1

Загадки века информации

Что получится, если скрестить компьютер с самолетом?

В декабре 1995 года рейс 965 компании American Airlines вылетел по регулярному маршруту из Майами в Кали, Колумбия. На подлете к посадочной полосе пилоту Боинга-757 потребовалось выбрать следующий радиомаяк по имени «ROZO». Он набрал букву «R» в своем навигационном компьютере. Компьютер отобразил перечень ближайших радиомаяков с именами на «R», а пилот выбрал первую позицию в списке, потому что широта и долгота показались ему верными. К несчастью, вместо «ROZO» пилот выбрал маяк «ROMEO», расположенный в 210 километрах к северо-востоку. Самолет направлялся на юг и находился в тот момент в долине, пролегающей с юга на север, так что любое отклонение от курса было опасно. Следуя показаниям полетного компьютера, пилоты начали корректировать курс к востоку, и самолет врезался в гранитный пик на высоте трех километров. Сто пятьдесят два пассажира и восемь членов экипажа погибли. Четыре пассажира выжили, получив серьезные травмы. Национальная комиссия по безопасности транспорта провела расследование и – как обычно – заявила, что причиной явился человеческий фактор. Вспомогательное навигационное средство, показаниями которого руководствовались пилоты, выдало корректную информацию, но не для посадки в Кали. Человеческий фактор, если следовать буквальному смыслу фразы, действительно был причиной – ведь именно пилот выбрал неправильный маяк. Однако если взглянуть на ситуацию в целом, вины пилота здесь не было.

Передняя панель навигационного компьютера самолета отображала выбранный навигационный маяк и индикатор отклонения от курса. Когда самолет находится на курсе, стрелка расположена по центру, но она никаким образом не указывает на правильность выбора радиомаяка. Индикатор выглядит примерно одинаково перед посадкой и перед катастрофой. Компьютер сообщил пилоту, что на выбранный маяк взят точный курс. К сожалению, компьютер упустил из виду, что такой выбор маяка смертелен.

⌘

Полученная информация может быть точной и полной, но при этом трагически некорректной. Это происходит слишком уж часто, когда мы общаемся с компьютерами, а компьютеры проникли во все аспекты современной жизни. От самолетов, на которых мы летаем, до потребительских товаров и услуг – везде компьютеры, везде присущее им поведение и способы взаимодействия.

В компьютерной индустрии широкое хождение имеет такой анекдот: человек, пилотирующий небольшой самолет, заблудился в облаках. Он снижается и замечает офисное здание неподалеку. «Не подскажете, где я нахожусь?» – кричит он человеку в открытом окне. Человек отвечает: «Вы в самолете, примерно в тридцати метрах над землей». Пилот немедленно ложится на верный курс, находит аэропорт и совершает посадку. Его пассажиры в изумлении интересуются, как он определил, куда лететь. И пилот говорит: «Ответ этого человека был абсолютно точен и правдив, однако совершенно бесполезен, поэтому я сразу понял, что это разработчик программного обеспечения из Microsoft, а я знаю, где находится здание Microsoft по отношению к аэропорту».

В свете трагедии рейса 965 анекдот звучит зловеще, однако профессионалы из цифрового мира рассказывают его часто и с удовольствием, потому что он отражает главную правду о компьютерах: они могут сообщать нам факты, но не информируют нас. Их указания точны, но не способны привести нас в нужное место. Полетный компьютер рейса 965 мог с легкостью сообщить пилотам, что «ROMEO» – неподходящий маяк для Кали. Даже простой намек, что выбор «необычен» или «незнаком» мог бы спасти самолет. Вместо этого компьютер, похоже, совершенно не интересовали пассажиры и собственно рейс. Его интересовали только собственные вычисления.

Сложные в применении компьютеры влияют на всех нас, временами фатально. Продукты, основанные на программном обеспечении, сложны в применении не *от природы*, но потому, что мы используем неверный процесс для их создания. В данной книге я намереваюсь показать следствия этого неверного процесса и объяснить его происхождение. Затем мы

поговорим о том, как следует изменить процесс, чтобы наши программные продукты стали дружелюбными, мощными и приятными. В этой главе я прежде всего покажу, насколько серьезна эта проблема.

Что получится, если скрестить компьютер с фотокамерой?

Вот загадка информационного века: что получится, если скрестить компьютер с фотокамерой? Ответ: компьютер! Тридцать лет назад в моем первом фотоаппарате, 35-миллиметровом Pentax H, была маленькая батарейка, питавшая экспонометр. Я просто менял батарейку каждые два года, как в наручных часах.

Пятнадцать лет назад в моей первой электронной фотокамере, 35-миллиметровом Canon T70, было две пальчиковых батарейки, приводивших в действие достаточно простой компьютерный блок экспонометра и питавших автоматическую прокрутку пленки. Простой выключатель на аппарате предотвращал ненужные затраты энергии батареек.

