*Лекция: Жизненный цикл программного обеспечения*

1. *Понятие технологии разработки программного обеспечения*

В современном мире всеобщей компьютеризации и информатизации требования, предъявляемые к программному обеспечению вообще и к программным продуктам (ПП), программным средствам (ПС) и программам в частности, весьма высоки, что очевидно. В связи с этим обеспечение удовлетворяющих пользователя потребительских качеств программы, таких как надежность, быстродействие, соответствие заявленным возможностям, полнота документации, возможность расширения, развития и т.д., без строгого соблюдения определенной технологии практически невозможно.

Рассмотрим сначала основные термины и определения. Политехнический словарь рассматривает словом «технология» (от греч. techne – искусство, мастерство, умение, и логия) в широком смысле как совокупность «методов обработки, изготовления, изменения состояния, свойств, формы сырья, материала или полуфабрикатов, применимых в процессе производства, для получения готовой продукции»; как науку «о способах воздействия на сырье, материалы и полуфабрикаты соответствующими орудиями производства. Разработка технологии осуществляется по отраслям производства». В Энциклопедическом словаре определение примерно то же, более того, задача технологии заключается в выявлении «физических, химических, механических и других закономерностей с целью определения и использования на практике наиболее эффективных и эконо­мичных производственных процессов». В «Толковом словаре» технология – это «совокупность производственных про­цессов в определенной отрасли производства, а также научное описание способов производства».

Таким образом, на основании анализа вышеприведенных определений под технологией программирования в широком смысле следует понимать технологию разработки программного средства как совокупность абсолютно всех технологических процессов его соз­дания – от момента зарождения идеи о данном ПС до составле­ния необходимой программной документации. Каждый процесс указанной совокупности базируется на использовании неких методов и средств, например компьютерных (в этом случае следует говорить о компьютерной технологии программирования).

В литературе имеются и другие, отличные от приведенного, понятия технологии программирования. Используется также близкое к обсуждаемому понятие программной инженерии, опре­деляемой как систематический подход к разработке, эксплуатации, сопровождению и изъятию из обращения программных средств. Главное различие между технологией программирова­ния и программной инженерией в качестве учебных дисциплин заключается в способах рассмотрения и систематизации мате­риала. В технологии программирования акцент делается на изу­чении процессов разработки ПС (технологических процессов) и порядке их прохождения – в этих процессах используются опре­деленные методы и инструментальные средства разработки ПС (их применение и образует технологический процесс), тогда как в программной инженерии изучаются прежде всего методы и инструментальные средства разработки ПС с точки зрения достижения определенных целей: они могут использоваться в раз­ных технологических процессах (и в разных технологиях программирования). Вопросом о том, каким образом эти методы и средства создают технологический процесс, в этом случае никто не задается.

Не следует также путать технологию программирования с методологией программирования, несмотря на то что в обоих случаях изучаются соответствующие методы. Дело в том, что в технологии программирования методы рассматриваются «сверху», т.е. с точки зрения организации технологических процес­сов, а в методологии программирования – «снизу», т.е. с точки зрения основ их построения.

Имея в виду, что надежность является неотъемлемым атрибутом ПС, технологию программирования здесь целесообразно рассмат­ривать как технологию разработки надежных ПС. Это означает обсуждение, во-первых, всех процессов разработки ПС (от идеи создания до «утилизации»), а во-вторых, вопросов построения программных конструкций, описания функций и принимаемых решений с точки зрения их человеческого восприятия. И, нако­нец, в качестве продукта технологии появится надежное про­граммное средство. Все вышеперечисленное будет существенно влиять на выбор методов и инструментальных средств при разработке ПС [2].

Целью программирования является выполнение систематической последовательности дей­ствий для достижения автоматической обработки данных. Таким образом, технология разработки программного обеспечения, или технология программирования, терминологически обозначает совокупность процессов для создания программного продукта требуемой функциональности.

Результатом таких процессов является программное средст­во – совокупность логически связанных программ на носителях данных, снабженных программной документацией и предназна­ченных для людей, не участвовавших в процессе разработки.

Поскольку технологический процесс разработки программного обеспечения аналогичен процессу разработки программного средства, далее будем рассматривать специфику технологии программирования именно по отношению к ПО.

1. *Основы разработки программного обеспечения*

В основе разработки и дальнейшего применения программ­ного обеспечения пользователем лежит понятие жизненного цикла, который, в сущности, является моделью его создания и использования, отражающей различные состояния, начиная с момента осознания необходимости появления данного ПО и за­канчивая моментом его полного выхода из употребления [5].

