Лекция 2. Стандарты и модели жизненного цикла программного обеспечения

Понятие жизненного цикла программного обеспечения

Жизненный цикл (ЖЦ) программного обеспечения (ПО) – это непрерывный процесс, который определяется как период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.

Существует ряд стандартов, регламентирующих ЖЦ ПО, а в некоторых случаях и процессы разработки.

Международный стандарт ISO/IEC 12207: 1995 “Information Technology - Software Life Cycle Processes” (ISO - International Organization for Standardization - Международная организация по стандартизации, IEC - International Electrotechnical Commission - Международная комиссия по электротехнике) - основной нормативный документ, регламентирующий состав процессов ЖЦ ПО.

ISO/IEC 12207:1995 - стандарт на процессы и организацию жизненного цикла. Распространяется на все виды заказного ПО. Стандарт не содержит описания фаз, стадий и этапов. Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

Среди других известных стандартов можно выделить следующие:

ГОСТ 34601 - 90. «Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания» - распространяется на автоматизированные системы и устанавливает стадии и этапы их создания. Также в стандарте содержится описание содержания работ на каждом этапе. Стадии и этапы работы, закрепленные в стандарте, в большей степени соответствуют каскадной модели жизненного цикла.

Custom Development Method (CDM) по разработке прикладных ИС - технологический материал, детализированный до уровня заготовок проектных документов, рассчитанных на использование в проектах с применением Oracle. Применяется CDM для классической модели ЖЦ (предусмотрены все работы/задачи и этапы), а также для технологий "быстрой разработки" или "облегченного подхода", рекомендуемых в случае малых проектов.

Rational Unified Process (RUP) предлагает итеративную модель разработки, включающую четыре фазы: начало, исследование, построение и внедрение. Каждая фаза может быть разбита на этапы, в результате которых выпускается версия для внутреннего или внешнего использования. Прохождение через четыре основные фазы называется циклом разработки, каждый цикл завершается генерацией версии системы. Суть работы - это создание и сопровождение моделей на базе UML.

Microsoft Solution Framework (MSF) сходна с RUP, также включает четыре фазы: анализ, проектирование, разработка, стабилизация. MSF является итерационной, предполагает использование объектно-ориентированного моделирования, ориентирована на разработку бизнес-приложений.

Extreme Programming (XP) - экстремальное программирование. В основе методологии командная работа, коммуникация между заказчиком и исполнителем в течение всего проекта по разработке ИС, а разработка ведется с использованием последовательно дорабатываемых прототипов.

Процессы жизненного цикла ПО ISO/IEC 12207

Международный стандарт ISO/IEC 12207: *ПО (или программный продукт)* определяется как набор компьютерных программ, процедур и, возможно, связанной с ними документации и данных.

Процесс ЖЦ определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные. Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами.

Каждый процесс разделен на набор *действий*, каждое действие – на набор *задач*. Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем не существует заранее определенных последовательностей.

В соответствии с ISO/IEC 12207: 1995 все процессы ЖЦ ПО разделены на три группы (рис. 1):

1. Основные процессы: приобретение; поставка; разработка; эксплуатация; сопровождение.

2. Вспомогательные процессы: документирование; управление конфигурацией; обеспечение качества; верификация; аттестация; совместная оценка; аудит; разрешение проблем.

3. Организационные процессы: управление; усовершенствование; создание инфраструктуры; обучение.

