Пять типов тестирования в процессе локализации

Ричард Сайкс (Richard Sikes)

Стараясь добиться совершенства конечного продукта, что само по себе далеко нетривиальная задача, мы обычно стремимся обеспечить баланс времени, стоимости и качества. При таком подходе не учитывается собственно процесс как фактор, объединяющий все три составляющие.

В итоге считается, что при правильной организации процесса качество не обязательно будет связано с дополнительными затратами. В действительности верным может оказаться как раз обратное, но только в том случае, если участники разработки системы доставки считают приоритет контроля качества высоким. Это не всегда очевидно, и менталитет, ориентированный на достижение ближайших целей, может побудить менеджеров по развитию игнорировать точку зрения на общую картину. Пора признать — рабочая методология, замедляющая отдачу, не особенно привлекательна. Однако, по словам великого американского хореографа Марты Грэхем (Martha Graham), можно добиться «свободы через дисциплину». Действительно, гибкость организации-разработчика, предполага-

ющая некоторый уровень собственной дисциплины, можно сравнить с гибкостью квалифицированного танцора. В контексте программного обеспечения возможная свобода состоит в успешной поставке ПО на любой мировой рынок по выбору при минимальных дополнительных затратах времени и средств. Как и в случае хорошо поставленного танца, поставка должна происходить без усилий, а качество — быть безупречным.

Возможно, читатели захотят узнать, что означает *высокая приоритетность (front-loading)*. Это методология, при которой баланс усилий смещен не к концу, а скорее к началу проекта. Многие рекомендации типа «как сделать это правильно» подчеркивают важность планирования на ранних этапах, и настоящая статья не будет слишком отступать от этого правила, хотя мы ограничимся точкой зрения на планирование и приоритетность некоторых типов тестирования, связанных с международной поставкой продуктов. Обосновываемое в данной статье отличие состоит в том, что для достижения долговременного успеха приоритетным должно стать и тестирование. Такой подход при распределении ресурсов разработки противоречит вполне человеческой склонности отложить на потом задачи, которые не считаются критически важными. В какой-то мере сюда можно включить и переопределение выходных критериев, рассматриваемых как критические вехи для этапов процесса разработки, которые выполняются раньше, чем это предписывают традиционные ценности.

Некоторые весомые аргументы можно привести в пользу подхода с фокусом на краткосрочных целях. В контексте локализации, согласно одному из таких путей рассуждения утверждается, что в первую очередь необходимо обеспечить поставку продукта на самые большие рынки, чтобы добиться максимального поступления прибыли. Любые действия, которые могут быть связаны с поставкой продукта в международном масштабе, откладываются на более поздние этапы процесса развития. Это равноценно низкой приоритетности. Другими словами, усилия в областях, критически важных для международной поставки, откладываются на будущее — при существенном снижении приоритета — или, что еще хуже, передаются другим ресурсам в рамках организации в целом. Обычно это те ресурсы, которые более непосредственно ощущают негативный эффект отсрочки и возьмут на себя нагрузку, поскольку без нее они не смогут выполнить свое предназначение. Такой подход не только увеличивает задержку поставки на неосновные рынки, общая прибыль от которых может быть равной или превышать прибыль от основных рынков, он может привести также к дорогостоящему дублированию усилий. Код, которому для подготовки к переводу требуется перепроектирование, должен повторно пройти через циклы разработки и контроля качества (QA). Между тем, исходная база кода будет продолжать развиваться. Таким образом, придется либо поддерживать две базы кода — одну неинтернационализированную, а другую международную, но не актуальную, либо изменения, внесенные для обеспечения международной поддержки, нужно будет добавить обратно

в первоначальный код и объединенный код повторно подвергнуть процедурам контроля качества даже для выпусков нелокализованного продукта.