Существует несколько моделей жизненного цикла (ЖЦ), каждая из которых определяет различную методологию созда­ния систем, тем не менее все без исключения модели ЖЦ включают пять этапов и связей между ними с детальным описанием действий, моделей и результатов каждого этапа. Приведем названия и краткое содержание каждого этапа в соответствии с ГОСТ 19.102-77.

1. Техническое задание:

* постановка задачи;
* выбор критериев эффективности;
* проведение предварительных научно-исследовательских работ (НИР);
* разработка ТЗ.

2. Эскизный проект:

* структура входных и выходных данных;
* уточнение методов решения;
* общий алгоритм;
* разработка документации эскизного проекта.

3. Технический проект:

* уточнение структуры входных и выходных данных;
* разработка алгоритмов;
* формы данных;
* семантика и синтаксис языка;
* структура программы;
* конфигурация технических средств;
* план работ.

4. Рабочий проект:

* программирование и отладка;
* разработка документов;
* подготовка и проведение испытаний;
* корректировка программы и документов по итогам испытаний.

5. Внедрение:

* передача программы и документов для сопровождения;
* оформление акта;
* передача в Фонд алгоритмов и программ (ФАП).

1. *Модели жизненного цикла*

Исторически в ходе развития теории проектирования про­граммного обеспечения и по мере его усложнения утвердились четыре основные модели ЖЦ.

Первой по времени появления и самой распространенной явилась *каскадная модель* (рис. 2.1).

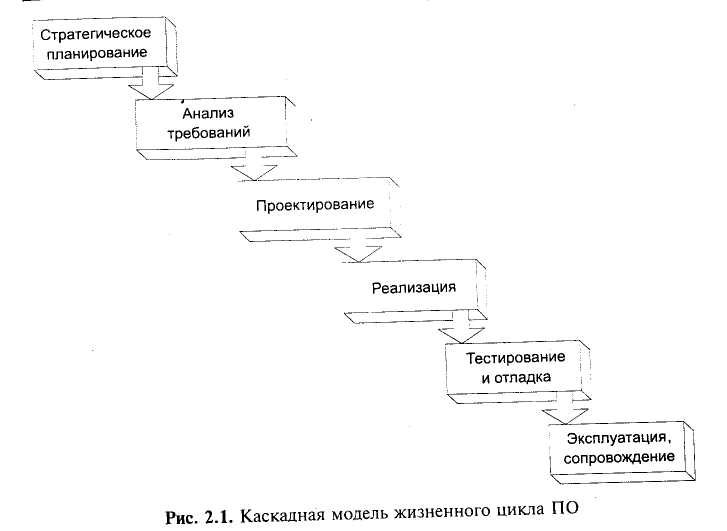


Рис. 1.1. Каскадная модель жизненного цикла

Каскадная модель характеризуется следующими основными особенностями:

* последовательным выполнением входящих в ее состав этапов;
* окончанием каждого предыдущего этапа до начала последующего;
* отсутствием временного перекрытия этапов (последующий этап не начнется, пока не завершится предыдущий);
* отсутствием (или определенным ограничением) возврата к предыдущим этапам;
* наличием результата только в конце разработки.

Выявление и устранение ошибок в каскадной модели произ­водится только на стадии тестирования, которая может растя­нуться во времени или вообще никогда не завершиться.

Следующей стадией развития теории проектирования ПО стала *итерационная модель* ЖЦ, или так называемая поэтапная модель с промежуточным контролем (рис. 2.2).

