МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Кафедра общей информатики

В. В. Шумкина

Методы адаптации пользовательского интерфейса

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
по направлению высшего профессионального образования

230100.68 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Тема диссертации утверждена распоряжением по НГУ № 64 от «12» марта 2012г.

Руководитель

Пальчунов Д. Е.

д.ф.-м.н., профессор

Новосибирск, 2013г.

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Кафедра общей информатики

УТВЕРЖДАЮ

Зав. кафедрой Пальчунов Д. Е.

…….
(подпись, дата)

ЗАДАНИЕ
на магистерскую диссертацию

студент Шумкина Вера Владимировна

факультета ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Направление подготовки 230100.68 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Магистерская программа ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНЫХ СИСТЕМ

Тема Методы адаптации пользовательского интерфейса

Цели работы: На основе имеющихся данных об обращениях пользователей к USSD сервису разработать и реализовать алгоритм, адаптирующий граф меню для различных групп пользователей с учетом ограничений, накладываемых иерархической структурой меню и спецификой использования USSD технологии.

Руководитель

Пальчунов Д. Е.

д.ф.-м.н., профессор

……..
(подпись, дата)

Содержание

ВВЕДЕНИЕ

Интерфейс имеет большое значение для любой программной системы и является неотъемлемой ее составляющей, ориентированной, прежде всего, на конечного пользователя. Пользовательский интерфейс – это разновидность интерфейсов, в котором одна сторона представлена человеком (пользователем), а другая - машиной/устройством. Пользовательский интерфейс предназначен для обеспечения взаимодействия между пользователем и процессом, выполняющим некоторое задание - прикладной программой. Интерфейс является двунаправленным - получив команды от пользователя и исполнив их, система выдаёт информацию пользователю наличествующими у неё средствами - визуальными, звуковыми, тактильными и т. п. Приняв эту информацию пользователь выдаёт устройству последующие команды предоставленными в его распоряжение средствами, такими как: кнопки, переключатели, регуляторы, сенсоры, голос, и т. д. Таким образом, основной задачей данного взаимодействия является передача информации, входных данных, от пользователя прикладной программе и выходных данных, результатов работы программы, пользователю.

Современным пользователям зачастую приходится сталкиваться с все возрастающим количеством различных программных продуктов, кроме того, сами информационные системы становятся более сложными, что не может не сказаться на эффективности взаимодействия конечного пользователя с информационной системой. Другой проблемой является то, что зачастую пользователи одной и той же системы имеют совершенно различные потребности при работе с ней, что делает затруднительным построение одного идеального интерфейса, который бы позволял каждому одинаково хорошо достигать своих целей. Одним из путей решения данной проблемы является использование адаптивных пользовательских интерфейсов.

Под адаптивным пользовательским интерфейсом понимают взаимосвязанную совокупность программных и технических средств, позволяющую конечному пользователю наиболее эффективно использовать все предоставленные системой возможности путем автоматически настраиваемого интерфейса под конкретного пользователя. Адаптивные системы обнаруживают общие пользовательские задачи и делают эти задачи более доступными. Самый простой пример таких систем – системы, создающие списки недавно открытых файлов или наиболее часто используемых приложений. Поскольку именно от интерфейса зависит эффективность деятельности пользователя, а соответственно и эффективность использования системы, задача разработки методов построения адаптивных интерфейсов весьма актуальна.

В данной работе в качестве адаптируемого интерфейса было взято меню USSD приложения, целью работы стала разработка алгоритма изменяющего данный интерфейс на основе статистики взаимодействия клиентов мобильной сети с данным сервисом. Полученный алгоритм должен учитывать, как ограничения, накладываемые иерархической структурой меню, так и те, что обусловлены спецификой использования USSD технологии.

Не смотря на внешнюю простоту организации интерфейса USSD приложения, взаимодействие с ним может вызывать определенные трудности: часто пользователям приходится посещать большое количество узлов меню, до того как он достигнет целевого пункта. При работе с USSD сервисом возникают дополнительные сложности, связанные прежде всего с тем, что пользователь видит только ту страницу меню, на которой он находится в данный момент, при этом смысл названия пунктов меню зачастую понятен только при наличии информации о названиях родительских узлов, и каждый переход по меню требует ввода тестовой информации при ограниченном времени на ответ. В целом, чем длиннее путь до целевой страницы, тем выше вероятность, что пользователю не удастся ее достигнуть, таким образом, основной целью адаптации меню является снижение количества переходов, необходимых для достижения наиболее часто используемых услуг.

Адаптированное меню должно снизить нагрузку на пользователя путем уменьшения количества страниц, прохождение которых требуется для достижения пользователем своей цели, и, возможно, в перспективе увеличит количество успешных сессий пользователя в USSD меню. Кроме того, снижение количества страниц также снизит сессионную нагрузку на мобильного оператора, обеспечивающего функционирование данного приложения.

