Департамент образования города Москвы

ГосударственноеБюджетное Образовательное Учреждение

среднего профессионального образования

Политехнический колледж №8

имени дважды Героя Советского Союза И.Ф. Павлова

КУРСОВАЯ РАБОТА

Предмет: Разработка и эксплуатация автоматизированных информационных систем

Тема: Модели жизненного цикла автоматизированных информационных систем

ИСПОЛНИТЕЛЬ: Антохин Игорь Николаевич

Москва, 2010г.

Содержание

Введение

Глава 1. Модели жизненного цикла автоматизированных информационных систем

1.1 Жизненный цикл АИС

1.2 Процессы жизненного цикла АИС

1.3 Основные процессы жизненного цикла АИС

1.4 Вспомогательные процессы жизненного цикла АИС

1.5 Организационные процессы жизненного цикла АИС

1.6 Модели жизненного цикла автоматизированных информационных систем

1.7 Каскадная модель

1.8 Спиральная модель

Глава 2. СASE-технологии

2.1 Основные методологии проектирования автоматизированных систем на основе CASE-технологий

2.2 Фазы жизненного цикла программного обеспечения. Фаза анализа и планирования требований

2.3 Фаза проектирования

2.4 Фаза построения

2.5 Фаза внедрения

Глава 3. Модели жизненного цикла программного продукта

3.1 Определение модели жизненного цикла АИС

3.2 Каскадная модель

3.3 V-образная модель

3.4 Модель прототипирования

3.5 Модель быстрой разработки приложений (RAD-модель)

3.6 Многопроходная модель

3.7 Спиральная модель

Заключение

Список использованной литературы

Введение

Современное общество невозможно представить без компьютера. Они настолько широко и глубоко внедрились в нашу жизнь, что очень трудно назвать какую-либо сферу деятельности человека, где они не использовались. В связи с этим серьезные требования предъявляются и к аппаратной части современных компьютеров, и к используемому программному обеспечению. В основном именно программное обеспечение, или, иными словами, программные продукты, обеспечивают возможность широкого использования компьютеров. Стоит нам переустановить программное обеспечение компьютера или добавить какой-либо новый программный продукт, и мы сможем решать на этом компьютеры совершенно новые задачи. Следовательно, используемые программные продукты должны соответствовать определенным критериям, обеспечивающим надежность работы компьютера и удобство работы пользователя.

В данной курсовой работе рассматриваются модели жизненного цикла автоматизированных информационных систем и программного продукта. Работа состоит из трех глав.

В первой главе рассказывается о моделях жизненного цикла автоматизированных информационных системах.

Жизненный цикл автоматизированных информационных систем - это непрерывный процесс, который начинается с момента принятия решения о необходимости создания ИС и заканчивается в момент ее полного изъятия из эксплуатации.

Модель жизненного цикла - структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ.

Наибольшее распространение получили две основные модели ЖЦ:

* каскадная модель (70-85 гг.);
* спиральная модель (86-90 гг.).

Во второй главе речь идет о CASE-технологиях. CASE-технология - технология, базирующаяся на методологиях подготовки информационных систем и соответствующих комплексах интегрированных инструментальных средств, а также ориентированная на поддержку полного жизненного цикла автоматизированной системы или его основных этапов.

Жизненный цикл программного обеспечения в соответствии с методологией RAD состоит из четырех фаз: анализа и планирования требований; проектирования; построения; внедрения.

В третьей главе - модели жизненного цикла программного продукта.

Под моделью жизненного цикла разработки программного продукта понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении жизненного цикла разработки программного продукта. Наибольшее распространение получили следующие модели жизненного цикла разработки программного продукта: каскадная модель, или водопад (waterfall model); v-образная модель (v-shaped model); модель прототипирования (prototype model); модель быстрой разработки приложений, или RAD-модель (RAD-rapid application development model); многопроходная модель (incremental model); спиральная модель (spiral model).

ГЛАВА 1. Модели жизненного цикла автоматизированных информационных систем

1.1 Жизненный цикл АИС

Жизненный цикл автоматизированных информационных систем - это непрерывный процесс, который начинается с момента принятия решения о необходимости создания ИС и заканчивается в момент ее полного изъятия из эксплуатации (рис.1).

Структура ЖЦ

Основные процессы ЖЦ

Вспомогательные процессы ЖЦ

Организационные процессы ЖЦ

Этапы ЖЦ

Жизненный цикл АИС

Модели ЖЦ

Спиральная модель

Каскадная модель

Рис.1 Структурная схема терминов

1.2 Процессы жизненного цикла АИС

Жизненный цикл - одно из базовых понятий методологии проектирования информационных систем. Это непрерывный процесс, который начинается с момента принятия решения о необходимости создания информационной системы и заканчивается в момент ее полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим жизненный цикл, является международный стандарт ISO/IEC 12207 (ISO - International Organization of Standardization - Международная организация по стандартизации, IEC - International Electrotechnical Commission - Международная комиссия по электротехнике). Он определяет структуру жизненного цикла, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания информационной системы.

Структура жизненного цикла по стандарту ISO/IEC 12207 базируется на трех группах процессов: основные, вспомогательные, организационные.

1.2.1. Основные процессы жизненного цикла АИС

Основные процессы включают в себя набор определенных действий и связанных с ними задач, которые должны быть выполнены в течение жизненного цикла программного продукта.

К основным относятся процессы приобретения, постаки, разработки, эксплуатации и сопровождения.

Процесс приобретения охватывает действия заказчика по приобретению ПП. К этим действиям относятся:

1. Инициирование приобретения включает в себя много задач, в том числе определение заказчиком своих потребностей в приобретении, разработки или усовершенствование системы ПП.
2. Подготовка заявочных предложений подразумевает разработку и составление предложений, которые должны содержать: требования к разрабатываемой или покупаемой системе; перечень необходимых ПП; условия и соглашения; технические ограничения.
3. Подготовка и корректировка договора включает в себя следующие задачи: выбор поставщиком критерия оценки предложений; выбор конкретного поставщика на основе анализа предложений; подготовка и заключение договора с поставщиком; внесение изменений (при необходимости) в договор в процессе его выполнения.
4. Надзор за деятельностью поставщика осуществляется в соответствии с действиями, предусмотренными в процессе совместной оценки аудита.
5. Приемка и завершение работ

В процессе приемки подготавливаются и выполняются необходимые тесты. Завершение работ по договору осуществляется в случае удовлетворения всем условиям приемки.

