ПРОГРАММИРОВАНИЕ, 2020, № 5, с. 16–24

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК 004.5

ПОДХОДЫ К РАЗРАБОТКЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

© 2020 г. В. Н. Лукин[a],*, А. Л. Дзюбенко[b],, Ю. Б. Чечиков[a,b],*****

[a] *Московский авиационный институт (национальный исследовательский университет)*
125993 Москва, Волоколамское шоссе, д. 4, Россия

[b] *Финансовый университет при Правительстве Российской Федерации*
125167 Москва, Ленинградский пр-т, д. 49, Россия

**E-mail: lukinvn@list.ru*

***E-mail: al_dz@list.ru*

****E-mail: yourych@mail.ru*

Поступила в редакцию 29.07.2019 г.
После доработки 01.10.2019 г.
Принята к публикации 01.11.2019 г.

В работе рассматриваются проблемы, связанные с современным пользовательским интерфейсом. Проводится анализ интерфейсов, построенных на парадигме реализации, и выявляются ее слабые стороны. Оцениваются существующие методологии разработки интерфейсов. Приводится обоснование использования метафорического или идиоматического подхода при создании (усовершенствовании) пользовательского интерфейса.

На ежегодном собрании Кеннет Олсен, инженер, основавший компанию Digital Equipment Corp. и в настоящее время руководящий ею, признался, что не знает, как приготовить кофе с помощью микроволновой печи компании.

DOI: 10.31857/S0132347420050052

1. ВВЕДЕНИЕ

Статья посвящена проблемам пользовательского интерфейса. Для того, чтобы программный продукт был востребован, он должен выполнять не только свои функции согласно техническому заданию [1], но и быть удобным и простым в использовании. Для этого должен быть тщательно спроектирован и реализован пользовательский интерфейс.

В распоряжении дизайнера интерфейса имеются разные подходы и методологии, но, наиболее проигрышная, на наш взгляд, методология, основанная на парадигме реализации.

Наиболее удачным, с точки зрения разработки комфортного пользовательского интерфейса, выглядит когнитивный подход, в рамках которого появились метафорическая и идиоматическая парадигмы. Они наиболее полно учитывают психологические и технологические требования пользователей, предъявляемые к интерфейсам, и именно в этом направлении целесообразно проводить дальнейшие исследования.

Создание удачного интерфейса требует изменения технологии разработки программного продукта, с учетом приоритетности качества взаимодействия, понимания важности комфорта пользователя руководством проекта. Следует опираться на результаты научных исследований [2–6] и на успешный практический опыт создания пользовательского интерфейса в области разработки прикладных программ, закрепленный отчасти в национальных стандартах [7–24].

Хорошие результаты могут получиться, если разработка пользовательского интерфейса, хотя бы на уровне макета, начинается сразу после анализа требований к системе, лучше до разработки программы, в крайнем случае, одновременно. Лучше, если в группу разработчиков включить профессиональных специалистов (не программистов!) [25], а, также, представителя заказчика, обладающего высокой квалификацией и административным ресурсом, чтобы вовремя высказать замечания и внести исправления в проект. На уровне технического задания необходимо детально прописать требования к интерфейсу. Это, помимо прочего, позволит более точно оценить время и стоимость разработки. Здесь же следует определить категории и количество групп пользователей, их цели при обработке информации. И, на протяжении всей работы по созданию интерфейса, необходимо обеспечить доминирова-

ние интересов пользователя над интересами программистов.

2. ИССЛЕДОВАНИЕ ВОПРОСА

Конкурентная борьба за увеличение продаж персональных компьютеров потребовала не только усовершенствования технических характеристик компьютеров и создания все более мощных операционных систем, но и разработки более совершенных средств взаимодействия пользователей и компьютеров. И, если на раннем этапе развития компьютерной индустрии, интерфейс предназначался для работы специально подготовленного инженерно-технического персонала, то в настоящее время, подавляющее число пользователей взаимодействует с вычислительным устройством, выполняя обычную работу. Таким образом, при проектировании интерфейса становится необходимостью учет интересов рядового пользователя [2, 7, 8, 26–28].

В связи с тем, что взаимодействие с персональным компьютером стало органичной частью работы пользователей, работающих в различных предметных областях, решение задач проектирования и реализации пользовательского интерфейса потребовало усилий специально подготовленных групп профессионалов. Соответственно, в крупных компаниях-разработчиках вычислительной техники и программного обеспечения появились подразделения, специализирующиеся на анализе потребностей пользователей программного обеспечения и создании эргономичного дизайна для решения ими своих профессиональных задач [10, 19]. Естественно, в ходе этих разработок возникает актуальная необходимость найти ответы на вопросы: “Как организовать взаимодействие пользователя с программными приложениями, чтобы он не ушел к конкуренту за более легким и удобным способом общения с компьютером? По каким критериям можно оценить качество интерфейса, и с каких позиций надо подходить к разработке успешного интерфейса?”