Таким образом основными требованиями, накладываемыми на разрабатываемый алгоритм стали:

- Алгоритм должен уменьшать количество страниц, которые необходимо пройти от начальной страницы до страницы с нужным сервисом.
- Прежде всего, алгоритм должен адаптировать наиболее используемые пути в меню, при этом также должна учитываться удаленность сервиса от текущей начальной страницы.
- Получаемое алгоритмом меню должно удовлетворять заданным ограничениям.

- Структура адаптированного меню по возможности должна быть близка к исходному меню, для того чтобы его взаимодействие с ним не вызывало сильных трудностей для пользователя.

В результате выполнения поставленных задач был разработан и реализован алгоритм, адаптирующий граф USSD меню в соответствии с требуемыми ограничениями, на основе имеющейся информации, о взаимодействии с сервисом. Также было проведено исследование взаимодействия адаптационного алгоритма с имеющимися алгоритмами кластеризации пользователей данного сервиса.

Разработанный алгоритм и его программная реализация были представлены на конференции Мальцевские чтения в 2012 г. в секции «Алгебро-логические методы в информационных технологиях» и на 51-й международной научно-студенческой конференции «Студент и научно-технический прогресс» в секции Информационные технологии, а также на семинаре «Прикладная логика». Кроме того, были опубликованы тезисы в сборнике материалов МНСК-2013 в секции «Информационные технологии»[1] и статья в журнале «Альманах современной науки и образования»[2].

Глава 1. Обзор предметной области

1.1 Описание USSD сервиса

USSD (Unstructured Supplementary Service Data) - это уникальная для сетей мобильной связи услуга двунаправленной сеансовой передачи неструктурированных данных дополнительных услуг, реализованная только в сетях стандарта GSM. Данная технология позволяет организовать высокоскоростное интерактивное взаимодействие между абонентом и сервисными приложениями оператора в режиме передачи данных. Используемая при этом одноименная технология имеет сходство с технологией SMS, прежде всего в том, что также использует формат коротких сообщений, однако, USSD имеет ряд существенных отличий.

USSD сервис, в основном, предназначен для обмена сообщениями между абонентом и дополнительными сервисами, в простейшем случае, службой автоинформатора расчётного счета, тогда как SMS в основном служит для обмена короткими сообщениями между абонентами. Также, USSD, в отличие от SMS, не имеет промежуточной базы данных и не гарантирует повторную доставку сообщений, что делает обмен сообщениями мгновенным. USSD технология является сессионно-ориентированной, весь диалог между абонентом и приложением ведётся в рамках одной сессии.

Для того чтобы воспользоваться USSD сервисом, абонент набирает на своем мобильном терминале номер, имеющий символ * в начале и символ # в конце. Наличие символа * в начале и символа # в конце набранного номера означает, что происходит не обычный звонок, а обращение к USSD-приложению. Нажатием клавиши звонка абонент осуществляет отправку запроса. После этого устанавливается соединение, и приложение отправляет нужную информацию в виде USSD-пакета на абонентский терминал. Содержимое этого пакета в виде текста отражается на экране терминала. Если логика USSD-приложения предусматривает продолжение общения между абонентом и приложением, то сессия не обрывается, и абонент может продолжить использование USSD приложения.

Максимальная длина USSD сообщения ограничена и составляет 182 символов для текстов, написанных латиницей и 80 символов для кириллицы. Если на какой-либо запрос абонента требуется совершить ответное действие, например, ввести пароль, на эту операцию будет отведено – 60 секунд. В рамках этого времени абонент должен отправить запрос в систему, в противном случае соединение будет разорвано и телефон выдаст

ошибку «Network error / Проблема сети», после чего абоненту будет необходимо заново выполнить операции по входу в USSD меню.

1.2 Адаптивные интерфейсы

Под адаптивным пользовательским интерфейсом понимают взаимосвязанную совокупность программных и технических средств, позволяющую конечному пользователю более эффективно использовать все предоставленные системой возможности путем автоматически настраиваемого интерфейса под конкретного пользователя. [3].

Адаптивные системы также включают системы, которые обнаруживают общие пользовательские задачи и делают эти задачи более доступными. Примеры таких систем – системы, создающие списки недавно открытых файлов, наиболее часто используемых приложений, история посещения страниц в интернете, статистика запросов в поисковых системах, различные рекомендательные системы.

Настройка функциональных возможностей и параметров интерфейса может осуществляться либо вручную самим пользователем, либо автоматически системой, на основании имеющейся информации о пользователе.

Для обеспечения возможности ручного редактирования интерфейса в системе должны присутствовать средства, позволяющие пользователю изменять меню, добавлять макрокоманды, панели инструментов, назначать действия кнопкам панелей инструментов и т. п. Недостаток данного подхода заключается в необходимости пользователя быть достаточно хорошо знакомым, как с самой системой, так и со средствами, позволяющими изменять ее интерфейс.