Процесс поставки охватывает действия и задачи поставщика при снабжении заказчика ПП или услугой.

К этим действиям относятся:

1. Инициирование поставки заключается в рассмотрении поставщиком заявочных предложений и принятия решения;
2. Подготовка ответа на заявочные предложения выполняются в соответствии с принятыми решениями;
3. Подготовка договора осуществляется после выбора заказчиком конкретного поставщика;
4. Планирование выполняется после заключения договора и включает в селя следующие задачи: принятие решения поставщиком относительно выполнения работ своими силами или с подключением субподрядчика; разработку поставщиком плана управления проектом, содержащего организационную структуру проекта, разграничение ответственности, технические требования к среде разработки, управление субподрядчиками.

Субподрядчик – организация, индивидуум или корпорация, заключившая договор с поставщиком на исполнение части работ, которые поставщик должен выполнить по договору с заказчиком.

1. Выполнение и контроль
2. Проверка и оценка
3. Поставка и завершение работ выполняется в соответствии с оговоренными в процессе инициирования действиями по приемки и завершении работ.

Процесс разработки охватывает действия и задачи разработчика и предусматривает следующие основные направления работ:

1. Создание ПП и его компонентов с заданными требованиями, включая оформление проектной и эксплуатационной документации
2. Подготовку материалов, необходимых для проверки работоспособности и качества ПП
3. Подготовку материалов, необходимых для организации обучения персонала и т.д.

Процесс эксплуатации охватывает действия и задачи оператора – организации, занимающейся эксплуатацией разработанного ПП. К этим действиям относятся: подготовительная работа, эксплуатационное тестирование, эксплуатация системы, поддержка пользователей заключается в оказании помощи и консультациях при обнаружении ошибок в процессе эксплуатации ПП.

Процесс сопровождения. Данный процесс активизируется при изменениях (модернизации) ПП и соответствующей документации, вызванных возникшими проблемами.

Основной целью этих процессов является создание надежного, полностью удовлетворяющего требованиям заказчика ПП в установленные договором сроки.

1.2.2 Вспомогательные процессы жизненного цикла АИС

Основной целью этих процессов является создание надежного, полностью удовлетворяющего требованиям заказчика ПП в установленные договором сроки. К вспомогательным относятся процессы документирования, управления конфигурацией, обеспечения качества, верификации, аттестации, совместной оценки, аудита, разрешения проблем.

Процесс документирования предусматривает формализованное описание информации, созданной в течении ЖЦ ПП.

Этот процесс включает в себя:

1. Подготовительную работу, которая требуется для определения и согласования необходимого перечня документов и документируемых процедур;
2. Проектирование и разработку документации, которые выполняются в процессе работы над ПП и завершается одновременно с завершением его ЖЦ;
3. Выпуск документации, который осуществляется по мере ее готовности;
4. Сопровождение включает в себя действия по корректировки и обновлению документации в процессе ЖЦ ПП.

Процесс управления конфигурацией предполагает применение административных и технических процедур на всем протяжении ЖЦ ПП.

Согласно стандарту IEEE-90 под конфигурацией ПП понимается совокупность его функциональных и физических характеристик, установленных в технической документации и реализованных в ПП.

Этот процесс включает в себя:

1. Подготовительную работу, которая заключается в планировании управления конфигурацией;
2. Идентификацию конфигурации - устанавливает правила, с помощью которых можно однозначно идентифицировать и различать компоненты ПП и их версии. Кроме того каждому компоненту и его версиям соответствует однозначно обозначаемый комплект документации;
3. Контроль конфигурации – предназначен для систематической оценки предполагаемых модификаций ПП и координированной их реализации с учетом эффективности каждой модификации и затрат на ее выполнение;
4. Учет состояния конфигурации - представляет собой регистрацию состояния компонентов ПП, подготовку отчетов обо всех реализованных и отвергнутых модификациях версий компонентов ПП;
5. Оценку конфигурации – заключается в оценки функциональной полноты компонентов ПП;
6. Управление выпуском и поставкой включает в себя изготовление эталонных копий программ и документации, их хранение и поставку пользователям в соответствии с порядком, принятом в организации.

Процесс обеспечения качества обеспечивает соответствующую гарантию того, что ПП и процессы его ЖЦ ПП соответствуют заданным требованиям и утвержденным планам.

Для получения достоверных оценок создаваемого ПП процесс обеспечения его качества должен происходить независимо от субъектов, непосредственно связанных с разработкой ПП.

Процесс верификации состоит в доказательстве, того, что ПП, являющийся результатом некоторого действия полностью удовлетворяет требования или условия, зависящих от предшествующих действий.

Верификация может проводиться как самим исполнителем, так и другим специалистом данной организации, а так же специалистом сторонней организации. Верификация в узком смысле означает формальное доказательство правильности ПП. Данный процесс может включать в себя анализ, оценку и тестирование.

Процесс аттестации предусматривает определение полноты соответствия заданных требований к создаваемой системе или ПП.

Под аттестацией обычно понимают подтверждение и оценку достоверности проведенного тестирования ПП. Аттестация должна гарантировать полное соответствие, а также возможность его безопасного и надежного применения пользователем.

Процесс совместной оценки предназначен для оценки состояния работ по проекту и ПП. Он заключается в основном в контроле за планированием и управлением ресурсами, персоналом, аппаратурой и инструментальными средствами проекта.

Процесс аудита представляет собой определение соответствия требованиям, планам и условиям договора как хода выполнения работ по созданию ПП, так и самого продукта.

Аудит служит для установления соответствия реальных работ и отчетов, поэтому аудиторы (ревизоры) не должны иметь прямой зависимости от разработчиков ПП.

Процесс разрешения проблем предусматривает анализ и решение проблем, обнаруженных в ходе разработки, эксплуатации и других процессов, независимо от их проблемы или источника. Каждая обнаруженная проблема должна быть идентифицирована, описана, проанализирована и разрешена.

Разрешение проблем проводится на всем протяжении ЖЦ ПП.

1.2.3 Организационные процессы жизненного цикла АИС

Основной целью организационных процессов является организация процесса разработки надежного, полностью удовлетворяющего требованиям заказчика программного продукта в установленные договором сроки и управление этим процессом. К организационным относятся процессы управления, создания инфраструктуры, усовершенствования, обучения.

Процесс управления проектами состоит из действий и задач, которые могут выполняться любой стороной, управляющей своими процессами. Данная сторона (менеджер) отвечает за управление за управление выпуска продукта, проектом и задачами соответствующих процессов, таких как приобретение, поставка, разработка, эксплуатация, сопровождение и др.