В период, когда в самом разгаре был инженерный подход к разработке программных систем, и программные комплексы разрабатывались в стиле индустриального материального производства, основными пользователями были операторы (мы не рассматриваем пласт научных и расчетных программ, пользователями которых были сами авторы). Это специалисты, обученные работе с аппаратно-программным обеспечением, они “прогоняли” программы согласно прилагаемым инструкциям, в которых расписывался сценарий деятельности. В некотором смысле, они были элементом технологического процесса, как рабочие в цеху или доярки на ферме. Разумеется, никому в голову не приходило особо заботиться о форме представления диалога (интерфейса), нужно было только для минимизации ошибок оператора заботиться об однозначности и корректности инструкции.

Таким образом, инженерно-технический (Machine-Centered) подход рассматривал процесс разработки с точки зрения функциональных возможностей компьютера, “изнутри-вовне”, когда сначала разрабатывается алгоритм, пишутся программы, затем проектируется ввод/вывод. По умолчанию предполагалось, что процесс решения задачи человеком подобен работе компьютера, только роль программы играла инструкция, которую следовало неукоснительно выполнять. Для достижения цели большая задача разбивалась на подцели, которые, при необходимости, также разбивались на подцели.

Этому подходу способствовало то, что тогда процесс решения сильно ограничивался техническими возможностями компьютеров, которые имели малый объем памяти и недостаточное быстродействие, так что все внимание разработчиков ЭВМ и программистов концентрировалось на усовершенствовании машин и повышении эффективности программ. Ресурсная эффективность считалась важнейшей характеристикой качества прикладной программы, поэтому из техники старались выжать все, что можно, и только потом принимались во внимание и учитывались адаптивные возможности человека. Надо сказать, они были достаточно высокими для компенсации слабого сопряжения человек-машина. Недостаток внимания к проблемам пользовательского интерфейса объясняется особенностью тогдашнего контингента пользователей, среди которых не было не подготовленных специально людей.

Программисты того времени неплохо разбирались в электронной технике, прекрасно ориентировались в существующем системном программном обеспечении и, волей-неволей, вырабатывали стиль профессионального мышления, сходный с алгоритмическим (стиль “центрального процессора”). Привычка мыслить в терминах машины помогала расшифровать специфические коды, играющие роль сообщения об ошибках: нередко это была комбинация неоновых лампочек, горящих на пульте. Понятно, они требовали такого же подхода и от операторов. Таким образом, инженерно-технический подход к созданию пользовательского интерфейса был основан на предположении, что человек работает с компьютером подобно самому компьютеру, следовательно, пользователь, независимо от профиля своей работы, должен думать как разработчик, а интерфейс, соответственно, ориентировался именно на функциональные характеристики программы.

Подобный подход и привел к созданию такой парадигмы разработки пользовательского интерфейса, как парадигма реализации, при которой перед пользователем стоит непростая задача:

разобраться в интерфейсе, не владея логикой построения программы.

Но, начиная с 1970-х годов, бурное развитие автоматизированных систем, в контуре управления которыми находился человек-оператор, показало, что игнорирование психофизиологических характеристик человека резко снижает эффективность систем и даже может быть причиной техногенных катастроф [21, 30–32].

Например, в середине 80-х годов была предпринята попытка автоматизировать одну из ведомственных поликлиник Мосгорисполкома. Работа была поручена крупному предприятию оборонного комплекса. Техническая сторона проекта была сделана грамотно: продумана архитектура сети, конфигурация рабочих мест врачей и медсестер, продублированы особо важные элементы системы.

С целью экономии народных средств, было решено не оснащать каждое рабочее место врача принтером, а поставить один качественный высокоскоростной принтер в холле каждого этажа. Именно это решение привело к тому, что, спустя месяц после запуска, система встала: медперсонал заявил, что отказывается от ее услуг, т.к. бегать за каждой бумажкой в холл и обратно невозможно, нет гарантии, что твой документ не унес кто-то другой и, поэтому, дешевле работать по старой ручной технологии.

Систему, конечно, переделали как положено, установив принтер на каждое рабочее место, но этот пример наглядно показывает, что даже технически грамотные, но не учитывающие интересы пользователей системы нежизнеспособны.

Таким образом, раз модель использования отражает подробности реализации программы в коде, понятно, что никто, кроме самого разработчика, не сможет ориентироваться в ней лучше него. Конструкция интерфейса полностью соответствует конструкции программы: по одной кнопке на каждую функцию, каждому модулю кода соответствует свое окно, внутренние структуры данных и алгоритмы порождают команды и процессы обработки данных пользователя.

Чтобы квалифицированно пользоваться таким интерфейсом, пользователь не только должен обладать профессиональными знаниями о том, как программа устроена, он должен думать, как программист и, фактически, быть программистом, что просто нереально. Получается, что таким образом интерфейс проектируется только для программистов.