Пять лет назад в моем Logitech, цифровом фотоаппарате первого поколения, тоже был подобный выключатель, однако на этот раз в камере уже появились зачатки компьютерных мозгов. Так что если я забывал выключить ее, она автоматически выключалась через минуту бездействия. Симпатично.

Год назад моя цифровая фотокамера второго поколения, Panasonic PalmCam, содержала еще более сообразительную компьютерную микросхему. Настолько сообразительную, что простой выключатель эволюционировал в переключатель Off/Rec/Play. Появились режимы: чтобы снимать, необходимо было перевести камеру в режим Rec, а чтобы просматривать фотографии на маленьком экране – в режим Play.

Моя последняя фотокамера – Nikon CoolPix 900 – цифровой фотоаппарат третьего поколения, и она еще умнее. Настолько умнее, что содержит полноценный компьютер, отображающий песочные часы а-ля Windows при «загрузке». Словно какая-то рыба-мутант с лишними головами, выключатель дорос уже до *четырех* позиций: Off/ARec/MRec/Play. ARec

обозначает автоматическую запись, а MRec – ручную. Насколько я могу судить, разницы между этими режимами нет. Режим On (включено) вообще отсутствует, и без подробных объяснений никто из моих друзей не может сообразить, как включить устройство.

Эта новая камера весьма прожорлива, поэтому создатели заботливо снабдили ее изощренной компьютерной программой, управляющей потреблением энергии батарей. Типичный сценарий выглядит так: я перевожу зловещий переключатель в положение MRec, жду примерно семь длинных секунд, пока камера загрузится, затем направляю ее на предмет съемки. Я нацеливаю камеру и выбираю увеличение, чтобы получить нужный кадр. В тот момент, когда я уже почти нажимаю спуск, камера внезапно осознает, что одновременная зарядка вспышки, увеличение и включение экрана окончательно исчерпали заряд аккумулятора. В порыве самозащиты камера временно отключает возможность снимать. Но я этого не знаю, потому что смотрю через видоискатель, машу руками, говорю «улыбочку» и нажимаю на спуск. Компьютер фиксирует нажатие на кнопку, но не способен повиноваться. В бестолковой попытке спасти ситуацию программа управления питанием мгновенно вмешивается и принимает ответственное решение: снизить нагрузку. Она отключает прожорливый LCD-экран. Я в недоумении смотрю на камеру, силясь понять, почему она не сделала снимок, пожимаю плечами и опускаю руку с камерой. Но после отключения экрана другие подсистемы получают больше энергии батареи. Программа управления питанием ощущает повышение электрического потенциала и осознает, что *вот теперь* электричества достаточно, чтобы сделать снимок. Она возвращает управление основной программе, которая терпеливо ожидает, когда можно будет выполнить мою команду сделать снимок, и делает замечательный цифровой снимок моего колена с автофокусом, экспозицией и с высоким разрешением.

В моем старом механическом Pentax была ручная фокусировка, ручная экспозиция, ручная выдержка, однако пользоваться им было гораздо менее мучительно, чем полностью компьютеризованным современным Nikon CoolPix 900, в котором фокусировка, экспозиция и выдержка автоматические. Эта фотокамера по-прежнему позволяет фотографировать, но *ведет* себя уже не как фотокамера, а как компьютер.

⌘

Лягушка, попавшая в кастрюлю с холодной водой на плите, не осознает, насколько убийственно повышение температуры. Напротив, тепло притупляет чувства лягушки. Подобно лягушке, я не осознавал медленного марша моих фотокамер от простого к сложному по мере их компьютеризации. Все мы переживаем точно такое же, медленное, анестезирующее вторжение компьютерного поведения в нашу повседневную жизнь.

Что получится, если скрестить компьютер с будильником?

Компьютер! Я только что купил для спальни новые дорогие часы со встроенным радиоприемником – JVC FS-2000. Прибор оснащен весьма изощренным компьютерным мозгом, высокоточными часами, цифровым звуком и вообще множеством функций. Он будит меня в заданное время, проигрывая музыку с компакт-диска, и обладает достаточно деликатным характером и достаточной сообразительностью, чтобы ме-е-е-е-едленно прибавлять звук, если дело происходит в шесть утра. Весьма приятная и довольно редкая особенность, компенсирующая мое желание вышвырнуть эту возмутительную машину из окна.

Очень сложно определить, когда будильник включен, поэтому время от времени он пропускает побудку в понедельник и выдергивает меня из кровати рано утром в субботу. Разумеется, в этих часах есть индикатор активности будильника, однако это не означает, что его можно использовать. Встроенный сложный алфавитно-цифровой жидкокристалличесский дисплей (LCD) отображает всевозможные функции часов. Так, маленькое изображение часов в левом верхнем углу дисплея указывает, что будильник включен, однако в полутьме спальни этот символ разглядеть невозможно. Дисплей оснащен подсветкой, благодаря которой символ часов становится видимым, однако подсветка включается, лишь если самостоятельно включить радио или проигрывание компакт-диска. Но тут есть одна тонкость – будильник (даже активизированный) ни за что не сработает, если оставить проигрыватель компакт-дисков включенным. Именно такое парадоксальное поведение часто застает меня врасплох.

