**Качество ПО: восемь мифов**

Джеффри Воас

Складывается впечатление, что компьютерное сообщество вполне удовлетворено нынешним состоянием дел c качеством ПО: мало не то, что новых идей, да и просто энтузиазма. Нам остается лишь вспоминать о тех далеких временах, когда специализироваться в области управления качеством ПО было весьма престижно, и на визитке зачастую встречались надписи типа: "software safety evangelist".

Неужели уже можно почивать на лаврах? В конце концов, ни одна из основных проблем создания качественного ПО так и не решена! Сегодня перед нами стоят все те же проблемы, что и десять лет назад. Я хочу обратить внимание коллег на некоторые общепринятые решения, чтобы не сказать панацеи (silver bullets, как после знаменитой книги Брукса стало принято это обозначать в профессиональной литературе), которые, собственно, и создают иллюзию движения вперед. Мне кажется, что на самом деле эти решения носят мифологический характер. Не претендуя на полноту, я выделил восемь таких мифов.

**Совершенствование процесса и модели зрелости разработки ПО**

Миф, связанный с этой тенденцией современной программной инженерии, состоит в следующем: измерение зрелости процесса разработки в некоторой организации эквивалентно измерению качества ПО, которое эта организация производит. Отсюда неявно следует, что построение более зрелого процесса разработки обеспечивает создание более зрелого ( более качественного) ПО.

Само по себе усовершенствование процесса разработки, так же как и ранжирование коллективов по уровню профессионализма (особенно по стандартизованной методике типа CMM) безусловно похвально. Однако практика доказывает, что предположение об однозначной связи между официально засвидетельствованными рейтингами зрелости процессов и качеством производимого ПО ошибочно. К сожалению, в компьютерной индустрии этот миф получил чрезвычайно широкое распространение.

**Формальные методы**

Этот миф можно считать в некоторой степени частным случаем предыдущего. Он гласит, что именно формальные методы способны стать движущей силой "усовершенствования процессов", в частности, облегчить решение проблем безопасности и надежности ПО. На деле формальные методы - это не более чем математически строгая демонстрация наличия у разрабатываемого ПО некоторых желаемых свойств абстрактной природы. Формальные методы позволяют делать заключения об отсутствии логически некорректного, плохо определенного и рассогласованного поведения, которое в принципе может присутствовать в спецификации.

Конечно, внедрение формальных методов имеет смысл, однако необходимо четко понимать, что возможности их применения весьма ограничены. Не говоря уже о том, что использовать формальные методы сложнее, и обходится это недешево.

**Языки и объектно- ориентированное проектирование**

Этот миф устанавливает, что, изменив языковую или проектную парадигму, можно решить те проблемы разработки, с которыми мы не могли справиться, используя существующие языки или стратегии проектирования. Однако замена одного языка программирования другим, более современным, вряд ли поможет решить проблемы, не связанные напрямую с особенностями применяемого языка. А именно с такой задачей обычно и приходится сталкиваться.

Современные программные системы становятся все сложнее. Для их реализации предлагается использовать опять-таки все усложняющиеся языки проектирования и программирования (которые содержат столько возможностей, что мало кто из специалистов способен корректно применять их). Особенно показательны в этом отношении ставшие сейчас очень популярными объектно-ориентированные языки. Понятия и концепции, лежащие в основе объектно-ориентированной парадигмы, весьма сложны и нуждаются в очень аккуратном обращении. Недостаточно корректное использование этой парадигмы сплошь и рядом приводит к возникновению серьезных проблем, что дает основания для внешне парадоксального вывода: создавать качественное ПО для многих легче с использованием "старых" языков, наделенных более скромными возможностями.

К тому же современные парадигмы проектирования, в основе которых лежат принципы абстракции (такие, как инкапсуляция), делают процесс тестирования на системном уровне более сложным и менее эффективным. Известно, однако, что чем тяжелее тестировать, тем меньше шансов получить в итоге высококачественное ПО.

**Метрики и измерения**

Данный миф утверждает возможность умозаключений, "хорошее" ли ПО разработано, на основе цифровых оценок, относящихся как непосредственно к коду, так и к процессу его разработки. Но можно ли точно определить, что значит "хороший" код?

Для большинства специалистов код хорош тогда, когда он реализует вычисление необходимой функции желаемым способом (например, корректно и с ограничениями, накладываемыми исполнением в реальном времени). При этом понятие "хороший" не связано напрямую с тем, как код структурирован и тем более с тем, как он выглядит - оно прежде всего относится к семантике функции, реализованной данным фрагментом кода.

Так как структурные методы семантику не измеряют, они не могут показать, насколько код хорош . В еще большей степени это относится к метрикам процессов. Одно время считалось, что метрики могут измерять семантику, но это оказалось не так. К тому же, как выяснилось, уже собранные метрики не так просто интерпретировать в практических терминах, что необходимо для реального улучшения процесса разработки. Интересно, что метрики кода больше подходят для оценки качества процесса разработки, чем самого кода.

