Самарин Юрий Николаевич
д-р техн. наук, профессор
Ложкина Дарья Дмитриевна
студентка
Высшая школа печати и медиаиндустрии
ФГБОУ ВО «Московский политехнический университет»
г. Москва

АНАЛИЗ СОВРЕМЕННЫХ МЕТОДИК ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

***Аннотация**: в работе рассматривается роль тестирования в процессе создания программного обеспечения. Анализируются модели разработки ПО и выбирается лучшая из представленных.*

***Ключевые слова**: тестирование, контроль качества, обеспечение качества, модель разработки, программное обеспечение.*

Каждый день человек использует десятки приложений, сайтов, сервисов в интернете. В процессе работы с ними, выявляются их положительные и отрицательные стороны. Происходит оценка качества и удобства.

Потребители и заказчики, покупая продукты компаний ожидают результат определенного качества: от «так сойдет» до качественной системы с минимальным количеством ошибок и недочетов. В большинстве своем, приобретая продукт предполагается его максимальное качество.

Но даже самой качественной разработке присущи ошибки, недочеты, в том числе «человеческий фактор», т.е. качество итогового продукта становится хуже. Разница между ожидаемым и получаемым качеством проявляется в рисках для компании-разработчика, а также для заказчиков, пользователей и других заинтересованных лиц.

Чтобы избежать таких рисков компании закладывают в разработку работу по улучшению продукта, инвестируя в тестирование. В итоге, уменьшаются затраты, которые связаны с проблемами качества; повышается доверие к

функциональности системы; снижается вероятность событий, угрожающих прибыли и жизни программы; повышается качество планирования, бюджетов будущих проектов; повышается уровень доверия к заказчику; увеличивается прибыль; снижается вероятность не сдачи системы заказчику, как следствие повышается конкурентоспособность компании.

Ключевые процессы тестирования протекают в контексте проекта тестирования, который в свою очередь является частью разработки проекта, сопровождения, интеграции и приемно-сдаточных испытаний системы.

В широком смысле тестирование – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом [5].

В узком смысле тестирование программного обеспечения – это предоставление отрицательной обратной связи [4].

«Контроль качества» – Quality Control, можно считать в широком смысле синонимом для термина «тестирование», потому что контроль качества – это предоставление обратной связи в самых разных ее разновидностях, на самых разных этапах программного проекта. Иногда тестирование подразумевается как некоторая отдельная форма контроля качества.

Существует термин QA (англ. Quality Assurance – Обеспечение качества) – это обеспечение качества на всех этапах разработки. QA имеет цели:

1. Демонстрация соответствия конечного продукта изначальным требованиям

2. Выявление ситуаций, когда программа, ее расчеты и результаты являются ошибочными, неправильными и не соответствуют спецификации.

Данный термин не является синонимом тестирования, поскольку термин «обеспечение» имеет положительный оттенок, это никак не предоставление отрицательной обратной связи, а наоборот предоставляются качество, предпринимаются меры, чтобы качество разработки ПО [4].

«Обратная связь» это какие-то данные, которые с выхода попадают обратно на вход, или какая-то часть данных, которые с выхода попадают обратно на вход. Эта обратная связь может быть положительной и отрицательной. Данное определение берется из науки – «теория систем».

Если рассматривать программное обеспечение, как конечный продукт. То при использовании мы можем получить корректную работу и положительную обратную связь от пользователей. И соответственно, увеличение положительных отзывов, увеличение объемов продаж и распространение продукта.

Отрицательная обратная связь возникает в результате ошибок в работе программы и приходит в виде отрицательных отзывов. Либо ее предоставляет отдел тестирования. Чем раньше предоставляется отрицательная обратная связь, тем меньше энергии необходимо для модификации этого сигнала. Именно поэтому тестировать нужно начинать как можно раньше, на самых ранних стадиях проекта, и предоставлять эту обратную связь и на этапе проектирования, и еще, может быть, раньше, еще на этапе сбора и анализа требований.

На самом верхнем уровне процесс тестирования состоит из 4-х шагов, которые могут пересекаться друг с другом:

1. Планирование – понимание места тестирования.

2. Подготовка – сбор людей и тестов.

3. Проведение – выполнение тестов и сбор результатов.

4. Совершенствование – предоставление информации о проекте и его улучшении [3].

Процесс тестирования основывается на методиках, которые определяют модель построения процесса, эффективность его, а также все ключевые моменты.