Отключить будильник просто: достаточно нажать кнопку Alarm один раз, и часики исчезнут с дисплея. Но чтобы включить будильник, я должен нажать эту кнопку ровно пять раз. После первого нажатия дисплей отображает время, когда сработает будильник. После второго – время, когда будильник перестанет работать. После третьего – источник звука: радио или компакт-диск. После четвертого – предустановленный уровень звука. После пятого часы возвращаются в нормальный режим работы с включенным будильником. Однако всего одно дополнительное нажатие

отключает будильник. В полусне, в темной спальне достаточно сложно правильно исполнить этот маленький цифровой балет.

Будучи занудным поклонником всяких штуковин, я продолжаю свои игры с прибором в надежде справиться с ним. А вот моя жена давно уже сдалась на милость дьявольской машины. Ей нравится гладкий современный дизайн и качество звука, но прибор не прошел аттестацию на часы-будильник, потому что его слишком сложно заставить работать. Этот будильник по-прежнему способен разбудить меня, но *ведет себя*, словно компьютер.

А мой старый некомпьютерный будильник за 11 долларов будил меня внезапным злобным жужжанием. Когда будильник был включен, горела красная лампочка. Когда он был выключен, лампочка не горела. По многим причинам этот старый будильник мне не нравился, однако я, по крайней мере, мог определить, собирается ли он меня будить.

⌘

Производителям гораздо дешевле использовать компьютеры для управления внутренними процессами устройств, чем более старые, механические методы, а потому проникновение компьютеров во все продукты и услуги в нашей жизни экономически неизбежно. Это означает, что поведение всех существующих продуктов скоро станет похожим на поведение самых отвратительных компьютеров, если только мы не применим другой подход.

⌘

Описанное явление не ограничено лишь продуктами для конечного потребителя. Прочти любое компьютеризованное устройство или услуга предлагают больше возможностей и вариантов использования, чем механический эквивалент. Однако на практике именно механическими устройствами мы пользуемся более гибко, более тонко, с большим пониманием, чем современными вариантами тех же устройств, основанными на кремниевых микросхемах.

Высокотехнологические компании, стремясь улучшить свои продукты, просто насыщают их сложными и никому не нужными возможностями. Поскольку ущербный процесс не способен решить проблему некачественных продуктов, а позволяет лишь добавлять новые функции, именно этим создатели продуктов и занимаются. Позже в этой книге я покажу, как усовершенствованный процесс разработки делает пользователей счастливыми, не требуя затрат времени на дополнительные ненужные функции.

Что получится, если скрестить компьютер с автомобилем?

Компьютер! Великолепный новый высокотехнологичный спорткар Boxster от Porsche оборудован семью компьютерами, которые управляют его сложными системами. Один из них занимается исключительно двигателем. В этот компьютер встроены специальные процедуры, позволяющие выходить из критических ситуаций. К сожалению, эти процедуры иногда становятся причиной странных эффектов. В некоторых ранних моделях, если уровень топлива в баке становился очень низким – около четырех литров, – центробежная сила в крутом повороте могла сместить бензин к одной стенке бака, из-за чего воздух попадал в топливную систему. Компьютер фиксировал серьезное изменение в поступающей топливной смеси и интерпретировал это как катастрофический сбой системы впрыска. Чтобы избежать повреждений, компьютер отключал зажигание и останавливал автомобиль. Кроме того, чтобы избежать повреждений, компьютер не разрешал водителю перезапускать двигатель, пока машину не отбуксируют в мастерскую и не починят.

Когда владельцы первых Boxster столкнулись с этой проблемой, компания Porsche смогла предложить им только одно решение: открыть капот и отсоединить батарею как минимум на пять минут, чтобы компьютер смог забыть о заминке. Эти спортивные машины позволяют носиться по двухполосным асфальтовым дорогам, но в острых поворотах теперь *ведут себя* точно как компьютер.

⌘

Прилагая достойные всяческих похвал усилия, чтобы обезопасить владельцев Boxster, программисты превратили этих владельцев в униженных жертв. Каждый приверженец спортивных автомобилей знает, что компания Porsche не скупится на уважение к своим клиентам и предоставляет им множество привилегий. Этот инцидент показывает, что программное обеспечение автомобиля создано не фирмой Porsche, сделавшей другие компоненты автомобиля. Оно создано компанией внутри компании: программистами, а не легендарными немецкими автостроителями. Каким-то образом появление новой технологии вынудило солидную компанию оступиться и упустить из виду некоторые из своих ключевых принципов. Стандарты качества у разработчиков программного обеспечения гораздо ниже, чем в традиционных инженерных дисциплинах.

Что получится, если скрестить компьютер с банком?