Естественно, никто не обвиняет программистов в преднамеренном желании создавать сложные для использования интерфейсы. Существует достаточно причин объективного и субъективного характера, из-за которых программисты проектируют интерфейсы, сложные именно для неспециалистов.

Во-первых, это диктуют формулировки технического задания (ТЗ). В нем четко прописано, какие именно функции должна выполнять система. Для удобства ее сдачи заказчику программисту проще каждую функцию оформлять в виде кнопки или окна: тогда сразу видно, что конкретная функция реализована. Естественно, строить систему с интерфейсом, ориентированным на реализацию, проще всего: при создании новой функции добавляется фрагмент интерфейса, отражающий данную функцию. Если что-то не работает, легко выяснить, в чем причина, и быстро исправить ошибку.

Что касается требований ТЗ к интерфейсу, то, как правило, в ТЗ в списке требований к системе стоит фраза: “Система должна иметь удобный и эргономичный интерфейс”. А что конкретно стоит за этим требованием и как заказчик будет проверять его выполнение – неизвестно. Поэтому заказчик вынужден принимать тот интерфейс, который ему предложили, и принято считать, что этого достаточно.

Во-вторых, как показывает опыт разработки программных проектов, в большинстве случаев плановые сроки разработки системы нарушаются, и для того, чтобы сдать заказчику систему в срок, все силы и внимание в первую очередь направляются на разработку функционала, а не на создание удобного интерфейса, но с неполным функционалом.

Если при сдаче системы приоритетнее функционал, разработчики рассчитывают, что претензии к интерфейсу можно будет отнести к категории замечаний, которые выявились в последний момент и не были оговорены заранее, и есть возможность уговорить заказчика перенести усовершенствование интерфейса на более поздний период, когда будут исправляться выявленные ошибки. Но здесь может таиться серьезная опасность для системы. Если интерфейс привязан к функционалу, систему заказчику сдать можно, особенно если систему принимают не потенциальные пользователи, которые с ней потом будут работать. Но, если в приемке системы участвует контингент, заинтересованный в качестве взаимодействия, то для соответствия реальной технологии работы потребуется серьезное изменение интерфейса. Реализация замечаний может привести к изменению структуры информационной базы системы, что, в свою очередь, потребует изменения алгоритмов обработки информации и даже структуры всей системы в целом. А это уже практически создание новой системы, что по срокам и трудоемкости не идет ни в какое сравнение с исправлением ошибок. Поэтому, скорее всего, интерфейс немножко подправят, но его суть останется прежней – это будет интерфейс, основанный на реализации.

В-третьих, при проектировании имеет большое значение субъективная система ценностей разработчика. Самый важный и принципиальный вопрос, на который должен явно ответить разработчик – чьи интересы важнее, пользователя или разработчика. Будет ли разработчик приближать программные продукты к образу мышления пользователей или заставит пользователей подстраиваться под чуждую им машинную логику, облегчая себе жизнь? Как правило, разработчик облегчит жизнь себе, особенно, если нужно успеть сдать систему в срок. Конечно, интересы пользователей оказываются на последнем месте, и в результате неудобный интерфейс готов.

В-четвертых, играет роль структура и состав коллектива разработчиков программного продукта. Если руководитель проекта учитывает важность удобного интерфейса для пользователя, то, для нивелирования субъективной системы ценностей отдельных программистов, в коллектив должен быть включен специалист по дизайну интерфейса, обязанность которого – обеспечить качество взаимодействия с пользователями. Он должен иметь право блокировать выпуск некачественного интерфейса, что позволит защитить интересы пользователя в конфликтных ситуациях с программистами. Однако, подобные решения в результате могут не только повысить стоимость проекта, но и увеличить конфликтность в команде. Тем не менее, руководитель проекта должен обладать административной мудростью и управленческой твердостью, чтобы с самого начала всесторонне учитывать интересы пользователя и снижать напряженность в коллективе. К сожалению, такие руководители встречаются нечасто.

В-пятых, для обеспечения надежности изделия инженерам всегда нужно знать, как именно оно устроено и работает, поэтому парадигма реализации, которая помогает понять, что происходит внутри системы, их устраивает. Тот факт, что это неоправданно усложняют жизнь пользователей, отходит на второй план и кажется разработчикам несущественным побочным эффектом.

Критика данного подхода стала появляться в литературе с середины 80-х годов, когда накопился мировой опыт использования ЭВМ в различных сферах и появились осознанные рекомендации по выбору тех или иных интерфейсных решений [29, 32, 33]. Эти вопросы рассматривались такими исследователями, как Б. Шнейдерман [34], М. Минанси [35], Д. Норман [3], Гультяев А. [36]. В результате этих исследований пришло понимание того, что интересы пользователя и программиста часто противоречат друг другу, и, если в рамках программного проекта этой проблемой не заниматься, то программист решит ее удобным для себя способом.

