis of our first sketch, we improved the design to the final version shown in Figure 2.

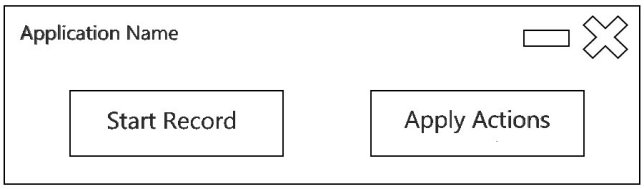


Figure 2. Revised Sketch of Application Interface

1. We removed the "Stop" button and action list to give the user a minimalism interface which only has two buttons. This window is small enough and can be hidden to the edge of the screen if the user drags it close to the edge.
2. We expanded the button text to make them easier to understand.
3. We added two prompt windows. One of them (Left in Figure 3) to notify the user that the program has detected and learned the repetitive task. The user can automate the detected repeated task or clear the action record. The other one (Right in Figure 3) is to notify the user successful completion of the tasks and let the user choose to apply the automated procedure to more targets or clear the process.

We verified the "Apply Actions" button is necessary. It makes sure the user is certain about what files will be affected to avoid undesirable modifications or the user accidentally select wrong files.

Figure 3. Two prompt windows

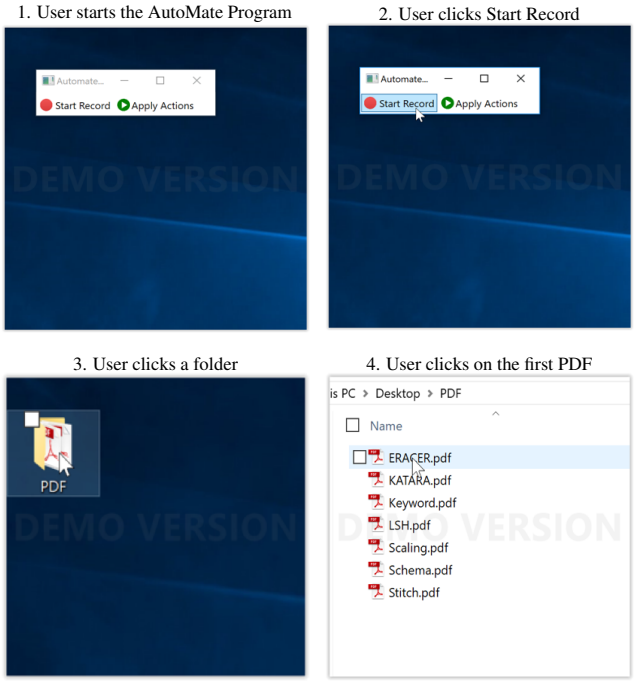
This design is better than the first sketch due to the following reasons: it eliminates the disadvantages of the first design; its simpler and suppressible interface minimizes effect to users' regular operations; its reinforced interaction with users upon detection of repetitive tasks and completion of repetitive tasks, makes the user have more control over the process.

The detailed functionality will be illustrated in Scenarios and Mock-ups sections.

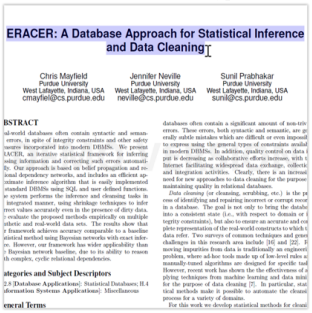
SCENARIOS/STORYBOARDS

One typical scenario is for extracting titles from PDFs. This is particularly useful to scholars and students who need to make citations or literature lists from the massive amount of academic papers.

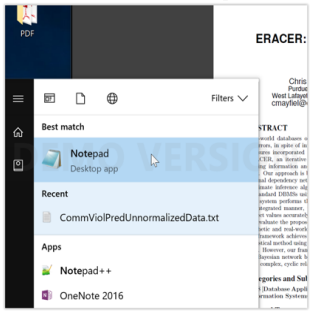
A narrative walk-through of how the automated program will work in this scenario is shown in Figure 4 in storyboard format [6].



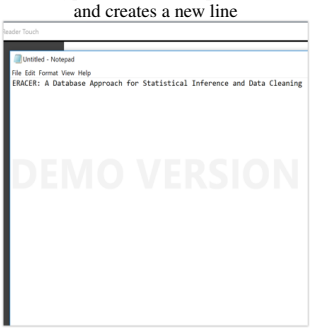
5. User selects the title and types Ctrl+C