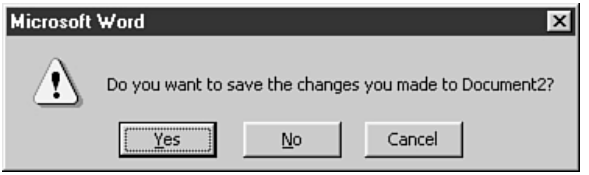
Компьютер! Всякий раз, снимая деньги в банкомате, я сталкиваюсь с одним и тем же угрюмым и сложным поведением, столь присущим компьютерам. Если я сделаю малейшую ошибку, банкомат блокирует всю транзакцию и вышвырнет меня из процесса. Я должен вытащить карту, снова вставить ее, повторно набрать свой PIN-код, а затем повторить запрос. Обычно и ошибка-то не моя, это компьютер банкомата дипломатично сбивает меня с толку. Он постоянно спрашивает, с какого счета я хочу снять деньги – с текущего, с депозитного или же с валютного – хотя счет у меня всего один – текущий. Я постоянно забываю, какого типа у меня счет, и такой вопрос сбивает меня с толку. Примерно раз в месяц я безо всякого умысла выбираю депозитный счет, и адская машина бесцеремонно заставляет меня начать все сначала. Чтобы отказать в снятии денег с депозитного счета, машина должна знать, что у меня такого счета нет, но она предлагает мне этот счет в качестве одного из вариантов. Единственная разница между мной, когда я выбираю банковский счет, и пилотом рейса 965, выбравшим «ROMEO»,– в масштабах последствий.

Кроме того, банкомат налагает «суточное ограничение» в размере двухсот долларов. Если я совершу все шаги – наберу свой код, выберу тип счета, укажу сумму – и это будет, скажем, сумма в 220 долларов, компьютер бесцеремонно откажет в проведении операции, грубо проинформировав меня, что я превысил суточное ограничение. Он не сообщит мне, каково это ограничение, и не позволит узнать, сколько денег на моем счету, и не даст возможности указать другую, меньшую, сумму. Он просто выплевывает мою карту и предоставляет мне возможность повторить процесс с самого начала, при том, что я обладаю ровно той же информацией, что и минуту назад, а очередь за мной растет, волнуется и вздыхает. Банкомат точен и правдив, но совершенно бесполезен.

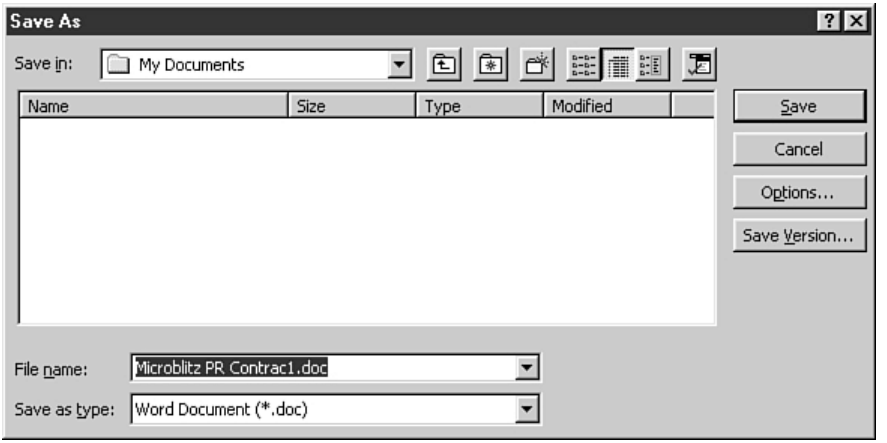
Этот банкомат следует заложенным в него правилам, и я тоже вполне готов им следовать, но это уж слишком компьютерный подход – не сообщить мне о правилах, дать мне противоречивые сведения, а затем бесцеремонно наказать меня за то, что я эти правила по незнанию нарушил. Такое поведение – столь типичное для компьютеров – не присуще им от природы. По сути дела, от природы компьютерам ничего не присуще: они просто действуют, повинуясь программе. А программы пластичны так же, как человеческая речь. Человек может говорить грубо или вежливо, угрюмо или любезно. Так же легко, как человек способен вежливо говорить, компьютер может уважительно и с почтением себя вести. Требуется лишь, чтобы кто-нибудь объяснил, каким образом. К сожалению, программисты не слишком успешно учат компьютеры подобным вещам.

Компьютер позволяет легко попасть в беду

Компьютеры на рабочих столах ведут себя тем самым вызывающим раздражение способом, который им так присущ; им даже не требуется какое-либо скрещивание. Моя подруга Джейн когда-то работала координатором в области связей с общественностью. Она работала в Microsoft Word на своем компьютере под управлением Windows 95 – редактировала записки и контракты. Файловая система в Windows 95 имеет иерархическую структуру. Все документы Джейн хранились в маленьких папках, которые хранились в других маленьких папках. Джейн этого не понимала, равно как не видела преимуществ в таком хранении информации. Вообще говоря, Джейн не слишком над этим задумывалась, а просто следовала по пути наименьшего сопротивления.