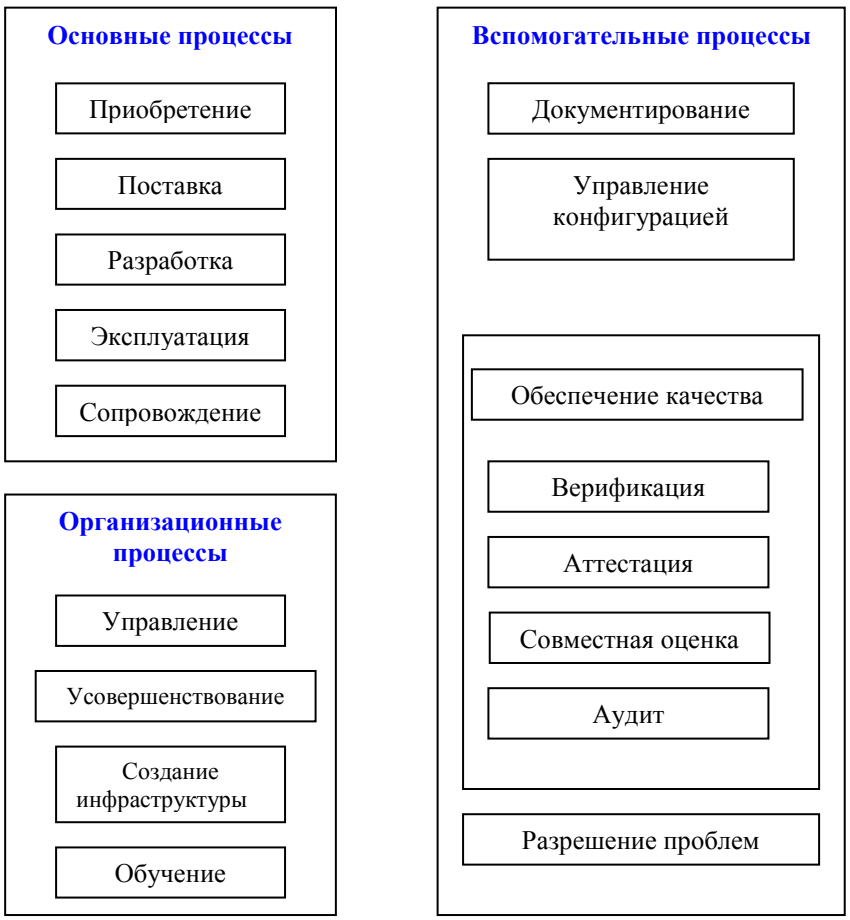

Рис.1. Структура процессов ЖЦ ПО

Основные процессы

Процесс приобретения - действия заказчика: инициирование приобретения; подготовка заявочных предложений, содержащих требования к системе; подготовка и корректировка договора; надзор за деятельностью поставщика; приемка.

Процесс поставки - действия поставщика: инициирование поставки (рассмотрение поставщиком заявочных предложений); планирование.

Процесс разработки предусматривает действия, выполняемые разработчиком:

- *Подготовка:* выбор модели ЖЦ, стандартов, методов, средств разработки, составление плана работ.

- *Анализ требований к системе:* определение ее функциональных возможностей, требований: пользовательских, к надежности и безопасности, к внешним интерфейсам и т.д.

- *Проектирование архитектуры ПО:* определение компонентов оборудования, ПО и операций, выполняемых персоналом.

- *Анализ требований к ПО:* определение функциональных возможностей (характеристики производительности; среды функционирования; внешних интерфейсов; спецификаций надежности и безопасности; эргономических требований; требований к используемым данным, к установке и приемке, к пользовательской документации, к эксплуатации и сопровождению).

- *Проектирование архитектуры ПО:* определение структуры ПО, документирование интерфейсов компонентов, разработка предварительной версии пользовательской документации, требований к тестам и плана интеграции.

- *Детальное проектирование ПО:* подробное описание компонентов ПО и интерфейсов между ними, обновление пользовательской документации, разработка и документирование требований к тестам и плана тестирования компонентов ПО; обновление плана интеграции ПО.

- *Кодирование и тестирование ПО:* разработка и документирование каждого компонента ПО и БД, совокупности тестовых процедур и данных для их тестирования; тестирование компонентов, обновление пользовательской документации и плана интеграции ПО.

- *Интеграция ПО:* сборка компонентов ПО в соответствии с планом интеграции и тестирование ПО на соответствие квалификационным требованиям.

- *Квалификационное тестирование:* тестирование ПО в присутствии заказчика для демонстрации его соответствия и готовности к эксплуатации, проверка полноты документации.

- *Интеграция системы:* сборка всех ее компонентов, включая ПО и оборудование. После интеграции система подвергается квалификационному тестированию на соответствие совокупности требований к ней. Производится оформление и проверка полного комплекта документации на систему.

- *Установка ПО* на оборудовании заказчика и проверка его работоспособности.

- *Приемка ПО:* оценка результатов квалификационного тестирования ПО и системы в целом и документирование результатов оценки совместно с заказчиком.

Процесс эксплуатации - действия и задачи оператора – организации, эксплуатирующей систему: подготовительная работа (планирование действий и работ); эксплуатационное тестирование;

эксплуатация системы (в предназначенной среде в соответствии с пользовательской документацией); поддержка пользователей (оказание помощи и консультаций при обнаружении ошибок).

Процесс сопровождения - выполняется службой сопровождения. В соответствии со стандартом IEEE-90: ***сопровождение*** - внесение изменений в ПО в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям. Вносимые изменения не должны нарушать целостность ПО. Процесс сопровождения включает перенос ПО в другую среду (миграцию) и заканчивается снятием ПО с эксплуатации.

Действия: подготовительная работа; анализ запросов на модификацию ПО; модификация ПО (внесение необходимых изменений); проверка и приемка; перенос ПО в другую среду; снятие ПО с эксплуатации (по решению заказчика при участии эксплуатирующей организации).

