АЛАН КУПЕР

ПСИХБОЛЬНИЦА В РУКАХ ПАЦИЕНТОВ

АЛАН КУПЕР ОБ ИНТЕРФЕЙСАХ

Алан Купер

Психбольница в руках пациентов. Алан Купер об интерфейсах

Серия «Библиотека программиста (Питер)»

epub предоставлен правообладателем
http://www.litres.ru/pages/biblio_book/?art=31736385
Издательский дом "Питер"; 2018
ISBN 978-5-4461-0674-5

Аннотация

Все мы – безумцы, живущие в технологическом сумасшедшем доме, и создали этот безумный мир мы сами. Своими руками сотворили этот кошмар: интерфейсы, которые нас раздражают и утомляют глаза, устройства, которые приводят к болям в спине и в запястьях. Эта книга стала манифестом и до сих пор не потеряла актуальность. Дверь на свободу распахнута. Почему же мы не замечаем выхода? Об этом и рассказывает Алан Купер, объясняя разницу между интерфейсом и взаимодействием.

Эй, ребята, у вас тут полно обозленных клиентов. Вам есть что им ответить?

В форматах a4.pdf и ios.epub сохранены издательские макеты.

Содержание

обеспечению тоже нелегко

Конец ознакомительного фрагмента.

А. Купер
Психбольница в руках пациентов. Алан Купер об интерфейсах
или Почему высокие технологии сводят нас с ума и как восстановить душевное равновесие

Alan Cooper
THE INMATES ARE RUNNING THE ASYLUM:
Whe High-Tech Products Drive Us Crazy And How to Restore the Sanity

Информация, содержащаяся в данной книге, получена из

источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги.

Серия «Библиотека программиста»

* * *

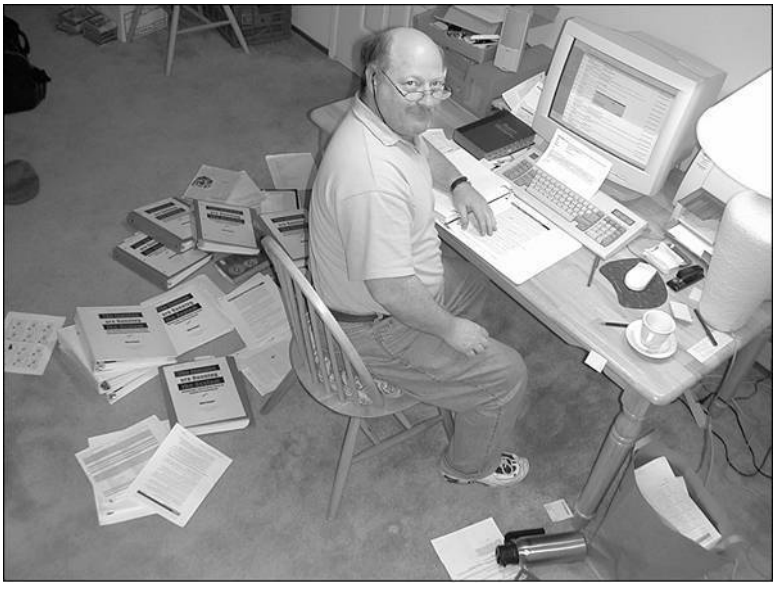
Сью, Скотту и Марти, с любовью

Благодарности

Этой книги не было бы без помощи и поддержки многих моих замечательных друзей и коллег. Некоторые из них, в частности, выполняли сложную работу со множеством требований – вычитывали и комментировали рукопись, иногда даже по несколько раз. Благодаря их комментариям я отвечал на нелегкие вопросы, предлагал новые темы, резюмировал свои убеждения, усмирял свой пыл и брал в узду свое возмущение. Еще лучше этой книге помогли стать Ким Гудвин (Kim Goodwin), Лейн Хэлли (Lane Halley), Келли Боуман (Kelly Bowman), Скотт Макгрегор (Scott McGregor), Дэвид Уэст (David West), Майк Нельсон (Mike Nelson), Марк Джирск (Mark Dziersk), Алан Карп (Alan Karp), Терри Суок (Terry Swack), Луи Вейцман (Louie Weitzman), Уэйн Гринвуд (Wayne Greenwood), Райан Ольшавски (Ryan Olshavsky), Джон Майер (John Meyer), Лайза Сондерс (Lisa Saunders), Винни Шоуз (Winnie Shows), Кевин Вандрайк (Kevin Wandryk), Глен Халстед (Glenn Halstead), Брайан О'Салливан (Bryan O'Sullivan), Чак Оуэн (Chuck Owen), Майк Суэйн (Mike Swaine), а также Скип Уолтер (Skip Walter). Я благодарен всем вам за ваше время, поддержку и мудрость. Особую ценность для меня также возымели советы и комментарии Джонатана Кормана (Jonathan Korman) – с их помощью я смог лучше выделить темы для книги. Я

также должен поблагодарить всех невероятно талантливых и трудолюбивых сотрудников из *Cooper Interaction Design*, которые взяли мои обязанности на себя, пока я занимался подготовкой книги. Особую благодарность выражаю директору по проектированию Уэйну Гринвуду (Wayne Greenwood), которому пришлось работать в жестких стрессовых условиях, сохраняя при этом высокое качество наших проектов и не утрачивая боевой дух.

Одним из самых интересных вызовов при создании этой книги стала задача подготовки иллюстраций. Чед Кубо (Chad Kubo), гениальный создатель иллюстраций, проделал невероятную работу по претворению моих смутных задумок в четкие и запоминающиеся изображения. Благодаря им книга заиграла другими красками. Но этих иллюстраций не было бы без интенсивного руководства художественной частью в исполнении Пенни Бейлес (Penny Bayless) и Дэвида Хейла (David Hale). Другие специалисты помогали с решением производственных задач. Я благодарю Брит Кацен (Brit Katzen) за выверку фактов и исследования, а также Майка Генри (Mike Henry) за редактуру текста.



Создание книги – это бизнес-проект, и за его успех я должен выразить благодарность моей команде подкованных в технологиях предпринимателей, во главе которой стоял мой агент Джим Левайн (Jim Levine), а саму команду составляли Глен Халстед (Glenn Halstead), Линн Боуман (Lynne Bowman), Келли Боуман (Kelly Bowman) и Сью Купер (Sue Cooper). Также поддержку по ходу всего проекта оказывало и издательство *Pearson* в лице Брэда Джоунса (Brad Jones), но самые большие благодарности я хотел бы высказать Крису Веббу (Chris Webb) за его целеустремленность, высочай-

шую концентрацию и тяжелый труд, без которых эта книга вообще бы не состоялась.

Для меня невероятно ценно внимание всех тех людей, что оказывали мне моральную поддержку, а также снабжали меня историями, советами и тратили на проект время. Огромное спасибо таким людям, как Дэниел Эпплман (Daniel Appleman), Тодд Баше (Todd Basche), Крис Байер (Chris Bauer), Джефф Безос (Jeff Bezos), Элис Блэр (Alice Blair), Мишель Борк (Michel Bourque), По Бронсон (Po Bronson), Стив Калде (Steve Calde), Дэвид Карлик (David Carlick), Джефф Карлик (Jeff Carlick), Кэрол Кристи (Carol Christie), Клэй Коллье (Clay Collier), Кендалл Косби (Kendall Cosby), Дэн Крэйн (Dan Crane), Роберт Крингели (Robert X. Cringely), Трой Дэниелс (Troy Daniels), Лайза Пауэрс (Lisa Powers), Филип Энглхарт (Philip Englehardt), Карен Ивенсен (Karen Evensen), Риджели Эверс (Ridgely Evers), Ройял Фаррос (Royal Farros), Пэт Флек (Pat Fleck), Дэвид Фор (David Fore), Эд Форман (Ed Forman), Эд Фридкин (Ed Fredkin), Жан-Луи Гассе (Jean-Louis Gassee), Джим Гэй (Jim Gay), Расс Голдин (Russ Goldin), Влад Горелик (Vlad Gorelik), Марсия Грегори (Marcia Gregory), Гаррент Грюнер (Garrett Gruener), Чак Хартледж (Chuck Hartledge), Тед Харвуд (Ted Harwood), Уилл Хирст (Will Hearst), Тамра Хизершоу-Харт (Tamra Heathershaw-Hart), Джей-Ди Хильдебранд (JD Hildebrand), Лори Хиллз (Laurie Hills), Питер Хиршберг (Peter Hirshberg), Ларри Кили (Larry Keeley), Гэ-

ри Краткин (Gary Kratkin), Дебора Курата (Deborah Kurata), Том Лефлер (Tom Lafleur), Пол Лотон (Paul Laughton), Элен Леви (Ellen Levy), Стивен Лист (Steven List), Ти-Си Манган (TC Mangan), Дэвид Майстер (David Maister), Роберт Мэй (Robert May), Дон Маккини (Don McKinney), Кэтрин Медоуз (Kathryn Meadows), Лайза Митчелл (Lisa Mitchell), Джеффри Мур (Geoffrey Moore), Брюс Мовери (Bruce Mowery), Нат Майерс (Nate Myers), Эд Нихаус (Ed Niehaus), Констанс Петерсен (Constance Petersen), Кит Плис (Keith Pleas), Роберт Райнман (Robert Reimann), Джон Ривлин (John Rivlin), Говард Рейнгольд (Howard Rheingold), Хайди Ройцен (Heidi Roizen), Нил Рубенкинг (Neil Rubenking), Пол Сафо (Paul Saffo), Джош Зайден (Josh Seiden), Расс Зигельман (Russ Siegelman), Донна Слоут (Donna Slote), Линда Стоун (Linda Stone), Тони Уокер (Toni Walker), Кевин Уикс (Kevin Weeks), Кевин Уэлш (Kevin Welch), Дэн Уиллис (Dan Willis), Хизер Уинкл (Heather Winkle), Стефан Уайлдстром (Stephen Wildstrom), Терри Виноград (Terry Winograd), Джон Цикер (John Zicker) и Пьерлуиджи Заппакоста (Pierluigi Zappacosta).