Процесс создания инфраструктуры охватывает выбор и поддержку (сопровождение) технологии, стандартов и инструментальных средств, выбор и установку аппаратных и программных средств, используемых для разработки, эксплуатации и сопровождения ПП.

Процесс усовершенствования предусматривает оценку, измерение, контроль, усовершенствование процессов ЖЦ ПП.

Усовершенствование процессов ЖЦ ПП направлено на повышение производительности труда всех участвующих в них специалистов за счет совершенствования используемой технологии, методов управления, выбора инструментальных средств и обучения персонала.

Процесс обучения охватывает первоначальное обучение и последующее повышение квалификации персонала. Основные процессы в значительной степени зависят от уровня знаний и квалификации персонала.

Для этого процесса должны быть запланированы необходимые ресурсы и технические средства автоматизации.

1.3 Модели жизненного цикла АИС

Модель жизненного цикла - структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ.

Наибольшее распространение получили две основные модели ЖЦ:

* каскадная модель (70-85 гг.);
* спиральная модель (86-90 гг.).

1.3.1 Каскадная модель

Каскадный способ - разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем (рис.2).

Положительные стороны применения каскадного подхода:

* на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
* выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении информационных систем, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования. В эту категорию попадают сложные расчетные системы, системы реального времени и другие подобные задачи.

Формирование требований к программному продукту

Прототипирование

Разработка

Тестирование

Эксплуатация и сопровождение

Рис.2 Схема каскадного подхода

Однако реально в процессе создания ИС постоянно возникает потребность в возврате к предыдущим этапам, уточнении или пересмотре ранее принятых решений. Реальный процесс создания информационной системы принимает следующий вид (рис.3):

Формирование требований к программному продукту

Прототипирование

Разработка

Тестирование

Эксплуатация и сопровождение

Рис.3 Реальный процесс создания ИС на базе каскадной модели

Одно из использовавшихся в западной литературе названий такой схемы организации работ: "водопадная модель" (waterfall model).

Основным недостатком каскадного подхода является существенное запаздывание с получением результатов. Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением. Другой недостаток - такое проектирование информационной системы ведет к примитивной автоматизации (по сути - "механизации") существующих производственных действий работников.

1.3.2 Спиральная модель

В спиральной модели жизненного цикла (рис.4) делается упор на начальные этапы ЖЦ: анализ и проектирование. Реализуемость технических решений проверяется путем создания прототипов.

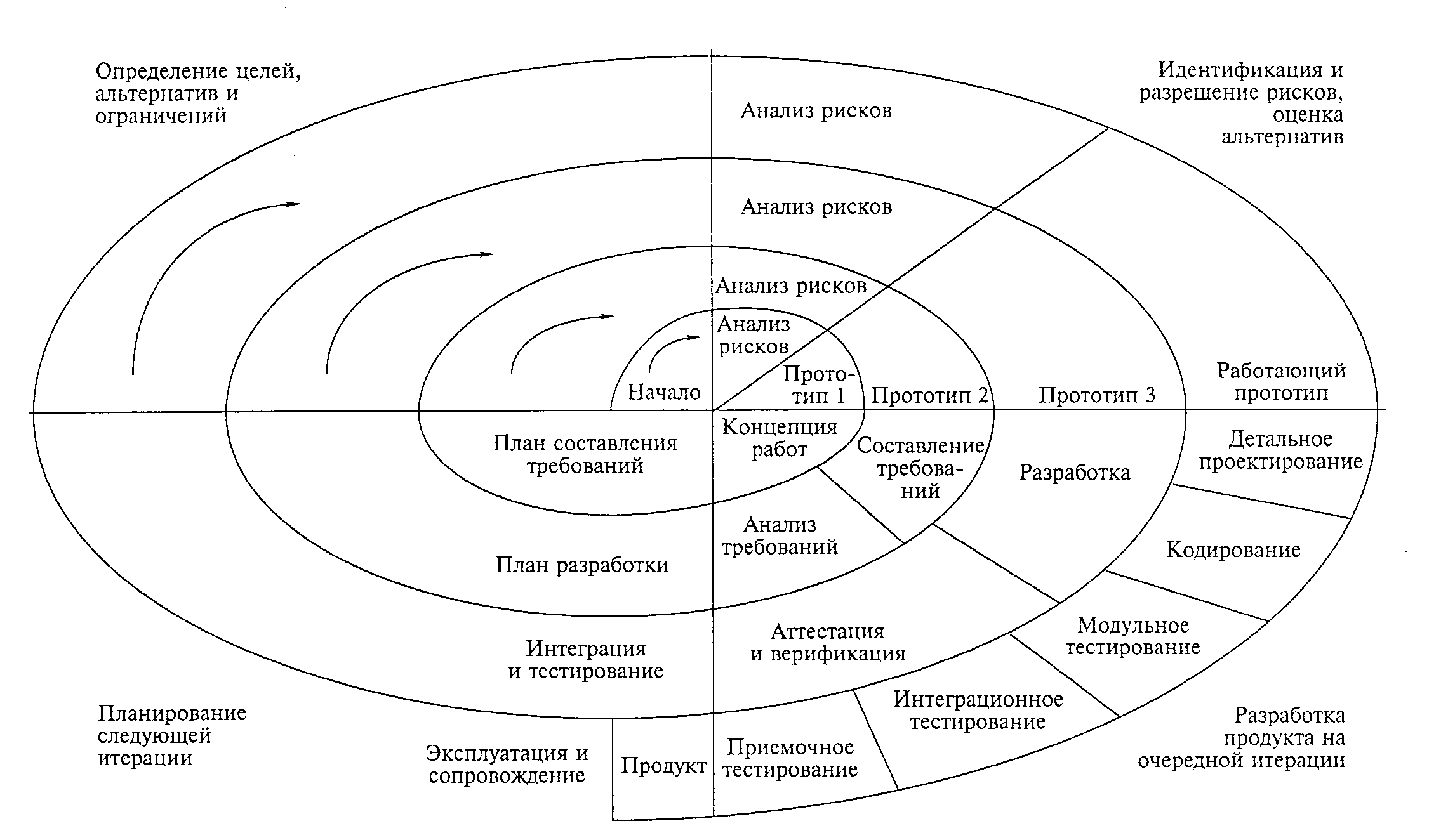


Рис 4. Спиральная модель

Каждый виток спирали соответствует созданию нового фрагмента или версии информационной системы, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Один виток спирали при этом представляет собой законченный проектный цикл по типу каскадной схемы. Такой подход назывался также "Продолжающимся проектированием". Позднее в проектный цикл дополнительно стали включать стадии разработки и опробования прототипа системы. Это называлось: "быстрое прототипирование", rapid prototyping approach или "fast-track".

