СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Уровень удобства использования программного интерфейса влияет на качество всего ПО в целом. Признаком недостаточного уровня удобства использования является наличие проблем взаимодействия пользователя с пользовательским интерфейсом. Они могут быть связаны либо со сложностью формулирования плана действий (принятия решений, что делать дальше), либо с непониманием ответа системы (как изменения в интерфейсе связаны с выполненными действиями) [1].

Проблемы взаимодействия в большинстве случаев можно определить по наличию в данных активности пользователей определенных последовательностей действий (шаблонов). Для их обнаружения применяются различные методы анализа собираемых данных – как требующие ручного анализа (например, тепловые карты [2, 3]), так и использующие алгоритмы автоматического анализа [1] на основе шаблонов, выявленных исследователями ранее [4–6]. Автоматический анализ экономит время и деньги, так как эксперты вместо анализа всех данных фокусируют внимание на отдельных областях пользовательского интерфейса, где были выявлены соответствующие шаблоны.

Цель работы – провести обзор существующих методов анализа пользовательской активности, сформулировать критерии для их оценки и провести сравнение рассмотренных методов.

Задачи работы:

- рассмотреть существующие решения в области анализа пользовательской активности;
- классифицировать методы анализа пользовательской активности;
- выбрать для них критерии оценки и сравнить.

1 Аналитическая часть

Тестирование удобства использования программного обеспечения обычно состоит из двух этапов. Первый этап заключается в сборе данных о действиях, совершаемых пользователями посредством взаимодействия с графическим интерфейсом программы (движение курсора мыши, нажатие клавиш мыши, нажатие клавиш клавиатуры и т.д.), и характеристиках действий (координаты курсора, частота нажатия, используемые клавиши и т.д.). Такие данные обозначаются устоявшимся термином «активность пользователей». Второй этап – анализ этих данных экспертом с целью выявления проблем связанных с удобством использования, что является трудоемкой задачей. Поэтому, встает вопрос о хотя бы частичной автоматизации этого этапа, для чего требуется наличие соответствующих моделей и алгоритмов.

1.1 Шаблоны поведения пользователя

По мнению многих исследователей (например, авторов [1, 4–6]), индикатором проблем удобства использования может являться наличие часто повторяемых одинаковых последовательностей действий. Они могут означать, что пользователь пытается достичь цели и каждый раз терпит неудачу. Например, пользователь пытается взаимодействовать с изображением, которое он принял за кнопку [1], или пользователь нажимает кнопку и каждый раз получает ошибку.

В работе [4] выделен ряд шаблонов, связанных с выполнением пользователем поставленных задач, например, шаблон «Отмена действия», когда пользователь отменяет действие сразу после его выполнения, или шаблон «Повторение действий», когда пользователь часто повторяет простые действия (клики мыши или нажатие клавиш). Наличие второго шаблона может означать недостаточную отзывчивость интерфейса, кото-

рая ошибочно приводит пользователя к мысли, что система не распознает его действие.

Отдельные исследователи предлагают отслеживать более простые индикаторы: количество вызовов онлайн-справки, количество действий отмены, частое открытие-закрытие выпадающих списков, нажатие одной и той же кнопки более одного раза и т.д. [5]. Другие исследователи основываются на обнаружении проблем поиска информации пользователем в процессе просмотра веб-сайта [6]. Например, выделяется шаблон вертикального или горизонтального перемещения курсора мыши. В процессе визуального поиска на странице пользователь обычно перемещает курсор вслед за элементами, а значит, тратит много времени на поиск элемента.

Перечисленные методы поиска шаблонов поведения пользователей имеют много общего с задачей поиска последовательных шаблонов из области интеллектуального анализа данных [7]. В большинстве случаев все шаблоны являются последовательными, варьируются лишь анализируемые события. Однако данные активности пользователей почти всегда представляют собой не короткие транзакции, а большие наборы действий, которые в большинстве случаев невозможно корректно разделить на поднаборы [2, 3].