Вспомогательные процессы ЖЦ ПО

Процесс документирования - формализованное описание информации, созданной в течение ЖЦ ПО. Действия: подготовка; проектирование и разработка; выпуск документации; сопровождение.

Процесс управления конфигурацией - позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях ЖЦ ПО.

Согласно стандарту IEEE – 90: ***конфигурация*** ПО - совокупность ее функциональных и физических характеристик, установленных в технической документации и реализованных в ПО.

Действия: подготовительная работа; идентификация конфигурации (каждому компоненту и его версиям соответствует однозначно обозначаемый комплект документации); контроль конфигурации; учет состояния и оценка конфигурации; управление выпуском и поставку.

Процесс обеспечения качества - обеспечивает гарантии того, что ПО и процессы его ЖЦ соответствуют заданным требованиям и утвержденным планам. ***Качество ПО*** - совокупность свойств, которые характеризуют способность ПО удовлетворять заданным требованиям.

Процесс обеспечения качества должен происходить независимо от субъектов, непосредственно связанных с разработкой ПО. Действия: подготовительная работа; обеспечение качества продукта (гарантирование соответствия программных продуктов и документации требованиям заказчика); обеспечение качества процесса (гарантирование соответствия процессов ЖЦ ПО, методов разработки, среды разработки, квалификации персонала условиям договора и стандартам). Обеспечение прочих показателей качества - в соответствии с условиями договора и стандартом ISO 9001.

Процесс верифиации - определение того, что программные продукты, являющиеся результатами некоторого действия, удовлетворяют требованиям или условиям, обусловленным предшествующими действиями. ***Верификаци***я в узком смысле означает формальное доказательство правильности ПО.

Верификация может проводиться с различными степенями независимости: от выполнения верификации самим исполнителем или специалистом организации до ее выполнения специалистом другой организации. Если процесс верификации осуществляется организацией, не зависящей от поставщика, разработчика или службы сопровождения, то - ***процесс независимой верификации***.

Процесс верификации включает действия: подготовительную работу; верификацию;

В процесс верификации проверяются условия:

- непротиворечивость требований к системе и степень учета потребностей пользователей;
- возможности поставщика выполнять заданные требования;
- соответствие выбранных процессов ЖЦ ПО условиям договора;
- адекватность стандартов, процедур и среды разработки процесса ЖЦ ПО;
- соответствие проектных спецификаций ПО заданным требованиям;
- корректность описания в спецификациях входных/выходных данных, интерфейсов, логики;
- соответствие кода проектным спецификациям и требованиям;
- тестируемость и корректность кода, его соответствие принятым стандартам кодирования;
- корректность интеграции компонентов ПО в систему;
- адекватность, полнота и непротиворечивость документации.

Процесс аттестации - определение полноты соответствия заданных требований и созданного программного продукта их конечному функциональному назначению. ***Аттестация*** - подтверждение и оценка достоверности проеденного тестирования. Аттестация должна гарантировать соответствие ПО спецификациям, требованиям и документации. Аттестацию выполняют тестированием во всех возможных ситуациях. Аттестация может осуществляться с различными степенями независимости. Если процесс аттестации выполняется организацией, не зависящей от поставщика, разработчика и др., то он называется ***процессом независимой аттестации***.

Процесс совместной оценки - для оценки состояния работ по проекту и ПО: контроль планирования, управления ресурсами, персоналом, аппаратурой и инструментальными средствами. Выполняется двумя любыми сторонами, участвующими в договоре, при этом одна сторона проверяет другую. Действия: подготовительная работа; оценка управления проектом; техническая оценка.

Процесс аудита - определение соответствия требованиям, планам и условиям договора. Аудит выполняется двумя любыми сторонами, участвующими в договоре, одна сторона проверяет другую. ***Аудит*** – это ревизия (проверка), проводимая компетентным органом (лицом) для обеспечения независимой оценки степени соответствия ПО установленным требованиям. Аудит служит для установления соответствия реальных работ и отчетов требованиям, планам и контракту. Аудиторы не должны прямо зависеть от разработчиков ПО.

Процесс разрешения проблем - анализ и решение обнаруженных проблем независимо от их происхождения. Проблема должна быть идентифицирована, описана, проанализирована и разрешена.

Организационные процессы ЖЦ ПО

Процесс управления - менеджер отвечает за управление выпуском продукта, управление проектом и задачами процессов. Действия: определение области управления, планирование (составление графиков выполнения работ; оценка затрат; выделение ресурсов; распределение ответственности; оценка рисков, создание инфраструктуры управления).

Процесс создания инфраструктуры - выбор и поддержка стандартов и инструментальных

средств, выбор и установка аппаратных и программных средств.

