**14 принципов создания вежливых программ**

Профессора Стэндфордского Университета Клиффорд Насс (Clifford Nass) и Байрон Ривз (Byron Reeves) занимались изучением реакции человека на компьютер. Применяя классические методы социальной психологии, они обнаружили в поведении людей нечто интересное. Результаты их работы, опубликованные в книге "The Media Equation", показывают, что человек реагирует на компьютер так же как на других людей.

Для нашего мозга компьютеры ведут себя больше как люди, а не как камни и деревья. Таким образом мы бессознательно относимся к ним, как к людям, даже тогда, когда считаем это бессмысленным. Другими словами, человек обладает определенными инстинктами, которые говорят ему, как вести себя в обществе других чувствительных созданий, поэтому как только некий объект проявляет определенные познавательные способности, эти инстинкты начинают действовать и мы начинаем реагировать на этот объект так же, как на других людей. Это бессознательная и неизбежная реакция срабатывает у каждого.

Насс и Ривз использовали для тестов множество студентов-компьютерщиков, достаточно опытных для того чтобы самим написать тестовую программу. Эти студенты были высокообразованными, зрелыми индивидуумами, но не смотря на это все без исключения отрицали, что благодаря своим познавательным способностям программы воздействуют на них эмоционально, хотя результаты наблюдений являлись неопровержимым доказательство обратного. Из этого исследования следует важный вывод: Если мы хотим, чтобы наша программа понравилась пользователям, мы должны сделать ее поведение похожим на поведение человека. Довольно просто, не правда ли? Насс и Ривз утверждают что программы должны быть “вежливыми”, потому что вежливость – это универсальный человеческий признак, - хотя действия, которые можно считать вежливыми различаются от одной культуры к другой, этот признак присутствует в любой культуре. Продукты производства с высокими познавательными способностями, такие как программы, тоже должны следовать этому правилу и быть вежливыми.

Некоторые продукты высоких технологий ведут себя так, как будто сказав “пожалуйста” или “спасибо”, можно быть грубым, но это не вежливость. Если программа скупа на информацию, скрывает результаты своей работу, заставляет пользователя искать где находятся простейшие функции, и винит его в своих собственных неудачах, то пользователю она точно не понравится. Это произойдет независимо от “пожалуйста” и “спасибо”. Это также не зависит и от того, насколько находчивой, представительной, метафоричной, наполненной содержанием или персонализированной она будет. Если же программа уважает пользователя и помогает ему, то она обязательно ему понравится. И снова, это произойдет независимо от ее интерфейса; интерфейс командной строки тоже будет нравиться, если он будет обладать вышеуказанными качествами. Что значит для программы быть дружественной и вежливой? Что значит для программы вести себя подобно человеку? Торговцы “Гербалайфом” одеты в красивые костюмы, широко улыбаются и полны впечатляющей информации, но разве они нам нравятся?

Человек склонен к ошибкам, медлителен и импульсивен, но это не значит, что программа с такими качествами будет считаться хорошей. Человек обладает множеством других качеств делающих его хорошо подходящим для роли служащего – роли, которую выполняют большинство программ. Большинство хороших программистов совершенно несведущи в вежливости. Программисты выразительны и точны до крайности, но только когда они так чувствуют. Их способ общения так точен, что они могут показаться практически неспособными к общению. Концепции вежливости или даже человечности, могут быть камнем преткновения для программистов. Они не в состоянии понять такие расплывчатые концепции. Они борются с идеей сделать компьютеры более похожими на людей, потому что они считают людей слабыми и несовершенными компьютерными устройствами.