Джейн только что вчерне закончила новый контракт на пиар для начинающей компании из Кремниевой долины. Она выбрала Закрыть из меню Файл. Вместо того чтобы сделать как сказано и закрыть документ, программа Word открыла диалог. Разумеется, речь идет о до боли знакомом запросе Do you want to save changes...? (Сохранить изменения?). Она ответила – как обычно – нажатием кнопки Yes. Она так часто давала этот ответ, что даже перестала смотреть на диалоговое окно.



За первым диалогом немедленно последовал еще один – тоже очень знакомый Save As (Сохранить как). Этот диалог явил Джейн множество непонятных кнопок, пиктограмм и текстовых полей. Единственное, что Джейн здесь понимала и чем пользовалась,– поле ввода для имени файла. Она набрала подходящее имя и нажала кнопку Сохранить. Программа сохранила контракт в папке *Мои документы*. Джейн настолько привыкла к этой ненужной процедуре, что проходила ее не задумываясь.

В обед, пока Джейн не было в офисе, Сунил, компьютерный специалист компании, установил на ее машину новую версию антивируса VirusKiller. Работая на компьютере Джейн, Сунил воспользовался программой Word, чтобы просмотреть файл Readme для VirusKiller. Просмотрев файл, Сунил закрыл его и привел компьютер Джейн в прежнее состояние. То есть он так думал.

После обеда Джейн понадобилось вновь открыть контракт, чтобы распечатать его и показать шефу. Джейн выбрала Открыть из меню Файл, и появилось диалоговое окно Открыть. Джейн ожидала, что здесь будут выведены в удобном алфавитном порядке все ее контракты и документы. Вместо этого она увидела кучу имен файлов, которые никогда не видела раньше и не могла опознать. Один из этих файлов назывался *Readme.doc*.

Разумеется, когда Сунил использовал Word для просмотра файла Readme, он велел программе заглянуть в загадочную папку на шестом уровне вложенности файловой системы и безо всякого умысла сбил привычную для Джейн настройку на *Мои документы*.

Джейн была озадачена. Первая и неизбежная мысль была о том, что весь ее тяжелый труд каким-то образом оказался стертым, поэтому она не на шутку разволновалась и позвала Рене, свою подругу и коллегу, однако Рене была сбита с толку не меньше Джейн. Наконец, в состоянии близком к панике, Джейн позвонила Сунилу, чтобы попросить о помощи. Сунила на месте не было, и только в понедельник утром он смог заглянуть к Джейн и все исправить. Джейн, Рене, Сунил и компания, в которой они работали, потеряли каждый по половине дня.

Иерархические файловые системы нужны операционным системам компьютеров, но не людям. Неудивительно, что программистам нравится видеть иерархические файловые системы, но точно так же нет ничего примечательного в том, что обычные пользователи, вроде Джейн, никакого удовольствия от этого не испытывают. Вернее говоря, нет ничего примечательного в этом для всех, кроме программистов, создающих для нас программное обеспечение. Они программируют поведение и отображение информации так, что это подходит для них самих, но это очень сильно отличается от того, что подходит для Джейн. За срыв планов

и низкую эффективность работы винят Джейн, а не программистов, которые ее подставили.

У Джейн, по крайней мере, есть работа. Ведь многих людей считают недостаточно «компьютерно образованными», а потому не подлежащими найму. Все больше и больше позиций требуют взаимодействия с компьютерами, так что пересекать границу между наймом и отсутствием такового становится все труднее. Политики могут требовать создания рабочих мест для неимущих, однако ни одна компания не позволит недостаточно компетентному человеку сесть за компьютер. Необходимо слишком серьезное обучение, и здесь слишком велик риск разрушения информации и порчи бесценных баз данных.

Возмутительное поведение и невразумительность взаимодействий, присущие продуктам, основанным на программном обеспечении, наделяют законным статусом режим, который я называю «апартеидом программного обеспечения». Этот режим не позволяет нормальным в целом людям выходить на рынок труда и жить в обществе, потому что они не могут эффективно использовать компьютеры. В нашем просвещенном обществе социальные активисты неустанно трудятся, чтобы разрушить расовые и классовые барьеры, в то время как технологи тяжким трудом воздвигают новые, еще более высокие барьеры. Целенаправленно проектируя наши программные продукты так, чтобы они были более гуманными и терпимыми к ошибкам людей, мы автоматически сможем сделать их менее восприимчивыми к социальному классу и цвету кожи.

Коммерческое программное обеспечение тоже страдает

Компьютеры захватывают в авиалайнерах не только кабины пилотов, но и пассажирские салоны, и ведут себя там столь же знакомо извращенным и сложным для восприятия способом. Современные реактивные самолеты оборудованы развлекательными системами, позволяющими пассажирам слушать в полете музыку и смотреть фильмы. Эти системы – обычные компьютеры, объединяемые локальными сетями, точно как в вашем офисе. Хорошие развлекательные системы устанавливаются обычно только на крупных самолетах, летающих трансокеанскими рейсами.