При автоматическом подходе сама система изменяет интерфейс для пользователя, согласно его потребностям. Система создает модель пользователя, на основании которой и выстраивается процесс адаптации. Однако такие системы могут вызвать у пользователя чувство потери контроля, возможны некоторые неточности в предсказании желаний и поведения пользователя, поэтому для представления пользователю свободы управления система должна спрашивать его о принятии тех или иных изменений, за пользователем остается право принять или отклонить адаптационные изменения.

В основе систем автоматической адаптации лежит модель пользователя, она представляет собой совокупность сведений о конкретном пользователе, позволяющую прогнозировать цели пользователя, его предпочтения или производить опознание

образцов поведения. Построение модели пользователя является ключевой частью процесса адаптации, ведь именно в ней накапливаются сведения о тех особенностях пользователя, на основе которых будет впоследствии происходить изменение интерфейса. Сбор информации может происходить явно - через анкетирование, тесты, либо неявно – как результат наблюдения за действиями пользователя.

При построении модели пользователя могут использоваться индивидуальный или стереотипный подход. При стереотипном подходе используется предполагаемая принадлежность пользователя к определенной модели (классу), количество которых строго ограниченно. Обычно создают несколько моделей пользователей. И. Бомон выделяет 3 модели: «модель новичка», «модель продвинутого пользователя» и «модель эксперта» [4]. Если каждому пользователю системы соответствует своя модель, отличная от других моделей пользователей, то такой подход является индивидуальным. Для каждого пользователя собирается статистика его действий в системе, и на ее основании принимается решение о возможных изменениях интерфейса, которые могут помочь обеспечить более удобное и эффективное использование системы данным пользователем. Построение индивидуальных адаптивных интерфейсов может стать проблемой для многопользовательских систем с большим количеством клиентов, в таких случаях целесообразно использовать стереотипный подход, однако, при этом возникает необходимость решения задачи кластеризации пользователей.

Модель пользователя может быть статичной, т. е. информация, представляющая пользователя, не меняется со временем, либо динамической, информация в ней может быть изменена, соответственно текущим потребностям пользователя. В динамических моделях интересы делятся на долгосрочные, которые не меняются со временем, и краткосрочные, которые часто меняются. Так как краткосрочные интересы часто изменяются, о них трудно собирать информацию, и в целом их труднее выделять по сравнению с долгосрочными интересами.

Обычно модели пользователя строятся путем приписывания весов ключевым словам, которые представляют интересы пользователя. Также они могут быть построены с использованием семантических сетей и ассоциативных правил. Модели на основе ключевых слов легче всего построить, но они требуют большое количество информации о пользователе. Модели на основе ключевых слов строятся извлечением подходящих ключевых слов из документов, представляющих интерес для пользователя, или предоставляются самим пользователем явно. Документы, представляющие интерес для пользователя извлекаются из истории браузера, закладок и загруженных файлов.

Извлеченным ключевым словам после приписывается вес, и этот вес означает степень заинтересованности пользователя в данной теме. Так, каждая пара (слово, счет) может использоваться для представления модели интересов пользователя. Другой подход может заключать в группировании ключевых слов по множествам и формировании модели интересов пользователя из этих множеств. Также существуют модели интересов пользователя на основе анализа понятий. Главное их отличие от моделей на основе ключевых слов в том, что вектор интересов содержит понятия, а не ключевые слова.

Существует множество систем, строящих модель пользователя, но большинство из них, в основном предназначено для извлечения информации о пользователе из посещенных им web-страниц или просмотренных файлов. Такие системы могут быть использованы в рекомендательных системах, системах адаптации результатов поиска и других системах, где требуется информация об общих интересах пользователя, а не об его взаимодействии с конкретной системой в целях изменения ее интерфейса. Примерами систем, строящих модель интересов пользователя, являются:

- Amalthea [5]. Строит профиль, основанный на терминах, извлекая их из веб страниц, посещенных пользователем и проставляя им вес. Эта система строит только один вектор интересов и не имеет различения для долгосрочных и краткосрочных интересов, также система использует информацию, предоставляемую непосредственно пользователем.
- PEA [6]. Здесь профиль пользователя представлен множеством векторов интересов, где каждый вектор представляет один из множества интересов пользователя. Каждая страница, добавленная в закладки, представляет собой интересы пользователя, так на одну страницу строится один вектор интересов. Но утверждение о том, что пользователь добавляет в закладки все страницы, содержащие его интересы, достаточно слабо, но, тем не менее, использование нескольких векторов интересов делает профиль более точным.
- WebPersonae [7]. Использует n последних посещенных страниц и строит вектор частоты упоминания терминов. После этот вектор берется в качестве текущих интересов пользователя.
- OBIWAN [8]. Строит профиль пользователя, используя иерархию понятий. Классифицирует страницы, посещенные пользователем по нескольким категориям, учитывая время, проведенное на страницах.