Этот проект «сроком на год» занял двадцать месяцев, и моя семья проявила величайшее терпение. Я в большом долгу любви и благодарности перед моей супругой Сью Купер и моими прекрасными юными сыновьями Скоттом и Марти. Люблю вас всем сердцем.

Предисловие

Спасайся кто может – мир захватывают компьютеры. Великолепные, мощные, они справляются с еще более важными задачами посредством ужасных, старомодных интерфейсов. Все больше проникая во все сферы жизни человека, они будут злить, раздражать нас, а некоторых способны даже убить. В ответ мы тоже будем безумно желать уничтожить наши компьютеры, но не осмелимся, потому что мы уже так сильно, неразрывно привязаны к ним, к этим монстрам, обещающим нам столь многое и без которых современная жизнь уже немыслима.

К счастью, выход есть. Мы должны фундаментально переосмыслить взаимодействие компьютера и человека. Изменить эти взаимоотношения на более глубоком уровне, применяя новаторские подходы, потому что причина наших все возрастающих проблем кроется вовсе не в машинах, а внутри нас самих. Именно люди создали интерфейсы, которые нас так раздражают; люди продолжают использовать ставшие нефункциональными машины, невзирая на то, что эти ужасные интерфейсы утомляют глаза, приводят к боли в спине и вредят сухожилиям в запястьях. Как следует из названия этой книги, все мы стали безумцами, обитающими в технологическом сумасшедшем доме, и создали этот безумный мир мы сами.

Эта книга – наш путь к спасению. А точнее, целью Алана Купера было показать, что дверь на свободу и без того широко раскрыта. Мы вольны выйти в любой момент, но, ослепленные безумием, мы до сих пор не замечали выхода. Весь секрет в том, чтобы переопределить способ, которым мы взаимодействуем с компьютерами, и сделать это в более широком контексте.

Сам Алан Купер – не просто такой же безумец, как и все мы; он еще и еретик, чьи взгляды способны разъярить тех, кто намерен держать нас в заключении. А желают этого инженеры, которые и разрабатывают столь ненавистные нам системы и по-прежнему придерживаются мнения, что единственный выход из этого ужаса – еще больше совершенствовать интерфейсы. Только сама концепция *интерфейса* – это реликвия из той поры, когда компьютеры были не столь распространены, их производительность была слабой, а сами они не слишком эффективно взаимодействовали с людьми, которые ими управляли. Интерфейсы были полезны, когда все взаимодействие с пользователем происходило через неизведанную территорию на стеклянном экране компьютера. Теперь же, в эпоху, когда компьютеры проникают в каждый уголок нашей жизни, мыслить категориями интерфейсов становится даже опасным. Компьютеры больше не используют интерфейсы лишь в качестве «прослойки» между пользователем и их внутренним устройством – теперь они взаимодействуют с людьми, и такое взаимодействие бу-

дет лишь переходить на более глубокий уровень, становиться все более незаметным и более критичным для здравомыслия нашего общества и всеобщего выживания.

Алану Куперу разница между интерфейсом и взаимодействием знакома как никому другому. Его идеи произрастают из многолетнего опыта по содействию в проектировании продуктов, мягко и ненавязчиво заполоняющих мир вокруг нас. Он апробировал свои идеи на практике в течение долгих лет, и теперь, наконец, у него появилось время преобразовать полученный практический опыт в четкое описание того вызова, с которым мы столкнулись, в методологию побега из сумасшедшего дома, который мы воздвигали с такой любовью. Читайте эту книгу, и вы станете свободными.

Пол Саффо (Paul Saffo), директор Института будущего

Предисловие к оригинальному изданию

Книга-обоснование

Изначально я собирался написать совершенно другую книгу – книгу-руководство о процессе проектирования взаимодействия. В мае 1997 года, во время поездки в Тоскану двое моих друзей, Дон Маккини (Don McKinney) и Дэйв Карлик (Dave Carlick), уговорили меня написать книгу, которую вы держите в руках. Они убедили меня, что следует, прежде всего, обращаться к деловой аудитории.

Они знали о моем намерении написать руководство, но сомневались в востребованности книги о проектировании взаимодействия (interaction design). Они хотели, чтобы я написал книгу о том, насколько важен и нужен этот процесс. Предложение друзей было, конечно, интересным, но я не был уверен, что смогу.

Однажды поздней ночью на веранде нашей общей виллы с видом на Флоренцию у меня состоялся серьезный разговор с Дэйвом и Доном. На столе несколько пустых бутылок из-под кьянти и остатки хлеба, сыра и оливок. Сияют звезды, светлячки танцуют на лужайке, а вдалеке подмигивают огни древних куполов столицы Тосканы. Дэйв снова предложил мне отказаться от идеи книги-руководства и вместо этого «дать деловое обоснование проектированию взаимодействия».

Я энергично запротестовал: «Дэйв, я же не знаю, как пи-

сать такую книгу». И начал загибать пальцы: «Мне придется объяснять, насколько мерзок тот процесс разработки, что существует сейчас, как компании теряют деньги на неэффективном создании программного обеспечения, как ненадежны неудовлетворенные клиенты и как все эти проблемы способен разрешить более совершенный процесс проектирования».

Дэйв перебил меня: «Алан, это называется главами».

Его ответ лишил меня сил сопротивляться дальше. Я вдруг осознал, что повторяю избитый сценарий и что Дэйв прав. Книга, содержащая «бизнес-обоснование», необходима и более своевременна, чем книга-руководство. Дэйв и Дон убедили меня, что я действительно способен написать такую книгу.

Инженер, разбирающийся в бизнесе, либо бизнесмен, разбирающийся в технологиях

Успешный профессионал XXI века – это либо инженер, разбирающийся в бизнесе, либо бизнесмен, разбирающийся в технологиях, и именно такому человеку адресована моя книга.

Бизнесмен, разбирающийся в технологиях, понимает, что его успех зависит от качества доступной ему информации и его умения воспользоваться этой информацией. Инженер, разбирающийся в бизнесе, – это предприимчивый конструктор или ученый, специализирующийся на технологиях, но при этом обладающий деловой хваткой и осознающий, какая огромная сила заключена в информации. Оба новых архетипа будут доминировать в современном бизнесе.

Всех деловых людей можно разделить на две категории: на тех, кто овладеет высокими технологиями, и тех, кто скоро выйдет из бизнеса. Руководители больше не могут делегировать обработку информации специалистам, ведь бизнес и есть обработка информации. Сегодня вы можете выделиться качеством своих систем обработки информации, но не качеством систем производства. Если вы производите что-либо, скорее всего, в вашем продукте есть микросхема. Если

вы предлагаете услугу, то, скорее всего, вы используете компьютеризованные инструменты. Попытка определить, какие предприятия зависят от высоких технологий, столь же глупая затея, как попытка определить, какие предприятия зависят от телефона. Революция высоких технологий затронула все сферы деловой деятельности, а цифровая информация стала основой ваших трудовых будней.

Кто-то сказал: «Человеку свойственно ошибаться, но чтобы провалить дело капитально, нужен компьютер». Неэффективные механические системы способны приносить незначительные убытки на каждой производимой детали, однако из-за некачественных информационных процессов можно потерять целую компанию. Влияние на вашу компанию продуктов, основанных на программном обеспечении, как и влияние инженеров, создающих эти продукты, огромно.

К сожалению, наши цифровые средства сложны для изучения, понимания и применения; они часто препятствуют достижению наших целей. Мы теряем деньги, время, возможности. Будучи инженером, понимающем в бизнесе, или же бизнесменом, понимающем в технологиях, вы создаете или потребляете продукты, основанные на программном обеспечении, а вернее всего, потребляете и создаете одновременно. Обладание более совершенными, более простыми в освоении и применении высокотехнологичными продуктами – в ваших личных и профессиональных интересах.

Более качественные продукты не требуют больших временных и денежных затрат на свое создание. Ирония в том, что они не должны быть сложными и являются таковыми лишь потому, что устарели процессы их создания. Лишь давние традиции, основанные на заблуждениях, мешают нам сегодня получить более качественный продукт. Эта книга покажет вам, как можно требовать и получать лучшие продукты – продукты, которых вы заслуживаете.

Идея этой книги незамысловата: мы можем создавать мощные и приятные продукты, основанные на программном обеспечении, используя простой прием: *проектируя* взаимодействие продукта с пользователем *до* этапа программирования. Сегодня мы этого не делаем. Проектирование интерактивных продуктов, использующих программное обеспечение, – специализация столь же сложная, как собственно разработка.