Предположим, что компания выбрала вариант высокой приоритетности своих процессов, согласившись с некоторой задержкой в поставке продуктов на исходном языке (ИЯ), чтобы обеспечить ускоренную поставку в мировом масштабе международных и локализованных продуктов и в долгосрочной перспективе быстрее добиться присутствия в более широком спектре рынков. Для этого в план развития продуктов на ИЯ необходимо включить два из пяти типов тестирования. По большому счету, самое важное из них — тестирование интернационализации. Вспомогательный тип — тестирование готовности к локализации.

Тестирование интернационализации

Тестирование интернационализации наиболее важно по нескольким причинам. Во-первых, полностью интернационализированные продукты можно поставлять как продукты на ИЯ в регионы по всему миру, где они, несмотря на отсутствие локализации, сохранят полную функциональность. Это может упростить продажи продукта большим корпорациям, которые в головном офисе используют языковый стандарт ИЯ и имеют множество филиалов по всему миру. Часто эти корпорации в качестве операционного используют прежде всего язык страны, в которой находится их головной офис, и поэтому готовы к внедрению еще не локализованного продукта, чтобы воспользоваться преимуществами новых возможностей или другими выгодами, которые можно получить, если персонал повсеместно работает с одной версией.

Тестирование интернационализации в этом контексте гарантирует соответствие продукта некоторому набору критериев, таких как различные форматы времени, дат и чисел, поддержка наборов символов, функциональность локализованных операционных систем и другие аспекты интернационализации. Без такой поддержки полноценное использование продукта в регионах, говорящих на неосновном языке, будет затруднено. Сюда включается также контроль правильности ввода, передачи, сохранения, обработки, извлечения и отображения данных, представленных на языках, отличных от языка интерфейса продукта.

Поскольку для вышеупомянутой функциональности, связанной с интернационализацией, характерна реализация глубоко внутри кода продукта, намного эффективнее встроить ее в начальный проект, а затем провести тестирование непосредственно, чем возвращаться и перепроектировать или переделывать код после выпуска продукта на ИЯ. Ключевое слово здесь — *тестирование*. Проверка успешной реализации интернационализированной функциональности должна быть включена в планы тестирования с самого начала и служить выходным критерием для каждой фазы каждого модуля. Это позволит избежать дополнительной существенной нагрузки, вызванной необ-

ходимостью в последующей поддержке двух баз кода или внедрения дополнительного и протестированного интернационализированного кода обратно в структуру основного кода. Разветвление и повторное объединение сами по себе могут стать источниками ошибок человека-оператора и поэтому требуют дополнительного регрессивного тестирования объединения ветви и основной структуры.

В идеале, сборки на ИЯ необходимо прежде всего протестировать с локализованными данными, а также на локализованных операционных системах и локализованных сценариях. При такой парадигме языковый стандарт ИЯ становится просто одним из многих заранее протестированных языковых стандартов. Причина этого в том, что изначально можно предположить: если полностью поддерживаются не очень хорошо знакомые разработчикам стандарты языков, то полностью поддерживаться будет и функциональность стандарта ИЯ. Такой подход представляет противовес естественной склонности разработчи-ков — в частности тех, кто находится в Северной Америке — делать необоснованные предположения о том, как обстоят дела где-либо еще в мире («Что? Символ валюты указывается *после* числа? Это *странно!*»).

Основная мысль здесь заключается в следующем: если проблемы этого типа обнаруживаются в результате тестирования локализованного продукта, то для процесса разработки уже слишком поздно, чтобы исправить их эффективно с точки зрения затрат времени и средств. К сожалению, в реальном мире такая ситуация возникает гораздо чаще, чем хотелось бы. Для менеджеров по локализации важно ставить во главу угла долгосрочные экономические преимущества заблаговременного тестирования интернационализации, и лучшим местом для этого является не уровень менеджера по разработке, а, скорее, С-уровень, оптимально — уровень финансового и технического директоров. Точка зрения этих людей на цели корпорации и финансовые эффекты обычно шире, чем у менеджеров по разработке и разработчиков, которые более склонны искать персональное удовлетворение в реализации изощренных возможностей. Парадокс состоит в том, что в любой компании размером больше среднего руководители С-уровня — это как раз люди, которых труднее заставить вникнуть, у которых мало времени и интереса, чтобы заниматься деталями стратегии разработки, но цель которых, тем не менее, — определять и проводить общую политику корпорации.