После того, как модель пользователя построена, начинается процесс собственно адаптации. В зависимости от системы и от представляемых ей возможностей алгоритмы адаптации могут сильно отличаться друг от друга, общим между ними является то, что результат их работы всегда направлен на повышение эффективности взаимодействия пользователя и информационной системы.

В целом, имеет смысл производить адаптацию интерфейса только для тех пользователей, поведение которых можно считать устойчивым. Стабильность поведения и интересов потребителя может быть определена путем разбиения информации, собранной о нем, на несколько подмножеств. Одно из этих подмножеств будет использовано в качестве обучающей выборки, на основе которой либо, в случае индивидуального подхода, будет подстроен адаптированный интерфейс, либо, при стереотипном подходе, пользователь будет отнесен к некоторой выделенной подгруппе. Далее, на оставшихся подмножествах проверяется достаточно ли хорошо подходит ему полученный интерфейс или попадает ли он в ту же группу. Пользователи, результаты которых не удовлетворяют требуемому условию, могут быть признаны нестабильными и исключены из процесса кластеризации и/или адаптации интерфейса. Определение стабильности интересов помогает в автоматическом режиме решить вопрос о том, когда необходимо включать в процесс адаптации интерфейса новых пользователей системы.

Последним шагом процесса адаптации интерфейса является введение результатов адаптации в работающую систему, при этом следует учитывать, что резкие изменения в интерфейсе системы могут вызвать недовольство пользователей, поэтому нужно предупреждать пользователя о принятых адаптациях и давать ему возможность принимать решение относительно принятия или отклонения предложенных изменений.

Глава 2. Постановка задачи

Среди огромного разнообразия существующих на данный момент видов пользовательских интерфейсов, остается широко применимым такой интерфейс, как текстовое иерархическое меню. Одним из примеров тестового интерфейса с древовидной структурой является меню USSD сервисов, разработке способа автоматической адаптации для которого и посвящена данная работа.

В качестве исходных данных имеется информация о взаимодействии пользователей с одним из существующих USSD сервисов. Информация представлена в виде логов сессий. Каждая сессия содержит идентификатор пользователя и последовательность страниц USSD меню, посещенных им, начиная с начальной страницы, точки входа пользователя в систему, и заканчивая последней посещенной страницей в этой сессии. В случае успешной сессии, эта последняя страница будет представлять собой ссылку на некую услугу, интересующую пользователя, назовем такие страницы финальными. Страницы, не являющиеся финальными, будем называть промежуточными.

В имеющихся логах явно не обозначены успешные и неуспешные сессии, поэтому была проведена предварительная работа по выявлению финальных страниц на основе анализа всего множества логов. В общем случае финальными принимались те страницы, которые никогда не были промежуточными.

Исследование имеющихся логов показало, что в последовательностях, проходимых пользователями, имеется большое количество возвращений на предыдущие страницы меню, а также имеется большое количество сессий, обрывающихся на промежуточных страницах. Все это позволяет сделать вывод о том, что имеющееся USSD меню является недостаточно удобным для большинства пользователей и существует необходимость в его адаптации. Также было замечено, что абоненты часто пользуются лишь некоторыми услугами, предоставляемыми USSD сервисом, и, сокращение длины пути, необходимого для достижения страницы с данными услугами, могло бы сделать USSD меню более удобным для пользователя, кроме того оператору пришлось бы отсылать меньшее число USSD сообщений в рамках одной сессии.

Основным отличием адаптации меню USSD сервиса от адаптации меню Web сайта или компьютерной программы, является то, что мобильный телефон чаще всего используется только одним человеком. Таким образом, можно с достаточно высокой вероятностью утверждать, что получая информацию о взаимодействии пользователя с USSD сервисом, мы всегда получаем информацию об одном и том же человеке.

Несмотря на то, что поведение различных абонентов в USSD меню является индивидуальным, построение отдельного адаптивного меню для каждого пользователя не является приемлемым. Прежде всего, это связано с огромным количеством пользователей в системе, адаптация меню для каждого в отдельности потребует большое количество ресурсов. Также у справочных служб оператора должна иметься информация о текущей структуре меню, для помощи абоненту в использовании сервиса в случае необходимости. Ясно, что при наличии индивидуального меню у каждого пользователя, оказание такой помощи будет крайне затруднительно. Кроме того, чтобы построить достаточно хорошее меню для каждого пользователя требуется собрать большое количество информации об его взаимодействии с сервисом в течение достаточно длинного промежутка времени.

