Глава 29

Как правильно построить “правильную” систему: общие положения

Основные положения

- Для правильного построения “правильной” системы необходимо постоянно убеждаться, что разработка проводится надлежащим образом, ее результаты корректны, и обрабатывать возникающие в процессе разработки изменения.
- Верификация (verification) позволяет удостовериться, что деятельности разработки неизменно соответствуют потребностям клиента.
- Проверка правильности (validation) демонстрирует, что продукт соответствует предъявляемым к нему требованиям и конечный результат получает одобрение заказчика.
- Поскольку изменения неизбежны, следует учитывать это при планировании и знать, как ими управлять.

Как мы уже неоднократно подчеркивали, крайне важно, чтобы каждый член команды понимал требования к проекту. Но на практике, к сожалению, невозможно ждать, пока абсолютно все станет ясно. Поэтому мы описали методы итеративной разработки, которые позволяют постепенно уточнять понимание системы по мере продвижения вперед. Но даже тогда проект, в котором до этого момента все шло хорошо, может разрушиться и превратиться в дезорганизованные действия. Причина в том, что команда *уже понимает* требования, но еще не в состоянии построить систему, которая *удовлетворяет* им. В данной главе мы рассмотрим несколько важных дополнительных концепций, которые позволят убедиться, что вы надлежащим образом строите “правильную” систему. Это подразумевает следующее.

- Необходимо постоянно убеждаться, что разработка находится на правильном пути.
- Необходимо убеждаться в корректности результатов разработки.
- Необходимо научиться обрабатывать изменения, возникающие в процессе разработки.

Рассмотрим каждый из этих пунктов более подробно.

Проверка того, что разработка находится на правильном пути

При проектировании и реализации команда должна иметь возможность постоянно проверять, не сбился ли проект "с пути", т.е. соответствуют ли результаты разработки потребностям клиента.

Постоянно выполняемый процесс проверки того, что каждый шаг разработки является корректным, удовлетворяет потребности последующей деятельности и не является излишним, мы будем называть *верификацией (verification*). К сожалению, в отрасли имеется множество различных толкований того, что же собой представляет верификация. Поэтому мы обсудим это важное понятие более подробно.

Принципы верификации программного обеспечения

СтандартIEEE (1994) определяет верификациюследующим образом.

> *Процесс оценивания системы или компонента с целью определить, удовлетворяют ли результаты некой фазы условиям, наложенным в начале данной фазы (IEEE 1012-1986, § 2, 1994).*

Таким образом, верификация в значительной мере является аналитической деятельностью, в ходе которой необходимо убедиться, что каждая стадия разработки (например, реализация в программном обеспечении одного или нескольких требований) соответствует заданным на предыдущей стадии требованиям. Как минимум, хотелось бы верифицировать следующее.

- Описанные нами функции действительно соответствуют потребностям.
- Производные от этих функций прецеденты и требования действительно поддерживают данные функции.
- Прецеденты реализуются при проектировании.
- Проектирование поддерживает функциональные и нефункциональные аспекты поведения системы.
- Код действительно соответствует результатам и целям проектирования.
- Тесты обеспечивают полное покрытие разработанных требований и прецедентов.

Итак, как узнать, что собой представляет наша разработка в целом? Нужен метод, который позволит гарантировать, что мы провели верификацию всего необходимого (и ничего лишнего!).

Для этого нужен план верификации и (желательно) некие автоматические средства, которые помогут выполнить его. Но самое главное, команде необходимо понять, что такое верификация, и взять на себя ответственность за ее проведение. Верификация — это *не только* деятельность, осуществляемая группой обеспечения качества проекта (QA team), которая, безусловно, играет огромную роль в этом процессе.

Одним из методов осуществления постоянного контроля верификационных действий является *трассировка (traceability).* Мы уже упоминали о трассировке в части 5, а в главе 31 обсудим, как ее можно использовать для организации верификационных действий.

Независимо от того, как организована верификация, необходимо помнить, что суть ее в том, чтобы убедиться, что шаг, над которым мы работаем, имеет соответствующую предысторию и выполняется непротиворечивым и надежным способом. Более того, необходимо удостовериться, что каждое предпринимаемое нами действие необходимо, а ненужные шаги отсутствуют.

В значительной мере этому способствует применение эффективных процессов разработки программного обеспечения, а также осуществление анализа. Например, можно исследовать требования и убедиться, что они корректно, полно и сжато выражают более высокоуровневые потребности пользователей. Затем можно убедиться, что проектирование проводилось на основании требований и прецедентов и полученный технический проект является полным и не имеет лишних элементов. В некоторых ситуациях анализ сокращается до *просмотров* и *ревизий*. В других случаях можно использовать модели и автоматизированные средства, чтобы проверить полноту, семантику и т.д. Напоминаем, что на данном этапе мы не пытаемся проверить работоспособность, мы займемся этим позднее. Сейчас же необходимо убедиться, что мы сделали то, что должны были сделать, и это следует логике разработки.

Итак, достаточно отступлений. Вернемся к обсуждению некоторых аспектов верификации.