К концу 80-х–началу 90-х годов в области проектирования пользовательских интерфейсов назрел методологический кризис, связанный с отсутствием ясного, единого понимания технологии их создания. Подход, основанный на теории деятельности, позволил проектировщикам найти путь из этого кризиса. Система “человек-компьютер” рассматривается в нем как комплекс деятельностных понятий и представлений. Этот подход (Activity-Centered Design, ACD) [11, 12, 28] близок к инженерно-технической парадигме реализации, но в нем учитываются интересы пользователей. Теория деятельности, лежащая в основе этого подхода, представляет компьютер уже в качестве инструмента, с помощью которого человек решает различные задачи, и здесь именно деятельность человека влияет на интерфейс.

Согласно принципам теории деятельности, выдвинутой советским психологом А.Н. Леонтьевым[1], человек рассматривается через призму того, как он взаимодействует с окружающим миром, и весь поток активности пользователя раскладывается на последовательность связанных задач и подзадач, логические этапы. Этот подход позволяет анализировать цели, внешние и внутренние задачи, порядок и вид операций пользователя, совершаемых для достижения итогового результата, и, по результатам анализа, разработать интерфейс, наиболее подходящий для данного вида деятельности.

В иерархии ACD деятельность состоит из задач, задачи состоят из действий, а действия составлены из операций. Такая структура упрощает программистам жизнь, так как, в результате, систему можно представить в виде набора функций, правда, уже не программных, а технологических, связанных с непосредственной деятельностью пользователей.

Схема ACD особо подчеркивает важность контекста пользователя. Метод ACD полезен при разделении на составные части того, что делает пользователь, но этот метод не отвечает на важный вопрос: почему пользователь приступает к этой активности, задаче, действию или операции? Он просто отражает технологию работы пользователя без анализа целей, которые преследует пользователь на каждом этапе работы.

Существенный недостаток подхода на основе анализа деятельности – невозможность анализировать сложные умственные виды деятельности пользователя, но более опасным видится риск повторить, а не модифицировать устаревшую ручную технологию работы пользователей при автоматизации предметной области [5, 37].

[1] А.Н. Леонтьев, Становление психологии деятельности, Москва, НПФ “Смысл”, 2003, 880 стр., илл.

Помимо прочего, стоит отметить, что системы с неудобным для заказчика интерфейсом имеют короткий жизненный цикл, а это проигрышно как для заказчика, так и для разработчика. Поэтому от подходов, приводящих к созданию неудобных для заказчика интерфейсов рациональнее отказываться.

Можно подумать, что в условиях, когда не было не только графического, но и даже цветного дисплея, добиться удобства взаимодействия просто невозможно. Но это не так. В середине 1980-х годов разрабатывалось довольно много систем с интерфейсом не только в инженерном стиле, но и в стиле деятельности, хотя достаточной теоретической базы еще не было. Некоторые из них были довольно удачными, для этого необходимо было внимательно изучить технологию работы потенциального пользователя и сделать так, чтобы ему было удобно. Так, в частности, разрабатывался интерфейс для системы Биолаб, предназначенной для автоматизации биохимической лаборатории Центральной клинической больницы [38]. Стиль взаимодействия был диалоговый, был реализован удобный и понятный пользователям язык общения с системой, отражающий все технологические процессы выполнения исследований, расчетов, отображения данных, выдачи документов. И, возможно, поэтому жизненный цикл системы длился 14 лет.

Но, даже в лучшем варианте, подходы со стороны программы или даже технологии, не дают пользователю достаточно свободы и не учитывают того, что называют психоэмоциональным комфортом пользователей [1, 13, 33, 39–41]. Возможно, для подготовленных и достаточно квалифицированных специалистов это не так важно, но при массовом использовании вычислительных средств на таких пользователей рассчитывать не приходится. Поэтому были предложены различные подходы к созданию пользовательского интерфейса, учитывающие этот самый комфорт [3, 4, 22, 34].

Что же конкретно предлагается взамен? Альтернатива – когнитивный подход или подход “извне-вовнутрь”, пришедший на смену алгоритмическому моделированию. Этот подход рассматривает пользователя как центральную фигуру процесса взаимодействия с системой. Конечно, такая возможность появилась с массовым производством персональных компьютеров, когда технические ограничения перестали быть определяющими, и можно было более полно учитывать психологические особенности человека и создавать более комфортные системы, учитывающие интересы пользователя – непрофессионала в области вычислительной техники. Ожидать от этих пользователей стремления адаптироваться под технику стало бессмысленным.

Действительно, пока компьютер был диковинкой, люди психологически были готовы адаптироваться к нему, а теперь появились все основания считать, что машину надо приспосабливать к человеку с помощью более тщательно сконструированного интерфейса.

Разработку, в таком случае, естественнее проводить по пути “извне-вовнутрь”. Извне находятся человек, дисплей, выводимая информация. Внутри – программа, алгоритм. Это подход ориентирован на пользователя, в настоящее время он господствующий. В общем случае, сначала определяются функции будущей системы, описывается диалог пользователя, готовится макет интерфейса, а уже потом под него пишется программа.