Тестирование готовности к локализации

Продукт, полностью поддерживающий интернационализированные сценарии с точки зрения функциональности на ИЯ, тем не менее может быть не готов к локализации. Основная причина этого — влияние перевода на компоновку экрана и правильность грамматики.

Разработчики программного обеспечения обычно склонны проектировать диалоги в расчете на ИЯ и, если не знают об увеличении длины строк в результате перевода на некоторые языки, часто не резервируют место, позволяющее учесть

такое увеличение при компоновке элементов диалога. К аналогичным проблемам может привести использование сокращений. Опять же, не будучи предупрежденными, разработчики иногда не могут осознать, что в некоторых языках просто невозможно создать осмысленные сокращения. Решения, как правило, сложны и требуют затрат времени. Часто простая перестановка компонентов диалога может создать пространственный буфер, позволяющий точно учесть увеличение длины строк. В долговременной перспективе намного эффективнее и дешевле сделать это один раз на ИЯ и более не вносить изменения для каждого локализуемого языка. Но следует еще раз подчеркнуть, что эта задача подразумевает подход с высоким приоритетом, который можно легко исключить из планов разработки ИЯ и передать в группы локализации или поставщикам услуг для настройки каждого языка отдельно, что в разы увеличивает затраты времени и средств.

К счастью, есть инструменты, с помощью которых можно сделать массу предположений о задаче изменения размера и перекомпоновки. В основных инструментах визуальной локализации реализована функция так называемой псевдолокализации, которая позволяет моделировать влияние перевода за счет подстановки символов целевого языка в строки программного обеспечения и эмуляции увеличения длины строк. Затем измененные строки возвращаются в скомпилированное и действующее программное обеспечение. Это можно сделать быстро и просто, предоставив при этом разработчикам веские визуальные доказательства того, что изменения в их проектах положительно повлияют на дальнейшие действия по локализации.

У псевдолокализации есть два других полезных применения в разработке пользовательского интерфейса (UI). Одно — это эмпирическая идентификация жестко закодированных строк, а другое — демонстрация сцепления строк. Жесткое кодирование и сцепление встречается в программном коде довольно часто. Для последующих действий по локализации они могут стать превосходным источником головной боли, приводя в лучшем случае к переработке кода, а в худшем — к некачественной локализации. В сущности, продукт, который выглядит вполне привлекательно на ИЯ, нельзя считать пригодным к локализации, пока все жестко закодированные строки не будут помещены в доступный для перевода сегмент ресурсов, а все сцепленные строки — переработаны с целью устранения сцепления.

Второе применение псевдолокализации в тестировании готовности к локализации попадает в пограничную зону между интернационализацией и готовностью к локализации. Это область, где перевод отрицательно влияет на функциональность продукта. Такое возможно, если элементы UI, в частности текстовые строки, использованы разработчиками для принятия в коде логических решений либо составлены из данных, участвующих в переносе, сохранении, обработке или повторном отображении. Если, например, локализованные строки UI будут повреждены в любом из вышеперечисленных процессов либо исполь-

зованы для посимвольного сравнения с другими строками с целью повлиять на поведение программы, то перевод этих строк может привести к сбоям. Псевдолокализация чрезвычайно полезна для заблаговременного выявления проблем такого сорта на ранних этапах жизненного цикла разработки, чтобы их можно было решить до начала фактической локализации.

Итак, интернационализация продукта и его готовность к локализации подтверждена. Что дальше?