Все это ведет к необходимости адаптации меню сразу для множества пользователей. В ходе исследования логов взаимодействия с сервисом было выявлено более чем семисот страниц, которые являются финальными на данном множестве логов. При таком большом количестве страниц с полезными сервисами построение одного меню, одинаково удобного для всех пользователей не является возможным.

Решением данной проблемы может быть разделение пользователей USSD сервиса на некоторое ограниченное количество групп и построение отдельного адаптивного меню для каждой группы. Ожидается, что степень возможности адаптации меню для подгруппы будет выше, чем у всего множества пользователей вместе взятого, это обуславливается использованием меньшего числа полезных сервисов внутри одной единицы кластеризации.

Основной частью работы является разработка и реализация алгоритма адаптации пользовательского USSD меню. На основании истории использования USSD меню определенным пользователем или группой пользователей алгоритм должен строить адаптированный граф USSD меню. Основная цель работы алгоритма – уменьшение количества страниц, которые требуется пройти от начальной страницы до финальной страницы. При этом нужно учитывать, что необходимость адаптации пути от начальной страницы до финальной, прежде всего, зависит от двух показателей: частоты использования пути и его длины, то есть количества страниц USSD меню, которые требуется пройти от начальной страницы до финальной страницы.

Также при разработке алгоритма адаптации USSD меню нужно учитывать такие ограничения, как:

- Максимальное количество символов на одной USSD странице.

- Максимальное количество пунктов меню на странице.
- Ограничения, накладываемые тематической структурой иерархического графа меню и другие.

Кроме того, должна быть возможность определить, как поступать с неиспользуемыми или мало используемыми пунктами меню: можно ли перемещать их в общий пункт меню (такой как, например – «9 далее») или же они должны быть сохранены на той странице, где были расположены изначально.

На заключительном этапе работы должно быть проведено тестирование эффективности работы алгоритма на различных подмножествах логов пользователей.

Глава 3. Алгоритм адаптации меню

Описываемый в данной главе алгоритм направлен не только на оптимизацию графа меню, но также имеет среди своих целей максимально возможное сохранение структуры исходного меню. В большинстве случаев все имеющиеся пути в меню остаются на месте после применения адаптаций, и пользователь может сам выбрать пойти ему по новому адаптированному пути или же воспользоваться уже знакомым старым.

Пример такого способа изображен на рисунке 3.1. В ходе адаптации было добавлено ребро из вершины 1 в вершину 5, но старый путь через вершину 3 также был сохранен.

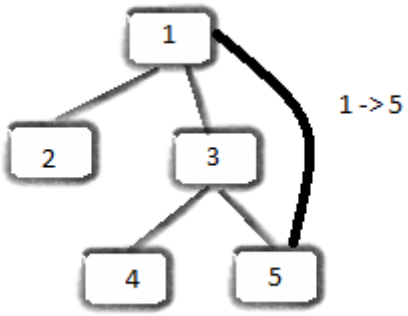

Рисунок 3.1. Пример адаптации с сохранением исходного пути.

Данный подход направлен на постепенное включение адаптаций в интерфейс приложения и снижения недовольства пользователей, которое может быть вызвано резкими изменениями в уже привычном интерфейсе.

3.1 Описание алгоритма

Основная цель работы алгоритма адаптации заключается в уменьшении длины пути, который нужно пройти пользователю для достижения своей цели при использовании USSD сервиса.

На вход алгоритму подается история взаимодействия пользователя или группы пользователей с USSD сервисом. В работе алгоритма адаптации делается ряд предположений:

- Считается, что пользователь имеет представление обо всей структуре меню, и использует все нужные ему услуги и только их.
- Предполагается, что все последовательности, поданные на вход алгоритма, и прошедшие последующую обработку, представляют собой успешные сессии, в ходе которых пользователь достиг своей цели.

Очевидно, что в адаптации нуждаются только те пути, длина которых превышает две страницы. Пути длины 1 являются прямыми переходами на финальные страницы, а пути длины 2 состоят только из точки входа и финальной страницы, расстояние между ними уменьшить нельзя, замена же страницы, являющейся точкой входа, на финальную страницу является недопустимой. Вообще, необходимость в оптимизации путей, проходимых пользователем, зависит от двух показателей: частота использования этого пути и удаленность финальной страницы пути от точки входа. Перед началом работы алгоритма адаптации следует определить минимальные значения, которые должны превышать эти показатели, чтобы последовательность была признана нуждающейся в оптимизации.

Входными данными для алгоритма являются:

- исходный граф меню;
- информация об имеющихся в меню финальных вершинах или же полное множество логов для построения списка таких вершин;
- множество логов для адаптации;
- заданные настройки для ограничений.

Перед началом работы алгоритма происходит обработка поданных логов. Прежде всего, извлекается информация о финальных вершинах графа. В качестве финальной вершины принимается любая вершина, присутствующая в исходном графе меню и никогда не использовавшаяся в качестве промежуточной в логах.