Развлекательная система одной авиакомпании оказалась столь неприятной в использовании, что многие стюардессы и стюарды пытались добиться перевода на более короткие местные рейсы, чтобы избежать необходимости изучать и использовать эту сложную вещь. Это примечательно, учитывая, что освященный временем процесс служебного роста на авиалиниях основан на старшинстве, так что именно эти дальние марш-

руты всегда считались наиболее лакомыми кусочками – благодаря продолжительным остановкам в экзотических местах вроде Сингапура и Парижа. Желание стюардов перевестись на непримечательную, неромантическую дерготню рейсов из Денвера в Даллас или Лос-Анджелеса в Сан-Франциско, лишь бы избежать общения с полетной развлекательной системой, свидетельствовало о серьезной проблеме с боевым духом. Любая компания, мучая плохим оборудованием своих наиболее ценных сотрудников, – именно тех, что провели больше всего времени с клиентами, – поступает глупо, расточительно тратя деньги, подрывая преданность клиентуры и собственного персонала.

Компьютерный комплекс другой крупной авиакомпании был еще хуже. Эта авиакомпания создала полетную развлекательную систему, связавшую показ фильма с функцией сбора денег. Раньше в закрытом реактивном самолете на высоте одиннадцати километров процедура сбора денег основывалась на принципах доверия: никто ведь не сбежит из самолета на такой высоте. Стюарды обеспечивали пассажиров товарами и услугами, когда это было удобно, а сбор платы был весьма слабо связан с этим процессом, и сотрудникам авиакомпании не приходилось без нужды бегать взад-вперед по узким проходам. Конечно же, время от времени случались ошибки, но их стоимость никогда не превышала нескольких долларов, а система в целом была достаточно человечной и способной прощать ошибки; все были довольны, и работа была не в тягость.

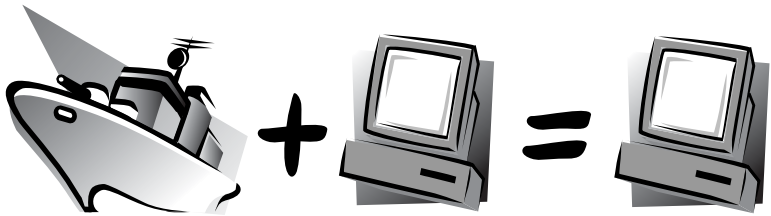
После компьютеризации процесса приема денег за услугу стюарду сначала приходилось получать плату от пассажиров, затем идти в самое начало салона, где находится консоль управления, набирать пароль обслуживающего персонала и выполнять транзакцию по регистрации оплаты. Лишь когда транзакция завершится, пассажир сможет посмотреть фильм или послушать музыку. Столь бестолковое проектирование продукта заставляло стюардов сотни раз впустую ходить взад-вперед по узким проходам – в течение одного рейса. От отчаяния стюарды устраивали короткое замыкание в начале каждого длинного рейса вскоре после взлета. Затем они вежливо объявляли пассажирам, что, к сожалению, система неисправна и в *этом* рейсе фильмов не будет.

Авиакомпания потратила миллионы долларов на создание системы столь отвратительной, что пользователи преднамеренно отключали ее, лишь бы не иметь с ней дела. Тысячи скучающих пассажиров – просто невинные жертвы. И все это на долгих трансокеанских рейсах, забитых, как правило, бесценными постоянными клиентами. Не могу назвать точную цифру потерь авиакомпании, но убежден, что все это обошлось ей катастрофически дорого.

Программное обеспечение полетных развлекательных систем работало с безупречной точностью, но не умело взаимодействовать с людьми, и поэтому его внедрение закончилось полным провалом. Почему компания не могла предвидеть столь печальный исход? Почему не увидела эту связь? Цель данной книги – ответить на такие вопросы и показать, как избежать подобных высокотехнологических катастроф.

Что получится, если скрестить компьютер с военным кораблем?

В сентябре 1997 года, участвуя в морских маневрах в Атлантике, корабль ВМФ США *Yorktown*, один из новых крейсеров с оборонительной системой Aegis,[1] замер на месте. Техник ВМФ, калибруя топливный клапан, ввел нулевое значение в один из управляющих компьютеров – с процессором Pentium Pro и операционной системой Windows NT. Программа попыталась разделить другое число на этот нуль, то есть выполнить операцию, не определенную в математике, что и стало причиной сбоя всей системы управления бортом. Без участия компьютеров двигатель прекратил работать, и корабль два часа сорок пять минут качался на волнах, пока не прибыл буксир. Хорошо, что это произошло не в зоне боевых действий.



Что получится, если скрестить компьютер с военным кораблем? Адмирал Нимиц[2] в гробу перевернулся бы! Несмотря на описанную неудачу в ВМФ приняли решение о компьютеризации всех кораблей, поскольку это позволяло сэкономить на персонале, а чтобы отразить критику, объявили причиной «происшествия» человеческий фактор. Раз процесс создания программного обеспечения вышел из-под контроля, индустрия высоких технологий должна либо привести этот процесс в порядок, либо продол-

[1] Aegis (англ. *эгида*) интегрирует системы обнаружения и боевые системы корабля с целью противостояния ракетным атакам. – *Примеч. перев.*

[2] Честер Нимиц, командующий Тихоокеанским флотом США во время Второй мировой войны. – *Примеч. перев.*

жать сваливать вину на обычных пользователей, в то время как все более грандиозные механизмы беспомощно плещутся в воде.