* * *

Сделав выбор в пользу книги-обоснования и отказавшись от книги-руководства, я прошу прощения у каждого проектировщика взаимодействия, который возьмет в руки эту книгу. Из уважения к деловой части аудитории она содержит лишь краткие отступления в область практической методологии проектирования взаимодействий (в основном в главах части IV). Я включил лишь необходимое количество инфор-

мации, чтобы показать, что такая методология существует, что она применима в любой предметной области и что ее преимущества очевидны всем, независимо от технических знаний.

Алан Купер,
Пало-Альто, Калифорния
www.cooper.com
inmates@cooper.com

Предисловие ко второму изданию

Недавно я познакомился с топ-менеджером одной из крупнейших технологических компаний мира. Официальное название его должности – вице-президент по вопросам простоты использования, и он несет ответственность за очень многие программные продукты, как небольшие, так и крупные. Это выдающийся и состоявшийся человек, выходец из сообщества HCI (Human-Computer Interaction, взаимодействие человека и компьютера), где принят формализованный подход. Он, как и его компания в целом, искушен в «юзабилити» – в тестировании и наблюдении из-за односторонних зеркал. Однако он пришел поговорить со мной не о тестировании, а о проектировании, и не о пользователях, а о персонах. Он рассказал, что в его компании полностью прекращено тестирование юзабилити на завершающей стадии разработки; вместо этого усилия прикладываются до начала разработки – при проектировании. Более того, он упомянул, что все его люди, умеющие и привыкшие наблюдать за пользователями в искусственных условиях, проходят переподготовку и будут заниматься этнографическими исследованиями в «полевых» условиях.

Этот топ-менеджер и его компания – символ тех огромных изменений, которые произошли в отрасли за пять коротких лет с момента выхода в 1999 году первого издания книги

The Inmates Are Running the Asylum («Психбольница в руках пациентов»). Книга стала одновременно и манифестом для революции, и учебником. Не сосчитать, сколько писем я получил от менеджеров среднего звена, в которых рассказывалось, как после прочтения книги они покупали по экземпляру каждому из своих руководителей. Между тем разработчики программного обеспечения и университеты восприняли три главы части IV как исходный материал для руководства «сделай сам» по претворению в жизнь целеориентированного (Goal-Directed®) проектирования с использованием персон.

Я глубоко признателен всем менеджерам, программистам, руководителям и специалистам по юзабилити за то, что они воспользовались моими идеями и вывели юзабилити из лаборатории в жизнь, сместили фокус с тестирования на проектирование. Благодаря их усилиям полностью изменилась профессия специалиста по юзабилити. Сегодня в большинстве организаций, с которыми я контактирую, на полной ставке работают один или несколько профессиональных проектировщиков взаимодействия, и влияние этих людей на качество и поведение программных продуктов и услуг постоянно растет. Мне приятно осознавать, что эта книга способствовала их успеху.

Помню, как читал программную речь на конференции программистов в 1999 году, вскоре после опубликования книги. Название речи совпадало с названием книги, и свою

речь я начал со слов, что «психбольница находится в руках пациентов, и эти пациенты – вы». В аудитории воцарилась гробовая тишина: 2500 инженеров пытались осмыслить мое обвинение. Я продолжил представлять основные идеи книги, и час спустя это сборище *Хомо логикус* настолько проникло моими аргументами, что мне аплодировали стоя. Удивительно, но большинство программистов с энтузиазмом восприняли идеи проектирования и проектировщиков взаимодействия. Они понимают, что нуждаются в помощи там, где речь идет о человеческой стороне конструирования программ, и счастливы получить, наконец, некоторые полезные указания. Программисты признают, что любая практика, повышающая качество и успех программ, не является для них угрозой.

В прошлом руководители считали проектирование взаимодействия задачей программирования и делегировали ее решение программистам, которые прилежно трудились над ней, хотя их опыт, подготовка, образ мышления и рабочий график не позволяли добиться успеха. В духе диагностики проблем эта книга подробно описывает такой провал, который всегда оказывается провалом программиста. Некоторые из программистов восприняли книгу враждебно, подумав, будто я злословлю или пытаюсь переложить на программистов вину за некачественный софт. Определенно, некачественные программы создаются при их *участии*, но они никоим образом не заслуживают осуждения. Я не виню про-

граммистов за сложные в использовании программы, и мне очень жаль, что у некоторых из них сложилось обратное впечатление. За некоторыми исключениями, знакомые мне программисты прилежны и добросовестны в своем желании угодить конечным пользователям, и стремятся неустанно повышать качество программ. Подобно пользователям, программисты – лишь жертвы несовершенного процесса, характеризующегося цейтнотом, противоречивыми указаниями и недостаточно эффективным руководством. Мне очень жаль, если у кого-то из программистов сложилось впечатление, будто я их обвиняю.

Жесткий процесс создания программ, в особенности высокая стоимость программирования и низкое качество взаимодействия, – проблема, попросту говоря, не технического плана. Это результат применения бизнес-практик в том направлении, в котором они устарели, – в разработке программ. С чистым сердцем, благими намерениями и с благословения руководства программисты пытаются решить эту проблему инженерным путем. Но работать интенсивнее здесь не помогает. Программисты ощущают все большую тщетность своих усилий, и их отчаяние нарастает.

Из нескольких своих недавних поездок я сделал вывод, что тревога в сообществе программистов нарастает. К сожалению, хуже всего чувствуют себя лучшие и опытные программисты. Они прикладывают титанические усилия, но излучают цинизм и впадают в тоску, понимая, что их умения

растрачиваются попусту. Они могут и не знать точно, как именно получается, что их квалификация не находит правильного применения, но они не могут не видеть очевидного. Многие из лучших программистов вообще перестали программировать, поскольку работа раздражает их. Они ушли в преподавание, религию, писательство, консультационную сферу, потому что эти занятия не оставляют ощущения пустой траты времени и сил. Но этих трагических потерь вполне можно избежать. (В некотором смысле движение свободных программ с открытым кодом можно назвать раем для этих отчаявшихся программистов – тут они могут писать код по своим стандартам и быть судимыми только равными, не выслушивая советы от маркетологов или руководителей.)

Программистам не хватает времени, четких указаний и адекватных планов, позволяющих добиться успеха. Эти три кита – на совести руководителей, именно они не дают всего этого программистам, причем вовсе не по злому умыслу или по глупости, а по причинам, которых можно было бы избежать. Они просто не вооружены адекватными инструментами, позволяющими решать сложные и уникальные проблемы информационной эпохи. Ну вот, звучит так, будто я снова кого-то критикую, но на этот раз в мой прицел попали бизнесмены, а не программисты. Повторюсь, чтобы решить проблему, ее следует сначала разобрать на составляющие. Я ищу решения, а не козлов отпущения.

Мудрый руководитель Питер Друкер (Peter Drucker) в

свои девяносто два года, большую часть которых он провел, направляя действия руководителей, смотрит на эту проблему со своей, уникальной точки зрения. В недавнем интервью журналу *CIO* он упомянул о наивном оптимизме руководителей в 1950-е – 1960-е годы, когда компьютеры только прокладывали дорогу в деловой мир. Эти руководители думали, что компьютеры «окажут огромное воздействие на способы ведения бизнеса», но Друкер утверждает: «Но так не случилось. Очень немногие руководители задавали вопрос: „Какая информация мне требуется для выполнения данной работы?“» Компьютеры дали руководителям небывалые объемы данных, но лишь немногие поинтересовались, подходят ли эти данные для управления корпорацией. Образ существования бизнеса менялся очень быстро, однако менеджмент остался прежним. Друкер нападает на наши устаревшие бухгалтерские системы, рожденные в эпоху меркантилизма, повзрослевшие в век пара и стали и угасающие на пороге XXI века, эпохи информации. Друкер говорит: «Самая нужная вам информация – информация о внешнем мире, и этой информации у вас нет».

В последние несколько лет XX века, по мере раздувания мыльного пузыря доткомов, целые цистерны чернил уходили на продажу идеи, что в интернете существует «новая экономика». Знатоки утверждали, что продажа вещей во Всемирной паутине, где магазины строят из страниц, а не из кирпичей, принципиально отличается от привычных стилей биз-

неса и что «старую экономику» уже не оживить. Разумеется, почти все компании новой экономики мертвы, финансисты, поддержавшие их, пережили шок, а эксперты, пропагандировавшие новую экономику, теперь заявляют, что то была пустая мечта. Новая-преновая линия такова, что нам суждено пока оставаться со старой-престарой экономикой.

Вообще говоря, я считаю, что мы живем при новой экономике. Более того, я думаю, что доткомы никогда не были ее частью. Напротив, они стали последним вздохом *старой* экономики, экономики производства.

В промышленную эпоху, до появления программ, продукты *создавались* из реальных материалов – из атомов. Затраты на добычу, плавку, приобретение, транспортировку, нагрев, формовку, сварку, окраску и снова транспортировку преобладали над всеми прочими расходами. В бухгалтерском учете эти расходы называются «переменными затратами», поскольку они различны для каждого продукта. «Фиксированные затраты», как вы, наверное, догадываетесь, очевидным образом не меняются и включают такие затраты, как корпоративное администрирование или начальная стоимость завода.

