электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.01

Каскадная модель жизненного цикла программного обеспечения

Желудков А. В., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Егерев А.А., ассистент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»
bauman@bmstu.ru

Описание проблемы

С развитием вычислительной техники стала появляться возможность написания программного обеспечения для массовых пользователей в промышленных масштабах, а, значит, появилась необходимость изменить старые подходы к программированию «по наитию» на более совершенные и универсальные. Как и в большинстве подобных ситуаций, была сделана попытка перенять опыт крупных производств из других отраслей [1] В частности, было заимствовано понятие жизненного цикла. Жизненный цикл программного обеспечения — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации [2].

Рассмотрим общую схему, по которой создаётся программное решение. Сначала команде разработчиков поступает заказ на реализацию программы или, чаще, системы программ. Далее, после согласования с заказчиком деталей, составляется проект, пишется код, и проводится его тестирование. Если конечный программный продукт удовлетворяет всем заявленным требованиям, то работа сдаётся заказчику. В отличие от физических объектов, программы не подвержены износу, как таковому, однако, по мере их эксплуатации, несмотря на успешное прохождение всех тестов, происходит выявление разнообразных ошибок и неточностей. Так же появляется необходимость в модернизации готового программного продукта путём добавления в него дополнительного функционала, в котором во время разработки ещё не было нужды. Все эти требуемые исправления вносятся на самом продолжительном этапе поддержки программного обеспечения (ПО).

Однако, к сожалению, предложенная выше схема хоть и описывает общую последовательность действий, но не отвечает на конкретные вопросы о том, как именно стоит организовать каждую из стадий работ. Например, сколько подстадий нужно выделить? Должна ли весь проект реализовывать одна команда разработчиков или стоит распределить работу между группами и как это сделать? Надо ли письменно описывать проделанные операции на каждой из стадий или конечной пользовательской документации вполне достаточно? Для ответа на эти и ещё многие вопросы были созданы и продолжают разрабатываться модели жизненного цикла ПО.

Анализ базовых моделей разработки ПО и обоснование актуальности работы

Рассмотрим основные особенности базовых моделей разработки программного обеспечения для обоснования актуальности изучения каскадной методологии. Начнём с краткого описания этой модели, так как она идёт первой по хронологическому порядку. Каскадная методология впервые описана в 1970 году и подразумевает строгое последовательное выполнение определённых этапов разработки (рис. 1): систематизация и анализ требований, проектирование, кодирование, тестирование, сопровождение и поддержка. Основная особенность данной модели заключается в простоте и большой схожести с общей схемой разработки программного обеспечения, что облегчало её внедрение.