Водопадная модель сейчас представляет скорее исторический интерес, т. к. в современных проектах практически неприменима. Она предполагает однократное выполнение каждой из фаз проекта, которые, в свою очередь, строго следуют друг за другом. Очень упрощённо можно сказать, что в рамках этой модели в любой момент времени команде «видна» лишь предыдущая и следующая фаза. В реальной же разработке ПО приходится «видеть весь проект целиком» и

возвращаться к предыдущим фазам, чтобы исправить недоработки или что-то уточнить [1].

V-образная модель является логическим развитием водопадной. Как водопадная, так и v-образная модели жизненного цикла ПО могут содержать один и тот же набор стадий, но принципиальное отличие заключается в том, как эта информация используется в процессе реализации проекта. При использовании v-образной модели на каждой стадии «на спуске» нужно думать о том, что и как будет происходить на соответствующей стадии «на подъёме». Тестирование здесь появляется уже на самых ранних стадиях развития проекта, что позволяет минимизировать риски, а также обнаружить и устранить множество потенциальных проблем до того, как они станут проблемами реальными.

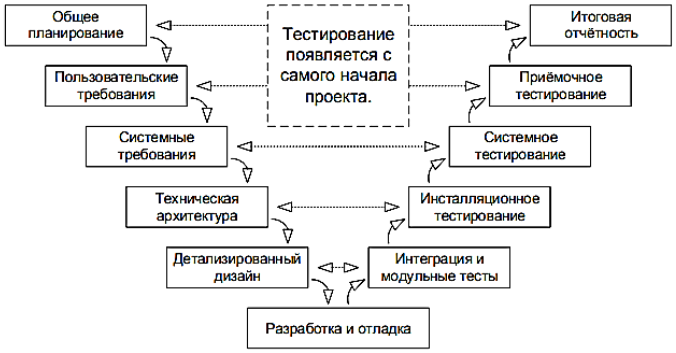

Рис. 1. V-образная модель разработки ПО

Итерационная инкрементальная модель является фундаментальной основой современного подхода к разработке ПО. Как следует из названия модели, ей свойственна определённая двойственность:

– с точки зрения жизненного цикла модель является итерационной, т.к. подразумевает многократное повторение одних и тех же стадий;

– с точки зрения развития продукта (приращения его полезных функций) модель является инкрементальной.

Длина итераций может меняться в зависимости от множества факторов, однако сам принцип многократного повторения позволяет гарантировать, что и тестирование, и демонстрация продукта конечному заказчику (с получением обратной связи) будет активно применяться с самого начала и на протяжении всего времени разработки проекта. Во многих случаях допускается распараллеливание

отдельных стадий внутри итерации и активная доработка с целью устранения недостатков, обнаруженных на любой из (предыдущих) стадий. Итерационная инкрементальная модель очень хорошо зарекомендовала себя на объёмных и сложных проектах, выполняемых большими командами на протяжении длительных сроков. Однако к основным недостаткам этой модели часто относят высокие накладные расходы, вызванные высокой «бюрократизированностью» и общей громоздкостью модели.