Я спросил моего друга Кейта Плиса (Keith Pleas), известного программиста-эксперта, занимающегося к тому же вопросами пользовательского интерфейса, что он думает о создании более человечных программ. Кейт воспринял добавление в программу человечности как добавление нечеткости во взаимодействие. Он ответил: Может ли компьютер “лгать” вам? Может ли компьютер сказать вам, что у вас на счету “около 500$”? Может ли компьютер дать вам другой ответ, чем только что кому-то еще? Если мы увеличиваем человечность, мы должны уменьшить некоторую “компьютерность”, по крайней мере в сравнении. С точки зрения программиста ответ Кейта выглядит естественным. Действительно, компьютер никогда не выдаст вам приблизительный баланс, но тогда компьютер не увидит разницы между выдачей сообщения о том, что у вас на счету “около 500$” за долю секунды, и точной суммы 503.47$ за 17 минут. Более вежливая, более человечная программа сразу бы сообщила, что у вас на счету “около 500$” а затем проинформировала бы вас, что даст более точный ответ через несколько минут. Тогда выбор будет за вами – стоит ли тратить время на дополнительную точность.

Человек обладает множеством качеств, которые делают его “вежливым”, но их определения туманны и расплывчаты. Вот мой список того, что улучшает качество взаимодействия как с другими людьми, так и с программами. Вежливая программа интересуется мной. Мой друг всегда интересуется мной, и моими предпочтениями. Он запомнит что я люблю, а что нет, чтобы в будущем доставить мне удовольствие. Любой человек, предлагающий какие-либо услуги, попытается запомнить имена и лица своих клиентов. Некоторым нравится, когда их приветствуют по имени, некоторым нет, но каждому нравится, когда к нему относятся в соответствии с его личными вкусами. Большинство программ не знает ничего о том, кто ею пользуется. Действительно, ни одна из программ на моем персональном компьютере не помнит ни меня, ни моих привычек, несмотря на то, что только я и никто другой постоянно, снова и снова пользуюсь ею. Каждая программа должна стараться запомнить мои привычки, и в частности, все, что я ей говорю.

Программист считает реальный мир миром информации, так что как только программе требуется какая-либо информация, она просто требует ее от пользователя. Но бездумная программа забывает эту информацию, считая что всегда может затребовать ее снова, если потребуется. Компьютеры и так лучше всего подходят для хранения информации, так что забывать ее по крайней мере невежливо. Например, в моей программе электронной почты есть 11 человек по имени Дейв. С большинством из них я общаюсь редко, но этот список включает в себя Дейва Карлика (Dave Carlick), которому я пишу письма постоянно. Когда я создаю новое письмо и печатаю “Дейв” в графе “Кому”, я ожидаю, что на основе моего поведения в прошлом программа догадается, что я имею в виду Дейва Карлика. Если бы я хотел послать письмо другому Дейву (например Дейву Форе), я бы написал Дейв Ф, Девид Форе или что-то еще, чтобы указать на мой неординарный выбор. Вместо этого, программа ведет себе довольно глупо, всегда предлагая мне окно для выбора нужного Дейва из 11 вариантов.

Программе попросту наплевать на меня, она относится ко мне, как к чужаку, хотя я – единственный человек, которого она знает. Вежливая программа относится ко мне иначе, чем к другим. Любой хороший представитель сферы услуг уважительно относится к своему клиенту. Он понимает, что тот, кому он оказывает услуги - это его босс, и что бы босс не хотел, он должен получить это. Когда владелец ресторана указывает мне на столик, я считаю это предложением, а не приказом. Если я вежливо возражу, и выберу другой столик в пустом ресторане, я ожидаю что меня немедленно там разместят. Если хозяин отказывается сделать это, я скорее всего покину этот ресторан и поищу другой, где мои желания имеют приоритет перед желаниями владельца. Невежливая программа контролирует действия пользователя, которые кажутся ей некомпетентными. Нет ничего страшного в том, чтобы программа выразила свое мнение о том, что я сделал ошибку, но она ни в коем случае не должна судить мои действия. Например, программа может утверждать, что я не могу отправить заказ, пока не введу свой номер социального страхования, но если я все таки захочу отправить заказ без него, программа должна сделать так как я сказал.