Техноярость

В недавнем номере *Wall Street Journal* появилась статья, посвященная анонимному видеоклипу, широко распространившемуся посредством электронной почты. В клипе «...Усатый Рядовой Гражданин в рубашке с коротким рукавом озадаченно склонился над компьютером. Внезапно, в порыве раздражения, он ударяет по своему монитору. Любопытствующий коллега заглядывает в его отсек, в то время как этот человек лупит клавиатурой по монитору, сшибая его на пол. Поднявшись со своего места, он добивает упавший монитор последним жестоким ударом».

В статье говорилось, что реакцию этот клип вызвал «значительную» и что он, очевидно, затронул «мощную скрытую тенденцию к техноярости».

По иронии судьбы, чтобы даже просто переслать этот видеоклип, необходима умеренная подготовка в компьютерной области. Человек на видео может быть и актером, но он затронул знакомую бизнес-миру струнку сочувствия. В нашей жизни раздраженность сложными и неприятными продуктами, основанными на программном обеспечении, быстро растет.

В закрытых списках рассылки имеют хождение шутки о «компьютерном синдроме Туретта». Это вариация на тему психического расстройства, известного как синдром Туретта. Некоторые из подверженных этому расстройству людей переживают неконтролируемые припадки сквернослo-

вия. Шутка в том, что можно пройти по коридорам практически любого современного офисного здания и услышать, как в целом нормальные люди, сидя за своими компьютерами и сжав зубы, постоянно и яростно ругаются. Кто знает, что вызвало такую вспышку: потерянный файл, недоступное изображение или же раздражающее взаимодействие. А может быть, программа просто вежливо стерла единственную копию пятисотстраничной рукописи пользователя, потому что он ответил «Да» на диалог с подтверждением, предположив, что ему предлагается сохранить изменения, когда в действительности предлагалось удалить работу.

Индустрия в «несознанке»

Наш мир опьянен высокотехнологичными инструментами. Компьютеры господствуют на рабочих местах и у нас дома, транспортные средства заполняются примочками, основанными на кремниевой технологии. Каждое из этих мощных, изощренных компьютеризованных устройств обескураживающе сложно и нелогично в применении.

Индустрия высоких технологий отказывается признать простой факт, очевидный каждому владельцу мобильного телефона или текстового редактора: *наши компьютеризованные инструменты слишком сложно применять.* Инженеры, создающие программное обеспечение и высокотехнологичные устройства, довольны собственными усилиями. Разработчики программного обеспечения[1] пытаются в меру возможностей сделать эти инструменты простыми в применении и немного в этом преуспели. Они полагают, что их продукты настолько просты в применении, насколько это технически возможно. Будучи инженерами, они доверяют технологии и верят в то, что лишь новая технология – скажем, распознавание голоса или искусственный интеллект – способна улучшить опыт для конечных пользователей.

По иронии судьбы, вероятно, *наименьший* вклад в простоту использования продуктов, основанных на программном обеспечении, внесет именно новая технология. *Технически* разницы между сложной, запутанной программой и простым, приятным, мощным продуктом практически нет. Вопрос скорее в культуре, подготовке, отношении людей, создающих эти продукты, нежели в микросхемах и языках программирования. Ущербен наш *процесс* разработки, а не инструменты.

Индустрия высоких технологий по недосмотру поставила во главу процесса программистов и инженеров, поэтому доминирует их сложная

[1] В компьютерной индустрии термин «разработчик программного обеспечения» употребляется в качестве синонима термина «программист»; то же самое делаю и я в этой книге.

в применении инженерная культура. Несмотря на кажущиеся полномочия, люди на руководящих постах попросту не контролируют индустрию высоких технологий. Этим шоу заправляют инженеры. В своем стремлении принять многочисленные преимущества кремниевых микросхем мы отреклись от ответственности. *Мы позволили пациентам завладеть психбольницей.*

Когда психбольница в руках пациентов, им сложно четко осознать природу собственных проблем. Смотрясь в зеркало, слишком уж просто сконцентрироваться на лучших своих чертах и забыть о недостатках. Когда создатели продуктов, основанных на программном обеспечении, изучают плоды своей ручной работы, они не понимают, насколько эта работа плоха. Они видят только грандиозную мощь и гибкость. Они видят, насколько продукт богат возможностями и функциями. Они игнорируют то, насколько мучительно сложно использовать продукт, сколько часов приходится через силу его изучать или как он унижает и деморализует людей, которым приходится использовать продукт ежедневно.