Поиск последовательных шаблонов давно и активно применяется в области торговли [8]. Поиск наиболее частых наборов позволяет получать информацию о том, через какой промежуток времени после покупки товара «А» человек наиболее склонен купить товар «Б» или в какой последовательности приобретаются товары. Получаемые закономерности в действиях покупателей можно использовать для персонализации клиентов, стимулирования продаж определенных товаров, управления запасами [8]. Это позволяет, с одной стороны, увеличить продажи, с другой – предложить клиентам товар, который, скорее всего, будет им интересен, а значит, минимизировать их временные затраты на поиск.

Как уже отмечалось, одной из возможных причин появления регулярно повторяющихся шаблонов в данных активности пользователей является наличие ошибок или затруднений при взаимодействии с интер-

фейсом. В этом случае может наблюдаться снижение и результативности, и эффективности пользователей. Следовательно, уменьшение числа подобных шаблонов снижает риск возникновения ошибок.

Другой возможной причиной наличия повторяющихся шаблонов в данных активности пользователей является потребность выполнения одних и тех же повторяющихся цепочек действий для выполнения поставленных задач. Закономерно, что автоматизация промежуточных действий уменьшает затраты ресурсов. Следовательно, чем меньше пользователь совершает однотипных цепочек действий, тем меньше он затрачивает ресурсов, а значит, тем эффективнее взаимодействие.

Конечно, при этом отмечается, что повторяющиеся шаблоны могут быть образованы из-за повторяющихся задач, которые либо невозможно или нецелесообразно автоматизировать, либо являются нормальным корректным поведением [1]. Поэтому требуется понимание семантики шаблонов и конкретных действий.

2 Существующие решения

2.1 Математическая модель пользовательской активности ПО

В статье [9] предлагается математическая модель активности пользователей ПО, основанная на теории последовательных шаблонов для предметной области оценки удобства использования. Данный способ совмещает анализ временных характеристик с вычислением уровней поддержки последовательных шаблонов.

Модель состоит из следующих элементов:

- множество событий с атрибутами;
- множество классов событий;
- функция классификации событий;
- множество сессий до классификации;
- множество сессий после классификации и фильтрации;
- множество последовательных шаблонов;
- множество значений поддержки последовательных шаблонов;
- функция преобразования класса событий в затрачиваемое время.

Данная модель может найти применение при оценке удобства использования пользовательских интерфейсов и для решения задач повышения эффективности взаимодействия пользователей с ПО.

Имея значения поддержки и затрачиваемого времени для каждого шаблона, эксперт может сконцентрироваться на наиболее значимых из них для процесса работы пользователей с ПО в целом. Набор шаблонов при этом будет зависеть от целей проводимого анализа.

Далее эксперт может выдвинуть гипотезы о необходимых изменениях в пользовательском интерфейсе для повышения эффективности взаимодействия пользователей с ПО. При принятии решений эксперту необходимо учитывать множество различных факторов: особенности ПО, психологические факторы использования ПО и особенности пользователей.

Изменение пользовательского интерфейса повлечет изменение множеств событий, сессий и последовательных шаблонов, так как изменится последовательность действий, необходимых для достижения пользователями поставленных целей.

Таким образом, можно утверждать, что задачей эксперта становится переход от текущей модели активности пользователей к новой, с иным составом сессий и шаблонов, следовательно, и иными значениями поддержки шаблонов и затратами времени пользователей.

После внесения изменений в программный интерфейс возможны повторный сбор и анализ данных активности пользователей, что может подтвердить либо опровергнуть выдвинутую ранее гипотезу.

Недостатком данной модели является использование заранее предопределенных шаблонов.

2.2 Алгоритм получения ассоциативных правил Apriori

Базовым алгоритмом, применяемым для получения ассоциативных правил, является алгоритм Apriori [10], автором которого является Ракеш Агравал (Rakesh Agrawal). Алгоритм Apriori использует стратегию поиска в ширину и осуществляет его снизу-вверх, последовательно перебирая кандидатов. В алгоритме используются две структуры данных: C_i — для хранения множества кандидатов в частые множества признаков длины i и F_i — для хранения частых множеств признаков длины i. Каждая структура имеет два поля — itemset, сохраняющее множество признаков, и support, которое хранит величину поддержки этого множества признаков.