Вежливая программа предусмотрительна. Если я спрашиваю у служащего аэропорта, через какой выход пройти на рейс 729, я ожидаю, что он не только ответит на мой вопрос, но и даст мне важную информацию о том, что вылет рейса 729 задерживается на 20 минут. Если делаю заказ в ресторане, должно быть ясно, что мне также нужен нож, вилка, ложка, соль, перец, и салфетки. Большинство программ не делает этого. Они лишь кратко отвечают на мои вопросы, не пытаясь проявить предусмотрительность в отношении другой информации, даже если она напрямую связана с моими целями. Если я попрошу свой текстовый процессор распечатать документ, он никогда не сообщит мне, что бумаги осталось мало, или что в очереди уже находятся 40 других документов. Вежливая программа обладает здравым смыслом. Хотя любой хороший ресторан позволит вам побывать на своей кухне, но когда вы впервые входите в него, официант, руководствуясь здравым смыслом, проводит вас в обеденный зал.

Большинство программ не делают различия между кухней и обеденным залом, помещая рядом часто-используемые и никогда не используемые элементы управления. В программе можно встретить пункты меню предлагающие простые, безвредные функции рядом с чрезвычайными, отменить которые невозможно. Это как если бы вас усадили за стол рядом с духовкой. Упомянутый ранее пример "около 500$"- хорошая иллюстрация наличия здравого смысла в интерфейсе. Мы то и дело слышим ужасные истории о компьютерных системах, которые шлют клиентам чеки на 0.00$ или счета на 8,943,702,624.23$. Изолировав клиентов от компьютерных систем, можно облегчить труд отдела обслуживания клиентов, однако служащим все равно приходится ими пользоваться. Служащим платят за это, поэтому они стараются не слишком сильно жаловаться, тем более что жаловаться то некому – отдел обслуживания клиентов явно не для них. Вежливая программа предвосхищает мои нужды.

Мой ассистент знает, что мне нужна комната в отеле, когда я еду в другой город на конференцию, несмотря на то, что я не говорил ему этого специально. Он знает, что мне нужна тихая комната и заказывает ее без всякого напоминания с моей стороны. Он предвосхищает мои нужды. Мой web-браузер проводит большую часть своего времени в пустом ожидании, пока я просматриваю загруженные страницы. Однако он может очень просто предвосхитить мои потребности и подготовится к ним, вместо того чтобы тратить время впустую. Почему бы не использовать это время, чтобы предварительно загрузить страницы, ссылки на которые видны в окне. Вполне вероятно, что я вскоре попрошу браузер загрузить ту или иную ссылку. Легче остановить запрос, если он окажется ненужным, чем ждать его выполнения. Вежливая программа отзывчива. Когда я обедаю в ресторане, я ожидаю, что официант будет реагировать соответствующим образом на мои намеки. Когда я веду оживленную беседу с соседями за столом, я ожидаю что официант в это время будет выполнять другие дела. Было бы неуместным, если бы он прервал наш разговор и сказал "Привет, меня зову Рауль, и я буду вашим официантом на этот вечер". С другой стороны, как только наша беседа закончится, и я начну вертеть головой в поисках Рауля, я жду, что он сразу же поспешит к моему столику, чтобы узнать, что мне нужно.

У меня на компьютере обычно установлено разрешение экрана 1024х768. Когда я провожу презентации, мне необходимо временно сменить разрешение на 800х600 чтобы оно соответствовало низкому разрешению моего видео-проектора. Многие из запущенных программ, включая Windows 95, реагируют на смену разрешения изменением размера, формы и положения на экране своих окон. Однако когда я меняю разрешение обратно, окна не возвращаются к своим размерам. Информация об этом легко доступна, однако программа не заботится о моих очевидных нуждах. Вежливая программа умалчивает о своих проблемах. В баре, салоне и психиатрическом кабинете бармен, парикмахер и доктор будут умалчивать о своих проблемах и показывать интерес в ваших. Такова природа сферы обслуживания. Программы тоже должны молчать о своих проблемах и интересоваться вашими. Так как у компьютеров нет собственного “я” и чувств, они отлично подходят для подобной роли, однако они обычно ведут себя наоборот.