Процесс усовершенствования - оценка, контроль и усовершенствование процессов ЖЦ ПО.

Процесс обучения - первоначальное обучение и последующее постоянное повышение квалификации персонала.

Действия ***процесса разработки*** можно сгруппировать, условно выделив следующие основные этапы разработки ПО (в скобках указаны соответствующие *стадии разработки* по ГОСТ 19.102-77 «Стадии разработки»):

- постановка задачи (стадия «Техническое задание»);
- анализ требований и разработка спецификаций (стадия «Эскизный проект»);
- проектирование (стадия «Технический проект»);
- реализация (стадия «Рабочий проект»).

Традиционно разработка также включала этап *сопровождения* (началу этого этапа соответствует стадия «Внедрение» по ГОСТ). Но по международному стандарту в соответствии с изменениями, произошедшими в индустрии разработки ПО, этот процесс теперь рассматривается отдельно.

Условность выделения этапов связана с тем, что на любом этапе возможно принятие решений, которые потребуют пересмотра решений, принятых ранее.

Постановка задачи - четко формулируют назначение ПО и определяют основные требования к нему. Каждое требование представляет собой описание необходимого или желаемого свойства программного обеспечения. Различают *функциональные требования,* определяющие функции, которые должно выполнять разрабатываемое программное обеспечение, и *эксплуатационные требования,* определяющие особенности его функционирования.

Требования к ПО, имеющему *прототипы,* определяют по аналогии, учитывая структуру и характеристики существующего ПО. Для формулирования требований к ПО, не имеющему аналогов, необходимо провести специальные *предпроектные* исследования, где определяют разрешимость задачи, разрабатывают методы ее решения (если они новые) и устанавливают наиболее существенные характеристики разрабатываемого ПО. Для выполнения предпроектных исследований обычно заключают договор. В любом случае этап постановки задачи заканчивается разработкой *технического задания,* фиксирующего принципиальные требования, и принятием основных проектных решений.

Анализ требований и определение спецификаций. *Спецификация* - точное формализованное описание функций и ограничений разрабатываемого ПО. Различают *функциональные* и *эксплуатационные* спецификации. Совокупность спецификаций представляет собой общую логическую модель проектируемого ПО.

Для получения спецификаций выполняют анализ требований ТЗ, формулируют содержательную постановку задачи, выбирают математический аппарат формализации, строят модель предметной области, определяют подзадачи и выбирают или разрабатывают методы их решения. Часть спецификаций может быть определена в процессе предпроектных исследований и зафиксирована в ТЗ.

Проектирование. Основная задача - определение *подробных* спецификаций разрабатываемого

ПО. Процесс проектирования сложного ПО включает:

• проектирование общей структуры – определение основных компонентов и их взаимосвязей;

• декомпозицию компонентов и построение структурных иерархий в соответствии с рекомендациями блочно-иерархического подхода;

• проектирование компонентов.

Результат проектирования - *детальная модель* разрабатываемого ПО вместе со спецификациями его компонентов всех уровней. Тип модели зависит от выбранного подхода (структурный, объектный или компонентный) и конкретной технологии проектирования. В любом случае процесс проектирования охватывает как проектирование программ (подпрограмм) и определение взаимосвязей между ними, так и проектирование данных, с которыми взаимодействуют эти программы или подпрограммы.

Различают также два аспекта проектирования:

• логическое проектирование - включает проектные операции, которые непосредственно не зависят от имеющихся технических и программных средств, составляющих среду функционирования будущего программного продукта;

• физическое проектирование – привязка к конкретным техническим и программным средствам среды функционирования, т.е. учет ограничений, определенных в спецификациях.

Реализация - процесс поэтапного написания кодов программы на выбранном языке программирования (кодирование), их тестирование и отладку.

Сопровождение – это процесс создания и внедрения новых версий программного продукта. Причинами выпуска новых версий могут служить:

- необходимость исправления выявленных ошибок;
- необходимость совершенствования предыдущих версий: улучшения интерфейса, расширения состава выполняемых функций и пр.;
- изменение среды функционирования: появление новых технических средств и/или программных продуктов, с которыми взаимодействует сопровождаемое ПО.

На этом этапе в программный продукт вносят необходимые изменения, которые также могут потребовать пересмотра проектных решений, принятых на любом предыдущем этапе. С изменением модели ЖЦ ПО (см. далее) роль этого этапа возросла, так как продукты теперь создаются итерационно: сначала выпускается сравнительно простая версия, затем следующая с большими возможностями, затем следующая и т.д. Это и послужило причиной выделения этапа сопровождения в отдельный процесс ЖЦ в соответствии с стандартом ISO/IEC 12207.