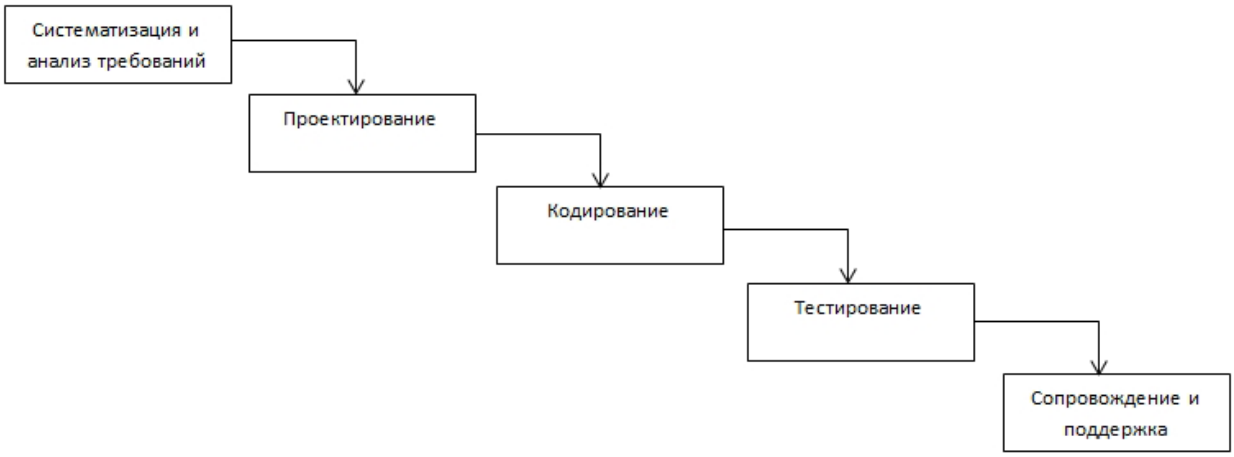

Рис. 1. Каскадная модель разработки ПО

На основе каскадной модели была выделена V-образная модель (рис. 2), в которой задачи реализации идут сверху вниз по левой стороне буквы V, а задачи тестирования — вверх по правой стороне буквы V [3]. Её основное отличие от каскадной модели заключается в наличии чётких связей между каждым этапом разработки и соответствующим этапом тестирования.

Рис. 2. V-образная модель разработки ПО

Спиральная модель представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции [4]. Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла (рис. 3).

Три описанные модели жизненного цикла являются базовыми, и на их основе уже создавались такие современные методологии, как экстремальное программирование (XP), SCRUM, модель RUP и другие. Многие компании при попытке внедрения этих методологий иногда (бывает, что и неосознанно) возвращаются к рассмотренным моделям и, чаще всего, именно к каскадной методологии из-за её интуитивной простоты и наглядности. Следовательно, для минимизации потерь и рисков появляется необходимость учитывать особенности именно этой, на первый взгляд, изжившей себя модели жизненного цикла. Если вовремя не обратить внимание на изменение подхода на «каскадный» из-за незнания его структуры и деталей, может реализоваться ситуация нехватки бюджета, срыва сроков проекта или вовсе необходимости начать всё заново. В результате проведённого анализа был сделан вывод, что каскадная методология лежит в основе базовых моделей разработки ПО, поэтому более подробное её рассмотрение остаётся актуальным вопросом и на сегодняшний день, которому и посвящена оставшаяся часть статьи.