Ниже представлена формальная запись алгоритма состоящего из двух частей: самого Apriori и вспомогательной процедуры AprioriGen.

Apriori($Context, min\_supp$). Context - набор данных, min_supp - минимальная поддержка, I_F – все частые множества признаков.

$C_1 \leftarrow \{1 - itemsets\}$
$i \leftarrow 1$
while$(C_i \neq 0)$
do $\begin{cases} SupportCount(C_i) \\ F_i \leftarrow \{f \in C_i | f.support \geq min\_supp\}//\text{F – частые сножества признаков} \\ C_{i+1} \leftarrow AprioriGen(F_i)//\text{C – кандидаты} \\ i++ \end{cases}$
return(I_F)

AprioriGen(F_i). F_i - частые множества признаков длины i, C_{i+1} – потенциальные кандидаты частых множеств признаков.

insert into C_{i+1} // объеденение
select $p[1]$,$p[2]$,...,$p[i]$,$q[i]$
from F_ip, F_iq
where $p[1] = q[1]$,...,$p[i-1] = q[i-1]$,$p[i] < q[i]$
for each $c \in C_{i+1}$ // удаление
do $\begin{cases} S \leftarrow (i-1) - \text{элементарные подмножства } c \\ \textbf{for each } s \in S// \text{ удаление} \\ \textbf{do} \begin{cases} \textbf{if } (s \notin F_i) \\ \textbf{then } C_{i+1} \leftarrow C_{i+1} \backslash c \end{cases} \\ \textbf{return}(C_{i+1}) \end{cases}$

Основной особенностью алгоритма можно считать использование свойства антимонотонности, которое гласит, что поддержка любого набора элементов не может превышать минимальной поддержки любого из его подмножеств. Именно благодаря этому свойству перебор не является «жадным» и позволяет обрабатывать большие массивы информации за секунды.

Существуют различные модификации алгоритма Apriori и иные алгоритмы [11], значительно оптимизированные под определенные ситуации. Можно утверждать, что применение существующего проработанного аппарата теории последовательных шаблонов позволит реализовать поиск новых (неизвестных ранее) шаблонов взаимодействия пользователей с информационной системой при меньших временных затратах.

2.3 Алгоритм GSP

Алгоритм GSP (англ. Generalized Sequential Pattern, обобщенный секвенциальный паттерн) является модификацией алгоритма AprioriAll, учитывающей ограничения по времени между соседними транзакциями [12, 13].

В случае с алгоритмом GSP требуется учитывать дополнительные условия, чтобы определить, содержит ли последовательность указанную подпоследовательность [14].

Введем такие параметры, как минимальное и максимальное допустимое время между транзакциями (min_gap и max_gap), а также понятие скользящего окна, размера win_size. Допускается, что элемент последовательности может состоять не из одной, а из нескольких транзакций, если разница во времени между ними меньше, чем размер окна.

Последовательность $d = \langle d_1 ... d_m \rangle$ содержит последовательность $s = \langle s_1 ... s_m \rangle$, если существуют такие целые числа $l_1 \leqslant u_1 < l_2 \leqslant u_2 < ... < l_n \leqslant u_n$, что:

1) s_i содержится в объединении d_k, где $l_i \leqslant k \leqslant u_i, 1 \leqslant i \leqslant n$.

2) $t_{\text{транзакции}}(d_{l[i]}) - t_{\text{транзакции}}(d_{u[i-1]}) \leqslant win\_size, 1 \leqslant i \leqslant n$.

3) $min\_gap \leqslant t_{\text{транзакции}}(d_{l[i]}) - t_{\text{транзакции}}(d_{u[i-1]}) \leqslant max\_gap$, $2 \leqslant i \leqslant n$.

Выполнение алгоритма GSP предусматривает несколько проходов по исходному набору данных. При первом проходе вычисляется поддержка для каждого предмета и из них выделяются частые. Каждый

подобный предмет представляет собой одноэлементную последовательность. В начале каждого последующего прохода имеется некоторое число ЧВП (часто встречающихся последовательностей), выявленных на предыдущем шаге алгоритма. Из них будут формироваться более длинные последовательности-кандидаты.