Рассматривая процессы и закономерности восприятия, переработки информации и принятия решения, когнитивная психология выявила факторы, определяющие успешность выполнения задачи оператором. И это оказались не функциональные характеристики системы, как предполагалось инженерами раньше, а качество предоставления и управления информацией с точки зрения возможностей и ограничений человека.

Ориентация на характеристики пользователя, исследование способностей к восприятию, когнитивных возможностей и ограничений человека, позволили выявить закономерности взаимодействия человека с автоматизированной системой. Однако, как оказалось, анализа только процессов восприятия и переработки информации человеком недостаточно для проектирования эргономичного интерфейса, поскольку он не позволяет определить состав и последовательность выводимой на экран информации.

В связи с этим, различные исследователи и организации-разработчики программного обеспечения предлагают для разработки интерфейса разные подходы. Так, появились методологии дизайна пользовательского интерфейса, основанные на когнитивном подходе, в частности, целеориентированный дизайн (Goal-oriented design) [37], дизайн, ориентированный на пользователя (User-Centered Design (UCD)) [14], эмоциональный дизайн [42, 44, 34] (рис. 1).

Рассмотрим эти методологии подробнее.

1. Целеориентированный дизайн (Goal-oriented design).

Эта методология разработки пользовательских интерфейсов предложена Аланом Купером [37], она основана на предположении о том, что тщательное изучение целей пользователя и понимание того, к чему он стремится, позволяет понять смысл его деятельности и, таким образом, создать более уместные и качественные продукты.

Цели побуждают людей вести некую деятельность; понимание целей позволяет понять ожидания и устремления пользователей, что, в свою

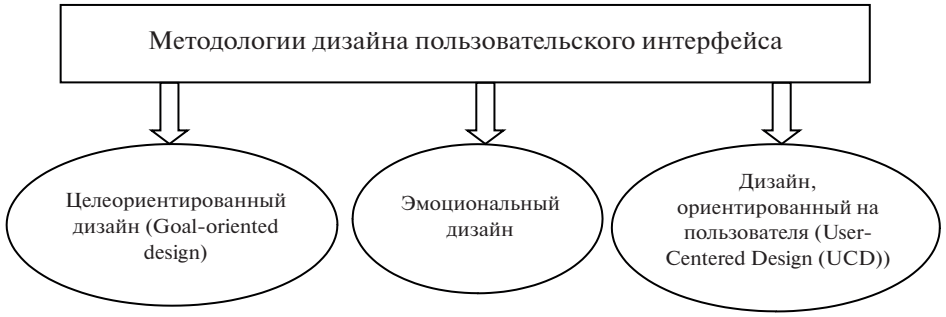

Рис. 1. Схема методологии дизайна пользовательского интерфейса.

очередь, помогает в определении видов деятельности, имеющих реальное отношение к дизайну интерфейса.

Цели определяются человеческими мотивами и, поэтому, со временем не меняются или меняются весьма незначительно. Деятельность и задачи преходящи, поскольку почти целиком основаны на имеющейся в данный момент технологии.

2. Дизайн, ориентированный на пользователя User-Centered Design (UCD) [14].

Суть этой методологии сводится к изучению потребностей и возможностей конечных пользователей и адаптации продукта под их нужды.

Принципы UCD:

• определение целевой аудитории – центральный этап разработки;

• понимание психологии пользователей, видение дизайна через призму взгляда представителей целевой аудитории;

• анализ информации о конкурентах;

• эволюционный дизайн, построенный на обратной связи с пользователями, которая поддерживается с помощью тестирования многочисленных прототипов и управляет процессом разработки;

Принципы UCD успешно реализуются на практике только при наличии постоянной тесной связи с целевой аудиторией. Как только совместно с заказчиком станет ясно, кто будет пользоваться продуктом, можно переходить к выяснению:

1) что представители целевой аудитории хотят от продукта?

2) что он должен делать для них?

3) в каких условиях они будут его использовать?

4) какие функции аналогичных продуктов конкурентов представители целевой аудитории используют чаще всего?

5) чего им не хватает в продуктах конкурентов?

На основе собранных входных данных проектировщики интерфейсов создают прототип, по которому дизайнеры разрабатывают предварительную версию. Версия тестируется представителями целевой аудитории. Их попытки выполнить пользовательские задачи с помощью прототипа, результаты, отзывы и впечатления ложатся в основу разработки. Затем проводится тестирование нового варианта, и, таким образом, процесс продолжается до того момента, пока рациональные предложения пользователей и их серьезные критические замечания не иссякнут.

Другими словами, это концепция создания программных продуктов, которыми люди хотели бы пользоваться.