Модели и стадии жизненного цикла программного обеспечения

Модель ЖЦ ПО – это структура, определяющая последовательность выполнения и взаимосвязи процессов, действий, задач на протяжении ЖЦ. Модель ЖЦ зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Международный стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО.

Его положения являются общими для любых моделей ЖЦ, методов и технологий разработки ПО. Стандарт только называет и определяет процессы ЖЦ ПО (описывает структуру), но не конкретизирует в деталях, как реализовать и выполнить действия и задачи, включенные в эти процессы. Эти вопросы регламентируются соответствующими методами, методиками и т.п.

Модель ЖЦ конкретного ПО определяет характер *процесса* его создания - совокупность упорядоченных во времени, взаимосвязанных и объединенных в стадии работ, выполнение которых необходимо и достаточно для создания ПО, соответствующего заданным требованиям.

Стадия создания ПО - часть процесса создания ПО, ограниченная временными рамками и заканчивающаяся выпуском какого-то конкретного продукта (моделей ПО, программных компонентов, документации).

На протяжении последних лет сменились три модели жизненного цикла ПО: каскадная, модель с промежуточным контролем и спиральная.

Каскадная модель (1970-1985). Первоначально была предложена и использовалась каскадная схема разработки ПО (рис.2), которая предполагала, что *переход на следующую стадию осуществляется только после того, как будет полностью завершена работа на текущей стадии, и возвратов на пройденные стадии не предусматривается*. Каждая стадия заканчивается получением результатов, которые служат в качестве исходных данных для следующей стадии.

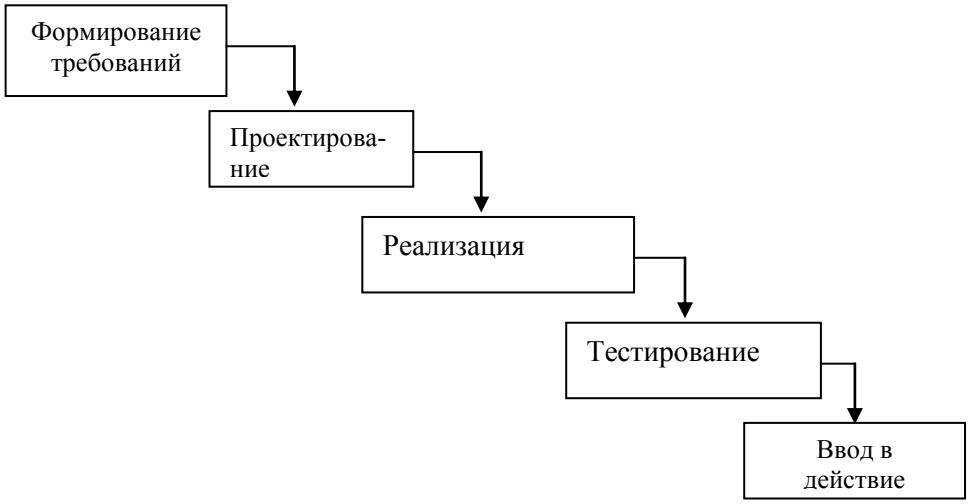

Рис.2. Каскадная модель

Каждая стадия завершается выпуском комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков. Критерием качества разработки при таком подходе является точность выполнения спецификаций технического задания.

Преимущества применения каскадной схемы:

- в конце каждой стадии формируется законченный набор проектной документации;
- простота планирования процесса разработки.

Каскадный подход хорошо зарекомендовал себя при построении программных систем, для которых в самом начале разработки можно точно и полно сформулировать все требования. На практике такие разработки встречается редко. Необходимость возвратов на предыдущие стадии обусловлена

причинами:

- неточные спецификации, уточнение которых в процессе разработки может привести к необходимости пересмотра уже принятых решений;
- изменение требований заказчика непосредственно в процессе разработки;
- отсутствие удовлетворительных средств описания разработки на стадиях постановки задачи, анализа и проектирования.

Недостатки каскадной схемы вызваны тем, что реальный процесс создания ПО никогда полностью не укладывается в такую жесткую схему. Результаты очередной стадии часто вызывают изменения в решениях, выработанных на предыдущих стадиях. Возникает потребность в возврате к предыдущим стадиям и уточнении или пересмотре ранее принятых решений. Реальный процесс разработки носит итерационный характер.

Модель с промежуточным контролем. Схема, поддерживающая итерационный характер процесса разработки, была названа схемой с промежуточным контролем (рис. 3). Контроль, который выполняется по данной схеме после завершения каждого этапа, позволяет при необходимости вернуться на любой уровень и внести необходимые изменения. Межстадийные корректировки обеспечивают большую надежность, хотя увеличивают весь период разработки.