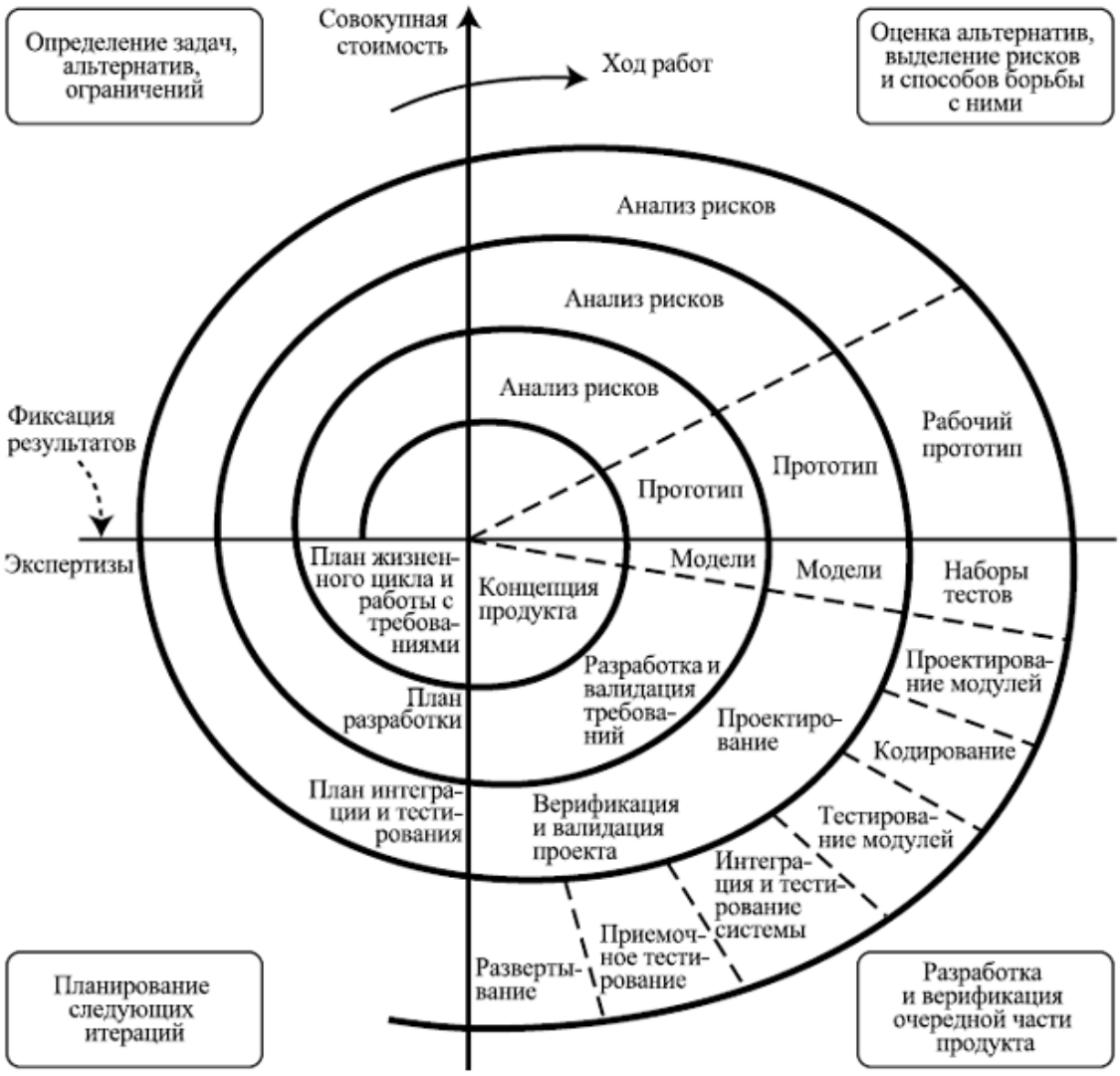

Рис. 3. Спиральная модель разработки ПО

Описание каскадной модели

В 1970 году Винстон Ройс [5] в своей статье описал в виде концепции то, что впоследствии стало именоваться как «каскадная модель», и на её примере продемонстрировал недостатки такого подхода к разработке.

В оригинальной каскадной модели Ройса, следующие фазы шли в таком порядке:

1. Определение требований
2. Проектирование
3. Конструирование (также «реализация» либо «кодирование»)
4. Интеграция
5. Тестирование и отладка (также «верификация»)
6. Инсталляция

7. Поддержка

Каскадная модель предусматривает последовательную организацию работ. За это, в дальнейшем, подобную модель стали именовать «водопад». Впервые в литературе этот термин использовали Белл и Тейер [6]. Переход от одной фазы к другой происходит только после полного и успешного завершения предыдущей и подразумевается, что переходов назад либо вперёд или перекрытия фаз не происходит. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Каскадная модель демонстрирует классический подход к разработке различных систем в любых прикладных областях. Именно поэтому она и стала одной из первых «позаимствованных» для программной сферы. Для разработки информационных систем данная модель широко использовалась в 70-х и первой половине 80-х годов и на сегодняшний момент считается устаревшей и применяется очень редко.

Однако из-за огромной популярности в своё время каскадная модель претерпевала существенные изменения, в основном, потому, что представления о стадиях, на которые должна разбиваться работа постоянно менялись. Тем не менее, можно выделить 5 различных этапов, которые в том или ином виде обязательно присутствовали во всех вариациях описываемой методологии:

1. Систематизация и анализ требований
2. Проектирование
3. Кодирование
4. Тестирование
5. Сопровождение и поддержка

Рассмотрим каждый из этапов подробнее. На первом из них производятся непосредственные переговоры с заказчиком, и выполняется анализ его требований. Здесь очень важно не ошибиться и корректно определить направление написания и развития программного продукта, поскольку ошибка может повлечь за собой катастрофические последствия, сведя всю последующую работу к нулю. В качестве результата на данном этапе, всеми заинтересованными сторонами подписывается техническое задание, в котором подробно и чётко описываются достигнутые договорённости и устанавливаются сроки и критерии качества продукта.