Программы всегда досаждают мне подтверждающими сообщениями и ненужными строками состояний. Я не хочу знать, насколько трудна работа компьютера. Меня не интересую затруднения программы в вопросе когда чистить “корзину”. Я не хочу слышать ее нытье о том, что она не уверена, в какое место на диске записать файл. Я не хочу слышать свист модема или наблюдать информацию о скорости передачи данных, так же как я не хочу слышать о разводе бармена, сломанном автомобиле парикмахера или алиментах доктора. Из этого следуют два вывода. Программа не только должна молчать о своих проблемах, но и должна уметь решать их сама. Вежливая программа хорошо информирована. С другой стороны, все мы должны знать, что происходит. Тот же бармен помогает мне, повесив список цен передо мной на стенке. Владельцы магазинов должны держать своих покупателей в курсе того, что может их заинтересовать. Я не хочу узнать 21 ноября в мясной лавке, что индюшки на День Благодарения кончились. Я хочу знать заранее, что объем партии ограничен и заказ нужно сделать заранее.

Когда я ищу информацию в Интернет через поисковую машину, я не могу быть уверен, что не наткнусь на неработающую ссылку. Я выбираю нужную мне ссылку и получаю противное сообщение об ошибке “404 Link Not Found”. Разве поисковая машина не может периодически проверять каждую ссылку? Если ссылка неверна, ее можно удалить из списка, и мне не придется тратить время на ожидание ее загрузки. Программы постоянно предлагают мне варианты выбора, которые, по разным причинам, в данный момент недоступны. Программа должна знать это, и не выводить их на экран.

Вежливая программа восприимчива. Консьерж Нью-Йоркского отеля, где я часто останавливаюсь, заметил мой интерес к Бродвейским шоу. И теперь, когда бы я ни остановился в этом отеле снова, консьерж – без моей просьбы – оставляет в моей комнате. Он оказался достаточно восприимчив к моим интересам, что позволило ему предположить мои желания и обеспечить меня нужной информацией до того, как я сам подумаю о ней. Для этого консьержу не требуется много усилий, но в результате я возвращаюсь в этот отель снова и снова. Я раскрываю на весь экран окно любой запущенной программы. Затем я использую Панель Задач для переключения между запущенными приложениями. Но программы, которые я запускаю, не замечают этот факт. Я так часто максимизирую окна, что мои предпочтения должны быть ясными и однозначными. Другие пользователи работают с программами в маленьком окне, чтобы видеть Рабочий Стол. Так просто для программы понять это и предугадать действия пользователя.

Вежливая программа уверена в себе. Я ожидаю от служащих, с которыми я общаюсь определенной смелости и уверенности в себе. Если они увидят, что я выхожу из уборной с расстегнутой ширинкой, я хочу чтобы кто-нибудь быстро, четко и ненавязчиво сказал мне об этом до того, как я войду в зал, чтобы произнести речь. Для этого требуется определенная смелость, но эта смелость будет оценена. Точно также, если мой ассистент не сможет взять билеты на нужный мне рейс, я ожидаю, что он закажет билеты на наиболее подходящий, не беспокоя меня из за таких мелочей. Если я прошу компьютер удалить файл, я не хочу чтобы он спрашивал у меня “Вы уверены?” Конечно я уверен, иначе бы я не спрашивал. Я хочу, чтобы он имел смелость в своих убеждениях и удалил файл. С другой стороны, если компьютер имеет какие-то подозрения, что я могу ошибаться – что может быть всегда – он должен предусмотреть это, и подготовиться к возможному восстановлению файла, если я вдруг передумаю. В любом случае, программа должна быть уверенной в своих действиях, а не перекладывать ответственность на меня. Очень часто после длительной работы с документом я нажимаю кнопку “Печать” и ухожу выпить чашечку кофе, пока документ распечатается. Затем я возвращаюсь чтобы обнаружить посреди экрана бессмысленное и пугающее окно диалога с вопросом “Вы действительно хотите печатать?” Такая неуверенность просто приводит в ярость, и это антитеза вежливого поведения.