3. В результате накопленного опыта разработки пользовательских интерфейсов появилась философия эмоционального дизайна, и произошло это в начале семидесятых годов прошлого века [42, 43]. Как показывает практика, зачастую, незначительные и, на первый взгляд, незаметные детали визуального дизайна вызывают у пользователей больший эмоциональный отклик, чем концепция системы в целом. Такого рода детали придают продукту неповторимую индивидуальность и помогают установить с пользователями связь на личностном уровне. Этот прием получил название “эмоциональный дизайн”.

Термин “эмоциональный дизайн” был введен (среди прочих) Аароном Уолтером[2]. В своей книге “Эмоциональный веб-дизайн” он описывает эмоциональный дизайн, используя иерархию человеческих потребностей А. Маслоу.

Воплощение философии эмоционального дизайна принадлежит японскому профессору Митсуо Нагамачи (Mitsuo Nagamachi), который предложил преобразовать эмоциональные переживания и ощущения человека в конкретные свойства продукта. В начале 90-х годов этот подход был применен при создании уникального японского автомобиля Mazda MX-5 Miata.

Впервые в разработке автомобиля на первое место поставили не технические характеристики (скорость разгона или количество лошадиных сил), а чувства, которые испытывает водитель автомобиля – нравится ли ему звук мотора, получа-

[2] http://aarronwalter.com/ ОРИГИНАЛЬНАЯ СТАТЬЯ: “THE PERSONALITY LAYER”.

ет ли он удовольствие от езды, от общения с машиной. Этот подход назвали кансей инжиниринг (kansei engineering) и он с успехом применяется во многих компаниях по всему миру в разных промышленных областях.

Таким образом, при создании продуктов важно учитывать не только конечные цели пользователей (ради чего они покупают продукт), но и их эмоциональные цели.

Эмоциональные цели выражают то, как человек хочет себя чувствовать, работая с продуктом. Когда продукт заставляет пользователей чувствовать себя глупо или неудобно, их самоуважение и производительность труда снижаются, а недовольство растет. Стоит чуть переборщить с таким отношением к пользователям – и они воспользуются первым же шансом, чтобы избавиться от этого продукта. Такой продукт потерпит неудачу – независимо от того, насколько хорошо он позволяет пользователям достигать всех прочих целей. В одной крупной организации, которая занималась анализом и прогнозированием загрузки авиарейсов в России, для этих целей была разработана удобная программная система. Со временем ее “ДОСовский” интерфейс показался одному из авторов устаревшим, и он заменил его современным “виндоувским”. Но пользователи отреагировали крайне отрицательно: у них стали сильно уставать глаза. Пришлось потратить время на проектирование интерфейса, адаптированного к удобной и не утомительной работе пользователя в течение рабочего дня.

Итак, делаем вывод, что при разработке программы, необходимо привести весь проект в соответствие с общими законами психологии, а не бездумно стремиться соответствовать так называемым общепринятым “промышленным стандартам”, которые, отчасти, построены без учета закономерностей мышления и поведения человека (например, раздвоение фокуса внимания, когда сообщение об ошибке и место, где произошла ошибка, находятся в разных частях экрана) [3].

Большая часть проблем, связанных с использованием компьютеров и подобных устройств, возникает, скорее, из-за низкого качества интерфейса, чем из-за сложности самой задачи или же недостатка старания или умственных способностей у пользователя [3, 25, 30, 35, 44–46].

Высшей степенью удобства считается ситуация, когда пользователь ощутил потребность в каком-либо ресурсе или сервисе и тут же появилась простая возможность легко эту потребность удовлетворить.

В результате эволюции когнитивного подхода появились еще две парадигмы интерфейсов: метафорическая и идиоматическая.

Метафорические интерфейсы основаны на интуитивных связях, которые пользователь устанавливает между визуальными элементами интерфейса и его функциональностью. Основная идеология этой парадигмы была заложена фирмой Xerox в конце 60-х годов прошлого века, а расцвет ее связан компьютерами компании Apple и появлением операционной системы Windows компании Microsoft.

По сравнению с парадигмой реализации, здесь пользователю не нужно разбираться в программировании. Пользователи применяют свою интуицию, чтобы через метафору понять предназначение объекта, а разработчики через метафоры структурируют интерфейс по понятным пользователям аналогиям.

Метафорическая парадигма была большим шагом вперед в построении интерфейсов, имеет массу достоинств и широко распространена в настоящее время. Но и здесь возникли проблемы: проявились ограничения и риски использования метафор. При увеличении количества объектов метафоры становятся неудобными, они с трудом подлежат масштабированию, для некоторых программных операций не удается найти метафору из реального мира, например, для изменения формата, а некоторые метафоры со временем перестают быть метафорами (изображение трехдюймовой дискеты для записи сейчас далеко не всем понятно).

Ответом на ограничения парадигм метафорической и реализации стало появление идиоматической парадигмы, которая сейчас развивается и широко исследуется.