На втором этапе выбираются оптимальные алгоритмы для реализации всех записанных в техническом задании требований. Выбранные методы подразделяются на элементарные подзадачи, которые можно реализовать тривиальным образом. Для каждой

такой подзадачи составляется список всех необходимых программных и технических решений для её реализации. При необходимости для наглядности представления, используются диаграммы и блок-схемы. Результатом данного этапа работы является пакет проектной документации.

Третий этап — кодирование. Здесь происходит непосредственная реализация программного обеспечения в соответствии с проектными решениями, выработанными на втором этапе. Результатом является готовый программный продукт.

На четвёртой стадии проводится проверка полученного ПО на предмет отсутствия в нём ошибок и соответствия требованиям, прописанным в техническом задании. Результатом является отчёт об успешном прохождении всех тестов.

Последний этап — написание сопроводительной документации, сдача готового проекта заказчику и последующая его поддержка в течение оговоренного на первом этапе периода времени.

Каскадная модель разработки ПО очень похожа на общую схему, по которой создаётся программное решение, однако, она существенно дополняет её. В частности, в данной методологии подразумевается чёткая последовательность выполнения этапов разработки. Каждая стадия в данной модели завершается выпуском определённого пакета документации для возможности выполнения последующих этапов другой командой разработчиков. Методику «Каскадная модель» довольно часто критикуют за недостаточную гибкость и объявление самоцелью формальное управление проектом в ущерб срокам, стоимости и качеству [7]. Тем не менее, при управлении большими проектами, формализация часто являлась очень большой ценностью, так как могла кардинально снизить многие риски проекта и сделать его более прозрачным.

На основе каскадной методологии построены российские стандарты и ГОСТы по созданию автоматизированных систем (34-й и 19-й серий), например

ГОСТы 34-й серии:

- ГОСТ 34.601-90 «Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания»;
- ГОСТ 34.201-89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем»;
- ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы»;
- ГОСТ 34.603-92 «Информационная технология. Виды испытаний автоматизированных систем»;

- ГОСТ 34.320-96 «Информационные технологии. Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы»;
- ГОСТ 34.321-96 «Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными».

ГОСТы 19-й серии:

- ГОСТ 19.001-77 «Единая система программной документации. Общие положения»;
- ГОСТ 19.101-77 «Единая система программной документации. Виды программ и программных документов»;
- ГОСТ 19.102-77 «Стадии разработки»;
- ГОСТ 19.103-77 «Обозначения программ и программных документов»;
- ГОСТ 19.104-78 «Основные надписи»;
- ГОСТ 19.105-78 «Общие требования к программным документам»;
- ГОСТ 19.106-78 «Требования к программным документам, выполненным печатным способом»;
- ГОСТ 19.201-78 «Техническое задание, требования к содержанию и оформлению»;
- ГОСТ 19.202-78 «Спецификация. Требования к содержанию и оформлению» и др.

Методические указания:

- РД-34.698-90 «Методические указания. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы требования к содержанию документов».

Данные стандарты подразумевают, что все требования должны быть сначала описаны, и только потом можно приступать к этапу кодирования, т.е. к процессу написания программного кода, скриптов с целью реализации определенного алгоритма на определенном языке программирования. В требования должны быть включены как функциональные задачи, так и все способы взаимодействия с другими системами и способы представления информации, поэтому достаточно часто описание всех этих требований по ГОСТу занимает длительное время (от нескольких месяцев до нескольких лет).

Достоинства каскадной модели

Каскадная модель имеет определённый перечень преимуществ, благодаря которым она и была столь популярна в своё время. Основными из них являются:

- Простота понимания модели. Это очень важная характеристика, позволяющая новым членам команды, которые не знакомы с методологией, быстро влиться в рабочий процесс.
- На каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности. На заключительных этапах также разрабатывается пользовательская документация, охватывающая все предусмотренные стандартами виды обеспечения информационной системы (организационное, методическое, информационное, программное, аппаратное).
- Выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения и соответствующие затраты.