Есть три типа задач тестирования, которые нужно выполнить после проведения локализации. Это косметическое тестирование, лингвистическое тестирование и функциональное тестирование. К счастью, если продукты тщательно протестированы с точки зрения интернационализации и готовности к локализации, затраты на последние три категории тестов будут минимальными. Тесты, тем не менее, необходимы.

Косметическое, лингвистическое и функциональное тестирование

Несмотря на все усилия по обеспечению готовности к локализации, иногда в локализованных версиях может наблюдаться усечение строк, и для учета увеличения длины строк, превышающей ожидаемую, необходимо изменить геометрию экрана. Достаточно сравнить английское слово *Reset* с его немецким эквивалентом *Wiederherstellen*, чтобы понять, как проявляется недооценка.

Предпосылкой для всестороннего косметического теста является наличие сценария, с помощью которого тестирующий может просмотреть все диалоги и меню в UI. На практике такие полные сценарии встречаются редко. Как правило, тестеры — это сотрудники компании, работающие в ней достаточно долго и хорошо знакомые с пользовательским интерфейсом. Они могут обнаружить явные проблемы, например текст, не помещающийся на кнопке, но при этом нет никакой гарантии, что их лингвистическая квалификация позволит распознать нехватку нескольких букв в конце слова, если для этого нет явных грамматических указаний, таких как точка в конце предложения. Если компании требуется максимальное косметическое качество, она должна либо привлечь внутренние ресурсы с необходимой языковой квалификацией и обучить их взаимодействию с программой, либо смириться с финансированием создания сценариев тестирования, чтобы подготовить материалы и передать их внешним ресурсам с необходимым лингвистическим опытом.

Хорошо спроектированный и готовый к локализации UI, подвергнутый переводу, тем не менее, может не соответствовать стандарту, если перевод неправильный или неоднозначный, содержит орфографические или грамматические ошибки, либо другие проблемы. Хотя наличие в переводе орфографических ошибок абсолютно недопустимо, могут легко возникать другие лингвистические ошибки, если переводчики вынуждены работать без знания контекста, в условиях неполных или неправильных данных, требуемых для принятия решений при переводе, либо сталкиваются с необходимостью разрешения подчас

неразрешимых лингвистических парадоксов, возникающих при сцеплении строк. Продукты с визуальной локализацией могут существенно помочь в уменьшении числа таких ошибок, которые, однако, могут и будут возникать, и поэтому требуют тщательной проверки. Вот некоторые выходные критерии для лингвистического тестирования такого типа: перевод имеет смысл и правильно отражает суть языка оригинала; орфографические и грамматические ошибки устранены; пунктуация правильна; инструкции по стилю и оформлению применены должным образом.

Еще один вид лингвистических проблем может возникать из-за ошибок в процессе сборки, связанных с человеческим фактором. При компиляции программного продукта часто используются сотни и даже тысячи файлов компонентов. Многие из этих файлов содержат переведенные ресурсы UI. В результате простой ошибки человека-оператора такие файлы могут легко попасть в не предназначенное для них место в структуре каталогов сборки. Результатом будет сборка продукта с двумя или несколькими языками. Выпуск такого продукта приведет к весьма неприятным для компании последствиям.

Не следует ожидать, что инженер по сборке сможет различать такие файлы по содержимому, особенно при наличии множества языков. Аналогично, нет веских оснований ожидать, что тестер в целом сможет различить некоторые языки, особенно родственные — например, финский и эстонский — или использующие наборы символов для языковых стандартов, неизвестных тестеру. Как и в случае с косметическим тестированием, лингвистическое тестирование можно передать внешним ресурсам с необходимой квалификацией. При этом для процесса контроля качества будет более чем желательным наличие сценария тестирования, с помощью которого эти ресурсы смогут получить доступ ко всем диалогам, меню и сообщениям об ошибках продукта.