Далее происходит обработка множества логов, которые будут использованы при адаптации. Для каждой сессии на первом этапе происходит очистка пути, включающая в себя удаление возвращений на предыдущие страницы пути и повторяющихся одинаковых страниц, таким образом, получается лог сессии, содержащий прямой путь из начальной вершины в целевую вершину. На следующем шаге проверяется, является ли последняя страница полученного пути финальной, и удовлетворяет ли путь заданному ограничению на длины адаптируемых путей. Успешно прошедший все проверки путь добавляется в список путей на адаптацию, здесь же увеличивается счетчик частоты использования пути.

По окончанию процесса обработки логов из списка путей, нуждающихся в адаптации, удаляются те, количество использований которых не превышает заданного порога.

На последнем шаге обработки логов множество путей разбивается по точкам входа, и для точек входа считается оценка количества использований.

Алгоритм адаптации путей начинает свою работу с точек входа пользователя. Адаптация происходит последовательно, с каждой группой в отдельности, в порядке убывания частоты использования начальных вершин.

Для каждой группы на основе последовательностей строится ориентированный подграф меню, у каждой промежуточной вершины вычисляется количество доступных символов на странице и свободных пунктов меню. Эти значения получают из заданных ограничений и полного графа меню. Также для каждой вершины считается ее удаленность от корня, а также количество посещений данной страницы, для промежуточных страниц это количество посещений финальных вершин, которые можно из них достичь.

Алгоритм проходит граф, начиная с начальной вершины – точки входа пользователя в систему, эта вершина заносится в список корневых вершин.

Для текущей корневой вершины генерируется множество возможных новых ребер. Новые ребра могут вести только на страницы, находящиеся в текущем подграфе ниже, чем текущая корневая вершина, т. е. являющиеся ее потомками. Генерируемое ребро не должно присутствовать в исходном графе, и добавление ссылки на конечную вершину ребра к текущей странице меню не должно вызывать нарушения ограничения на количество символов. Для каждого ребра считается выгода, которую должно принести его добавление – произведение разницы глубин конечной и стартовой вершины ребра на количество использований конечной вершины.

После генерации множества возможных ребер, происходит выбор подмножества ребер с максимальной суммарной выгодой, при этом выбранное множество не должно нарушать ограничения на количество символов на странице и разрешенное количество добавляемых новых ссылок. Таким образом, здесь решается линейная задача поиска максимума с ограничениями.

На следующем шаге ребра из найденного рекордного подмножества добавляются как принятые адаптации меню. После чего текущая обрабатываемая вершина удаляется из списка корневых вершин, а все ее дети заносятся в этот список. Таким образом, обеспечивается спуск по адаптируемому поддереву.

Далее алгоритм берет следующую вершину из списка корневых вершин и повторяет процесс адаптации для нее. Когда обработаны все вершины текущего поддерева, то есть список корневых вершин пуст, алгоритм переходит к обработке следующей точки входа в меню.

После обработки всех точек входа в граф, для каждой адаптируемой пары: точка входа – финальная страница ищутся новые кратчайшие пути в графе, и, на основе различий длин полученной и исходной последовательности, строится оценка уменьшения количества необходимых действий.

Выходными данными алгоритма являются:

- адаптированный граф меню;
- оценка уменьшения количества необходимых действий и статистика проведенных адаптаций.

Алгоритм не гарантирует построения оптимального адаптированного меню, кроме того структура меню может в итоге достаточно сильно измениться, что будет непривычно для пользователя, но он гарантированно не увеличивает пути до финальных вершин, а если адаптация возможна, то уменьшает их.

3.2 Поддерживаемые ограничения

Кроме ограничений, описанных в нижеследующих подразделах, алгоритм также использует ограничения связанные со статистическими параметрами адаптируемых путей, такими как минимальное количество страниц в пути и минимальное количество посещений.

Так как оценки, используемые в процессе адаптации, представляют собой произведение длины пути и частоты его использования, то малоиспользуемые, но длинные пути могут негативно влиять на результат адаптации. В качестве пути решения данной проблемы может быть использовано ограничение на минимальное количество посещений.

Ограничение на количество страниц в пути может быть полезно, когда необходимо особое внимание уделить улучшению доступности более удаленных от точки входа финальных страниц.

3.2.1 Ограничения, накладываемые спецификой использования USSD технологии

Как уже было описано выше - максимальная длина USSD сообщения ограничена и составляет 182 символов для текстов, написанных латиницей и 80 символов для кириллицы. Данный факт приводит к естественному ограничению: при добавлении новых ссылок на существующие страницы меню, общее количество символов на USSD странице должно не превышать указанного максимума.