Классические правила управления бизнесом обусловлены производственными традициями промышленной эпохи. Однако эти правила не учитывают новые реалии эпохи информационной, в которой продукты уже не создаются из атомов, а состоят в основном из программного кода, из наборов би-

тов. А биты не подчиняются тем же экономическим правилам, что атомы. Некоторые фундаментальные истины остаются справедливыми и для новой экономики. Цель любого бизнеса – стабильная прибыль, и есть лишь один законный способ получать ее: продавать товары или услуги дороже, чем обходится их создание. Из этого следует, что есть два пути повышения прибыльности: снижение затрат или рост прибыли. В старой экономике лучшим способом было снижение затрат. В новой экономике куда лучше работает рост прибыли.

Сегодня наиболее нужные и дорогие продукты состоят (полностью или почти полностью) из программного обеспечения. Они не требуют сырья. У них нет стоимости производства. Они не требуют затрат на транспортировку. Не требуют литья, обработки, покраски. Вот она, реальная разница между экономикой промышленной эпохи и экономикой эпохи информационной: в информационную эпоху практически или совсем отсутствуют переменные затраты, тогда как в позднюю промышленную эпоху именно эти затраты были главным фактором. Очевидно, что именно отсутствие переменных затрат делает новую экономику новой.

Зарплата программистов в вашем штате – фиксированные затраты или переменные? Один час работы программиста нельзя связать с одной продажей продукта – один и тот же код можно продавать много раз. Вложение в программирование можно амортизировать продажей миллионов копий

продукта – точно так же, как продажа продуктов, созданных на заводе, амортизирует вложения в этот завод.

Стоимость создания программ не переменна, но и не фиксирована тоже. Разработка программ – это непрерывный процесс для компании, приносящий прибыль, и это совсем не то же самое, что строительство завода. Высокооплачиваемые строители завода после завершения работ уходят на другую рабочую площадку. Программисты стоят гораздо дороже плотников и сварщиков и никогда не исчезают, потому что, по всей видимости, их работа никогда не кончается. Кто-то может сказать, что программирование – это научно-исследовательские работы, и сходство действительно есть. Однако же научно-исследовательская работа – это гипотезы и эксперименты, призванные оценить теоретическую жизнеспособность продукта, и они происходят совсем не так, как настоящее создание продуктов. Эта мысль подтверждается тем, что традиционный бухгалтерский учет разделяет исследования/разработку и ежедневную деятельность, приносящую прибыль. Создание программ не попадает ни в одну из этих категорий учета, приемлемых для прежних предприятий.

Да, этим маленьким терминологическим несоответствием можно было бы и пренебречь как придиркой, уместной в беседе счетоводов за кружкой пива, но в действительности оно оказывает огромное влияние на финансирование разработки программ, на управление проектами по разработке и, что

самое важное, на то, как к разработке программ относятся топ-менеджеры.

Программисты создают приложения, а руководители создают потоки прибылей и структурные подразделения. Программисты оценивают свой успех по качеству продукта, а руководители – по прибыльности вложений. Эту прибыльность они оценивают на языке математических терминов, позволяющем учитывать фиксированные затраты, переменные затраты, затраты на корпоративное администрирование, исследования и разработку, но, к сожалению, не описывающем подходящие модели для программ и программирования. Бухучет – основной язык бизнеса, и перечисленные категории настолько фундаментальны для всех оценок предпринимательской деятельности и коммуникации в бизнесе, что современные руководители полностью их усвоили. Программирование для них – еще одна статья корпоративных расходов, которую следует причислить к одной из существующих категорий. На практике большинство руководителей расценивают программирование как производственный процесс, имеющий переменные затраты. (Для целей налогообложения в большинстве компаний-разработчиков ПО программирование проходит по статье исследований и разработок, но во всех остальных отношениях расценивается как деятельность с переменными затратами.) Это худший выбор из всех возможных, поскольку он наносит серьезный ущерб процессу принятия бизнес-решений.

В промышленную эпоху основным преимуществом была массовость, которая позволяла снижать цены и делать продукцию доступной для широких слоев населения. Покупатель при этом получал функции, которые ранее были не доступны или доступны для богатых людей, поскольку производились вручную. Компании конкурировали в области продажных цен, непосредственно связанных с переменными затратами – затратами на производство и доставку. В информационную эпоху доступность продукции по разумным ценам считается обстоятельством само собой разумеющимся. В конце концов, программы можно распространять через интернет – практически бесплатно и почти не прилагая усилий.

Как вы помните, бизнес может повышать доходность за счет роста прибыли или сокращения расходов. Другими словами, предприятие может увеличивать инвестиции в области фиксированных затрат, повышая качество продукции и укрепляя таким образом ценовые позиции, или же снижать переменные затраты, что означает снижение стоимости производства. В старой, «атомной» экономике снижение затрат давалось легко и было эффективным и предпочтительным. Сегодня же руководители, ставящие программирование на одну доску с производством, воображают, будто снижение стоимости программирования дается так же легко и оказывается таким же эффективным. К сожалению, старые правила больше не действуют.

Поскольку производство программ сопровождается незначительными переменными затратами, снижение этих затрат не дает преимущества в бизнесе. С точки зрения бухгалтера зарплаты программистов – переменные затраты, однако в действительности их зарплаты представляют собой долгосрочные вложения, фиксированные затраты. Снижение стоимости программирования и снижение стоимости производства – разные вещи. Первое можно сравнить, скорее, с раздачей работникам дешевых инструментов, чем со снижением зарплат. Компании, заказывающие разработку в других странах с целью снижения зарплат, просто не понимают сути дела.

Более того, единственно возможный вариант для экономического подъема – это повышение качества и, как следствие, привлекательности продукта или услуги, а повышения качества невозможно добиться, сокращая затраты на проектирование и программирование. Правда в том, что в исследования, анализ, планирование и проектирование следует вкладывать больше времени и денег, чтобы полученный результат лучше соответствовал потребностям потребителя.

Разумеется, такой подход требует мышления, непривычного для предпринимателей XXI века. Им следует не *снижать* затраты на создание *каждого* объекта в отдельности, а *повышать* затраты на создание всей *совокупности* объектов. В этом суть новой экономики, и именно об этом говорит Питер Друкер.

Современные фармацевтические компании, разрабатывающие высокотехнологичные лекарства, имеют кое-что общее с новой экономикой программного обеспечения. Действительная стоимость производства одной таблетки минимальна, однако разработка лекарства может обойтись в миллиарды долларов и продолжаться более десяти лет. Подъем после выхода на рынок нового волшебного препарата может длиться до бесконечности, а вот выпуск недоработанного препарата способен принести только катастрофический спад. Фармацевтические компании знают, что снижение издержек на разработку – нежизнеспособная стратегия.

Как и разработка лекарственных препаратов, разработка ПО совсем не похожа на строительство завода. Завод – физический актив, который принадлежит компании, а работников завода, как правило, легко заменить другими. Неосязаемые, но невероятно сложные паттерны мыслей, составляющие программное обеспечение, обладают ценностью только вместе с написавшим код программистом. Ни одна компания не может себе позволить относиться к программистам так же, как к заводу. Программисты требуют постоянного внимания и поддержки, причем гораздо большей, чем любой завод.

Чаще всего пытаются сэкономить на архитектуре программного продукта, а эта часть проектирования (во время которого изучаются пользователи, определяются сценарии работы, проектируется взаимодействие, определяется фор-

ма, описывается поведение) выполняется человеком. Конечно, иногда проектированию уделяют слишком большое внимание, но сокращение этой фазы пользы точно не приносит. Каждый доллар и каждый час, потраченные на архитектуру, принесут десятикратную экономию на этапе программирования. Кроме того, вложения в достаточно качественное проектирование сделают ваш продукт привлекательным, а это означает, что продукт принесет больше прибыли. Его привлекательность станет основой для вашего бренда, создаст возможности для повышения цен, сделает клиентов лояльными, подарит вашему продукту более долгую и плодотворную жизнь. И хотя здесь нет экономии средств, вы получаете большое преимущество в смысле качества. По иронии судьбы лучший способ увеличить прибыльность в информационную эпоху состоит в том, чтобы больше потратить.

К сожалению, в руководителях живет практически неистребимое желание расходовать на программирование поменьше времени и средств. Они не считают тактику сокращения затрат устаревшей и не понимают, что сокращение инвестиций в программирование крайне негативно влияет на качество, привлекательность и прибыльность продукта в долгосрочной перспективе. Разумеется, простым повышением затрат не добиться улучшений. Зачастую ситуация даже ухудшается, если деньги вкладываются бездумно, без анализа и правильного руководства. Мой первый наставник, Дэн Хоакин (Dan Joaquin), любил повторять, что старую истину

«получаешь то, за что платишь» нужно поменять на «не получаешь того, за что не платишь». Действия без планирования всегда чреваты риском потратить слишком много. Фокус в том, чтобы потратить правильное количество денег, а для этого нужно разбираться в управлении созданием программного обеспечения. Еще для этого нужны процессы, обеспечивающие руководителей пониманием и информацией для принятия верных решений. Дать компаниям такие процессы – цель этой книги.

В буме доткомов участвовали компании с бизнес-моделями, полностью ориентированными на снижение переменных затрат. Хотя многие доткомы рекламировали преимущества онлайн-покупок, их сайты, тяжеловесные и неудобные, были плохой альтернативой обычной поездке в торговый центр. Основатели доткомов просто лучились от восторга (и пресса, кстати, тоже), потому что им удалось создать предприятия розничной торговли с невероятно низкими переменными затратами. Феерический провал этих предприятий, несомненно, доказывает, что информационной эпохой правят иные экономические правила, чем промышленной эпохой.