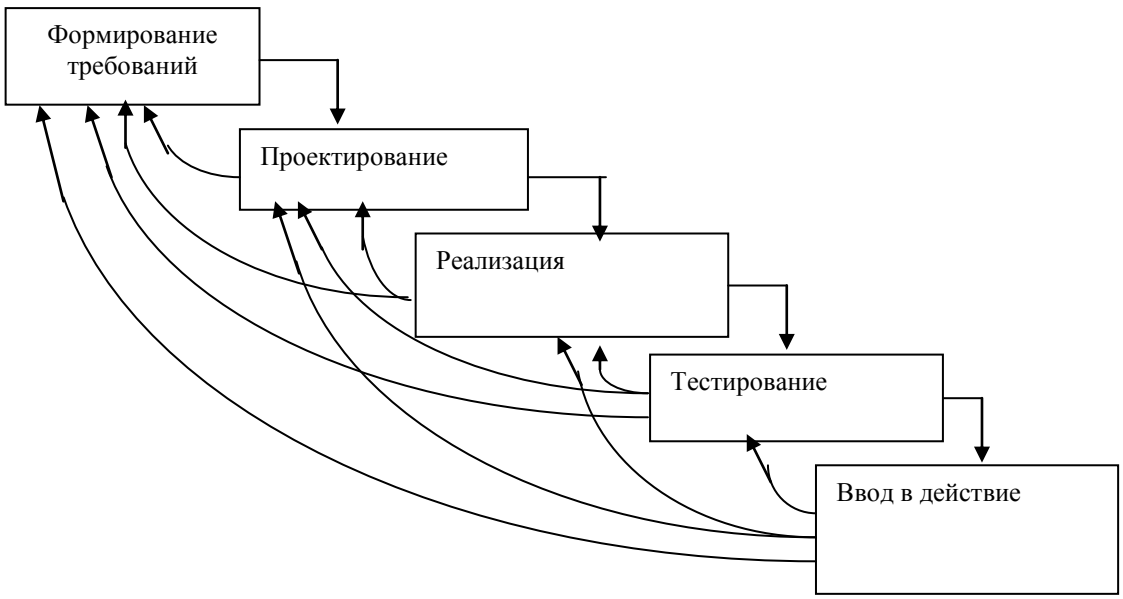

Рис. 3. Итерационный процесс создания программного обеспечения

Основная опасность использования такой схемы связана с тем, что разработка никогда не будет завершена, постоянно находясь в состоянии уточнения и усовершенствования.

Недостаток каскадной модели - существенное запаздывание с получением результатов и, как следствие, высокий риск создания системы, не удовлетворяющей и изменившимся потребностям пользователей. Это объяснятся причинами:

- пользователи не в состоянии сразу изложить все свои требования и не могут предвидеть, как они изменятся в ходе разработки;
- за время разработки могут произойти изменения во внешней среде, которые повлияют на

требования к системе.

Спиральная модель (1986-1990). Для преодоления перечисленных проблем в середине 80-х годов была предложена спиральная модель ЖЦ (рис.4). Ее особенность**:** *прикладное ПО создается не сразу, как в случае каскадного подхода, а по частям с использованием метода прототипирования***.**

Прототип - действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого ПО.

Создание прототипов осуществляется в несколько итераций, или витков спирали. Каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации. На каждой итерации производится оценка риска превышения сроков и стоимости проекта, чтобы определить необходимость выполнения еще одной итерации, степень полноты и точности понимания требований к системе, а также целесообразность прекращения проекта.

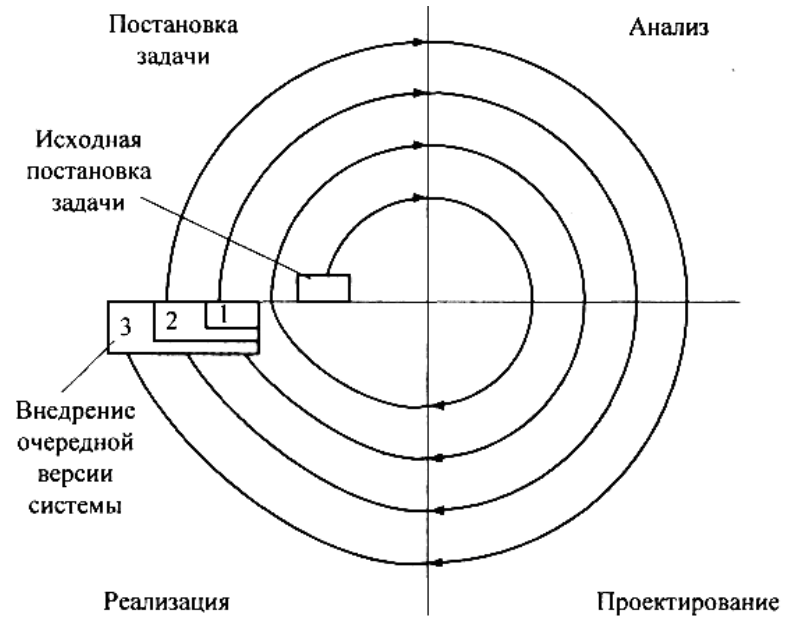

Рис. 4. Спиральная модель

Спиральная модель избавляет пользователей и разработчиков от необходимости точного и полного 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.