Идиома – это свойственное только данному языку устойчивое словосочетание, значение которого не определяется значением входящих в него слов, взятых по отдельности (“собаку съесть”, “пускать пыль в глаза”, “залить на сервер”). Смысл идиомы нельзя понять путем анализа или интуитивно. У идиом нет аналогов в реальном мире, и, поэтому, сначала необходимо узнать смысл идиомы, выучить ее, и только потом использовать ее дальше. Многие известные нам элементы метафорического интерфейса на самом деле являются идиомами. Например, компьютерная мышь – это идиома, хотя ничто в реальной мыши не подсказывает нам, как надо использовать компьютерную мышь, и к тому же, в старых лингафонных кабинетах устройство, похожее на компьютерную мышь, было микрофоном. Но если один раз показать, как пользоваться компьютерной мышью – проблемы отпадут сами собой.

3. ЗАКЛЮЧЕНИЕ

Проблема создания пользовательского интерфейса, зародившись на заре компьютерной эры, продолжает оставаться на удивление актуальной. Нельзя не отметить общее повышение комфортности интерфейса, что связано и с развитием тех-

ники, и с пониманием запросов пользователей, и с развитием методов проектирования. Но жизнь идет вперед, появляются новые коммуникационные возможности и с ними – новые проблемы.

В заключение, хотелось бы привести слова известного специалиста в области разработки пользовательских интерфейсов компании Apple Алана Купера: “Без сомнения, все эти решения потребуют от программистов большей работы. Это нисколько меня не волнует. Я не хочу, чтобы программисты больше работали, но, выбирая из сложности работы программистов и сложности работы пользователей, я немедленно заставляю программистов работать. Работа программиста – удовлетворить пользователя, а не наоборот”.

СПИСОК ЛИТЕРАТУРЫ

1. *Андреев В.Н.* Примерное содержание технического задания по разработке пользовательского интерфейса и тезисов по ведению переговоров // Интернет-источник http://www.usability.ru/toader/articles.htm Центр практичных программ.
2. *Грачев Н.Н.* Психология инженерного труда. М.: Высшая школа, 1998. 331 с., рекомендация Минвуза.
3. Норман Д. Дизайн привычных вещей: Пер. с англ. М.: Издательский дом “Вильямс”, 2006. 384 с.: ил. Парал. тит. англ.
4. Основы инженерной психологии. М.: Высшая школа, 1986. 270 с., учебное пособие, рекомендация Минвуза.
5. *Сугак Екатерина*. Автореферат диссертации на соискание ученой степени кандидата психологических наук (специальность 19.00.03-психология труда, инженерная психология) на тему “Эргономические аспекты проектирования пользовательского интерфейса”.
6. Хрестоматия по инженерной психологии /Сост.: Б.А. Душков, Б.Ф. Ломов, Б.А. Смирнов / Под ред. Б.А. Душкова. Уч. пособие. М.: Высшая школа, 1991. 287 с.
7. ISO 9241-10-1996 Ergonomic requirements for office work with visual display terminals (VDTs). P. 10. Dialogue principies.
8. ISO 9241-12-1998 Ergonomic requirements for office work with visual display terminals (VDTs). P. 12. Presentation of information.
9. ISO 9241-16-1998 Ergonomic requirements for office work with visual display terminals (VDTs). P. 16. Direct manipulation dialogues.
10. *Арлов Л.* Как создать хороший интерфейс пользователя? // Интернет-источник http://www.usability.ru/toader/articles.htm Центр практичных программ.
11. *Батенькина О.В.* Дизайн пользовательского интерфейса информационных систем: учеб. пособие. Омск: Изд-во ОмГТУ, 2014. 112 с.
12. *Гарретт Дж.* Веб-дизайн: книга Джесса Гарретта. Элементы опыта взаимодействия. Пер. с англ. СПб.: Символ-Плюс, 2008. 192 с.: ил.
13. *Герасимов Ю.* Улетный интерфейс // Интернет-источник
14. *Головач В.* Книги и статьи о пользовательских интерфейсах Электронный ресурс. [2018]. Режим доступа: http://www.usethics.ru/lib
15. *Головач В.* Юзабилити-тестирование по дешевке. Электронный ресурс. [2018]. Режим доступа: https://medium.com/usethics-doc/юзабилити-тестирование-по-дешевке-2e853250960f
16. *Головач В.В.* Дизайн пользовательского интерфейса // Интернет-источник http://www.uibook1.ru, 146 с. Центр практичных программ
17. ГОСТ Р 56274-2014. Общие показатели и требования в эргономике. Национальный стандарт Российской Федерации: изд. офиц.: введен 2016-01-01 / НИИ экономики связи и информатики “Интерэкомс”. Москва: Стандартинформ, 2015. IV. 26 с.
18. ГОСТ Р ИСО МЭК 9126-93 Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.
19. ГОСТ Р ИСО/МЭК 12119-2000 Информационная технология. Пакеты программ. Требования к уровню качества и тестирование.
20. *Нильсен Я.* Элементарные основы юзабилити. Электронный ресурс. [2018]. Режим доступа https://www.promo-webcom.by/analytics/usability/1429-elementarnyie-osnovyi-yuzabiliti/
21. *Седельников А.* Пять распространенных ошибок при разработке интерфейсов программ // Интернет-источник http://www.usability.ru/toader/mycolumn/five_errors.htm Центр практичных программ
22. *Соловьев С.В., Цой Р.И., Гринкруг Л.С.* Технология разработки прикладного программного обеспечения. Издательство: Академия Естествознания 2011 https://monographies.ru/en/book/view?id=141
23. *Сполский Дж.* Программистам о разработке пользовательских интерфейсов // Интернет-источник http://www.usability.ru/toader/articles/uid4p_1.htm Центр практичных программ
24. Человеко-машинный интерфейс. Правила организации http://genew.ru/cheloveko-mashinnij-interfejs-pravila-organizacii.html?page=6.
25. *Платт Д.* Софт-отстой и что с этим делать. Симбо, С. Петербург-Москва, 2008.
26. *Донской М.* Пользовательский интерфейс // Интернет-источник http://www.usability.ru/toader/articles/user_interface.htm Центр практичных программ
27. Интеллект человека и программы ЭВМ. Под ред. О.К. Тихомирова. М., 1979. С. 230.
28. *Лукин В.Н., Зотова А.А.* Пользовательский интерфейс: история и современность. “Новые информационные технологии”. Тезисы докладов XV Международной студенческой школы-семинара М.: МИЭМ, 2007. С. 50–55.
29. *Денинг В., Эссиг Г., Маас С.* Диалоговые системы “Человек-ЭВМ”. Адаптация к требованиям пользователя. М.: Мир, 1984. 112 с., научное издание.