Другой проблемой работы с USSD является то, что данная технология поддерживается на подавляющем большинстве современных мобильных устройств, имеющих самые различные по размеру дисплеи, поэтому использование длинных списков приведет к дополнительному неудобству для устройств с небольшими экранами: для просмотра всей страницы клиенту необходимо будет пользоваться прокруткой.

Кроме того, время, отводимое на ожидание ответа пользователя, также ограничено, длинный список ссылок может привести пользователя в замешательство и препятствовать получению ответа за установленное время, что приведет к обрыву сессии.

Для преодоления этих проблем в алгоритме используется еще одно задаваемое ограничение – максимальное количество пунктов на странице USSD меню.

3.2.2 Работа с выделенными тематическими подграфами меню

В силу ограниченного максимума символов на USSD странице названия пунктам меню присваиваются короткие, и, зачастую, понять смысл имеющейся ссылки можно только при наличии знания об именах ссылок, предшествующих данной в дереве меню.

При разработке алгоритма было предложено работать с такими сильно связанными подграфами меню, как с отдельными тематическими меню.

Для каждого выделенного тематического подграфа меню работа происходит следующим образом: при построении вспомогательных меню, используемых в алгоритме адаптации, все пути, включающие в себя страницы из тематического подграфа, но начинающиеся вне данной темы, рассматриваются как пути, финальной страницей которых является корень тематического подграфа. Корень имеет то же количество посещений, что и финальные страницы, достижимые из него. Таким образом, в основном процессе адаптации участвуют только корни тематик.

Сами тематические подграфы адаптируются позже, как подменю с точкой входа, являющейся корневой страницей данной тематики.

В результате, в процессе адаптации меню изменяется таким образом, что не существует вершин тематических подграфов, которые были бы ближе к точкам входа в граф меню, чем корень тематики.

3.2.3 Перемещение редко используемых меню

Еще одним параметром алгоритма является то, можно ли перемещать имеющиеся на странице, но неиспользуемые или редко используемые пункты меню в подпункт общим названием для которого, могут стать такие слова как «Еще», «Далее» и другие.

Перемещение пунктов меню происходит перед началом работы алгоритма, сразу после обработки логов. И далее алгоритм адаптации просто игнорирует пункты, находящиеся под ссылками «Еще».

Разрешение использования «Еще» ссылок положительно влияет на итоговые оценки адаптации, в особенности при адаптации для отдельных пользователей или подгрупп, но получаемый при этом выходной граф сильно отличается по структуре от исходного меню, кроме того, перемещение ссылок в подпункты «Еще» делает невозможным сохранение путей из неадаптированного графа.

3.3 Особенности реализации

Программная реализация адаптационного алгоритма осуществлена на языке программирования Java, с использованием среды разработки IntelliJ IDEA community edition.

Полученная программа позволяет работать с тремя форматами для входного графа меню: граф может быть описан на языке DOT, либо сохранен в формате TGF (trivial graph format – простой формат графов), или же граф может быть восстановлен самостоятельно из имеющегося множества логов. На рисунке 3.3.1 изображен пример структуры файла с графом, описанным на языке DOT.

```
digraph graphname {
    // Список вершин графа
    a;
    b;
    // Список ребер графа
    a -> b [label=Ссылка1];
}
```

Рисунок 3.3.1 Пример структуры файла с графом,

описанным на языке DOT.

Возможность хранения атрибутов для вершин и ребер, обеспечиваемая в формате DOT обеспечивает возможность автоматической загрузки и сохранения такой информации о графе меню, как названия ссылок меню, и разбиении графа по тематическим блокам. При использовании графов в TGF формате или восстановлении графа из логов такой информации не может быть предоставлено, поэтому пользователю будет нужно либо вручную заполнить эти данные, используя интерфейс программы, либо использовать автоматически сгенерированные тексты ссылок и отсутствием разбиения по темам.

Далее пользователю нужно загрузить множество логов, на котором будет происходить адаптация. Они могут храниться как в одном файле, так и в нескольких,

расположенных в определенной папке. Из загруженных путей удаляются все повторения и возвращения. Пути, длины которых оказались меньше трех, а также пути, используемые недостаточно часто, удаляются из списка путей, нуждающихся в адаптации.

Далее пользователю необходимо задать нужные ему ограничения алгоритма адаптации (все ограничения перечислены в разделе 3.2 данной работы) и запустить процесс адаптации.

В результате работы строится новый граф меню, который может быть сохранен либо в DOT, либо в TGF формате, а также подробный отчет о примененных адаптациях и оценка уменьшения числа необходимых действий, этот отчет также может быть сохранен.

3.4 Результаты

Полученный алгоритм был протестирован на различных подмножествах логов пользователей. При тестировании использовались значения ограничений алгоритма по умолчанию:

- Максимум символов на странице – 182, пунктов меню – 8.
- Использование перемещения в подпункт «Еще» запрещено.
- В качестве имен пунктов меню использовались идентификаторы страниц.