Затраты на верификацию

Неприятным моментом верификации является то, что можно потратить время на верификационную деятельность, которая не принесет отдачи. Поэтому нужен некий способ вычисления "экономических дивидендов" (return on investment, ROI) верификационной деятельности. Необходим подход, который поможет *правильно* вложить средства в проверку наших действий. Не хочется выполнять лишние проверки, но нельзя пропускать проверку жизненно важных аспектов. В главе 33 содержится краткое описание методов оценки и анализа рисков и предлагаются рекомендации по использованию этих методов для экономии средств при проведении проверок.

Команда должна сконструировать и реализовать систему, соответствующую требованиям. Другими словами, команда должна иметь возможность убедиться, что планы реализации соответствуют потребностям.

Но как на основании требований осуществить проектирование и реализацию? Нельзя же передать требования компьютеру и ожидать, что проектирование и реализация произойдут сами собой! В главе 30 рассматриваются способы перехода от разработки требований к проектированию и описывается, как можно использовать полученные в процессе создания требований артефакты при проектировании и реализации системы.

Верификация на всех уровнях

Верификацию можно применять на всех стадиях жизненного цикла разработки; методы не ориентированы на какую-либо одну фазу. Можно (и нужно!) верифицировать элементы требований, проектирования, реализации, тестирования и другие важные элементы разработки, основываясь на результатах ROI-анализа. В той или иной степени верификацией придется заниматься на *каждой* стадии разработки. Она необходима, чтобы удостовериться в правильности следующих переходов.

- От потребностей пользователя к функциям продукта
- От функций продукта к требованиям
- От требований к архитектуре
- От архитектуры к модели проектирования
- От модели проектирования к реализации
- От реализации к планированию тестов

(*Замечание.* Мы будем рассматривать тестирование как часть проверки правильности (validation).)

Хотелось бы вновь подчеркнуть, что верификация чрезвычайно важна на стадии проектирования, так как возникшие на этой стадии ошибки очень сложно устранять на стадии реализации.

Доводы в пользу верификации

Мы рекомендуем во всех разработках проводить процесс верификации, который должен начинаться в начале разработки проекта и продолжаться на протяжении всего ее жизненного цикла. При работе над проектом, в котором процесс разработки регулируется, верификацию, вероятно, придется проводить независимо от вашего желания. Верификация обязательна при разработке систем высокой надежности (систем жизнеобеспечения или таких систем, стоимость сбоя в которых недопустимо высока), в противном случае вы подвергаете недопустимому риску себя или других. Но каждый нетривиальный проект лишь выиграет от хорошо спланированной верификации.

Проверка корректности результатов разработки

По мере достижения важных вех (этапов) разработки, таких как исполняемые итерации, важно проверить правильность созданных функциональных возможностей, т.е. убедиться, что они работают корректно и соответствуют требованиям. Мало пользы в прохождении определенного этапа, если созданные части не работают так, как предполагалось, хотя даже в этом случае можно кое-чему научиться.

Особое внимание следует уделить этапу завершения проекта. В успешном процессе это просто еще один шаг, подтверждающий, что система сконструирована и реализована в соответствии с потребностями клиента. Кроме того, на этом этапе необходимо показать, что система действительно работает так, как предполагалось.

> Последней контрольной точкой является демонстрация работы системы в среде клиента.

В процессе разработки существует последняя и *очень* важная контрольная точка. Нужно продемонстрировать, как продукт работает в среде клиента, и продукт должен понравиться заказчику настолько, чтобы тот принял его. Суть в том, чтобы по окончании проекта клиент был доволен или, в крайнем случае, не слишком недоволен.

Работа по проверке правильности разработки (validating) имеет *два* важных аспекта: (1) показать, что продукт соответствует предъявляемым к нему требованиям и (2) сосредоточить внимание на приемке клиентом конечного результата. Хотя теоретически это

одно и то же, на практике заключительные приемочные испытания часто демонстрируют, что мы следовали требованиям, но потребности пользователя “сместились” по сравнению с более ранними стадиями проекта. В главе 32 обсуждаются оба аспекта проверки правильности, а также высказываются различные мнения о том, что она подразумевает.

Как и при верификации, здесь тоже нужен способ, позволяющий убедиться, что затраты времени и средств на проверку правильности или тестирование являются эффективными. И снова в этом может помочь анализ дивидендов (ROI).

Обработка изменений, возникающих в процессе разработки

Наконец, необходимо рассмотреть влияние изменений требований. Приходилось ли вам когда-нибудь работать над системой, в которой требования *ни разу* не менялись с первого дня?

Необходимо строить свои планы с учетом возможности изменений и знать, как этими изменениями управлять. В главе 34 обсуждается возможное разрушительное влияние неконтролируемых изменений и предлагается организованный способ их выявления, оценки воздействия, а также их внесения упорядоченным образом.

Далее...

Начнем с того, как на основании требований осуществить проектирование и реализацию решения имеющейся проблемы. Этому посвящена следующая глава.