Для заблаговременного снижения риска создания смешанных сборок и уменьшения объема последующего тестирования инженерные ресурсы могут предпринять также некоторые дополнительные действия. Одно из них — формирование ясного и интуитивно понятного соглашения по именованию исходных файлов, запланированных для перевода. В имена файлов необходимо включить атрибуты для идентификации языка содержимого файла. Например, файл «resources.rc» можно назвать «enu_ resources.rc» для ИЯ, «deu_resources.rc» для версии на немецком и «chs_resources.rc» для версии на упрощенном китайском языке.

Однако интуитивно понятного именования файлов недостаточно, на ранних этапах жизненного цикла разработки необходимо создать использующую эти файлы среду сборки, в идеале — на стадии ее написания для продукта на ИЯ. Инженер по сборке должен реализовать выбор режимов, чтобы при необходимости выполнить сборку на любом нужном языке. Затем среду сборки необходимо тщательно протестировать. Псевдолокализация и в этом случае может быть весьма полезной, хотя при отсутствии инструмента для визуальной локализа-

ции достаточно творчески отредактировать файлы, используемые в многоязыковых сборках, например, изменить заголовок основного диалога, чтобы отразить специфику различных языков перевода. Разумеется, это необходимо сделать для всех файлов ресурсов. Если их много, затраты времени могут быть значительными. Реально уменьшить затраты времени поможет способность инструментов для визуальной локализации выполнять операции такого рода автоматически и в пакетном режиме.

Кто-то может справедливо возразить, что тестирование среды сборки относится не к категории лингвистического тестирования после перевода, а скорее к тестированию интернационализации. Именно в этой части статьи оно рассматривается только потому, что отрицательные последствия его игнорирования проявляются на этапе лингвистического тестирования. Тестирование среды многоязыковой сборки определенно должно быть частью тестирования интернационализации. Фактически, без него всестороннее эмпирическое тестирование интернационализации выполнить невозможно.

Если тестирование интернационализации и готовности к локализации было всесторонним и, в частности, поддерживалось программой бета-тестирования в мировом масштабе, потребность в функциональном тестировании локализованных продуктов будет минимальной. Тем не менее, возможны ситуации, которые не были учтены в заблаговременных, предшествующих локализации циклах тестирования, поэтому очень желательно хотя бы ограниченное функциональное тестирование локализованного продукта, запланированного для выпуска. Опять же, из-за реальных обстоятельств, таких как ограничения во времени и средствах, функциональное тестирование локализованных сборок может быть кратковременным, поверхностным или просто исключено из графика. Таковы реалии, поэтому заблаговременное тестирование интернационализации становится тем более важным.

Другие связанные применения: документация

Типы тестирования можно с успехом применить и к другим компонентам продукта в целом. Отличный пример — контроль качества документации. Тестирование интернационализации для документации может включать оценку приемлемости содержимого для всех целевых рынков. Кроме того, оно должно гарантировать структуру файлов документов, при которой снимки связываются ссылками, а не вложением. Снимки экрана необходимо разделить на две группы и разместить их в разных каталогах: для локализованных снимков экрана и тех, которые не требуют локализации, например, значки или другие графические объекты с возможностью повторного использования в переведенной документации в режиме «как есть». Если есть рисунки с текстом, они должны создаваться в программах, допускающих размещение текста в отдельных слоях, чтобы переводчик мог работать в новом слое, а затем отключить текст на ИЯ.

Тестирование готовности к локализации в контексте документации ори-

ентировано в основном на компоновку и использование пробелов. Для снижения издательских расходов необходимо оставлять достаточно места, чтобы учесть увеличение длины строк и минимизировать необходимость повторного разбиения переведенного документа на страницы. Это особенно важно, если описательный текст располагается вокруг снимка экрана. Естественно желание избежать причуды локализации, когда в результате увеличения объема текста при переводе снимок экрана принудительно перемещается на новую страницу, а текст описания остается на предыдущей странице. Это особенно нежелательно для несмежных страниц. Не следует избегать повторного разбиения на страницы за счет уменьшения размера шрифта. Если некоторый размер шрифта приемлем для англоговорящих читателей, то он приемлем и для читателей, говорящих на немецком. Одной из реалий, с которой могут столкнуться менеджеры по локализации, состоит в том, что иногда бывает трудно убедить некоторых технических писателей использовать побольше пробелов.