Недостатки каскадной модели

- Существенная задержка в получении результатов из-за того, что согласование каждого из этапов может происходить только после его завершения в связи с последовательностью выполнения работ и, как следствие, высокий уровень риска и ненадежность инвестиций. По сведениям консалтинговой компании The Standish Group в США более 31 % проектов корпоративных информационных систем (IT-проектов) заканчивается неудачей; почти 53 % IT-проектов завершается с перерасходом бюджета (в среднем на 189 %, то есть почти в два раза); и только 16,2 % проектов укладывается и в срок, и в бюджет.
- Ошибки и недоработки на любой из стадий проявляются, как правило, на последующих этапах работ, что приводит к необходимости нарушить основное правило последовательности и снова обратиться к более ранним стадиям.
- Невозможность параллельного ведения работ по проекту
- Чрезмерная информационная перенасыщенность каждого из этапов, поскольку зачастую, не требуется создание столь подробных пакетов документации, на изучение которых требуется много времени.
- Сложность управления проектом из-за необходимости разрешения большого числа конфликтов, часто возникающих при выяснении причин и поиске виновных в ситуации возврата на одну из предыдущих стадий работ.

Применение каскадной модели

Не смотря на то, что описываемая методология считается морально устаревшей, она до сих пор может успешно применяется в следующих ситуациях:

- при внедрении проектов длительностью от нескольких недель до 2–3 месяцев, т. к. описанные требования не успевают устареть;
- при внедрении систем, где нет подзадач и нескольких этапов разработки функционала (например, после разработки основного функционала системы электронных денег нужно будет доработать ее взаимодействие с системой бухгалтерского учета, а требований к этому взаимодействию пока нет);
- когда требования к создаваемой системе четко определены и зафиксированы.

Выводы и рекомендации

Проанализировав представленный выше материал, был сделан вывод, что при использовании каскадной модели разработки жизненного цикла программного обеспечения необходимо учитывать её особенности, чтобы проект не оказался бессмысленной затеей. Во-первых, следует внимательно следить за настроениями в коллективе и вовремя разрешать конфликтные ситуации, которые часто будут неявные, так как однозначно определить по чьей вине произошла ошибка при данной методологии в большинстве ситуаций невозможно и попытки поиска ответственных могут сильно усложнить отношения в группе. Во-вторых, попытки начать параллельное ведение работ при каскадной модели приведут к стремительному увеличению числа ошибок и, как следствие, к необходимости возвращаться назад и росту напряжённости. В-третьих, каскадная модель, зачастую, приводит к необходимости увеличить бюджет проекта, и, следовательно, нужно это учитывать и более тщательно контролировать расходы.

Чтобы была возможность учесть вышеописанные особенности, надо чётко понимать отличительные черты каскадной методологии, чтобы успеть вовремя заметить её использование, несмотря на попытки внедрения других, более современных и оптимальных, моделей разработки ПО.

Список литературы

1. Скопин. И. Н. Модели жизненного цикла программного обеспечения. Режим доступа: http://www.computer-museum.ru/books/n_collection/models.htm (дата обращения 29.03.2015)

2. Жизненный цикл программного обеспечения. Режим доступа: https://ru.wikipedia.org/wiki/Жизненный_цикл_программного_обеспечения (дата обращения 29.03.2015)
3. V-Model. Available at: https://ru.wikipedia.org/wiki/V-Model, accessed 29.03.2015.
4. Igor Mats. Модели жизненного цикла программного обеспечения. Режим доступа: http://habrahabr.ru/post/111674/ (дата обращения 29.03.2015)
5. Waterfall model. Available at: https://en.wikipedia.org/wiki/Waterfall_model, accessed 29.03.2015.
6. Bell Thomas E., Thayer. T. A. Software requirements: Are they really a problem? // Second international conference on Software engineering. IEEE Computer Society Press proceedings: 1976. P. 61-68.
7. Нужно ли заказчику понимание проблем разработки ПО? Режим доступа: http://becmology.ru/blog/4c/soft_dev01.htm (дата обращения 29.03.2015).