Однако применение таких методов наряду с быстрым эффектом дает снижение управляемости проектом в целом и стыкуемости различных фрагментов информационной системы. Основная проблема спирального цикла - определение момента перехода на следующий этап. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

ГЛАВА 2. CASE-технологии

2.1 Основы методологии проектирования автоматизированных систем на основе CASE-технологий

Возрастающая сложность современных автоматизированных систем управления и повышение требовательности к ним обуславливает применение эффективных технологий создания и сопровождения автоматизированных систем в течение всего жизненного цикла. Такие технологии, базирующиеся на методологиях подготовки информационных систем и соответствующих комплексах интегрированных инструментальных средств, а также ориентированные на поддержку полного жизненного цикла автоматизированной системы или его основных этапов, получили название CASE-технологий и CASE-средств.

Для успешной реализации проекта автоматизированной системы должны быть построены полные и непротиворечивые, функциональные и информационные модели системы управления. Накопленный опыт проектирования указанных моделей показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако во многих случаях проектирование автоматизированной системы выполняется в основном на интуитивном уровне с применением неформальных методов, основанных на искусстве, практическом опыте и экспертных оценках. Кроме того, в процессе создания и функционирования АС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение автоматизированных систем управления. От перечисленных недостатков в наибольшей степени свободны подходы, основанные на программно-технических средствах специального класса - CASE-средствах, реализующих CASE-технологии создания и сопровождения АС.

Под термином CASE (Computer Aided Software Engineering) понимаются программные средства, поддерживающие процессы создания и сопровождения автоматизированной системы, включая анализ и формулировку требований, проектирование прикладного программного обеспечения и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным программным обеспечением и техническими средствами образуют полную среду разработки автоматизированной системы.

Одним из базовых понятий методологии проектирования автоматизированной системы является понятие жизненного цикла ее программного обеспечения.

Жизненный цикл программного обеспечения - это непрерывный процесс, который начинается с момента принятия решения о необходимости создания программного обеспечения автоматизированной системы и заканчивается в момент его полного изъятия из эксплуатации

Структура жизненного цикла программного обеспечения базируется на трех группах процессов: основные процессы жизненного цикла программного обеспечения (приобретение, поставка, разработка, эксплуатация, сопровождение); вспомогательные процессы, обеспечивающие выполнение основных процессов (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем); организационные процессы (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого жизненного цикла, обучение).

Разработка охватывает все работы по созданию ПО и его компонентов (анализ, проектирование и программирование) в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и качества программных проектов, материалов, необходимых для организации обучения персонала, и т.д.

Эксплуатация включает в себя работы по внедрению компонентов программного обеспечения (конфигурирование базы данных и рабочих мест пользователей, обеспечение эксплуатационной документацией, проведение обучения персонала и др.), локализацию проблем, возникающих при эксплуатации с устранением причин их возникновения, модификацию программного обеспечения в рамках установленного регламента, подготовку предложений по совершенствованию, развитию и модернизации системы. Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами. Результатами анализа, в частности, являются функциональные модели, информационные модели и соответствующие им диаграммы.

Жизненный цикл программного обеспечения носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах.

Известно несколько моделей жизненного цикла программного обеспечения. Под моделью жизненного цикла программного обеспечения понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении всего цикла. Модель жизненного цикла зависит от специфики автоматизированной системы и специфики условий, в которых система создается и функционирует.

К настоящему времени наибольшее распространение получили следующие две основные модели жизненного цикла: каскадный способ и спиральная модель.

Каскадная модель применяется, как правило, для разработки однородных автоматизированных систем, представляющих собой единое целое. Ее основной характеристикой является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем (рис.1). Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков. Преимущества применения каскадного способа заключаются в следующем: на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности; выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты. Каскадный подход хорошо зарекомендовал себя при построении автоматизированных систем, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования, с тем чтобы предоставить разработчикам свободу реализовать их технически как можно лучше. В эту категорию попадают сложные расчетные системы, системы реального времени и др. В то же время этот подход обладает рядом недостатков, вызванных, прежде всего тем, что реальный процесс создания автоматизированной системы никогда полностью не укладывается в такую жесткую схему, постоянно возникает потребность в возврате к предыдущим этапами уточнении или пересмотре ранее принятых решений.

Такую трансформацию каскадной схемы разработки автоматизированной системы можно рассматривать как "моделирование с промежуточным контролем". Межэтапные корректировки обеспечивают большую надежность каскадной модели, хотя и увеличивают весь период разработки. Основным недостатком каскадного подхода является существенное запаздывание с получением результатов. Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ, требования к автоматизированной системе "заморожены" в виде технического задания на все время ее создания. Таким образом, пользователи могут вносить свои замечания только после того, как работа над системой будет полностью завершена. В случае неточного изложения требований или их изменения в течение длительного периода создания автоматизированной системы пользователи получают систему, не удовлетворяющую их потребностям. Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением. От перечисленных недостатков свободна спиральная модель разработки автоматизированной системы (рис.3), в которой делается упор на начальные этапы жизненного цикла: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии программного обеспечения, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации. Разработка итерациями отражает объективно существующий спиральный цикл создания автоматизированной системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача - как можно быстрее показать пользователям автоматизированной системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований. Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков автоматизированных систем. В рамках спиральной модели жизненного цикла широкое распространение получил один из подходов к разработке программного обеспечения, известный как методология быстрой разработки приложений RAD (Rapid Application Development). Эта методология включает в себя три составляющие: небольшая команда программистов (от 2 до 10 человек); короткий, но тщательно проработанный производственный график (от 2 до 6 мес.); повторяющийся цикл, при котором разработчики по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком. Команда разработчиков должна представлять собой группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании программного обеспечения с использованием CASE-средств, способных хорошо взаимодействовать с конечными пользователями и трансформировать их предложения в рабочие прототипы.

Жизненный цикл программного обеспечения в соответствии с методологией RAD состоит из четырех фаз: анализа и планирования требований; проектирования; построения; внедрения.