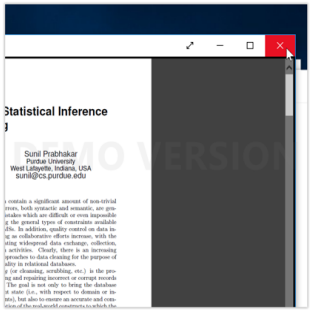
6. User launches notepad



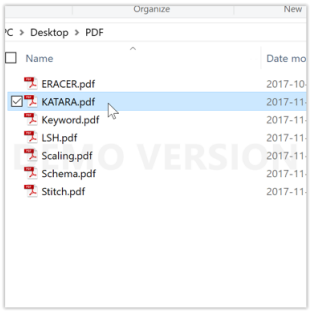
7. User types control-v and pastes the title and creates a new line



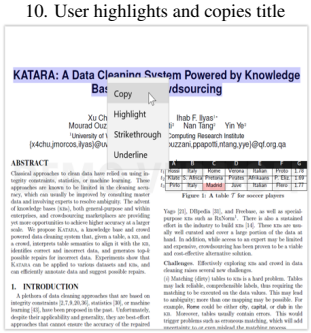
8. User closes the PDF window



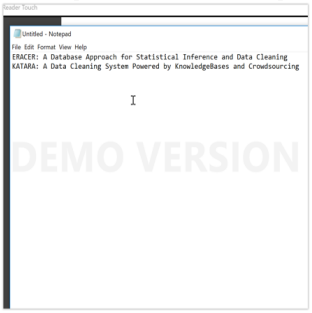
9. User clicks the next PDF



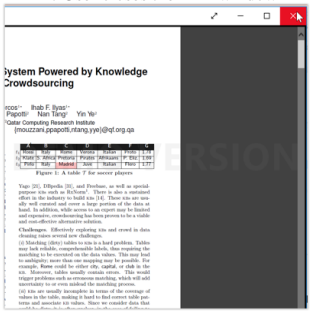
10. User highlights and copies title



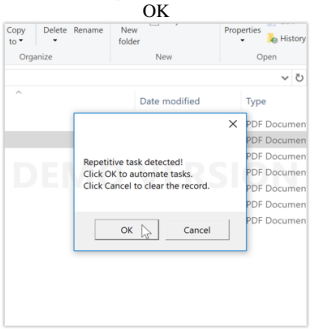
11. User pastes the title into the notepad



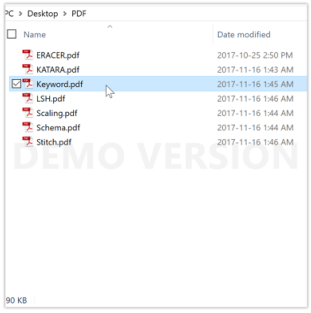
12. User closes the PDF window



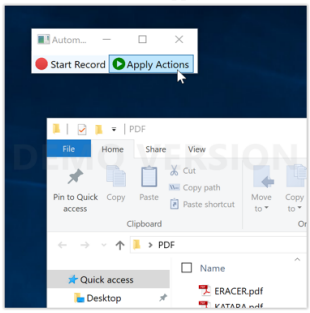
13. User sees a prompt window and clicks OK



