**Жизненный цикл по ИС**

Одним из базовых понятий методологии проектирования ИС является понятие жизненного цикла ее программного обеспечения (ЖЦ ПО). ЖЦ ПО - это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим ЖЦ ПО, является международный стандарт ISO/IEC 12207 [5] (ISO - International Organization of Standardization - Международная организация по стандартизации, IEC - International Electrotechnical Commission - Международная комиссия по электротехнике). Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

Структура ЖЦ ПО по стандарту ISO/IEC 12207 базируется на трех группах процессов:

* основные процессы ЖЦ ПО (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 ЖЦ, обучение).

Разработка включает в себя все работы по созданию ПО и его компонент в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала и т.д. Разработка ПО включает в себя, как правило, анализ, проектирование и реализацию (программирование).

Эксплуатация включает в себя работы по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование базы данных и рабочих мест пользователей, обеспечение эксплуатационной документацией, проведение обучения персонала и т.д., и непосредственно эксплуатацию, в том числе локализацию проблем и устранение причин их возникновения, модификацию ПО в рамках установленного регламента, подготовку предложений по совершенствованию, развитию и модернизации системы.

Управление проектом связано с вопросами планирования и организации работ, создания коллективов разработчиков и контроля за сроками и качеством выполняемых работ. Техническое и организационное обеспечение проекта включает выбор методов и инструментальных средств для реализации проекта, определение методов описания промежуточных состояний разработки, разработку методов и средств испытаний ПО, обучение персонала и т.п. Обеспечение качества проекта связано с проблемами верификации, проверки и тестирования ПО. Верификация - это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа. Проверка позволяет оценить соответствие параметров разработки с исходными требованиями. Проверка частично совпадает с тестированием, которое связано с идентификацией различий между действительными и ожидаемыми результатами и оценкой соответствия характеристик ПО исходным требованиям. В процессе реализации проекта важное место занимают вопросы идентификации, описания и контроля конфигурации отдельных компонентов и всей системы в целом.

Управление конфигурацией является одним из вспомогательных процессов, поддерживающих основные процессы жизненного цикла ПО, прежде всего процессы разработки и сопровождения ПО. При создании проектов сложных ИС, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, возникает проблема учета их связей и функций, создания унифицированной структуры и обеспечения развития всей системы. Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях ЖЦ. Общие принципы и рекомендации конфигурационного учета, планирования и управления конфигурациями ПО отражены в проекте стандарта ISO 12207-2 [5].

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. ЖЦ ПО носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах.

**Об этапах проектирования**

Когда мы говорим о проектировании баз данных, надо иметь в виду, что оно является лишь частью проектирования всей информационной системы. Мы довольно подробно рассматриваем вопросы проектирования самой важной части любой информационной системы – информационного хранилища или базы данных. Действительно проектирование базы данных имеет несомненный приоритет по отношению к функциональной части системы. На это есть несколько причин, одна из которых заключается в том, что при проектировании функциональной части системы желательно уже определится с именами, используемыми для обозначения хранящихся в базе данных объектов. Кроме этого заметим, что после начала эксплуатации информационной системы не желательно, а часто и довольно сложно менять структуру базы данных, в отличие от ее программной части.

Многолетний опыт показывает, что можно выделить три уровня проектирования баз данных: **концептуальный, логический и физический**. Кратко остановимся на каждом из этих уровней.