Важно также понимать, что метрики обеспечивают косвенную меру, вообще говоря, неизмеряемых свойств. Например, невозможно измерить "тестируемость" и "сопровождаемость" программы. Зато легко установить количество строк кода. Ясно, что программа длиной в одну строку будет иметь лучшие характеристики тестируемости и сопровождаемости по сравнению с программой в миллион строк, и метрика "число строк кода" это покажет. Лучше следовать эмпирическому правилу (для многих неочевидному): метрики не могут дать универсальных рецептов построения качественного ПО; они обеспечивают лишь направление дальнейшего поиска решений.

**Стандарты ПО**

Многие полагают, что если есть стандарт, то для получения качественного ПО достаточно просто четко ему следовать. Корни этого мифа - в необоснованном переносе в программную инженерию практики, сложившейся в традиционных индустриях. В последние 20 лет мы наблюдаем прямо-таки экспоненциальный рост числа разнообразных стандартов.

Организации, занимающиеся созданием и распространением стандартов (ISO, IEEE, IEC), пока не слишком преуспели в объективной оценке их достоинств с точки зрения практики разработки. К примеру, было бы неплохо хотя бы приблизительно знать, что если следовать стандарту "A" потребуются дополнительные затраты на величину "B", но при этом вы приобретете выгоды величиной "C". Если бы каждый стандарт сопровождался методикой оценки выгод его использования для некоторого типичного проекта, то стандарты было бы намного легче сравнивать и адаптировать.

Мои основные сомнения, касающиеся стандартизации, таковы:

стандарты редко появляются вовремя: как правило, процесс их создания растягивается на столь долгое время, что к моменту публикации они теряют актуальность;

бытует мнение (и оно не лишено оснований), что иногда под предлогом борьбы за качество стандарты вводятся сильными мира сего как средства конкурентной борьбы;

стандарты не содержат методик количественной оценки выгод от их применения;

не всегда понятна взаимосвязь вводимого стандарта с лучшими образцами практики и с другими стандартами.

Безусловно, следовать стандартам в той или иной степени необходимо, но разработка программн - не та область, где это гарантирует улучшение качества продукта в каждом конкретном случае.

Тестирование

Только неофиты от программирования верят мифу, что на этапе тестирования можно выявить и решить все накопившиеся в процессе разработки проблемы. По данным руководителя фирмы Software Productivity Research К. Джоунса вероятность благополучного завершения проблемного проекта не превышает 15%. Вывод очевиден: если разработчики ожидают фазы тестирования в надежде исправить обнаруженные недостатки ПО, шансов на спасение такого проекта очень мало.

CASE

Суть следующего мифа (к которому я особенно неравнодушен) заключается в том, что программирование спецификации с использованием диаграммного или визуального языка гарантирует более высокое качество и надежность кода. CASE-средства переживали бум в начале 90-х, когда на короткое время получил распространение миф о выгодах автоматической генерации кода из визуальной спецификации. Сторонники этого подхода исходили из того, что человек делает меньше ошибок, рисуя картинки. Практика, однако, подтвердила только то, что из некорректных картинок можно получить некорректный код.

Тотальное Управление Качеством

Последний миф связан с тенденцией регламентировать все возможные аспекты разработки ПО. Тотальное Управление Качеством (Total Quality Management - TQM) - это очень показательный пример воплощения данного мифа. В традиционных областях индустрии следование четко определенной методике управления качеством действительно приносит успех. Однако промышленное программирование остается сугубо творческим процессом со значительным элементом неопределенности, и прямое приложение отработанных в иных условиях методик представляется ошибочным.

Вывод - больше прагматизма!

Именно осознание малой практической ценности популярных когда-то направлений сделало многих практиков скептиками, не верящими в новые идеи. Это опасно, потому что так можно дойти до неприятия действительно полезных идей, не говоря уже о более приземленных вещах, таких как сокращение финансирования исследовательских работ.

Перечисленные идеи потеряют мифологический статус и действительно обеспечат реальную помощь в создании качественного и надежного ПО только в том случае, если будут подвергнуты критическому анализу. Кроме того, они должны использоваться не по отдельности, а в сочетании друг с другом

Однако я должен еще раз повторить уже не однажды высказанное предупреждение: исследователям не следует предлагать свои разработки практикам до того, как будут получены убедительные доказательства их действительной полезности. Важно, чтобы при этом рассматривались и вопросы, связанные с ограничениями новых технологий и стоимостью их адаптации.

У каждого из нас - своя роль. Практики должны более четко объяснять исследователям свои реальные проблемы, используя для этого и компьютерную прессу, и трибуны научных конференций. Исследователи должны демонстрировать эффективность своих идей на реальных, а не игрушечных задачах, и только потом выходить с ними на рынок.