В старой экономике более низкие переменные затраты приводили к более широкому распространению товара и снижению розничных цен. Это двойное преимущество было выгодно покупателям, а покупатели – фундамент экономического успеха промышленной революции. В новой экономике успех бизнеса зависит от способности дать потреби-

телю что-то новое и более качественное. Реальное качество каждого шага транзакции – от просмотра страниц до сравнения товаров – должно быть ощутимо более высоким для пользователя. Гораздо приятнее сделать покупку обычным путем, чем продираться через 11 экранов и в конечном итоге выяснить, что все равно придется звонить в компанию. Покупки в сети становятся совершенно ненужными и непривлекательными, если требуется набрать свое имя, свой адрес и ввести информацию о кредитной карте три или четыре раза, а затем обнаружить, что сайт не позволяет купить все необходимое и все равно придется ехать в обычный магазин, сделанный из атомов. Сегодня простое снижение цены на продукт уже не дает гарантии успеха.

Компания Pets.com, специализирующаяся на продаже корма для собак через интернет, не предлагала более качественный корм, как не предлагала и шопинг, более приятный, чем в обычном зоомагазине; она не предлагала новую информацию, новые возможности, новую уверенность. Она предлагала лишь дешевую доставку, складирование и торговлю (все это переменные затраты) на сайте *Pets.com*. Компания применила классическую тактику снижения затрат, характерную для промышленной эпохи, проигнорировав фундаментальные принципы новой экономики. Конечно, это было еще не первое дыхание новой экономики, но для старой это были последние судороги. Я совершенно убежден, что любой товар можно продавать через интернет

успешно и прибыльно. Для этого всего лишь нужно, чтобы в онлайн-магазине было намного приятнее покупать, чем в конкурирующих розничных сетях, и цена здесь – всего лишь один из факторов. Есть лишь один способ добиться этого: архитектуру системы следует создавать с целью максимально удовлетворить конечного пользователя. Нельзя относиться к любому аспекту проектирования и создания программного обеспечения как к производственному процессу – это повлечет за собой провал. Проектирование и программирование – это неподходящие цели для традиционных методов сокращения затрат. Конечно, можно потратить на создание программ слишком много времени и денег, но опасность потратить меньше необходимого гораздо серьезнее.

Скорее всего, эта опасность вам знакома и удивления не вызывает, но большинству топ-менеджеров крупных компаний сама идея кажется неприемлемой. Эти руководители до сих пор работают по моделям бухучета, вошедшим в моду в эпоху паровых машин, тогда как все аспекты жизни их компаний – функционирование, принятие решений, коммуникации и финансы – полностью зависимы от программного обеспечения. Термины и понятия, которыми оперируют эти руководители, не учитывают уникальную природу бизнеса в эпоху, когда инструменты и продукты торговли являют собой неосязаемые переплетения битов вместо железнодорожных составов, груженных сталью.

Корпорации уже нанимают проектировщиков взаимодей-

ствия и начинают применять целеориентированный подход, однако качество программных продуктов не слишком улучшилось. Более того, высокие затраты на программирование и жесткий процесс создания программ никуда не делись. Почему?

Перемены невозможны, пока топ-менеджеры компаний не осознают, что проблемы с программами – это не технические сложности, а важные бизнес-вопросы. Наши проблемы останутся неразрешенными до тех пор, пока мы не изменим *процесс* и *организацию*.

Компании живут не только по устаревшим финансовым моделям, но еще и по негодной организационной модели. Эта модель скопирована с научной, в ней создание программы смешивается с планированием и решением инженерных задач. Такова природа исследований. Прискорбно, что эта парадигма без предупреждения была перенесена в неизменном виде в мир бизнеса, где ей не место.

Корни всех современных производственных дисциплин уходят в доиндустриальные времена. Всех, кроме дисциплины программного обеспечения, которая возникла, когда индустриализация уже закончилась. Только программирование вышло сразу из университетской среды, где время исследований не ограничивалось, студенческой рабочей силы было хоть отбавляй, о прибылях вообще не говорили, а неработающая программа могла сойти за весьма удачный эксперимент. Неслучаен тот факт, что *Microsoft, IBM, Oracle* и другие

ведущие компании-разработчики ПО расположены в «кампусах». Университетам не нужны прибыли, они не стараются успеть создать привлекательный и полезный продукт к определенному сроку.

Любой бизнес, не связанный с программным обеспечением, начинается с исследований и заканчивается распространением продуктов или предложением услуг. Компания тщательно планирует время между этими двумя событиями, осознавая, что преждевременный выпуск непродуманного продукта опасен как для банковского счета, так и для репутации. Руководители знают, что время, размышления и деньги, вложенные в планирование, обернутся крупными дивидендами – отлаженным и быстрым процессом производства, популярностью и прибыльностью конечных продуктов.

Во всех других конструкторских дисциплинах инженеры создают стратегию, а ремесленники претворяют ее в жизнь. Инженеры не строят мосты сами, этим занимаются монтажники. Только в области программного обеспечения перед инженером стоит задача создать собственно продукт. Только в области программного обеспечения перед «монтажником» стоит задача определить, как следует создавать продукт. Только в области программного обеспечения эти две задачи решаются не последовательно, а одновременно. Однако компании-разработчики ПО, похоже, не осознают существования такой аномалии. Инженерное дело и конструкторское дело так тесно пересекаются, что специалисты и ру-

ководители их не разделяют и, вероятно, не различают. Планированием любого рода здесь пренебрегают или откладывают его до тех пор, пока не станет слишком поздно. Считается нормальным откладывать решение очень сложных инженерных проблем до момента, когда экономически станет накладно откатываться к фазе проектирования, потому что полным ходом пишется код для коммерческой версии продукта.

Проектирование архитектуры следует начинать на ранних стадиях инженерного планирования. Более того, именно оно должно быть движущей силой на этих стадиях, но такие разработки обычно откладываются до момента старта проекта и ведутся параллельно с созданием кода, поэтому не занимают должного места в процессе конструирования продукта. Компании нанимают проектировщиков взаимодействия и обучают специалистов по юзабилити создавать персон, однако работа этих людей почти не влияет на стоимость разработки и качество завершенного продукта.

Решение проблемы – в руках президентов и генеральных директоров корпораций. Делегируя решение своим техническим директорам или вице-президентам по разработке, они поступают неверно. Эти достойные исполнители – технари, а проблема не техническая. Как сказал Друкер, инструменты для бухгалтерии, на которые полагаются директора компаний, попросту не отражают истинной природы этих компаний. Нельзя ведь на основе точных показаний спидомет-

ра утверждать, что автомобиль движется в нужном направлении. В мире бизнеса цифровых технологий такой подход не может быть эффективным.

Одна из серьезных проблем применения неверных методов бухучета и организации для разработки программ состоит в том, что руководители не видят, сколько денег, израсходованных на программирование, потрачено впустую. Точная система показала бы, что из каждого доллара около пятидесяти центов тратится неправильно и что еще два или три доллара требуется, чтобы исправить проблему, вызванную этим некорректным вложением средств. В любом другом бизнесе подобная статистика вызвала бы революцию, однако отрасль программного обеспечения продолжает жить в состоянии блаженного неведения.

За последние тринадцать лет моя компания *Cooper* проконсультировала сотни компаний. Мои талантливые проектировщики создали для большинства клиентов «чертежи» продуктов, позволяющие коренным образом изменить ситуацию, но лишь немногие сумели воспользоваться всеми полученными преимуществами. В большинстве этих компаний проектирование взаимодействия и архитектуру программ считают лишь рекомендацией, в этих компаниях последнее слово всегда остается за программистами и инженерами. Ни один из президентов этих компаний не имеет ни малейшего представления о том, что происходит в офисах инженеров, и потому расписание ужимается безо всякой

причины. Программисты постоянно работают в условиях дефицита ресурсов, и прежде всего – времени на хорошее программирование, а также времени, чтобы определить, где вообще требуется программирование. Они вынуждены защищаться, отвергая советы, и изворачиваться, общаясь со своими менеджерами.

На мой взгляд, существует два типа исполнителей: инженеры и боящиеся инженеров. Первые множат знакомые проблемы, поскольку их точка зрения безнадежно затуманена конфликтом интересов. Вторые множат проблемы, поскольку не умеют говорить на языке программистов. И я не имею в виду языки Java и C#. Я имею в виду, что у предпринимателей и программистов нет общих инструментов и общих целей. *Хомо сапиенс* делегирует человеческие проблемы *Хомо логикус*, не осознавая, что решение могло бы оказаться куда лучше в случае применения – на исполнительном уровне – уместных финансовых и организационных моделей.

У компаний есть прекрасная возможность сдвинуться с мертвой точки и сосредоточить усилия на удовлетворении потребностей клиентов, а не на программах, на персонах, а не на технологиях, на выгоде, а не на программистах. Я с нетерпением жду, когда появится просвещенный руководитель, который ухватится за эту возможность и навсегда изменит способ создания программного обеспечения, подав смелый и успешный пример.

Алан Купер,
Менло-Парк, Калифорния,
октябрь 2003 г.
www.cooper.com
inmates@cooper.com

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

Часть I. Компьютерная неграмотность

1. Загадки информационной эры

Что будет, если скрестить компьютер и самолет?