Мотивы создания этой книги

Двадцать пять лет я изобретал и разрабатывал продукты, основанные на программном обеспечении. Многие годы я ломал голову над проблемой сложного в применении программного обеспечения. Наконец, в 1992 году, я прекратил программировать, чтобы посвятить все свое время компаниям-разработчикам, помогая им делать свои продукты более простыми в применении. И случилась удивительная вещь! Избавившись от требований, которые диктовало занятие программированием, я впервые осознал, насколько мощными и всеподчиняющими были эти требования. Программирование – задача настолько всепоглощающая и сложная, что она доминирует над всеми иными соображениями, включая и заботу о пользователе. Я смог понять это лишь после того, как освободился из капкана программирования.

Совершив такое открытие, я начал понимать, почему программные продукты настолько плохи с точки зрения пользователя. В 1995 году в книге «About Face: The Essentials of User Interface Design»[1] я рассказал о том, что узнал, и она оказала существенное влияние на разработку некоторых программ.

Чтобы стать хорошим программистом, необходимо сочувственно относиться к природе и потребностям компьютера. Однако природа и потреб-

[1] А. Купер, Р. Рейман, Д. Кронин «Алан Купер об интерфейсе. Основы проектирования взаимодействия» (перевод «About Face 3: The Essentials of Interaction Design», изданной в 2007 г.). – СПб.: Символ-Плюс, 2009.

ности компьютера совершенно чужды природе и потребностям человеческого существа, которому придется в конечном итоге этот компьютер использовать. Создание программного обеспечения требует таких интеллектуальных усилий, так поглощает программистов, что им приходится полностью погружаться в объективно чуждый человеку мыслительный процесс. Для программиста потребности процесса программирования получают приоритет перед любыми потребностями пользователей из внешнего мира, и более того – даже языки этих двух миров конфликтуют.

Процесс программирования оттесняет на обочину процесс создания легких в использовании продуктов по той простой причине, что цели программиста и пользователя коренным образом различаются. Программист желает, чтобы процесс создания протекал гладко и легко. Пользователь желает, чтобы легко и гладко происходило взаимодействие с программой. Эти две цели практически никогда не приводят к созданию одной и той же программы. В современной компьютерной индустрии программисты отвечают за создание взаимодействий, приятных для пользователя, однако, находясь в безжалостном капкане конфликта интересов, они просто не могут это делать.

В области программного обеспечения обычно невозможно увидеть результаты, пока работа не завершена, и это значит, что любая рецензия со стороны непрограммиста появится слишком поздно, чтобы дать какой-то эффект. Программное обеспечение для настольных компьютеров имеет дурную репутацию потому, что является исключительно продуктом деятельности программистов; между пользователем и программистами других людей нет. Вещи вроде телефонов и камер всегда имели видимые механические детали, которые делали их легкими для изучения. Однако, как мы установили, если скрестить компьютер практически с чем угодно, стиль поведения компьютера одерживает абсолютную победу.

Ключ к решению этой проблемы – проектирование взаимодействия. Нам нужен новый вид профессиональных проектировщиков взаимодействия, которые станут проектировать поведение программного обеспечения. Сегодня программисты сознательно проектируют «код» программ, но лишь непреднамеренно проектируют взаимодействие с людьми. Они проектируют возможности, но не то, как программа *ведет себя*, *общается* или *уведомляет*. Напротив, проектировщики взаимодействия сосредотачиваются непосредственно на том, как пользователи воспринимают продукты, основанные на программном обеспечении, и взаимодействуют с ними.

Ремесло проектирования взаимодействия – новое, оно не знакомо программистам, так что – если программисты вообще это признают – ему уделяется внимание лишь после того, как программирование завершено. Но в этот момент уже слишком поздно.

Люди, управляющие созданием продуктов, основанных на программном обеспечении, либо являются заложниками программистов, будучи недостаточно подкованными технически, либо слишком сочувствуют программистам, потому что сами таковыми являются. Пользователи этих продуктов просто не имеют понятия, что эти продукты могут быть приятными в применении и такими же мощными, как любой другой качественно спроектированный инструмент.

Программисты вовсе не злодеи. Они много работают, чтобы сделать свои *программы* легкими в использовании. К сожалению, судят они по себе, так что программы получаются легкими в использовании лишь для других разработчиков программного обеспечения, но не для обычных людей.

Стоимость некачественно спроектированных программ неисчислима. Стоимость времени Джейн и Сунила, стоимость раздражения пассажиров авиалайнера, стоимость жизней пассажиров рейса 965 просто невозможно измерить. А наибольшая растрата – потерянная возможность. Позволяя продуктам приводить нас в отчаяние, увеличивать наши затраты, запутывать, раздражать и убивать нас, мы не пользуемся реальным преимуществом программных продуктов, которые обещали стать наиболее человечными, мощными и приятными творениями из когда-либо выдуманных. Поскольку программное обеспечение сделано из самого податливого материала, оно обладает и потенциалом превзойти ожидания даже самого безумного мечтателя. И требуется лишь разумное сотрудничество проектировщиков взаимодействия и программистов.