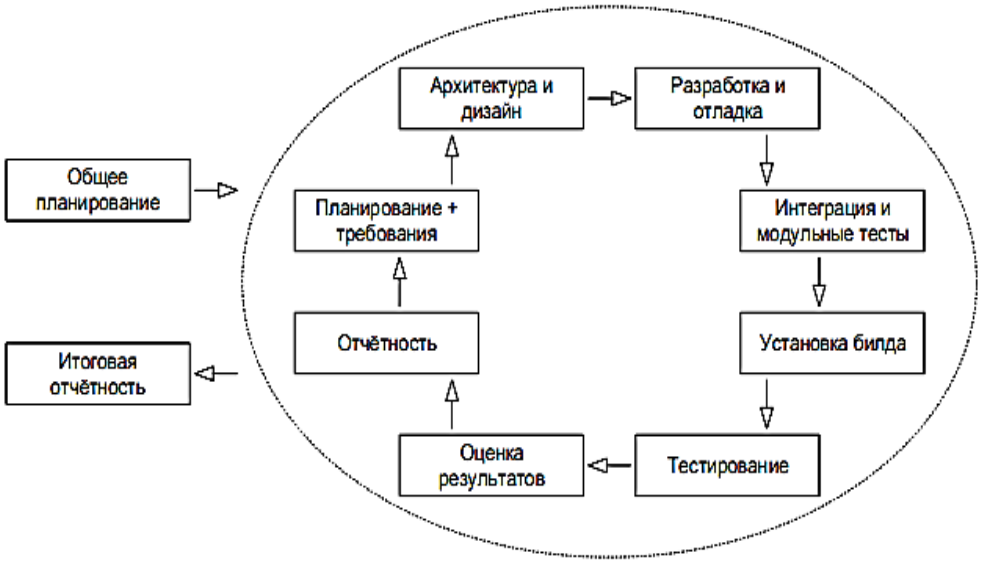

Рис. 2. Итерационная инкрементальная модель разработки ПО

Спиральная модель представляет собой частный случай итерационной инкрементальной модели, в котором особое внимание уделяется управлению рисками, в особенности влияющими на организацию процесса разработки проекта и контрольные точки [2]. Нужно обратить внимание на то, что здесь явно выделены четыре ключевые фазы:

– проработка целей, альтернатив и ограничений;

– анализ рисков и прототипирование;

– разработка (промежуточной версии) продукта;

– планирование следующего цикла.

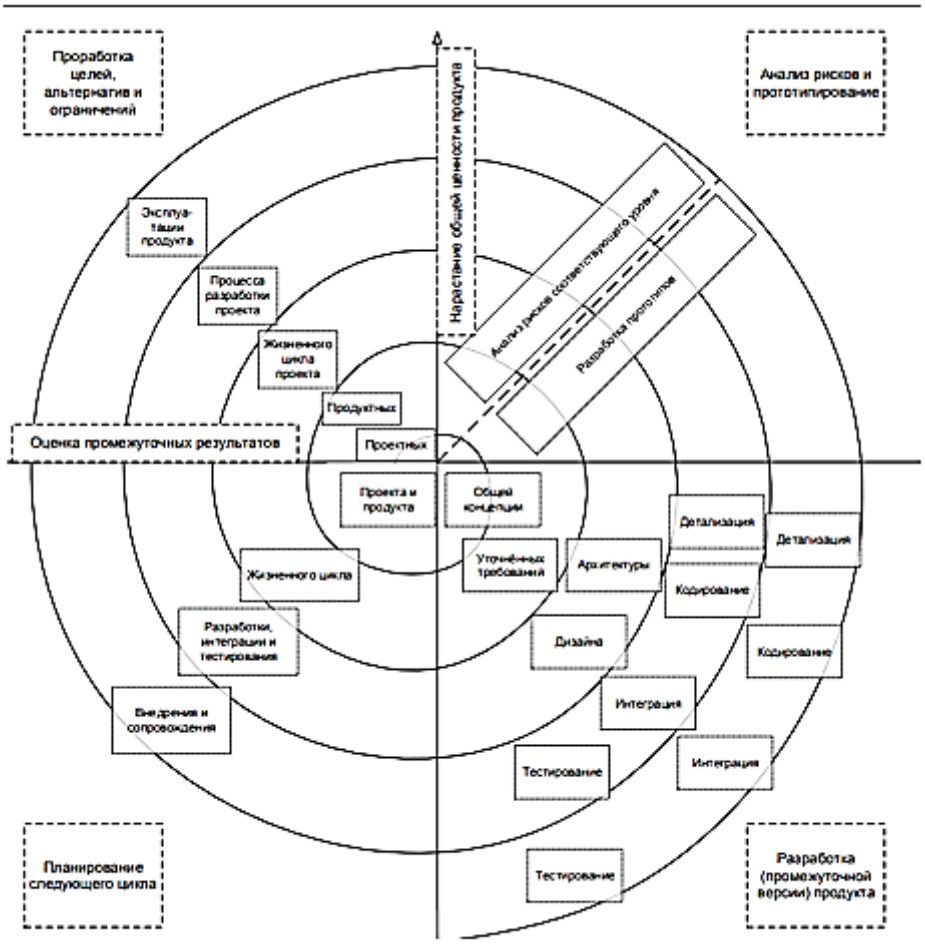

Рис. 3. Спиральная модель разработки ПО

Гибкая модель представляет собой совокупность различных подходов к разработке ПО и базируется на т.н. «agile-манифесте»:

1. Люди и взаимодействие важнее процессов и инструментов.
2. Работающий продукт важнее исчерпывающей документации.
3. Сотрудничество с заказчиком важнее согласования условий контракта.
4. Готовность к изменениям важнее следования первоначальному плану.

Из этих принципов важно понять, что главная цель – удовлетворить заказчика. Следует также знать, что единственной метрикой того, как идет разработка продукта – является сам продукт (работающий продукт). Никакие метрики, никакие отчёты напрямую не показывают насколько вы успешны в достижение требуемой цели.