Декабрь 1995 года. Самолет авиакомпании *American Airlines* совершает вылет по регулярному рейсу 965, следуя из Майами в колумбийский город Кали. Подлетая к взлетно-посадочной полосе, пилот «Боинга-757» должен был выбрать радиомаяк ROZO, чтобы определить ее координаты. Воспользовавшись бортовым навигатором, пилот ввел первую букву названия радиомаяка – R. Навигатор незамедлительно вывел список ближайших радиомаяков, названия которых начинались с R, – пилоту оставалось лишь выбрать нужный. Он выбрал самый первый, ориентируясь на указанные рядом с названием, казалось бы, верные координаты маяка. К несчастью, это оказался маяк ROMEO, который находился в 212 километрах к северо-востоку. Рейс же летел в южном направлении, а в момент выбора маяка вошел в долину, которая тянулась с севера на юг, так что отклоняться от курса было крайне рискованно. Ориентируясь на показания бортового навигатора, пилоты приступили к корректировке курса в направлении востока, а спустя несколько ми-

нут «Боинг-757» на высоте 3000 метров врезался в вершину гранитного утеса. Погибли 152 пассажира и 8 членов экипажа. Выжили только четверо пассажиров, все из них получили тяжелые травмы. Как обычно бывает в подобных ситуациях, причиной крушения самолета, объявленной Национальной комиссией по безопасности, был человеческий фактор. Бортовой навигатор, который использовали пилоты для ориентировки, отобразил верные данные, только не те, что нужны были для успешной посадки самолета в аэропорту Кали. Фактически человеческий фактор действительно имел место – ведь именно пилот выбрал маяк, совершив роковую ошибку, однако общая оценка ситуации позволяет утверждать, что вины пилота здесь не было.

На передней панели бортового навигационного компьютера самолета был показан радиомаяк, на который ориентировался рейс, и индикатор отклонения от курса. Если курс взят верно, стрелка будет указывать в центр шкалы, но правильность выбора самого маяка не проверяется. Внешний вид индикатора одинаков как в ситуации посадки, так и при возможном крушении. В данном случае навигатор отобразил данные о верно взятом курсе, но не учел, что неверно выбранный маяк может стать причиной катастрофы.

* * *

Полнота и точность передаваемых данных может быть аб-

солютной, но при этом повлечь трагический исход из-за их некорректности в текущей ситуации. При взаимодействии с компьютерами это происходит повсеместно, а ведь в современном мире все сложнее найти сферу, в которой бы не применялись компьютерные устройства. Начиная от пассажирских самолетов и заканчивая потребительскими товарами и услугами – везде используются компьютеры и везде их применение сопровождается характерным способом взаимодействия и поведения.

Существует такая шутка, широко известная в компьютерной среде: пилот небольшого самолета заблудился в облаках. Он идет на снижение, пока не оказывается рядом с офисным зданием, и кричит человеку в открытом окне: «Не подскажете, где я?» На что человек выдает ответ: «Вы в самолете, примерно в тридцати метрах над землей». Пилот тут же берет верный курс и спустя некоторое время благополучно приземляется в аэропорту. Пассажиры самолета удивленно спрашивают, как он понял, куда лететь. И пилот говорит: «Тот человек ответил мне совершенно точно и правдиво, но эта информация была абсолютно бесполезна, поэтому я сразу догадался, что этот человек – разработчик программного обеспечения из *Microsoft*, а я знаю, где расположено здание *Microsoft* по отношению к аэропорту».

В свете описанной выше трагедии рейса 965 эта шутка звучит зловеще, тем не менее профессионалы из цифрового мира не упускают возможности в очередной раз рассказать

ее и посмеяться, потому что она подчеркивает главную особенность компьютеров: они могут выдавать нам факты, но не разъясняют их. Их информация отличается точностью, но только лишь ее нам недостаточно, чтобы достичь намеченных целей. В бортовой навигатор рейса 965 легко можно было заложить функцию сообщать пилотам о неверном выборе маяка. Один простой намек на то, что выбор маяка ROMEO «нетипичен» или «незнаком», мог бы предотвратить крушение. Только пассажиры и сам рейс, вероятно, не были тем, что занимало ум бортового компьютера. Он был занят исключительно собственными вычислениями.

Сложности во взаимодействии с компьютерами влияют на всех нас, временами приводя к фатальным последствиям. Но программные продукты сложны в применении не из-за сложности самих компьютеров, а потому, что в основу их разработки заложен неверный процесс. Данная книга призвана не только продемонстрировать следствия такого неверного процесса, но и прояснить причины его возникновения. Далее мы рассмотрим, как следовало бы изменить процесс, чтобы наше программное обеспечение обрело дружелюбный вид и мощный функционал. В этой главе основной акцент прежде всего делается на серьезность рассматриваемой проблемы.

Что будет, если скрестить компьютер и фотокамеру?

Вот одна из загадок эры информации: что будет, если скрестить компьютер и фотокамеру? Верный ответ: компьютер! Тридцать лет назад[1] моим первым фотоаппаратом был 35-миллиметровый Pentax H, в него вставлялась маленькая батарейка, от которой запитывался датчик освещенности (экспонометр). От меня требовалось только менять батарейку раз в два года, как в наручных часах.

Пятнадцать лет назад моей первой электронной фотокамерой был 35-миллиметровый Canon T70, в который вставлялось уже две пальчиковые батарейки, за счет чего достаточно простой электронный блок экспонометра приводился в действие, а также подавалось питание для автопрокрутки пленки. Сэкономить заряд батареек помогал простой выключатель на корпусе фотоаппарата.

[1] Считая от 2006 года, когда была издана книга. – *Примеч. ред.*

Пять лет назад у меня появился Logitech – цифровой фотоаппарат первого поколения; он тоже имел подобный выключатель, но в дополнение обладал чуть более продвинутой электронной начинкой, напоминавшей зачатки компьютерного разума. Теперь, когда я забывал его отключить, он сам автоматически выключался спустя минуту бездействия. Довольно мило.

Год назад внутри моей Panasonic PalmCam – цифровой фотокамеры второго поколения – находилась куда более умная компьютерная микросхема. Ее ум был настолько велик, что простой рычажок «Вкл/Выкл» эволюционировал в переключатель Off/Rec/Play, который включал разные режимы работы камеры. Режим Rec предназначался для фотосъемки, а просмотр фотографий на небольшом экране осуществлялся в режиме Play.

Последней фотокамерой, которой я обзавелся, стал цифровой фотоаппарат третьего поколения Nikon CoolPix 900, и он еще более сообразительный. Его электронный мозг настолько умный, что представляет собой полноценный компьютер, который даже показывает песочные часы в стиле Windows во время загрузки. Его переключатель, словно рыба-мутант, отрастил дополнительные «головы», и теперь их стало *четыре*. Каждая из них отвечает за свою функцию: Off/ARec/MRec/Play. ARec – это режим автоматической записи, а MRec – режим записи вручную. Как по мне, разницы

между ними нет. При этом у камеры нет режима On (включено), так что даже мои друзья не смогли понять, как ее включить, без моих объяснений.

Эта новая камера потребляет достаточно много энергии, так что создатели предусмотрительно наделили ее изощренной функцией контроля уровня заряда аккумулятора. Теперь все обычно происходит так: я перевожу злосчастный переключатель Off в режим MRec, мучительно жду приблизительно семь секунд, пока камера загрузится, и только потом могу направить ее на предмет съемки. Далее я выбираю подходящий ракурс и приближение камеры для получения наилучшего кадра. И в тот самый миг, когда мой палец почти нажал на спуск, камера вдруг решает, что одновременное увеличение кадра, питание вспышки и поддержание яркости экрана способны вконец исчерпать заряд батареи. Порываясь сохранить последние капли жизненных сил, камера временно отключает функцию съемки. Но мне об этом неизвестно, мой взгляд направлен в видоискатель. Я воодушевляюще машу руками, прошу сказать «Сы-ы-ыр!» и нажимаю на спуск. Компьютерный блок фотокамеры отслеживает нажатие кнопки, но не способен ничего предпринять. Программа управления питанием делает еще одну слабую попытку спасти ситуацию и в мгновение ока принимает ответственное решение: уменьшить нагрузку. В результате этого энергоемкий LCD-экран гаснет. Я недоуменно гляжу на камеру, пытаясь понять, почему не получилось сделать сни-

мок, пожимаю плечами и опускаю руку, в которой держу камеру. Однако сэкономленная на отключении экрана энергия позволяет дополнительно запитать другие элементы камеры. У программы управления питанием открывается «второе дыхание», и она решает, что *прямо сейчас* энергии хватит, чтобы сделать снимок. Она посылает сигнал основной программе, которая терпеливо ждет момента, когда можно будет продолжить процесс фотосъемки и выполнить ранее посланную мной команду сделать снимок. В результате камера делает великолепно сфокусированное, безупречно экспонированное высококлассное цифровое фото моего колена.

На моем старом механическом фотоаппарате Pentax все настройки фокусировки, экспозиции и выдержки нужно было устанавливать вручную, тем не менее это вызывало куда как меньше мучений, чем процесс фотосъемки на современный, полностью компьютеризованный Nikon CoolPix 900, в котором все эти функции автоматические. Nikon по-прежнему позволяет делать снимки, но его *поведение* характерно для компьютера, а не для фотокамеры.