Вежливая программа не задает лишних вопросов. Когда я заказываю салат в хорошем ресторане, я получаю хороший салат. В третьесортном ресторане мне подают его несвежим вместе с кучей вопросов “Со шпинатом? С луком? С гренками? Посыпанный сыром? Полную порцию или среднюю? Масло французское или итальянское? Подавать до или после главного блюда?” Даже самый требовательный гурман не захочет никакого салата с таким отношением, однако программы все время ведут себя подобным образом. Adobe Photoshop например осыпает пользователя множеством ненужных маленьких вопросов, каждый в отдельном окне. Невежливая программа задает множество раздражающих вопросов. Когда выбор предлагают насильственно – это тяжелое испытание. Выбирать можно разными способами. Возьмем, к примеру, разглядывание витрин магазинов. Мы обычно смотрим на витрины праздно, обдумывая, выбирая, или игнорируя товары, которые нам предлагают. С другой стороны, иногда нам насильственно предлагают выбор, как например на таможне: “У вас есть что-нибудь нуждающееся в декларации?”. Если нас поймают, последствия могут оказаться значительными. Но мы не знаем, что последует за этим вопросом. Будут нас обыскивать или нет? Если мы знаем, что обыска не избежать, мы никогда не будем лгать. А если мы знаем, что никакого обыска не будет, нас будет одолевать искушение провезти лишнюю пачку Мальборо.

Вежливая программа является подстраиваемой. Когда система ручной обработки информации переносится на компьютеры, что-нибудь всегда теряется. Чаще всего система компьютеризируется для увеличения объемов обрабатываемой информации, а не для изменения свой функциональности. Однако системам работающим вручную присуща гибкость – свойство, которое на так просто выделить среди остальных. Хотя автоматизированная система ввода заказов может обработать в миллион раз больше заказов, чем простой служащий, служащий всегда может подстроить работу системы. В автоматизированной системе эта возможность исчезает. Нет практически никакой возможности изменить работу той или иной функции. Когда служащему звонит знакомый из отдела продаж и объясняет, что можно получить большую выгоду если вот этот заказ обработать быстрее, он ускоряет его обработку. Когда поступает какой-нибудь заказ с неполной информацией, он может обработать его и вернуться к нему позже, чтобы восполнить информацию. Такая гибкость начисто отсутствует в компьютеризированных системах.

Компьютерные системы могут находиться только в двух состояниях: не-существование или полное соответствие; любые промежуточные состояния не принимаются или игнорируются. Любая “ручная” система имеет важное но парадоксальное состояние неопределенности (о котором не говорят, но часто пользуются), когда транзакция может быть принята, но полностью еще не обработана. Оператор создает это состояние у себя в голове. Например, автоматизированной системе требуется информация и о клиенте и о его заказе для того, чтобы выписать счет. Тогда как служащий может выписать счет не имея полной информации о клиенте, автоматизированная система не сможет выполнить такой запрос, не желая выписывать счет. Я называю эту способность человека действовать вне существующей схемы обработки информации до того, как все необходимы реквизиты будут доступны подстраиваемостью. Отсутствие это качества – одна из главных причин нечеловечности компьютерных систем. Это прямое следствие модели воплощения. Программисты не видят никакого резона в том, чтобы создавать промежуточные состояния, потому что в компьютере они не нужны. Но пользователь должен иметь возможность слегка “подправить” систему. Одно из больших преимуществ подстраиваемой системы – уменьшение числа ошибок. Допуская существование в системе временных маленьких ошибок и доверяя человеку в том, что он позже исправит их, можно избежать более серьезных ошибок. Однако большинство правил исходящих от компьютерных систем направлены на то, чтобы не допустить этих маленьких ошибок. Эти негибкие правила делают программу и человека соперниками, и поскольку человеку не дают подстраиваться, чтобы избежать больших ошибок, он вскоре перестает заботится о защите программы от более серьезных проблем. Когда такие правила применяются к человеку, страдают обе стороны.