* Концептуальный уровень. Строится общая модель базы данных. За основу берется описание предметной области. В качестве языка описания может быть и естественный язык, и какой либо формализованный язык, например язык UML (Unified Modeling Language – универсальный язык моделирования). Описание предметной области называют еще **внешним уровнем** проектирования. На концептуальном уровне проектирования должна быть представлена схема, включающая единое описание всех элементов данных (имена элементов) и отношений между ними. В дальнейшем элементы будут называться нами **сущностями**. Схема представляет собой некоторый общий взгляд на предмет проектирования, позволяет единым взглядом охватить информацию, которые будут храниться в базе данных. При создании и анализе концептуальной модели следует особо обратить внимание на возможность осуществления на основе модели основных операций (транзакций), которые должна выполнять информационная система.
* Логический уровень проектирования представляет собой детализацию элементов концептуальной схемы: описание характеристик элементов (атрибутов) и связей между элементами. Именно на этом уровне могут появиться новые сущности, и нам придется возвратиться к концептуальному уровню проектирования базы данных. Логический уровень проектирования уже весьма близок к схеме реляционной базы данных. Именно здесь мы должны определиться с первичными ключами, обозначить типы связи их имена и характеристики, определить ограничения, накладываемые на хранимые в базе данные.
* На физическом уровне проектирования логическая модель данных преобразуется к реляционной базе данных конкретной СУБД. Происходит преобразование сущностей логической модели в конкретные таблицы, атрибуты сущностей в атрибуты (столбцы) таблицы, связи между сущностями в связи между таблицами, первичные ключи сущностей в первичные ключи таблиц. Кроме этого создаются индексы таблиц, а также реализуются ограничения целостности базы данных средствами конкретной СУБД. Заметим также, что возможно на данном этапе может потребоваться нормализация некоторых таблиц.

После завершения проектирование базы данных на физическом уровне реализуются другие задачи создания информационной системы: проектируется система безопасности информационной системы и создается программная составляющая на стороне сервера (триггеры, хранимые процедуры, функции, представления). Наконец после создания программной части на стороне сервера можно приступать к программированию клиента.

**Модели жизненного цикла ПО**

Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО (под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ. Модель ЖЦ зависит от специфики ИС и специфики условий, в которых последняя создается и функционирует). Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки. Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

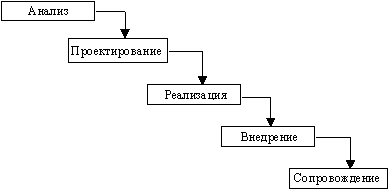
К настоящему времени наибольшее распространение получили следующие две основные модели ЖЦ:

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

В изначально существовавших однородных ИС каждое приложение представляло собой единое целое. Для разработки такого типа приложений применялся каскадный способ. Его 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. 1.1). Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

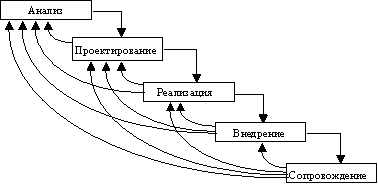
Положительные стороны применения каскадного подхода заключаются в следующем [2]:

* 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.



*Рис. 1.1. Каскадная схема разработки ПО*

Каскадный подход хорошо зарекомендовал себя при построении ИС, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования, с тем чтобы предоставить разработчикам свободу реализовать их как можно лучше с технической точки зрения. В эту категорию попадают сложные расчетные системы, системы реального времени и другие подобные задачи. Однако, в процессе использования этого подхода обнаружился ряд его недостатков, вызванных прежде всего тем, что реальный процесс создания ПО никогда полностью не укладывался в такую жесткую схему. В процессе создания ПО постоянно возникала потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ПО принимал следующий вид (рис. 1.2):



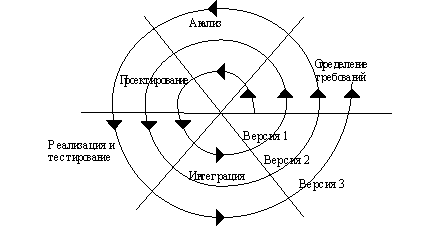
*Рис. 1.2. Реальный процесс разработки ПО по каскадной схеме*

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 ИС "заморожены" в виде технического задания на все время ее создания. Таким образом, пользователи могут внести 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 ПО, пользователи получают систему, не удовлетворяющую их потребностям. Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением.

Для преодоления перечисленных проблем была предложена спиральная модель ЖЦ [10] (рис. 1.3), делающая упор на начальные этапы ЖЦ: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, 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 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 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.



*Рис 1.3. Спиральная модель ЖЦ*