30. *Раскин Дж.* “Интерфейс: новые направления в проектировании компьютерных систем”: Символ-Плюс; М.; 2005. 161 с., ил.

31. *Шнейдерман Б.* Психология программирования: Человеческие факторы в вычислительных и информационных системах. М.: Радио и связь, 1984. 303 с., рекомендация Минвуза.

32. Эмоциональный дизайн с примерами, перевод материала Smashing magazine, https://naikom.ru/blog/archives/6382

33. *Коутс Р., Влейминк И.* Интерфейс “Человек-Компьютер”. М.: Мир, 1990. 501 с., научное издание.

34. *Шнейдерман Бен.* Разработка пользовательского интерфейса: стратегии эффективного взаимодействия человека и компьютера, 1-е издание. Addison-Wesley, 1986; 2-е изд. 1992; 3-е изд. 1998; 4-е изд. 2005; 5-е изд. 2010; 6-е изд., 2016.

35. *Минанси М.* Графический интерфейс пользователя. Секреты проектирования. М.: Мир, 1996 г., 160 с., научное издание.

36. *Гультяев А., Машин И.* Проектирование и дизайн пользовательского интерфейса. СПб.: Корона принт, 2000. 352 с.

37. *Купер А., Рейман Р., Кронин Д.* Алан Купер об интерфейсе. Основы проектирования взаимодействия. Пер. с англ. СПб.: Символ Плюс, 2010. 688 с., ил.

38. *Лукин В.Н., Скобеева М.В., Чечиков Ю.Б. и др.* Автоматизированная биохимическая лаборатория. Перспективное развитие // Лабораторное дело. 1989. № 9. С. 26–28.

39. *Лукин В.В., Лукин В.Н., Лукин Т.В.* Технология разработки программного обеспечения. Уч. пособие, 4-е изд. М.: Вузовская книга, 2019. С. 294.

40. *Вятчин К.* Определение пользовательских профилей. Электронный ресурс. [2018]. Режим доступа: http://www.sdteam.com/t15690

41. *Латышев В.Л., Клыпина И.А.* Представление информации в системе “Человек-компьютер”. М.: Изд. МАИ, 1993. 22 с. Уч. пособие

42. *Кузнецов А.* Эмоциональный дизайн или тайна четвертой волны. Электронный ресурс. [2017]. Режим доступа: https://uexpert.ru/emotsionalnyj-dizajn-ili-tajna-chetvyortoj-volny/

43. *Латышев В.Л., Клыпина И.А.* Представление информации в системе “Человек-компьютер”. М.: Изд. МАИ, 1993. 22 с. Уч. пособие

44. *Уолтер А.* Эмоциональный веб-дизайн. Пер. с англ. Изд.: Манн, Иванов и Фербер, 2012. 93 с.: ил.

45. *Tognazzini B.* Вежливый интерфейс или принципы создания диалогов // Интернет-источник http://www.usability.ru/toader/articles.htm Центр практичных программ

46. *Нильсен Я.* Веб-дизайн. Книга Якоба Нильсена. Пер. с англ. СПб.: Символ-Плюс, 2003. 504 с.: ил.

47. *Эндрю вэн Дам* Пользовательские интерфейсы нового поколения // Открытые системы. 1997. № 6.