Как и в случае с программным обеспечением, косметическое тестирование документации относится к задачам, выполняемым после перевода. Его основная цель — обеспечить правильность компоновки. Для снимков экрана необходимо проверить окружающий текст. Это особенно важно, если выполняется перевод программного обеспечения, разработка которого продолжается. В результате снимки экрана могут стать более актуальными, чем ссылающийся на них текст. Если в исходном документе есть текст с условиями, рецензент должен убедиться, что переведенный текст правильно отражает условие и не допускает непредусмотренных переходов. Если взаимные ссылки не являются динамическими гиперссылками, их следует проверить на изменение в разбиении на страницы, вызванное увеличением объема текста при переводе. Собственно гиперссылки необходимо также проверить и убедиться, что при переводе они не были нарушены.

Лингвистическое тестирование документации ориентируется на орфографические, грамматические и стилистические аспекты. Переведенные документы необходимо проверить на соответствие руководству по стилю и любым другим связанным инструкциям, а затем вернуть поставщикам услуг для переработки, если обнаружены несоответствия.

В целом, функциональное тестирование документов определяет точность их перевода с исходного языка. Однако это возможно не всегда. При разработке программного обеспечения документация на ИЯ часто создается в очень жестких временных рамках и *vis-à-vis* с меняющейся целью, мифической фиксацией UI. В документации часто возникают ошибки, которые нередко обнаруживаются переводчиками, тщательно изучающими оригинал, чтобы обеспечить правильность его перевода.

При этом возникает парадокс: должен ли перевод соответствовать неправильному источнику или он должен точного отражать поведение обновленного продукта? На это нет простого ответа. В идеале, пользователь, заплативший за продукт деньги, имеет право получить документацию, правильно описываю-

щую этот продукт. Но если письменный перевод отличается от оригинала, чтобы обеспечить точное описание поведения продукта, база памяти переводов будет неточной, что приведет к ущемлению Питера в пользу Пауля. К счастью, мы живем в эпоху Интернета, при помощи которого можно просто и дешево обновлять по меньшей мере интерактивную документацию.

Кроме того, смягчению парадокса способствуют достижения в технологии ТМ, когда элементы перевода можно пометить атрибутами и повторно использовать ранее переведенные сегменты оригинала. Но сегодня парадокс вполне реален, и это не очень радует. Лучшее решение — создание тесной обратной связи между ресурсами перевода и создателями оригинального содержимого, чтобы ошибки, обнаруженные в документации, можно было перехватывать и эффективно устранять, а также внедрение технологических улучшений, облегчающих такое взаимодействие.

Вывод: качество — крепкий орешек, и стремление к качеству может быть связано со значительными затратами с точки зрения времени, привлечения человеческих ресурсов и финансов. Использование жизненного цикла разработки с высокоприоритетными заблаговременными мерами по снижению последующих расходов , усилий и задержек выпуска должно оставаться бизнес-решением, принимаемым каждой компанией самостоятельно. Высшее руководство должно осознавать свою роль в определении политики качества — что их решения непосредственно влияют на финансовое положение и, как следствие, на их собственный успех как руководителей.

Ресурсы С-уровня должны быть осведомлены о взаимосвязях и зависимостях, существующих между пятью типами тестирования локализации, чтобы принимаемые ими решения были максимально компетентными и мудрыми. Эта ответственность лежит на тех, кто больше других ощущает груз проблем — на менеджерах по локализации.

Ричард Сайкс *— соучредитель и главный консультант компании Localization Flow Technologies. Занимается локализацией с 1989 года. В настоящее время его основные интересы — технологии перевода и отраслевой опыт управления.*