* * *

Лягушке, оказавшейся в кастрюле с холодной водой на плите, невдомек, что повышение температуры смертельно. Это происходит оттого, что нарастающий жар притупляет ее ощущения. Мои фотокамеры медленно эволюционировали

от простого к сложному, становясь все более компьютеризированными, а я не осознавал этого, совсем как та лягушка. И это характерно для всех нас, испытывающих на себе это медленное, убаюкивающее чувства посягательство компьютеров и присущего им поведения на нашу жизнь.

Что будет, если скрестить компьютер и будильник?

Компьютер! Недавно я поставил в своей спальне новый дорогой будильник со встроенным радио марки JVC FS-2000. Внутри устройства кроется довольно сложная компьютерная начинка, обещающая высокую точность. Кроме того, оно обладает цифровым звуком и еще множеством функций. Теперь я могу просыпаться в точно заданное время под мелодию с компакт-диска и, более того, будильник настолько умен и деликатен, что звук этой мелодии не взорвет барабанные перепонки в шесть утра, а будет нарастать постепенно. Эта редкая в будильниках опция действительно приходится весьма кстати, и только мысль о ней усмиряет мое желание вышвырнуть это дьявольское изобретение из окна.

Понять, на какой день недели установлен звонок будильника, чертовски сложно, так что периодически он не будит меня в понедельник и нагло врывается в мой мирный сон ранним субботним утром. Конечно, в этом устройстве име-

ется индикатор активности будильника, но это вовсе не значит, что он пригоден для использования. Встроенный алфавитно-цифровой LCD-дисплей запутанно отображает всяческие обозначения функций в часах. Например, маленький значок часов в левом верхнем углу LCD-дисплея сообщает о том, что будильник активен. Только в полутемной спальне опознать этот значок весьма затруднительно. У дисплея имеется функция подсветки, благодаря которой значок часов становится заметным, но включается она лишь в том случае, когда работает радио или играет компакт-диск. При этом есть один нюанс: будильник, даже если его установить, не прозвенит при включенном проигрывателе компакт-дисков. Именно такое противоречивое поведение часто ставит меня в тупик.

Отключить будильник проще простого: для этого нужно только один раз нажать кнопку Alarm, и значок часов пропадет с дисплея. А вот чтобы включить будильник, потребуется нажать эту кнопку не меньше пяти раз. Когда нажи-

маешь ее первый раз, на дисплее появляется время, в которое должен прозвенеть будильник. На второй раз отображается время, когда звук будильника должен выключиться. На третьем нажатии нужно выбрать, зазвучит ли радио или же компакт-диск. А на четвертом – определить желаемый уровень громкости. Завершающее пятое нажатие закрывает меню установки будильника и приводит часы к обычному режиму. Здесь главное – не нажать еще один, лишний раз, ведь тогда будильник вовсе отключится. Однако полусонному человеку в темноте спальни безошибочно исполнить этот маленький цифровой кордебалет бывает весьма затруднительно.

В силу своей упрямой симпатии к подобного рода гаджетам я не оставляю попыток разобраться в этом устройстве, надеясь однажды победить. Чего не скажешь о моей супруге – она уже давно перестала пытаться найти подход к этому дьявольскому агрегату. Она восхищается сглаженными линиями в его современном дизайне и качеством звучания, но экзамен в качестве будильника он не прошел, потому что требуется слишком много усилий, чтобы заставить устройство делать то, что нужно. Эти часы хотя и могут разбудить меня, но *ведут себя* как компьютер.

В отличие от него, мой древний 11-долларовый некомпьютеризированный будильник разливался внезапным жутким дребезжанием, когда требовалось меня разбудить. О том, что он включен, можно было узнать по яркой красной

лампочке. А когда он выключался, лампочка гасла. Тот старый будильник не нравился мне по ряду причин, но по меньшей мере мне было ясно, собирается он меня будить или нет.

* * *

По той причине, что производителям намного выгоднее применять для контроля внутренних процессов устройств компьютеры, нежели механизмы старого образца, компьютеризация устройств и процессов оказания услуг в нашей жизни экономически неотвратима. Это значит, что все существующие устройства скоро начнут вести себя как самые несносные компьютеры, если только мы не попробуем подойти к этому иначе.

* * *

Продуктами для конечного потребителя описанное выше явление не ограничивается. Устройства, снабженные компьютерной начинкой, и компьютеризированные сервисы отличаются от своих простых собратьев с ручным управлением широтой вариантов применения и количеством опций. Но в реальности люди чаще склонны использовать именно механические устройства и делать это более гибко, искусно и осознанно, нежели при обращении с их современными вер-

сиями, работающими на кремниевых микросхемах.

Высокотехнологичные компании, в стремлении улучшить свои разработки, сводят этот процесс к наполнению их сложным и часто ненужным функционалом. Но такой некорректный процесс не помогает повысить качество продуктов, а только добавляет новые функции, так что это именно то, чем в действительности и занимаются его создатели. Далее в этой книге я продемонстрирую, как совершенствование процесса разработки приносит больше счастья пользователям, без лишних затрат на никому не нужные дополнительные функции.

Что будет, если скрестить компьютер и автомобиль?

Компьютер! Роскошный высокотехнологичный спортивный автомобиль Boxster от *Porsche* оснащен семью компьютерами, посредством которых управляются его сложные подсистемы. Один из них целиком предназначен для управления двигателем. В память этого компьютера заложены специальные программы, которые помогают преодолевать критические ситуации. К несчастью, эти программы сами периодически становятся причиной странных сбоев в работе автомобиля. Некоторые ранние модели отличались такой особенностью: когда уровень топлива в баке доходил до предельно низкой отметки – где-то около четырех литров, – бензин

скапливался у стенки бака под действием центробежной силы при крутых поворотах. Это приводило к попаданию воздуха в топливную систему. Компьютер определял, что в топливной смеси произошли серьезные изменения, и интерпретировал это как фатальный сбой системы топливного впрыска. Для предотвращения критических последствий компьютер отключал зажигание, и автомобиль останавливался. Более того, во избежание тех же последствий компьютер не позволял заново запустить двигатель до тех пор, пока не будет произведена буксировка машины в автомастерскую для устранения неисправностей.

Все, что могла посоветовать тогда компания *Porsche* владельцам ранних Boxster, столкнувшихся с этой проблемой, – это открыть капот и отсоединить аккумулятор минимум на пять минут, тогда компьютер удалил бы из памяти сведения об инциденте. Несмотря на то что спортивные автомобили позволяют разъезжать по двухполосным асфальтированным дорогам на предельной скорости, в крутых поворотах они начинают *вести* себя в точности как компьютер.

* * *

В похвальном стремлении уберечь владельцев Boxster от всяческих бед разработчики программ для автомобиля только унизили их и заставили страдать. Каждый, кто любит гоночные автомобили, знает, что клиентская политика компа-

нии *Porsche* направлена на выражение уважения своим клиентам и предоставление большого количества привилегий. А случаи, подобные описанному выше, показывают, что программное обеспечение для автомобиля создавалось не той же самой *Porsche*, что выпускает другие его компоненты. Его создала компания внутри компании: программисты, а не легендарные немецкие инженеры автомобилестроения. По какой-то причине внедрение новых технологий привело к тому, что солидная компания с долгой историей пренебрегла своими ключевыми ценностями. Стандарты качества для проектировщиков программного обеспечения намного ниже, чем для более традиционных инженерных дисциплин.

Что будет, если скрестить компьютер и банк?

Компьютер! В те моменты, когда я снимаю денежные средства в банкоматах, я каждый раз наблюдаю одно и то же угнетающе запутанное поведение, столь характерное для компьютеров. Малейшая допущенная мной ошибка приводит к отмене всей транзакции и выкидывает меня из сеанса. Мне приходится вытаскивать карту, снова ее вставлять, снова набирать PIN-код и снова вводить запрос. И ошибку-то обычно я допускаю не по своей воле, а потому, что компьютер банкомата мастерски вводит меня в заблуждение.

Каждый раз он задает мне один и тот же вопрос: каким

счетом я хотел бы воспользоваться для снятия наличных – текущим, сберегательным или депозитным, – и это учитывая то, что у меня всего один счет – текущий. Каждый раз я забываю правильный ответ, так что такой вопрос сбивает меня с толку. Приблизительно раз в месяц я нечаянно выбираю сберегательный счет, и адский агрегат без долгих размышлений прерывает сеанс, так что мне приходится начинать сначала. При этом для отказа в снятии наличных со сберегательного счета банкомат должен знать, что такой счет у меня отсутствует, однако он все равно предлагает мне его как один из вариантов. Единственное, что отличает меня в момент выбора банковского счета от пилота рейса под номером 965, который ошибочно выбрал маяк ROMEO, – это масштабы последствий.

Помимо этого, банкомат ограничивает мои действия, налагая суточный лимит снятия наличных, который составляет 200 долларов. Так что, даже если я успешно преодолею весь путь – аутентификацию, выбор типа счета, запрос нужной суммы – и это будет, к примеру, сумма, равная 220 долларам, компьютер банкомата, не церемонясь, отклонит транзакцию, выдав грубое сообщение о превышении суточного лимита. Он не скажет мне, какова величина этого лимита, не отобразит остаток на моем счете и даже не позволит ввести другую сумму меньшего порядка. Вместо этого он просто выплюнет мою карту и предоставит мне возможность начать все с «чистого листа», не снабдив меня ни каплей дополнительной ин-

формации, заставляя растущую очередь за моей спиной беспокоиться и раздраженно вздыхать. Реакции банкомата точны и основаны на фактах, но толку от них никакого.