При итеративном способе недостающую часть работы можно выполнять на следующей итерации. Главная задача - как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

На первой итерации обычно специфицируют, проектируют, реализуют и тестируют интерфейс пользователя. На второй – добавляют некоторый ограниченный набор функций. На последующих этапах этот набор расширяют, наращивая возможности данного продукта.

Основное *достоинство спиральной модели* – с некоторой итерации, на которой обеспечена

определенная функциональная полнота, продукт можно предоставлять пользователю, что позволяет:

• сократить время до появления первых версий программного продукта;

• заинтересовать большое количество пользователей, обеспечивая быстрое продвижение следующих версий продукта на рынке;

• ускорить формирование и уточнение спецификаций за счет появления практики использования продукта;

• уменьшить вероятность морального устаревания системы за время разработки.

Основная проблема - определение моментов перехода на следующие стадии. Для ее решения обычно ограничивают сроки прохождения каждой стадии, основываясь на экспертных оценках.

Изменение ЖЦ ПО при использовании CASE-технологий.

CASE-технологии представляют собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных программных систем, основанных на структурном и на объектном подходах, которые поддерживаются комплексом взаимосвязанных средств автоматизации. В основе любой CASE-технологии лежит парадигма методология/метод/нотация/средство.

Методология строится на базе некоторого подхода и определяет шаги работы, их последовательность, а также правила распределения и назначения методов. Метод определяет способ достижения той или иной цели - выполнение шага работы.

Нотацией называют систему обозначений, используемых для описания некоторого класса моделей. Нотации бывают графические (предоставление моделей в виде графов, диаграмм, таблиц, схем и т.п.) и текстовые (описания моделей на формальных и естественных языках). В CASE-технологиях нотации используют для описания структуры проектируемой системы, элементов данных, этапов обработки и т.п.

Средства - инструментарий для поддержки методов: средства создания и редактирования графического проекта, организации проекта в виде иерархии уровней абстракции, а также проверки соответствия компонентов разных уровней. Различают:

• CASE-средства анализа требований, проектирования спецификаций и структуры, редактирования интерфейсов (первое поколение CASE-I);

• CASE-средства генерации исходных текстов и реализации интегрированного окружения поддержки полного жизненного цикла разработки программного обеспечения (2ое поколение CASE-II).

CASE-I в основном включают средства для поддержки графических моделей, проектирования спецификаций, экранных редакторов и словарей данных. CASE-II отличается большими возможностями, обеспечивая: контроль, анализ и связывание системной информации и информации по управлению процессом проектирования, построение прототипов и моделей системы, тестирование, верификацию и анализ сгенерированных программ.

Автоматизируя трудоемкие операции, CASE-средства существенно повышают производительность труда программистов и улучшают качество создаваемого ПО. Они:

• обеспечивают автоматизированный контроль совместимости спецификаций проекта;

• уменьшают время создания прототипа системы;

• ускоряют процесс проектирования и разработки;

• автоматизируют формирование проектной документации для всех этапов ЖЦ;

• частично генерируют коды программ для различных платформ разработки;

• поддерживают технологии повторного использования компонентов системы;

• обеспечивают возможность восстановления проектной документации по исходным кодам.

Появление CASE-технологий изменило все этапы ЖЦ ПО, наибольшие изменения касаются анализа и проектирования, которые предполагают строгое и наглядное описание разрабатываемого ПО.

В табл. 1 показано, какие качественные изменения процесса разработки ПО происходят при переходе к использованию CASE-средств.

Таблица 1.

Традиционная разработка	Разработка с использованием CASE-средств
Основные усилия на кодирование и тестирование	Основные усилия на анализ и проектирование
«Бумажные» спецификации	Быстрое итерационное прототипирование
Ручное кодирование	Автоматическая генерация кодов
Ручное документирование	Автоматическая генерация документации
Тестирование кодов	Автоматический контроль проекта
Сопровождение кодов	Сопровождение спецификаций проектирования

Использование CASE-средств позволяет существенно снизить трудозатраты на разработку сложного ПО в основном за счет автоматизации процессов документирования и контроля. Следует иметь в виду, что современные CASE-средства дороги, а их использование требует более высокой квалификации разработчиков. Их имеет смысл использовать в сложных проектах, причем, чем сложнее разрабатываемое ПО, тем больше выигрыш от использования CASE-технологий. Сегодня практически все промышленно производимое сложное ПО разрабатывается с использованием CASE-средств.