2.2 Фаза анализа и планирования требований

На фазе анализа и планирования требований пользователи автоматизированной системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности. Формулирование требований к автоматизированной системе осуществляется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта автоматизированной системы, устанавливаются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации проекта в заданных размерах финансирования, на имеющихся аппаратных средствах и т.д. Результатом этого этапа должен быть список расставленных по приоритету функций будущей автоматизированной системы, а также предварительные функциональные модели автоматизированной системы.

2.3 Фаза проектирования

На этапе проектирования часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков. CASE-средства используются для быстрого получения работающих прототипов приложений. Пользователи, непосредственно взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. Анализируется и при необходимости корректируется функциональная модель. Каждый процесс рассматривается детально. При необходимости для элементарного процесса создается частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности. Устанавливаются требования разграничения доступа к данным. На этой же фазе происходит определение необходимой документации. После детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении автоматизированной системы на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время (60 - 90 дней). С использованием CASE-средств проект автоматизированной системы распределяется между различными командами (делится функциональная модель). Результатом данного этапа должны быть: общая информационная модель системы; функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков; точно определенные с помощью CASE-средств интерфейсы между автономно разрабатываемыми подсистемами; построенные прототипы экранов, отчетов, диалогов. Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что в традиционном подходе при передаче информации о проекте с этапа на этап нередко происходит неконтролируемое искажение данных. Применение единой среды хранения данных о проекте позволяет этого избежать. В отличие от обычных подходов, при которых используются специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасываются после устранения неясностей в проекте автоматизированной системы, в подходе RAD каждый прототип передается будущей системе. Таким образом, на следующую фазу передается более полная и полезная информация.

2.4 Фаза построения

На этапе построения осуществляется непосредственно сама быстрая подготовка приложения. При этом разработчики выполняют итеративное построение реальной автоматизированной системы управления на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется CASE-средствами автоматически. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять указанным ранее требованиям. Тестирование автоматизированной системы осуществляется в процессе разработки. После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения, а затем тестирование АС в целом. Завершается физическое проектирование автоматизированной системы, включающее: определение необходимости распределения данных; анализ использования данных; физическое проектирование базы данных; определение требований к аппаратным ресурсам и способов увеличения производительности, завершение разработки документации проекта. Результатом данного этапа является готовая автоматизированная система, удовлетворяющая всем согласованным требованиям.

2.5 Фаза внедрения

На фазе внедрения автоматизированной системы производится обучение пользователей и вносятся организационные изменения. Для этого этапа характерно то, что одновременно с внедрением новой АС осуществляется работа с существующей системой управления до полного внедрения новой. Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы. Приведенная схема разработки автоматизированной системы не является окончательной. Возможны различные варианты, зависящие, например, от начальных условий, в которых ведется создание автоматизированной системы: а) разрабатывается совершенно новая система; б) было проведено обследование предприятия и существует модель его деятельности; в) на предприятии уже существует автоматизированная система, которая может быть использована в качестве начального прототипа или должна быть интегрирована с вновь разрабатываемой системой управления.

ГЛАВА 3. Модели жизненного цикла программного продукта

3.1 Определение модели ЖЦ АИС

Под моделью жизненного цикла разработки программного продукта понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении жизненного цикла разработки программного продукта. Наибольшее распространение получили следующие модели жизненного цикла разработки программного продукта (таблица1. Краткие характеристики моделей жизненного цикла АИС): каскадная модель, или водопад (waterfall model); v-образная модель (v-shaped model); модель прототипирования (prototype model); модель быстрой разработки приложений, или RAD-модель (RAD-rapid application development model);многопроходная модель (incremental model); спиральная модель (spiral model).

Таблица 1.Краткие характеристики каждой из перечисленных моделей

|  |  |
| --- | --- |
| Название | характеристики |
| Каскадная модель | Прямолинейная и простая в использовании. Необходим постоянный жесткий контроль за ходом работы. Разрабатываемое программное обеспечение не доступно для изменений |
| v-образная модель | Простая в использовании. Особое значение придается тестированию и сравнению результатов фаз тестирования и проектирования |
| Модель прототипирования | Создается «быстрая» частичная реализация системы до составления окончательных требований. Обеспечивается обратная связь между пользователями и разработчиками в процессе выполнения проекта. Используемые требования не полные |
| Модель быстрой разработки приложений | Проектные группы небольшие (3… 7 человек) и составлены из высококвалифицированных специалистов. Уменьшенное время цикла разработки (до 3 месяцев) и улучшенная производительность. Повторное использование кода и автоматизация процесса разработки |
| Многопроходная модель | Быстро создается работающая система. Уменьшается возможность внесения изменений в процессе разработки. Невозможен переход от текущей реализации к новой версии в течение построения текущей частичной реализации |
| Спиральная модель | Охватывает каскадную модель. Расчленяет фазы на меньшие части. Позволяет гибко выполнять проектирование. Анализирует риски и управляет ими. Пользователи знакомятся с программным продуктом на более раннем этапе благодаря прототипам |

3.2 Каскадная модель

В однородных информационных системах 1970-х и 1980-х годов прикладные программные продукты представляли собой единое целое. Для разработки такого типа программного продукта применялось каскадная модель, или «водопад».

Каскадная модель программного продукта подобна модели автоматизированной системы управления (см. главу 1, рис.1).

Этот процесс носит, как правило, итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних стадиях. Таким образом, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс разработки принимает иной вид (см. глава 1, рис.2)

3.3 V-образная модель

Эта модель (рис.5) была разработана как разновидность каскадной модели, в которой особое внимание уделяется верификации и аттестации программного продукта. Модель показывает, что тестирование продукта обсуждается, проектируется и планируется, начиная с ранних этапов жизненного цикла разработки.

От каскадной модели v-образная модель унаследовала последовательную структуру, в соответствии с которой каждая последующая фаза начинается только после успешного завершения предыдущей фазы.

Данная модель основана на систематическом подходе к проблеме, для решения которой определены четыре базовых шага: анализ, проектирование, разработка и обзор. При выполнении анализа осуществляются планирование проекта и составление требований. Проектирование разделяется на высокоуровневое и детальное (низкоуровневое). Разработка включает в себя кодирование, обзор – различные виды тестирования.

На модели хорошо просматриваются взаимосвязи между аналитическими фазами и фазами проектирования, которые предшествуют кодированию и тестированию. Штриховые стрелки показывают, что эти фазы надо рассматривать параллельно.