Подстраиваемость – одно из немногих качеств человека, связанных с вежливостью, которое трудно встроить в компьютерную систему. Подстраиваемость требует от интерфейса больших возможностей. Для того, чтобы быть подстраиваемой система должна сделать свой внутренний процесс доступным опытному пользователю. Служащий не сможет переместить документ в начало очереди, пока он не будет четко видеть саму очередь, ее начало и конец, документ и его положение в очереди. Далее ему должны быть доступны инструменты для того чтобы вытащить документ из очереди и поместить его в самое начало. Физическая реализация подстраиваемости требует специальных средств для хранения записей в состоянии неопределенности, но похожие средства требуются для операций отмены (undo). Настоящая проблема в том, что подстраиваемость делает возможным мошенничество и злоупотребление. Подстройку системы можно расценить как мошенничество. Технически это нарушение правил. В реальном мире на это закрывают глаза, потому что это специальный случай, и подразумевается что тот, кто “подстроил” систему приведет все счета в порядок до конца рабочего для или данной работы. Все подобные примеры должны конечно быть “подчищены” до того, как придет проверка.

Если бы процесс временной приостановки правил стал бы широко известен, у людей появилось бы желание злоупотребить этим. Можно привести множество рациональных и логичных причин не использовать подстраиваемые системы. Но к несчастью, такое идеализированное состояние дел не является точным описанием работы реального мира. Люди использует подстраиваемость “ручных” систем во всех видах бизнеса чтобы удержаться на плаву. Несмотря на все препятствия, наполнение автоматизированных систем подобным качеством является жизненно важным вопросом. Для предотвращения мошенничества можно воспользоваться возможностями компьютера следить за всеми действиями пользователя и показывать эту информацию специальному наблюдателю. Принцип здесь простой – позволить пользователю делать все что он захочет, но подробно записывать все его действия, так что ответственность остается.

Вежливая программа приносит удовлетворение. В программировании трудно достичь удовлетворения. Компьютер не сделает ничего, пока вы не затратите огромных усилий на написание программы. Поэтому программисты создают свои программы точно такими же. Их программы заставляют пользователя вводить всю возможную информацию, а сами и пальцем не пошевелят. Если бы кто-то из людей вел себя подобным образом, вам бы он очень не нравился. Мы можем сделать программы значительно более вежливыми, если они будут работать для того, чтобы предоставить пользователю всю возможную информацию без значительных усилий с его стороны. Вежливая программа внушает доверие. Друзья доверяют друг другу, потому что зависят друг от друга, и всегда готовы пожертвовать собой. Когда компьютеры ведут себя совершенно неуправляемо и неохотно выполняют задания пользователей, ни о каком доверии и речи быть не может. Я доверяю банковскому служащему, потому что он улыбается мне, но я всегда пересчитываю свои деньги после банкомата, потому что попросту не доверяю тупой машине.

Программы раздражают нас не из-за недостатка возможностей, а из-за отсутствия вежливости. Как показывает приведенный список характеристик, в большинстве случаев сделать вежливую программу ничуть не труднее, чем невежливую. Просто кто-то должен предусмотреть взаимодействие, имитирующее качества чувствительного и заботливого друга. Ни одна из этих характеристик не отличается от других, более очевидных целей информационного бизнеса. Более человеческое поведение – самая очевидная цель из всех.

**Список литературы**

Alan Cooper. 14 принципов создания вежливых программ.