Каждый кандидат представляет собой последовательность, длина которой *на один больше* чем у последовательностей, из которых кандидат был сформирован. Таким образом, число элементов всех кандидатов одинаково. После формирования кандидатов происходит вычисление их поддержки. В конце шага определяется, какие кандидаты являются ЧВП. Найденные ЧВП послужат исходными данными для следующего шага алгоритма. Работа алгоритма завершается тогда, когда не найдено ни одной новой ЧВП в конце очередного шага, или когда невозможно сформировать новых кандидатов. Таким образом, в работе алгоритма можно выделить следующие основные этапы:

1. Генерация кандидатов.

   1.1. Объединение.

   1.2. Упрощение.

2. Подсчет поддержки кандидатов.

Недостатками подобных алгоритмов, существенно снижающими вычислительную эффективность обработки данных, являются:

- большое количество обращений к базе данных, соответствующее длине максимального кандидата-последовательности;

- большое число генерируемых кандидатов-последователь-ностей.

2.4 Метод оценки эффективности интерфейса GOMS

Метод GOMS (сокращение от Goals, Operators, Methods and Selection Rules – Цели, Операторы, Методы и Правила выбора) – это семейство методов, позволяющих провести моделирование выполнения

той или иной задачи пользователем и на основе такой модели оценить качество интерфейса.

Идея метода заключается в разбиении взаимодействия пользователя с интерфейсом на атомарные физические и когнитивные действия. Обладая знаниями о метриках каждой из таких составляющих, можно делать заключение об эффективности взаимодействия в целом: оценка эффективности интерфейса сводится к разбиению типовых задач на элементарные действия и сложению метрик каждого из них.

Метод GOMS включает в себя модель Keystroke-level Model (KLM) [15], которая выделяет следующие элементарные задачи и длительность каждой из них (рассчитанные на основе усредненных данных лабораторных испытаний):

- K – нажатие на клавишу в зависимости от уровня владения клавиатурой: профессиональный наборщик – 0.08 сек., эксперт – 0.12 сек., частая работа с текстом – 0.20 сек., продвинутый пользователь – 0.28 сек., неуверенный пользователь – 0.5 сек., не знакомый с клавиатурой – 1.2 сек.;
- P – указание курсором мыши на объект – 1.1 сек.;
- B – нажатие или отпускание мыши – 0.1 сек.;
- M – умственная подготовка, выбор действия – 1.2 сек.;
- H – перемещение руки в исходное положение на клавиатуре – 0.4 сек;
- R – ожидание ответа системы, зависящее от времени выполнения системой запрошенной операции.

Оценка времени на решение задачи сводится к сложению продолжительностей каждой из простейших составляющих. Например, задача, состоящая из классов (P, P, B), потребует для завершения 2.3 сек. (1.1 сек. + 1.1 сек. + 0.1 сек.).

2.5 Классификация

Рассмотрев вышеописанные методы анализа пользовательской активности, можно их разделить на следующие категории:

- поиск ассоциативных правил;
- поиск последовательных шаблонов;
- сбор и анализ временных характеристик выполнения пользователем действий и промежутков между ними;
- вычисление уровней поддержки шаблонов поведения пользователя.

Вычисление уровней поддержки шаблонов поведения пользователя позволяет ранжировать их по степени приоритета для детального анализа. Методы поиска ассоциативных правил и последовательных шаблонов позволяют найти новые (неизвестные ранее) шаблоны взаимодействия пользователей с программным обеспечением. А анализ временных характеристик, позволяет оценить эффективность взаимодействия пользователя с интерфейсом.

3 Сравнение

Поскольку рассмотренные методы используются для решения разных задач, то сравнивать их по критериям нецелесообразно. В таблице 3.1 приведено сравнение входных и выходных данных для данных методов.