Модель включает в себя следующие фазы:

Составление требований к проекту и планирование – определяются системные требования и выполняется планирование работ;

Составление требований к продукту и их анализ – составляется полная спецификация требований к программному продукту;

Высокоуровневое проектирование – определяется структура программного обеспечения, взаимосвязи между основными его компонентами и реализуемые ими функции;

Детальное проектирование – определяется алгоритм работы каждого компонента;

Кодирование – выполняется преобразование алгоритмов в готовое программное обеспечение;

Модульное тестирование – выполняется проверка каждого компонента или модуля программного продукта;

Интеграционное тестирование – осуществляются интеграция программного продукта и его тестирование;

Системное тестирование – выполняется проверка функционирования программного продукта после помещения его в аппаратную среду в соответствии со спецификацией требований;

Эксплуатация и сопровождение – запуск программного продукта в производство. На этой фазе в программный продукт могут вноситься поправки и может выполняться его модернизация.

Составление требований к проекту и планирование

Составление требований к продукту и их анализ

Высокоуровневое проектирование

Интеграционное тестирование

Системное тестирование

Эксплуатация и сопровождение

Детальное проектирование

Модульное тестирование

Кодирование

Рис.5 V-образная модель

Преимущества v-образной модели:

1) Большая роль придается верификации и аттестации программного продукта, начиная с ранних стадий его разработки, все действия планируются;

2) Предполагаются аттестация и верификация не только самого программного продукта, но и всех полученных внутренних и внешних данных;

3) Ход выполнения работы может легко отслеживаться, так как завершение каждой фазы является контрольной точкой.

Кроме перечисленных достоинств модель обладает и рядом недостатков:

не учитываются итерации между фазами; нельзя вносить изменения на разных этапах жизненного цикла; тестирование требований происходит слишком поздно, поэтому внесение изменений влияет на выполнение графика работ.

Данную модель целесообразно использовать при разработке программных продуктов, главным требованием для которых является высокая надежность.

3.4 Модель прототипирования



Рис.6 Модель прототипирования

Модель прототипитования позволяет создать прототип программного продукта до или в течение этапа составления требований к программному продукту. Потенциальные пользователи работают с этим прототипом, определяя его сильные и слабые стороны, о результатах сообщают разработчикам программного продукта. Таким образом, обеспечивается обратная связь между пользователями и разработчиками, которая используется для изменения или корректировки спецификации требований к программному продукту. В результате такой работы продукт будет отражать реальные потребности пользователей.

Жизненный цикл разработки программного продукта начинается с разработки плана проекта, затем выполняется быстрый анализ, после чего создаются база данных, пользовательский интерфейс и выполняется разработка необходимых функций. В результате этой работы получается документ, содержащий частичную спецификацию требований к программному продукту. Данный документ в дальнейшем является основой для итерационного цикла быстрого прототипирования.

В результате прототипирования разработчик демонстрирует пользователям готовый прототип, а пользователи оценивают его функционирование. После этого определяются проблемы, над устранением которых совместно работают пользователи и разработчики. Этот процесс продолжается до тех пор, пока пользователи не будут удовлетворены степенью соответствия программного продукта, поставленным перед ним требованиям. Затем прототип демонстрируют пользователям с целью получения предложений по его усовершенствованию, которые включаются в последовательные итерации до тех пор, пока рабочая модель не окажется удовлетворительной. После этого получают от пользователей официальное одобрение (утверждение) функциональных возможностей прототипа и выполняют его окончательное преобразование в готовый программный продукт.

Модель протипирования обладает целым рядом преимуществ:

1. Взаимодействие заказчика с разрабатываемой системой начинается на раннем этапе;
2. Благодаря реакции заказчика на прототип сводится к минимуму число неточностей в требованиях;
3. Снижается вероятность возникновения путаницы, искажения информации или недоразумений при определении требований к программному прдукту, что приводит к созданию более качественного программного продукта;
4. В процессе разработки всегда можно учесть новые, даже неожиданные требования заказчика;
5. Прототип представляет собой формальную спецификацию, воплощенную в программный продукт;
6. Прототип позволяет очень гибко выполнять проектирование и разработку, включая несколько итераций на всех фазах жизненного цикла разработки;
7. Заказчик всегда видит прогресс в процессе разработки программного продукта;
8. Возможность возникновения противоречий между разработчиками и заказчиками сведена к минимуму;
9. Уменьшается число доработок, что снижает стоимость разработки: возникающие проблемы решаются на ранних стадиях, что резко сокращает расходы на их устранение; заказчики принимают участие в процессе разработки на протяжении всего жизненного цикла и в конечном итоге в большей степени довольны результатом работы.

Кроме указанных достоинств модели прототипирования присущ и целый ряд недостатков:

1. Решение сложных задач может отодвигаться на будущее;
2. Заказчик может предпочесть получить прототип, а не законченную полную версию программного продукта;
3. Прототипирование может неоправданно затянуться;
4. Перед началом работы неизвестно, сколько итераций придется выполнить.

Модель прототипирования рекомендуется применять в следующих случаях:

1. Требования к программному продукту заранее неизвестны;
2. Требования не постоянны или неудачно сформулированы;
3. Требования необходимо уточнить;
4. Нужна проверка концепции;
5. Существует потребность в пользовательском интерфейсе;
6. Выполняется новая, не имеющая аналогов разработка;
7. Разработчики не уверены в том, какое решение следует выбрать.

3.5 Модель быстрой разработки приложений (RAD-модель)

В RAD-модели (рис.7) конечный пользователь играет решающую роль. В тесном взаимодействии с разработчиками он участвует в формировании требований и апробации их на работающих прототипах. Таким образом, в начале жизненного цикла на конечного пользователя выпадает большая часть работы, но в результате этого создаваемая система формируется более быстро.

В традиционном жизненном цикле разработки большую часть работы составляют программирование и тестирование. При автоматизации программирования и повторном использовании кода, применяемых в RAD-модели, большую часть работы составляют планирование и проектирование.

На рисунке (рис.7), поясняющем принцип RAD-модели, указаны этапы процесса разработки и отображено участие заказчиков (штриховая линия) на каждом из них.

Модель включает в себя следующие фазы:

Составление требований и планирование - осуществляются с использованием, так называемого метода совместного планирования требований (планирование работ по созданию программного продукта и составление требований к программному продукту выполняются одновременно), который заключается в структурном анализе и обсуждении решаемых задач;

Описание пользователя – проектирование программного продукта, выполняемое при непосредственном участии заказчика;

Создание – детальное проектирование, кодирование и тестирование программного продукта, а также поставка его заказчику;

Сопровождение – приемочные испытания, установка программного продукта и обучение пользователей.

