Вы работаете программистом и практически каждый день пишете код. Скажите, как часто вы чувствуете удовлетворение от выполненной работы и гордость за результаты своего труда? Случалось ли вам выпускать работающий, но некачественный и «некрасивый» код только для того, чтобы уложиться в сроки? Есть ли у вас мотивация писать оптимальный код, зная, что через пару месяцев он станет неактуальным и бесполезным?

Попробуем разобраться, как же так получилось, что программирование из красивого искусства и творчества превратилось в повседневный рутинный конвейер.

Потогонный конвейер

Если вы работаете в коммерческой фирме, то вам, наверняка, знаком термин «time to market» — временной промежуток от появления идеи продукта или функциональности до выхода на рынок готового продукта. Нынче все стараются этот промежуток сократить. В коммерческой разработке правит бал ускорение процессов.

Релизы следуют один за другим, все участники разработки ПО вечно не укладываются в намеченные сроки и работают сверхурочно. И всё это делается с одной целью — поскорее продать потребителю готовый продукт. Тут уже не до изысков разработки и не до качественного кода — конвейер не должен останавливаться.

Продукт выпускается в срок, возможно, даже без критических ошибок: цель достигнута. Но каждому, кто участвовал в разработке, прекрасно видно, что творится за красивым рекламным фасадом. В системе постоянно накапливается технический долг, увеличивается количество «хардкода», все временные решения становятся постоянными. Попытки хоть как-то исправить ситуацию обычно ни к чему не приводят: «Очень хорошо, что ты это заметил, но сейчас времени на исправления нет, но, возможно, в будущем мы это поправим». Всё остаётся по-прежнему и на основе всей этой шаткой конструкции продолжается развитие системы, реализуется новая функциональность.

Пластмассовый мир

С другой стороны, кому сейчас нужны долговечность и качество? Большая часть написанного кода будет использоваться в системе максимум несколько месяцев, а затем будет заменена или переработана. Какой тогда смысл писать

этот код идеально? Представьте, что вы каждый день занимаетесь тем, что штукатурите комнату в доме, прекрасно зная, что через день этот дом снесут. Тут у кого угодно опустятся руки.

Складывается впечатление, что многим фирмам просто невыгодно выпускать качественный долговечный продукт. Если сделать смартфон удобным и надёжным, то кто из потребителей через год захочет покупать новый? Так появилось явление, которое называется «запланированное устаревание». Мы все живём в пластмассовом мире недолговечных вещей.

Вспомним, например, как Microsoft случайно выпустила довольно сносную Windows XP. Пользователям она настолько понравилась, что они ни в какую не хотели переходить на следующую версию ОС. Затем история отчасти повторилась с Windows 7. Но больше таких «ошибок» Microsoft не допускала — переход на следующую версию системы стал добровольно-принудительным.

Видимо, именно по этим причинам сегодня всё реже задумываются о красоте и об оптимальности выпускаемых систем. В коммерческой разработке главенствует принцип «Вам шашечки или ехать?»

Набор «Сделай сам»

Современная среда разработки ПО тоже не поощряет программистов к написанию быстрых и эффективных приложений. Для создания даже простейших решений и систем используются многочисленные фреймворки, платформы и тулкиты. Эти инструменты, безусловно, сильно упрощают и ускоряют разработку. Но при этом процесс программирования превращается в сборку программ из готовых кубиков-блоков. Оптимизация работы приложения при таком подходе отходит на второй план.

Это немного напоминает первые сотовые телефоны. На первый взгляд эти телефоны выглядели довольно компактно. Но на поверку эта компактность оказывалась мнимой. К телефону прилагался увесистый чемодан-аккумулятор, который постоянно нужно было носить с собой.

Программы, написанные в ранних версиях Visual Basic, тоже выглядели довольно компактно — сам исполняемый файл программы был небольшого размера. Но он был неработоспособен без дополнительного файла библиотеки MSVBVMXX.DLL с виртуальной машиной для запуска приложения.

Чем ситуация с сегодняшними приложениями лучше? От разработки на ассемблере и низкоуровневого программирования нас отделяет не один и не два, а десяток уровней абстракций и инструментов. Решение собирают из разрозненных полуготовых модулей, которые плохо подходят друг к другу. Чтобы всё это кое-как держалось и работало, используют клей и изоленту.

Судя по всему, сейчас разрабатывать коммерчески успешные приложения по-другому не получится. Приведу прекрасный комментарий с Хабра, который полностью отражает сложившуюся ситуацию: «Я раньше поражался тому, как уродливы изнутри „взлетевшие“ проекты. Сейчас я знаю: красивые проекты не взлетают, потому что они не успевают взлететь. Пока инженеры в белых халатах прикручивают красивый двигатель к идеальному крылу, бригада взлохмаченных придурков во главе с безумным авантюристом пролетает над ними на конструкции из микроавтобуса, забора и двух промышленных фенов, навстречу второму туру инвестиций. Авантюрист любезно раздаёт восторженным пассажирам талоны и бумажные пакетики».

Искусство программирования

Так кто же такой современный разработчик: ремесленник с удовлетворительным знанием инструмента или мастер, достигший вершин своего искусства? Этому вопросу уже много лет. Похоже, что в современном мире разработки творческих мастеров совсем не ждут. Часто приходится встречать такое отношение руководства: «По мнению моего начальника, программирование — это ремесло. И спорить с ним бесполезно. Он считает, что программист с творческим подходом к делу вообще вреден и ни в коем случае его брать на работу не надо».

Многие из мастеров, чтобы выжить, даже вынуждены маскироваться под ремесленников: иначе не получится уложиться в запредельно сжатые сроки разработки.

Безусловно, в любом деле нужны не только высококлассные мастера. Везде найдётся дело для простого работяги, окончившего полугодовые курсы программирования на одном из обучающих сайтов. Но проблема в том, что при этом неуклонно снижается качество программ. Установка новой версии какого-нибудь мобильного приложения превращается в игру «Угадай, что они ещё умудрились сломать». Если на стройке не будет знающего образованного инженера, то даже самые усердные и дисциплинированные рабочие смогут построить добротный надёжный дом разве что случайно.

Дональд Кнут назвал свою книгу «Искусство программирования». Первое издание этой книги вышло ещё в далёком 1968 году. Тогда программирование

было ещё искусством. Жаль, что профессия программиста постепенно становится всё более прагматичной и приземлённой. Но хотя бы в своих pet-проектах мы всегда можем заниматься свободным творчеством и не играть по корпоративным правилам.