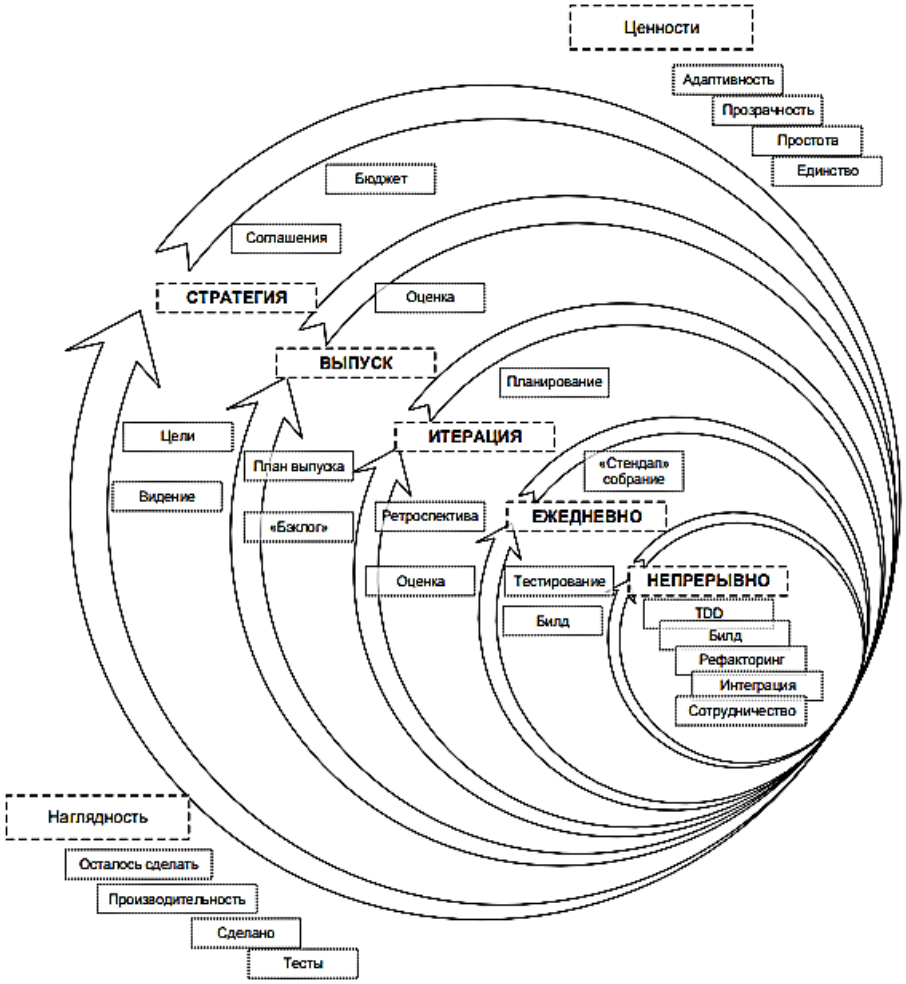

Рис. 4. Гибкая модель разработки ПО

На основании знаний о существующих моделях жизненного цикла ПО можно сделать вывод: в настоящее время наиболее предпочтительной является гибкая модель, поскольку положенные в основу гибкой модели подходы являются логическим развитием и продолжением всего того, что было за десятилетия создано и опробовано в водопадной, v-образной, итерационной инкрементальной, спиральной и иных моделях. Причём здесь впервые был достигнут ощутимый результат в снижении бюрократической составляющей и максимальной адаптации процесса разработки ПО к мгновенным изменениям рынка и требований заказчика. Кроме того, гибкая модель позволяет приспособиться к новым требованиям заказчика и доработкам на разных стадиях развития проекта даже после того, как проект был запущен.

Список литературы

1. Ключевые процессы тестирования. Планирование, подготовка, проведение, совершенствование // Рэкс Блэк – Лори, 2006. – 566 с.

2. Куликов С. Тестирование программного обеспечения. Базовый курс. – 2-е издание. – 2015–2018 – 298 с.

3. Основные положения тестирования [Электронный ресурс]. – Режим доступа – https://habrahabr.ru/post/110307/ (дата обращения: 23.03.2017).

4. ISO/IEC TR 19759:2005 (SWEBOOK).

5. QA engineer, с чего начать? [Электронный ресурс]. – Режим доступа: https://proglib.io/p/qa-engineer-how-to-begin/ (дата обращения: 23.03.2018).