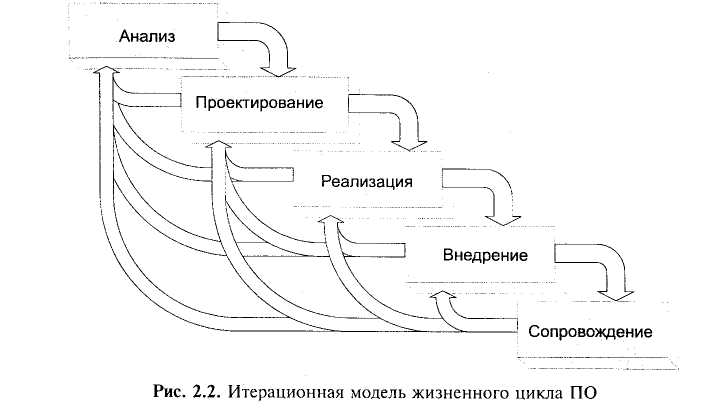


Рис. 1.2. Итерационная модель жизненного цикла

Основной ее осо­бенностью является наличие обратных связей между этапами, вследствие чего появляется возможность проведения проверок и корректировок проектируемой ИС на каждой стадии разработки. В результате трудоемкость отладки по сравнению с каскадной моделью существенно снижается. Итерационность модели про­является в обработке ошибок, выявленных промежуточным кон­тролем. Если на каком-либо этапе в ходе промежуточной про­верки обнаружена ошибка, допущенная на более ранней стадии разработки, необходимо повторить весь цикл работ этой стадии. При этом анализируются причины ошибки и корректируются в случае необходимости исходные данные этапа или его содержа­ние (последовательность действий).

К сожалению, в процессе разработки системы могут изме­ниться начальные требования, и в этом случае итерационная мо­дель может оказаться неэффективной.

Третья модель ЖЦ ПО – *спиральная (spiral) модель* (рис. 2.3) поддерживает итерации поэтапной модели, но осо­бое внимание здесь уделяется начальным этапам проектирова­ния: анализу требований, проектированию спецификаций, пред­варительному проектированию и детальному проектированию. Каждый виток спирали соответствует поэтапной модели созда­ния фрагмента или версии ПО, уточняются цели и требования к программному обеспечению, оценивается качество разработан­ного фрагмента или версии и планируются работы следующей стадии разработки (витка). Таким образом, углубляются и кон­кретизируются все детали проектируемого ПО, в результате получается продукт, который удовлетворяет всем требованиям за­казчика.

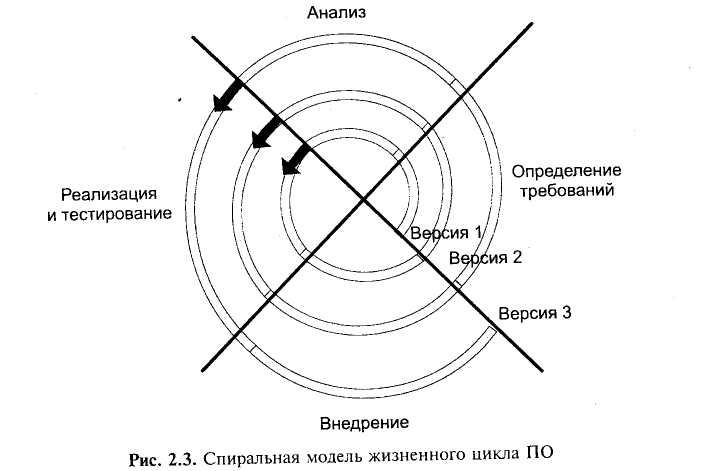


Рис. 1.3. Спиральная модель жизненного цикла

*Rational Objeciory Process -*– *модель жизненного цикла (методология объектно-ориентированного программирования)*

Объектно-ориентированное проектирование программного обеспечения стало результатом появления объект­но-ориентированного программирования (ООП), т.е. примене­ние новой методологии началось с этапа кодирова­ния. Ранние стадии описания предметной области и разработки архитектуры системы не поддерживались, первые варианты использования объектно-ориентированной методологии в большой степени являлись повторением принципов ООП. Такие вопросы, как декомпозиция предметной области, спецификация требований, интерфейс пользователя, не рассматривались, одна­ко успехи объектно-ориентированного программирования заста­вили распространить новую технологию на весь жизненный цикл ПО. В результате все преимущества подхода применяются не только в процессе кодирования, но и на более ранних этапах. Таким образом, были определены основные компоненты мето­дологии:

* модель жизненного цикла;
* действия;
* нотация языка.

Жизненный цикл UML (Rational Objectory Process). Фирма Rational Software, разработавшая язык UML, предложила также и свою модель ЖЦ, которая называется Rational Objectory Process (ROP). Означенная технология прямого перевода не име­ет, так как, во-первых, rational в данном случае употребляется и в значении «рациональный», и как название фирмы одновременно, во-вто­рых, слова objectory в английском языке не существует, его лингвообразование аналогично слову repository (накопитель).

Основные свойства ROP-технологии.

1. Rational Objectory Process – итеративный процесс, в течение которого происходит последовательное уточнение результатов;
2. Rational Objectory Process направлен именно на создание мо­делей, а не на разработку каких-либо других элементов проекта (например, текстовых документов);
3. Действия Rational Objectory Process определяются в первую очередь блоками использования (use case) (рис. 2.4);
4. Rational Objectory Process разбит на циклы, каждый из кото­рых, в свою очередь, состоит из четырех фаз:

* начальная стадия (Inception);
* разработка (Elaboration);
* конструирование (Construction);
* ввод в эксплуатацию (Transition).