Модель обладает следующими достоинствами:

1. Использование современных инструментальных средств позволяет сократить время цикла разработки;
2. Привлечение к работе заказчика сводит к минимуму риск того, что он останется недоволен готовым программным продуктом;
3. Повторно используются компоненты уже существующих программ.

В то же время ей присущи и недостатки:

1. Если заказчики не могут постоянно участвовать в процессе разработки, то это может негативно сказаться на программном продукте;
2. Для работы нужны высококвалифицированные кадры, умеющие пользоваться современными инструментальными средствами;
3. Существует риск, что работа над программным продуктом никогда не будет завершена, так как может быть зациклена, поэтому всегда надо вовремя остановиться.

Рассмотренную RAD-модель можно применять при разработке программных продуктов, которые хорошо поддаются моделированию, когда требования к программным продуктам хорошо известны, а заказчик может принять непосредственное участие в процессе разработки.

Приложенные усилия

Время

Заказчики

Разработчики

Сопровождение

Создание

Описание пользователя

Составление требований и планирование

Рис.7 Модель быстрой разработки приложений

3.6 Многопроходная модель

Многопроходная модель (рис.8) – это несколько итераций процесса построения прототипа программного продукта с добавлением на каждой следующей итерации новых функциональных возможностей или повышением эффективности программного продукта.

Предполагается, что на ранних этапах жизненного цикла разработки (планирование, анализ требований и разработка проекта) выполняется конструирование программного продукта в целом. Тогда же определяется и число необходимых инкрементов и относящихся к ним функций. Каждый инкремент затем проходит через оставшиеся фазы жизненного цикла (кодирование и тестирование). Сначала выполняются конструирование, тестирование и реализация базовых функций, составляющих основу программного продукта. Последующие итерации направлены на улучшение функциональных возможностей программного продукта.

Преимущества многопроходной модели: в начале разработки требуются средства только для разработки и реализации основных функций программного продукта; после каждого инкремента получается функциональный продукт; снижается риск неудачи и изменения требований; улучшается понимание как разработчиками, так и пользователями программного продукта требований для более поздних итераций; инкременты функциональных возможностей легко поддаются тестированию.

Недостатки многопроходной модели: не предусмотрены итерации внутри каждого инкремента; определение полной функциональности должно быть осуществлено в самом начале жизненного цикла разработки; может возникнуть тенденция оттягивания решения трудных задач; общие затраты на создание программного продукта не будут снижены по сравнению с другими моделями; обязательным условием является наличие хорошего планирования и проектирования.

Многопроходная модель может быть применена, если большинство требований к программному продукту будут сформулированы заранее, а для выполнения проекта будет выделен большой период времени.

Приемочное тестирование у продавца/поставщика ПП

Анализ требований

кодирование

Высокоуровневое проектирование

Детальное проектирование

Модульное тестирование

Системное тестирование

Приемочное тестирование у заказчика

Создание

(проход 1)

Создание

(проход 2)

Создание

(проход 3)

Интеграция прототипа

(проход 3)

Интеграция прототипа

(проход 2)

Интеграция прототипа

(проход 1)

Детальное проектирование (проход 3)

Детальное проектирование

(проход 2)

Детальное проектирование

(проход 1)