Таблица 3.1 — Сравнение входных и выходных данных методов

Метод	Входные данные	Выходные данные
Математическая модель пользовательской активности ПО	Множество событий, функция классификации событий, множество сессий, множество последовательных шаблонов	Множество значений поддержки последовательных шаблонов
Алгоритм получения ассоциативных правил Apriori	Транзакции с набором элементов и минимальный уровень поддержки	Ассоциативные правила
Алгоритм получения последовательных шаблонов GSP	База данных с полями: id последовательности, id и время транзакции, набор элементов и минимальный уровень поддержки	Последовательные шаблоны
Метод оценки эффективности интерфейса GOMS	Последовательность действий	Длительность выполнения

ЗАКЛЮЧЕНИЕ

По итогу проделанной работы была достигнута цель - проведен обзор существующих методов анализа пользовательской активности, сформулированы критерии для их оценки и проведено сравнение рассмотренных методов.

Также были решены все поставленные задачи, а именно:

- рассмотрены существующие решения в области анализа пользовательской активности;
- классифицированы методы анализа пользовательской активности;
- выбраны критерии для их оценки и проведено сравнение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Siochi A.C., Ehrich R.W Computer Analysis of User Interfaces Based on Repetition in Transcripts of User Sessions. ACM Transactions on Information Systems, 1991, vol. 9, no. 4, pp. 309–335.

2. Данилов Н.А., Шульга Т.Э. Метод построения тепловой карты на основе точечных данных об активности пользователя приложения // Прикладная информатика. 2015. Т. 10. № 2. С. 49–58.

3. Danilov N., Shulga T., Frolova N., Melnikova N., Vagarina N., Pchelintseva E. Software usability evaluation based on the user pinpoint activity heat map. Advances in Intelligent Systems and Computing, 2016, vol. 465, pp. 217–225.

4. Balbo S., Goschnick S., Tong D., Paris C. Leading Usability Evaluations to WAUTER. Proc. 11th Australian World Wide Web Conf. (AusWeb), Gold Coast, Australia, Southern Cross Univ., 2005, pp. 279–290.

5. Swallow J., Hameluck D., Carey T. User interface instrumentation for usability analysis: a case study. CASCON'97, Toronto, Ontario, 1997.

6. Shah I. Event patterns as indicators of usability problems. Jour. of King Saud Univ., Comp. and Inform. Sci., 2008, pp. 31–43.

7. Mabroukeh N.R., Ezeife C.I. A taxonomy of sequential pattern mining algorithms. ACM Computing Surveys (CSUR), 2010, vol. 43, no. 1, article no. 3.

8. Aloysius G., Binu D. An approach to products placement in supermarkets using prefixspan algorithm. Jour. of King Saud Univ. Comp. and Inform. Sc., 2013, vol. 25, no. 1, pp. 77–87.

9. Сытник А.А., Шульга Т.Э., Данилов Н.А., Гвоздюк И.В. Математическая модель активности пользователей программного обеспечения. // Программные продукты и системы. 2018. Т. 31. № 1. С. 79-84

10. Agrawal R., Imielinski T., Swami A.N. Mining Association Rules between Sets of Items in Large Databases // Proceedings of the 1993 ACM SIGMOD international conference on Management of data, SIGMOD '93. – Washington, D.C., USA, 1993. – Vol. 22(2). – Pp. 207-216.

11. Agrawal R., Srikant R. Fast algorithms for mining association rules // Proceedings of the 20th International Conference on Very Large Data Bases, VLDB. – Santiago, Chile, September 1994. – Pp. 487-499.

12. Agrawal R., Srikant R. Mining Sequential Patterns // Proc. of the 11th Int'l Conference on Data Engineering. 1995. P. 3–14.

13. Srikant R., Agrawal R. Mining Sequential Patterns: Generalizations and Performance Improvements // EDBT. Springer Berlin Heidelberg, 1996. P. 1–17.

14. Интеллектуальный анализ данных: учеб. пособие. – Томск: Издательский Дом Томского государственного университета, 2016. – 120 с.

15. Card S., Moran T., Newell A. The keystroke-level model for user performance time with interactive systems. Communications of the ACM, 1980, vol. 23, no. 7, pp. 396–410.