Результатом работы каждого цикла является своя вер­сия программной системы.

Каждая стадия завершается в четко определенной точке (milestone). В этот момент должны быть достигнуты важные результаты и приняты критически важные решения о дальнейшей разработке.

*Начальная стадия* может принимать множество разных форм. Для крупных проектов – это всестороннее изучение всех возможностей реализации на протяжении нескольких месяцев. Здесь же вырабатывается бизнес-план проекта, определяется его стоимость, примерный доход, а также ограничения ресурсов – иными словами, выполняется некоторый начальный анализ оценки проекта.

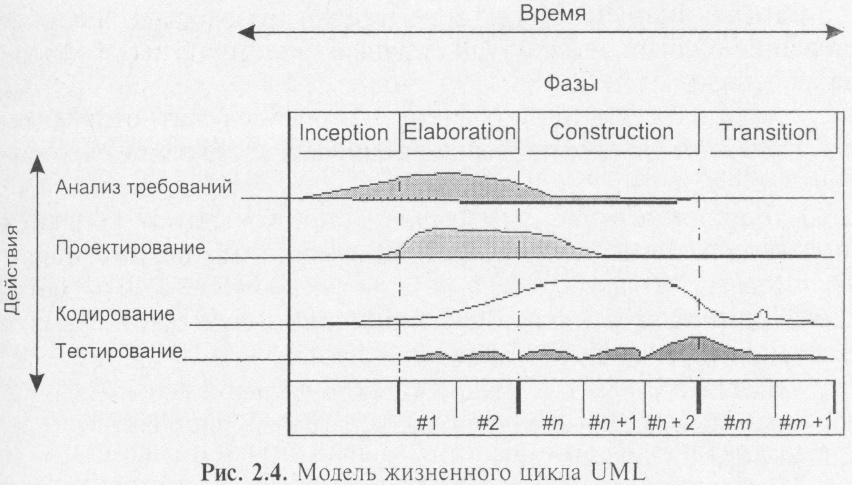


Рис. 1.4. Модель жизненного цикла UML

Окончанием начального этапа могут служить следующие ре­зультаты:

* начальный проектный словарь терминов;
* общее описание системы: основные требования к проекту, его характеристики и ограничения;
* начальная модель вариантов использования;
* начальный бизнес-план;
* план проекта, отражающий стадии и итерации;
* один или несколько прототипов.

На стадии *разработки* выявляются более детальные требова­ния к системе, выполняется высокоуровневый анализ предмет­ной области и проектирование базовой архитектуры системы, создается план конструирования и устраняются наиболее риско­ванные элементы проекта.

Самым важным результатом стадии разработки является описание базовой архитектуры будущей системы. Эта архитекту­ра включает:

* модель предметной области, которая служит отправным пунктом для формирования основных абстракций предметной области;
* технологическую платформу, определяющую основные элементы технологии реализации системы и их взаимодей­ствие.

Стадия разработки занимает примерно пятую часть времени создания проекта. Ее результатом являются:

* оценка времени реализации каждого варианта использования;
* идентификация всех наиболее серьезных рисков и возможности их ликвидации.

Сущность *стадии конструирования* заключается в определе­нии последовательности итераций конструирования и вариантов использования, реализуемых на каждой итерации, которые явля­ются одновременно инкрементными и повторяющимися.

При этом необходимо отметить следующее:

* итерации являются инкрементными в соответствии с вы­полняемой функцией. Каждая итерация добавляет очеред­ные конструкции к вариантам использования, реализованным во время предыдущих итераций;
* итерации являются повторяющимися по отношению к разрабатываемому коду. На каждой итерации некоторая часть существующего кода переписывается, для того чтобы сделать его более гибким.

Результатом стадии конструирования является продукт, гото­вый к передаче пользователям, содержащий, как правило, ру­ководство пользователей и готовый к интеграции на требуемых платформах.

Назначением стадии *ввода в эксплуатацию* является переда­ча готового продукта в полное распоряжение конечных пользо­вателей. Данная стадия включает:

* бета-тестирование, позволяющее убедиться, что новая сис­тема соответствует ожиданиям пользователей;
* параллельное функционирование с существующей (legacy) системой, которая подлежит постепенной замене;
* оптимизацию производительности;
* обучение пользователей и специалистов службы сопровождения.