Интеграционное тестирование

Рис.8 Многопроходная модель

3.7 Спиральная модель

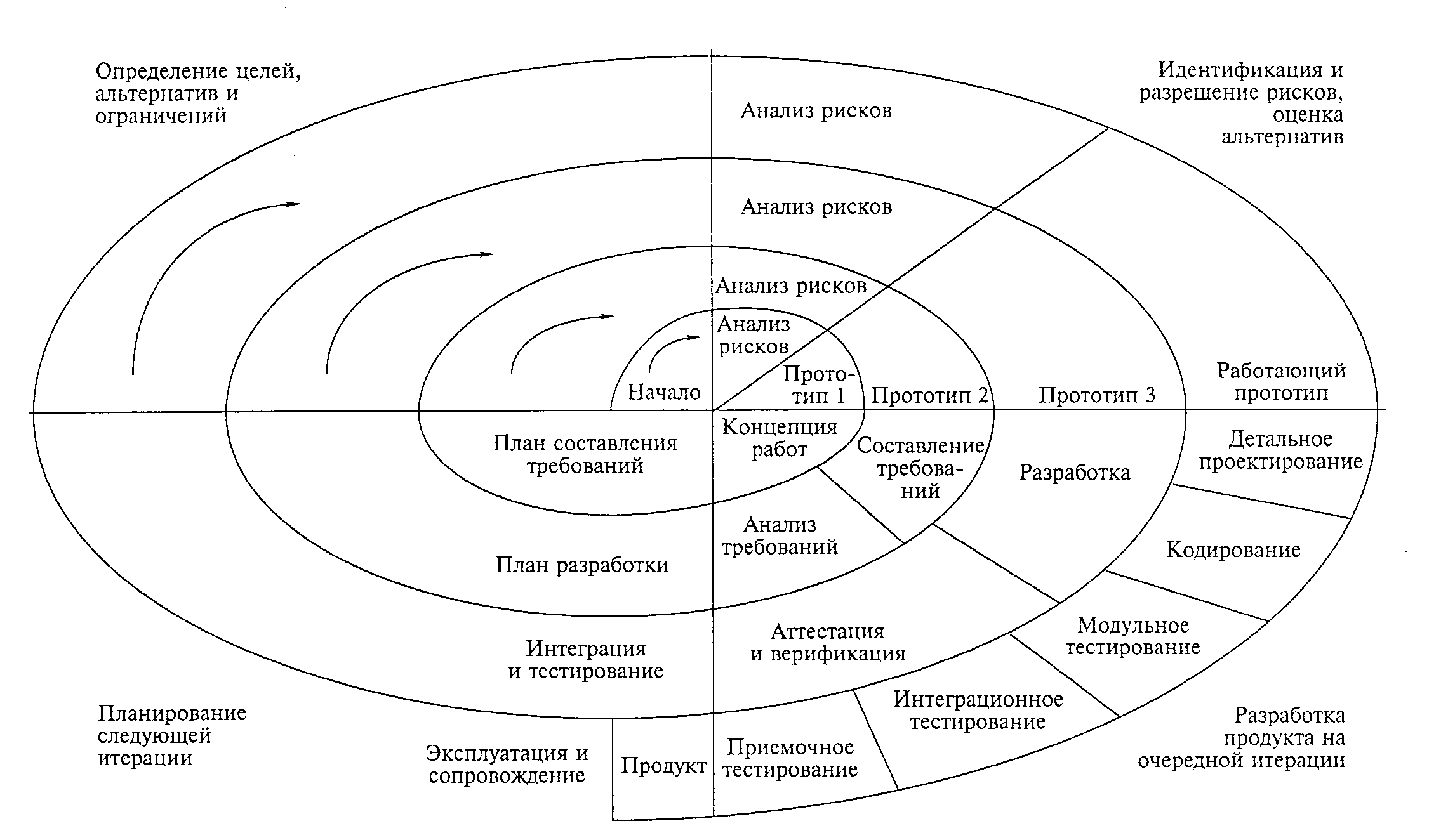


Рис.9 Спиральная модель

Для преодоления проблем, связанных с использованием многопроходной модели, в середине 1980-х годов была предложена спиральная модель жизненного цикла. Ее принципиальная особенность заключается в том, что прикладной программный продукт создается не сразу, как в случае каскадного подхода, а по частям с использованием метода прототипирования. Под прототипом понимается действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого программного продукта. Создание прототипов осуществляется за несколько итераций, или витков спирали. Каждая итерация соответствует созданию фрагмента, или версии программного продукта, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируется работа следующей итерации. На каждой итерации производится тщательная оценка риска превышения сроков и стоимости проекта с целью определения необходимости выполнения еще одной итерации, степени полноты и точности понимания требований к системе, а также целесообразности прекращения проекта. Спиральная модель (рис.9) избавляет пользователей и разработчиков программного продукта от полного и точного формулирования требований к системе на начальной стадии, поскольку они уточняются на каждой итерации. Таким образом, углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

Разработка итерациями отражает объективно существующий спиральный цикл создания системы, позволяя переходить на следующую стадию, не дожидаясь полного завершения работы на текущей стадии, поскольку при итеративном способе разработки недостающую работу можно выполнить на следующей итерации. Главная задача такой разработки – как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Спиральная модель не исключает использования каскадного подхода на завершающих стадиях проекта в тех случаях, когда требования к системе оказываются полностью определенными.

Основная проблема спирального цикла – определение момента перехода на следующую стадию. Для ее решения необходимо ввести временные ограничения на каждую из стадий жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Спиральная модель обладает следующими достоинствами: заказчик имеет возможность увидеть разрабатываемый программный продукт на ранних стадиях разработки; заказчики принимают активное участие в разработке программного продукта; в модели воплощены преимущества каскадной и многопроходной модели.

Недостатки спиральной модели: спираль может продолжаться до бесконечности, так как каждая ответная реакция заказчика может породить новый цикл.

В качестве модели жизненного цикла разработки программного продукта большое распространение получила улучшенная спиральная модель, показанная на рис. 10.



Рис.10 Улучшенная спиральная модель с указанием вспомогательных процессов

В отличии от ранее рассмотренной спиральной модели эта модель использует каскадный подход на завершающих этапах разработки программного продукта.

Использование спиральной модели целесообразно, если существует хотя бы одна из следующих причин: целесообразно создание прототипа; организация обладает навыками, требуемыми для адаптации модели; требуется выполнять проекты со средней и высокой степенями риска; заказчики не уверены в своих потребностях; требования слишком сложные; проект очень большой.

Заключение

В данной курсовой работе я рассмотрел понятия жизненного цикла автоматизированных информационных систем и программного продукта.

Работа состоит из трех глав.

В первой главе рассказывалось о моделях жизненного цикла автоматизированных информационных системах.

Жизненный цикл автоматизированных информационных систем - это непрерывный процесс, который начинается с момента принятия решения о необходимости создания ИС и заканчивается в момент ее полного изъятия из эксплуатации.

Модель жизненного цикла - структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ.

Наибольшее распространение получили две основные модели ЖЦ:

* каскадная модель (70-85 гг.);
* спиральная модель (86-90 гг.).

Структура жизненного цикла базируется на трех группах процессов:

* + основные процессы жизненного цикла (приобретение, поставка, разработка, эксплуатация, сопровождение);
  + вспомогательные процессы (документирование, управление конфигурацией, обеспечение качества, аттестация, аудит, решение проблем);
  + организационные процессы (управление проектами, создание инфраструктуры проекта, улучшение самого жизненного цикла, обучение).

Во второй главе речь шла о CASE-технологиях. CASE-технология - технология, базирующаяся на методологиях подготовки информационных систем и соответствующих комплексах интегрированных инструментальных средств, а также ориентированная на поддержку полного жизненного цикла автоматизированной системы или его основных этапов.

Под термином CASE (Computer Aided Software Engineering) понимаются программные средства, поддерживающие процессы создания и сопровождения автоматизированной системы включая анализ и формулировку требований, проектирование прикладного программного обеспечения и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным программным обеспечением и техническими средствами образуют полную среду разработки АС.

В третьей главе - модели жизненного цикла программного продукта.

Под моделью жизненного цикла разработки программного продукта понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении жизненного цикла разработки программного продукта. Наибольшее распространение получили следующие модели жизненного цикла разработки программного продукта: каскадная модель, или водопад (waterfall model); v-образная модель (v-shaped model); модель прототипирования (prototype model); модель быстрой разработки приложений, или RAD-модель (RAD-rapid application development model); многопроходная модель (incremental model); спиральная модель (spiral model).

Список использованной литературы

1. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. – 352с.
2. Канер С., Фолк Д., Кен Нгуен Е. Тестирование программного обеспечения: Пер. с англ. – Киев: ДиаСофт, 2000. – 544с.
3. Соммервилл И. Инженерия программного обеспечения. – М.: СПБ.: Киев: Изд. дом «Вильямс», 2002. – 624с.
4. Фридман А.Л. Основы объектно-ориентированной разработки программных систем. – М.: Финансы и статистика, 2000. – 200с.
5. Гагарина Л.Г. Основы технологии и разработки программных продуктов: Учебник – М ФОРУМ – ИНФРА – М. 2006.
6. Семенов М.И. Трубилин И.Т., Лойко В.И. Барановская Т.П. Архитектура компьютерных систем и сетей Учебное пособие – М Финансы и статистика, 2004. – 320с.
7. Советов Б.Я. Цеханковский В.В. Информационные технологии – М Высшая школа, 2003.
8. Информатика: Учебник под редакцией Макаровой Н.В. – М.:Финансы и статистика., 2005. – 480с.