При проведении адаптации на всем множестве пользователей общее уменьшение числа необходимых действий составило 22 процента. При запусках алгоритма для отдельных пользователей уменьшение числа необходимых действий достигало 58 процентов на некоторых пользователях, и в общем составило 43 процента.

На заключительном этапе было проведено тестирование алгоритма на подмножествах логов, представляющих действия выделенных групп пользователей. Уменьшение количества необходимых действий достигло 39 процентов, что лишь на 4 процента ниже результата адаптации индивидуальных пользователей.

Подмножество пользователей	Общее уменьшение количества необходимых действий, в процентах
Все пользователи	22
Выделенные группы пользователей	39
Индивидуальные пользователи	43

Таблица 3.4.1. Результаты тестирования алгоритма.

Как видно из таблицы 3.4.1 алгоритм уменьшил количество посещений промежуточных страниц во всех случаях, при этом наибольшая эффективность была достигнута при адаптации для групп пользователей и для индивидуальной адаптации графа меню для каждого пользователя, при этом различие результатов для этих случаев незначительно.

Заключение

При выполнении данной работы были получены следующие результаты:

- Изучена литература, касающаяся адаптации пользовательских интерфейсов и специфики использования USSD технологий.
- Разработан алгоритм адаптации графа меню, уменьшающий количество промежуточных страниц, которые необходимо проходить пользователям для получения желаемых услуг. Алгоритм обеспечивает сохранение структуры меню близкой к исходной, при этом учитывая в своей работе ограничения, накладываемые как использованием USSD технологии, так и иерархической структурой меню.
- Реализована программа, осуществляющая адаптацию графа меню описанным алгоритмом.
- Программная реализация алгоритма протестирована на различных множествах пользователей.
- Произведена апробация алгоритма адаптации меню для выделенных групп пользователей.

Дальнейшее развитие данной работы может быть связано с исследованием возможности автоматического выделения тематических блоков меню, основываясь на анализе названий ссылок, присутствующих на страницах. Кроме того, возможно произвести обобщения в алгоритме, которые позволили бы применять его не только для USSD сервисов, но и для любых интерфейсов с древовидной структурой.

Литература

1. Шумкина В. В., Методы адаптации пользовательского интерфейса // Материалы 51-й международ. науч. студ. конф. «Студент и научно-технический прогресс», Информационные технологии. Новосибирск, 12–18 апреля 2013 г. – С. 221.
2. Шумкина В. В., Алгоритм автоматической адаптации древовидных пользовательских интерфейсов // Альманах современной науки и образования. Тамбов: «Грамота», 2013, 6 (73).
3. Findlater L., Design Space and Evaluation Challenges of Adaptive Graphical User Interfaces // AI Magazine, 2009, 30(4). Pp. 68–73.
4. Ballard B., Designing the Mobile User Experience – Chichester, England: John Wiley & Sons Ltd, 2007.
5. Ходаков В.Е., Пользовательский адаптивный интерфейс: задачи исследования и построения // Восточно-Европейский журнал передовых технологий, 2004, 2. – С. 20–29.
6. Денинг В., Диалоговая система «человек-ЭВМ». Адаптация к требованиям пользователя – М.: Мир 1984.
7. Moukas A., Amalthaea: An evolving multi-agent information filtering and discovery system for the www // Autonomous Agents and Multi-Agent Systems, 1998, 1. Pp. 59–88.
8. Montebello M., Evolvable intelligent user interface for www // Knowledge-based systems. In IDEAS, 1998. Pp. 224–233.
9. Gowan J. M., A Multiple Model Approach to Personalized Information Access // Master Thesis in computer science, Faculty of Science, University College of Dublin, 2003.
10. Gauch S., Profusion: Intelligent fusion from multiple, distributed search engines. // Journal of Universal Computer Science, 1996, 2. Pp. 637–649.
11. Тидвелл Д., Разработка пользовательских интерфейсов – СПб: Питер 2008.
12. Раскин Д., Интерфейс: новые направления в проектировании компьютерных систем – СПб: Символ-Плюс 2005.
13. Сайт о новостях в мире мобильных коммуникаций: http://amobile.ru/info/tech/ussd/work.htm [Электронный ресурс]. Доступ: 09.02.2013.
14. Unstructured Supplementary Services Data (USSD): http://www.telecomspace.com/messaging-ussd.html [Электронный ресурс]. Доступ: 09.02.2013.

15. Технология USSD: http://ru.eyeline.mobi/ussd/ussd-white-paper/ [Электронный ресурс]. Доступ: 09.02.2013.
16. USSD запросы в сети Мегафон Москва – технология USSD: http://blog.megafoncity.ru/2007/04/28/ussd-zaprosyi-v-seti-megafon-moskva-tehnologiya-ussd/ [Электронный ресурс]. Доступ: 28.04.2013.