14. User the third file

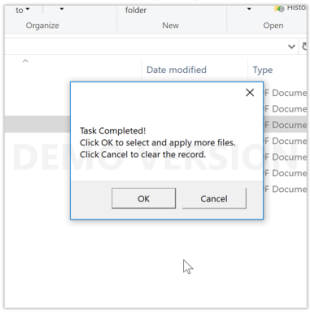


15. User clicks Apply Actions

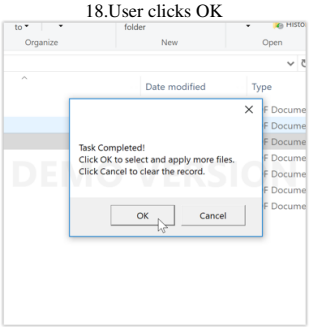


16. User sees the title in selected file was copied to Notepad

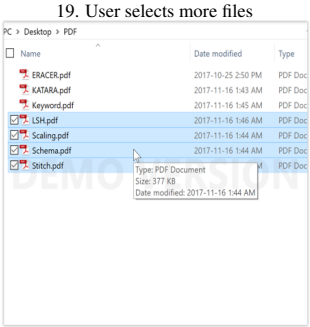
17. User sees a prompt window



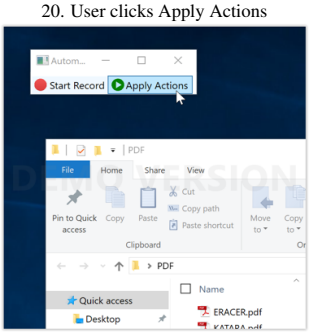
18.User clicks OK



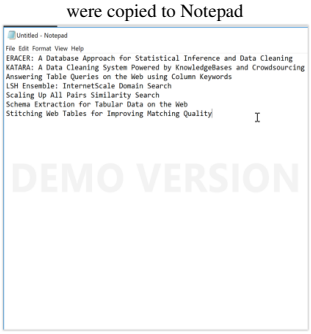
19. User selects more files



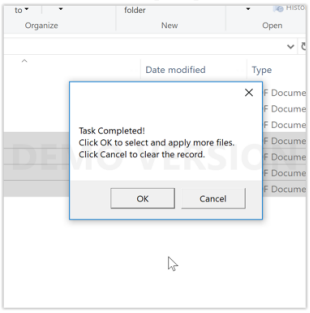
20. User clicks Apply Actions



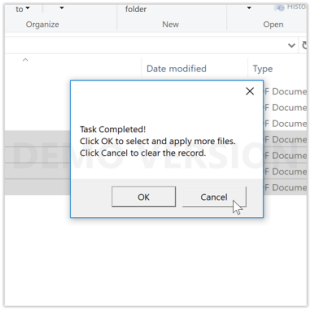
21. User sees the titles in selected files were copied to Notepad



22. User sees a prompt window



23.User clicks Cancel



24. Done

Figure 4. SCENARIOS/STORYBOARDS

MOCK-UPS

The protocol of our user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user's goals. We only placed two buttons for the user to facilitate their finishing the task at hand without distracting the user's attention. The button "Start Record" is the trigger to activate the application. The button "Apply Actions" is for the user to apply the automation of repetitive tasks. The mock-up is shown in Figure 5.

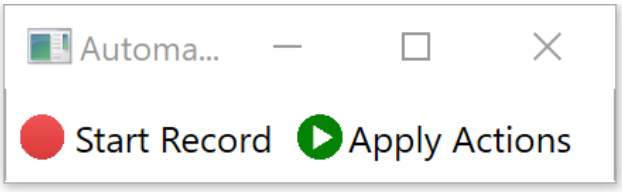


Figure 5. Interface mock-up

We designed the pop-up notification (Left in Figure 6) for users to confirm whether they would like to automate repetitive tasks when the application detects the repeated actions.

When the application completes the task, there is a notification (Right in Figure 6) for the user to choose to continue or quit the operation.

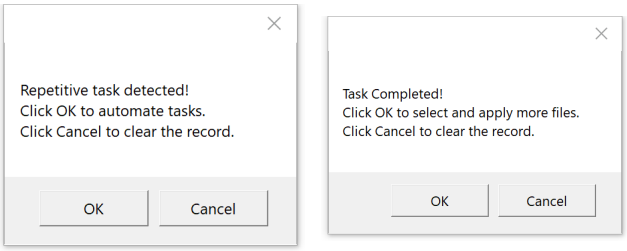


Figure 6. Two prompt windows

IMPLEMENTATION PLAN

We divided the system into four modules, including actions recorder, repetitive actions identifier, automation of repetitive actions and user interface. The workload will be split among three group members. The duration of development for this system is about two weeks including developer test [2]. This system will be developed based on Windows 10 operating system, and we will use Microsoft.Net Common Language Runtime(CLR) as our developer-platform to implement this application.

EXPERIMENT PLAN

The experiment design will happen along with the implementation of the prototype. The experiment is going to take place within one week after we completed the prototype. Each participant will be asked to perform tasks both with and without our prototype. The order of these two tasks will be randomly assigned to each participant to assure fairness in testing.

REFERENCES

1. Peter Achten, Marko Van Eekelen, and Rinus Plasmeijer. 2003. Generic graphical user interfaces. In *Symposium on Implementation and Application of Functional Languages*. Springer, 152–167.
2. B Braithwaite. 2015. Developer Testing - Why should developers write tests? (10 May 2015). Retrieved November 16, 2017, from `http://www.bradoncode.com/blog/2015/05/10/developer-testing`.
3. Urmila Kukreja, William E Stevenson, and Frank E Ritter. 2006. RUI: Recording user input from interfaces under Windows and Mac OS X. *Behavior Research Methods* 38, 4 (2006), 656–659.
4. James A. Landay and Brad A. Myers. 1995. Interactive Sketching for the Early Stages of User Interface Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 43–50. DOI: `http://dx.doi.org/10.1145/223904.223910`
5. J McKenzie. 1988. Guidelines and principles of interface design. (12 1988).
6. Mark W. Newman and James A. Landay. 2000. Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice. In *Proceedings of the 3rd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '00)*. ACM, New York, NY, USA, 263–274. DOI: `http://dx.doi.org/10.1145/347642.347758`
7. J Porter. n.d. Principles of User Interface Design. (n.d.). Retrieved November 16, 2017, from `http://bokardo.com/principles-of-user-interface-design/`.
8. Jean Scholtz, Jeff Young, Jill L Drury, and Holly A Yanco. Evaluation of human-robot interaction awareness in search and rescue. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 3. IEEE, 2327–2332.
9. K Tufts. 2014. Why Sketching Is An Important Part of The Design Process. (7 Jan 2014). Retrieved November 16, 2017, from `http://www.dnnsoftware.com/blog/why-sketching-is-an-important-part-of-the-design-process`.