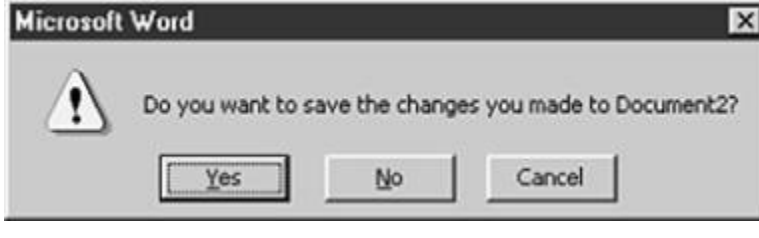
В память банкомата заложены определенные правила, которым он должен следовать. Я бы тоже рад следовать им, но это так по-компьютерному неблагоразумно – не сообщать мне об этих правилах, снабжать меня противоречивыми указаниями, а затем бесцеремонно наказывать меня за нарушение этих правил по чистому незнанию. Такое столь характерное для компьютеров поведение вовсе не является их истинной натурой.

Если быть точнее, никакой натуры у них нет: они просто выполняют операции по заданной программе. А программы такие же гибкие, как человеческая речь. Человек может говорить грубыми или вежливыми словами, выражать поддержку или высказывать недовольство. Компьютер может так же легко, как и человек, сообщать сведения с должным уважением и учтивостью. Его только требуется этому научить. Увы, программисты не те люди, которые способны успешно обучить компьютеры таким вещам.

Как легко попасть в беду с помощью компьютера

Компьютер, воцарившийся на вашем рабочем столе, обладает именно таким несносным раздражающим поведением,

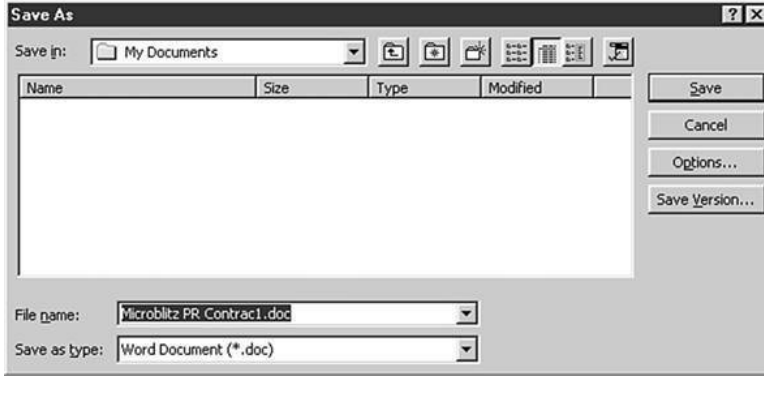
какое и полагается компьютеру, для этого его даже не нужно ни с чем скрещивать. Моя подруга Джейн когда-то занимала должность координатора по связям с общественностью. На ее компьютере была установлена операционная система Windows 95 и программа Microsoft Word, в которой Джейн редактировала заметки и договоры. Основой Windows 95 являлась иерархическая файловая система. Все документы Джейн помещались в маленькие папки, которые, в свою очередь, помещались в другие маленькие папки. Джейн не понимала этого, так же как и не видела необходимости хранить файлы именно таким образом. Если быть точнее, Джейн вообще не слишком задумывалась об этом, а просто выбирала путь наименьшего сопротивления.



Получилось так, что Джейн только что закончила набрасывать черновик нового договора на пиар-услуги для стартапа из Кремниевой долины. Она выбрала опцию Закрыть в меню Файл. Вместо того чтобы просто выполнить команду и закрыть документ, программа Word вывела диалоговое ок-

но. Как вы догадались, речь идет о всем знакомом вопросе Сохранить изменения в документе…? (Do you want to save changes…?). Джейн, как и обычно, нажала кнопку Да (Yes). Она слишком часто видела это окно, так что нажимала кнопки почти не глядя.

За первым диалоговым окном немедленно последовало следующее – и снова слишком хорошо знакомое Сохранить как… (Save As). В этом окне было великое множество непонятных для Джейн кнопок, значков и текстовых полей. Все, что она здесь могла понять и чем активно пользовалась, – это поле для ввода имени файла. Набрав подходящее имя, Джейн нажала кнопку Сохранить. Файл с договором сохранился в папке **Мои документы**. Джейн столько раз проделывала эту нудную процедуру, что уже даже не задумывалась о своих действиях.



В обеденное время, когда Джейн не было в офисе, Сунил, системный администратор компании, установил на ее компьютер обновленную версию антивируса VirusKiller 2.1. Находясь за рабочим компьютером Джейн, Сунил открыл программу Word, чтобы просмотреть файл Readme для антивируса. Окончив просмотр файла, Сунил закрыл его и вернул компьютер Джейн в то состояние, в каком тот находился до обеда. По крайней мере, так ему казалось.

После обеда Джейн потребовалось снова открыть файл договора для распечатки и демонстрации руководителю. Джейн выбрала опцию Открыть в меню Файл, после чего на экране возникло диалоговое окно Открыть. Согласно ожиданиям Джейн, все ее договоры и другие документы должны были удобно отобразиться в этом окне в алфавитном порядке. Однако вместо этого ее взору явилось множество файлов, имен которых она никогда до этого не видела и не смогла опознать. Название одного из этих файлов значилось как **Readme.doc**.

Конечно, все дело было в том, что, когда Сунил открыл Word для просмотра файла Readme, он поручил программе отыскать неведомую папку на шестом уровне вложенности файловой системы и нечаянно изменил привычное для Джейн местоположение **Мои документы** на другое.

Ситуация на экране поставила Джейн в тупик. Первое, о чем она нерадостно подумала, что вся большая работа, кото-

рую она проделала, просто не сохранилась. Джейн взволнованно обратилась за помощью к Рене – своей подруге и коллеге, но Рене пришла в не меньшее замешательство. В итоге, дойдя почти до полного отчаяния, Джейн набрала номер Сунила, чтобы попросить помощи у него. Но Сунила на месте не было, а поймать его и вернуть все на круги своя удалось только в понедельник утром. В результате Джейн, Рене, Сунил и PR-компания в целом потеряли каждый по половине рабочего дня.

И хотя такие иерархические файловые системы нужны операционным системам компьютеров, людям они без надобности. Нет ничего удивительного в том, что больше всего древовидная структура каталогов нравится программистам, как нет и ничего примечательного в том, что обычным пользователям, таким как Джейн, это не приносит никакого удовольствия. А точнее, это непримечательно для всех, кроме тех самых программистов, которые разрабатывают подобное программное обеспечение. Они закладывают в программы такое поведение и отображение информации, что оно понятно для них самих, но между ним и тем, что подходит для Джейн, простирается большая пропасть. Вина за несоблюдение сроков и низкую эффективность труда ложится на Джейн, а не на разработчиков программного обеспечения, чей продукт привел к провалу проекта.

Как бы то ни было, Джейн работой обеспечена. Но ведь ко многим другим людям отношение как к недостаточно «ком-

пьютерно грамотным», так что их отказываются принимать на работу. Чем для большего количества должностей требуются навыки взаимодействия с компьютерами, тем шире становится разрыв между людьми, потенциально подходящими для этих позиций, и теми, кто не подходит, и тем сложнее его преодолеть. Политики могут сколько угодно требовать создания рабочих мест для малоимущих, только компании не дадут ни единого шанса сесть за их компьютеры людям, не имеющим достаточного опыта в этом деле. Их нужно слишком долго обучать, и остается большой риск порчи информации и причинения вреда бесценным базам данных.

Несносное поведение и невнятность взаимодействий, которыми отличаются продукты, работающие на программном обеспечении, уже настолько вошли в норму, что я называю происходящее «софтверным апартеидом». Из-за такого положения совершенно нормальные в целом люди не могут найти работу и устроиться в обществе, так как не владеют компьютером на приемлемом уровне.

Социальные активисты нашего просвещенного общества трудятся не покладая рук, пытаясь преодолеть расовые и классовые барьеры, пока специалисты индустрии технологий не менее тяжким трудом воздвигают еще более высокие заслоны. Но целенаправленная деятельность по проектированию таких программных продуктов, которые бы обладали большей человечностью и меньшей требовательностью, автоматически сделает их более близкими к народу, вне зави-

симости от класса и цвета кожи пользователей.

Коммерческому программному обеспечению тоже нелегко

Компьютеры распространяют свое влияние не только на кабины пилотов в самолетах, но и на пассажирские салоны, показывая там свой упрямый и своевольный характер, так хорошо знакомый нам. В современных авиалайнерах устанавливают системы развлечений в полете, через которые пассажиры могут слушать музыку и смотреть фильмы. Такие системы представляют собой обычные компьютеры, соединенные в локальную сеть, как те, что стоят в вашем офисе. Системы развлечений с расширенным набором опций обычно устанавливают только на больших самолетах, следующих трансокеанскими рейсами.

Конец ознакомительного фрагмента.