Ускорение разработки программного обеспечения. Технология RAD

Разработка спиральной модели ЖЦ ПО и CASE-технологий позволили сформулировать условия, выполнение которых сокращает сроки создания ПО.

Технологии проектирования, разработки и сопровождения ПО, должна отвечать требованиям:

• поддержка полного ЖЦ ПО;

• гарантированное достижение целей разработки с заданным качеством и в установленное время;

• возможность выполнения крупных проектов в виде подсистем, разрабатываемых группами исполнителей (3-7 человек) с последующей интеграцией составных частей;

• минимальное время получения работоспособной системы;

• возможность управления конфигурацией проекта, ведения версий проекта и автоматического выпуска проектной документации по каждой версии;

• независимость выполняемых проектных решений от средств реализации (СУБД, операционных систем, языков и систем программирования);

• поддержка комплексом согласованных CASE-средств, обеспечивающих автоматизацию процессов, выполняемых на всех стадиях жизненного цикла.

Этим требованиям отвечает ***технология RAD*** (Rapid Application Development – Быстрая разработка приложений). Эта технология ориентирована на максимально быстрое получение первых версий разрабатываемого ПО. Она предусматривает выполнение следующих условий:

• ведение разработки небольшими группами разработчиков (3-7 человек), каждая из которых проектирует и реализует отдельные подсистемы проекта – позволяет улучшить управляемость проекта;

• использование итерационного подхода способствует уменьшению времени получения работоспособного прототипа;

• наличие четко проработанного графика цикла, рассчитанного не более чем на три месяца, существенно увеличивает эффективность работы.

Процесс разработки при этом делится на следующие этапы: анализ и планирование требований пользователей, проектирование, реализация, внедрение.

На этапе *анализа и планирования требований* формулируют наиболее приоритетные требования, что ограничивает масштаб проекта.

На этапе *проектирования,* используя CASE-средства, детально описывают процессы системы, устанавливают требования разграничения доступа к данным и определяют состав необходимой документации. Для сложных процессов создают частичный прототип: разрабатывают экранную форму и диалог. По результатам анализа процессов определяют количество функциональных точек и принимают решение о количестве подсистем и, соответственно, команд, участвующих в разработке.
Функциональная точка в RAD – это любой из функциональных элементов разрабатываемой системы:

• входной элемент приложения (входной документ или экранная форма);

• выходной элемент приложения (отчет, документ или экранная форма);

• запрос (пара «вопрос/ответ»);

• логический файл (совокупность записей данных, используемых внутри приложения);

• интерфейс приложения (совокупность записей данных, передаваемых другому приложению или получаемых от него).

Рассчитываются нормы, исходя из экспертных оценок, для систем со значительной повторяемостью кодов. В соответствии с ними разрабатываемую систему делят на подсистемы, слабо связанные по данным и функциям, и точно определяют интерфейсы между различными частями. Использование CASE-средств позволяет избежать неконтролируемого искажения данных при передаче информации о проекте со стадии на стадию.

Далее разработка ведется группами разработчиков, которые продолжают прорабатывать свои

части системы. Действия различных групп разработчиков должны быть хорошо скоординированы.

На этапе *реализации* выполняют итеративное построение реальной системы, при этом для контроля над выполнением требований к создаваемой системе привлекаются будущие пользователи.

Части постепенно интегрируют в систему, причем при подключении каждой части выполняют тестирование. На завершающих этапах разработки определяют необходимость создания соответствующих баз данных, которые разрабатываются и подключаются к системе. Далее формулируют требования к аппаратным средствам, устанавливают способы увеличения производительности и завершают подготовку документации по проекту.

На этапе *внедрения* проводят обучение пользователей и осуществляют постепенный переход на новую систему, эксплуатация старой версии продолжается до полного внедрения новой системы.

Технология RAD хорошо зарекомендовала себя для относительно небольших проектов, разрабатываемых для конкретного заказчика. Такие системы не требуют высокого уровня планирования и жесткой дисциплины проектирования. Однако эта технология не применима для построения сложных расчетных программ, операционных систем или программ управления сложными объектами в реальном масштабе времени, т.е. программ с большим процентом уникального кода. Не годится она и в случае создания приложений, от которых зависит безопасность людей, например, систем управления самолетами или атомными электростанциями, так как технология RAD предполагает, что первые несколько версий не будут полностью работоспособны, что в данном случае полностью исключается.