**Лекция №1. Модели жизненного цикла программных средств**

Комплексы программ создаются, эксплуатируются и развиваются во времени. Жизненный цикл ПС включает в себя все этапы развития от возникновения потребности в программе определенного целевого назначения до полного прекращения использования этого ПС, вследствие его морального старения или потери необходимости решения задачи.

В настоящее время можно выделить 5 основных подходов к организации процесса создания и использования ПС.

Водопадный подход. При таком подходе разработка ПС состоит из цепочки этапов. На каждом этапе создаются документы, используемые на последующем этапе. В исходном документе фиксируются требования к ПС. В конце этой цепочки создаются программы, включаемые в ПС.

Исследовательское программирование. Этот подход предполагает быструю (насколько это возможно) реализацию рабочих версий программ ПС, выполняющих лишь в первом приближении требуемые функции. После экспериментального применения реализованных программ производится их модификация с целью сделать их более полезными для пользователей. Этот процесс повторяется до тех пор, пока ПС не будет достаточно приемлемо для пользователей. Такой подход применялся на ранних этапах развития программирования, когда технологии программирования не придавали большого значения (использовалась интуитивная технология). В настоящее время этот подход применяется для разработки таких ПС, для которых пользователи не могут точно сформулировать требования (например, для разработки систем искусственного интеллекта).

Прототипирование. Этот подход моделирует начальную фазу исследовательского программирования вплоть до создания рабочих версий программ, предназначенных для проведения экспериментов с целью установить требования к ПС. В дальнейшем должна последовать разработка ПС по установленным требованиям в рамках какого-либо другого подхода (например, водопадного).

Формальные преобразования. Этот подход включает разработку формальных спецификаций ПС и превращение их в программы путем корректных преобразований. На этом подходе базируется компьютерная технология (CASE-технология) разработки ПС.

Сборочное программирование. Этот подход предполагает, что ПС конструируется, главным образом, из компонент, которые уже существуют. Должно быть некоторое хранилище (библиотека) таких компонент, каждая из которых может многократно использоваться в разных ПС. Такие компоненты называются *повторно используемыми* (*reusable*). Процесс разработки ПС при данном подходе состоит скорее из сборки программ из компонент, чем из их программирования.

Рассмотрим более подробно водопадный подход. Именно этот подход рассматривается в качестве индустриального подхода разработки программного обеспечения. Исследовательское программирование исходит из взгляда на программирование как на искусство. Оно применяется тогда, когда водопадный подход не применим из-за того, что не удается точно сформулировать требования к ПС. Прототипирование рассматривается как вспомогательный подход, используемый в рамках других подходов, в основном, для прояснения требований к ПС.

В рамках водопадного подхода различают следующие стадии жизненного цикла ПС (см. рисунок 1.1): разработку ПС, производство программных изделий (ПИ) и эксплуатацию ПС.



Рисунок 1.1 - Стадии и фазы жизненного цикла ПС

Стадия *разработки* (*development*) ПС состоит из этапа его внешнего описания, этапа конструирования ПС, этапа кодирования (программирование в узком смысле) ПС и этапа аттестации ПС. Всем этим этапам сопутствуют процессы документирования и управления ПС. Этапы конструирования и кодирования часто перекрываются, иногда довольно сильно. Это означает, что кодирование некоторых частей программного средства может быть начато до завершения этапа конструирования.

Этап *внешнего описания* ПС включает процессы, приводящие к созданию некоторого документа, который мы будем называть *внешним описанием* (*requirements document*) *ПС*. Этот документ является описанием поведения ПС с точки зрения внешнего по отношению к нему наблюдателя с фиксацией требований относительно его качества. Внешнее описание ПС начинается с анализа и определения требований к ПС со стороны пользователей (заказчика), а также включает процессы спецификации этих требований.

*Конструирование* (*design*) ПС охватывает процессы: разработку архитектуры ПС, разработку структур программ ПС и их детальную спецификацию.

На этом этапе определяется потребность в ПС, его назначение и основные функциональные характеристики, оцениваются затраты и возможная эффективность применения такого комплекса программ.

*Кодирование* (*coding*) ПС включает процессы создания текстов программ на языках программирования, их отладку с тестированием ПС.

На этапе *аттестации* (*acceptance*) ПС производится оценка качества ПС. Если эта оценка оказывается приемлемой для практического использования ПС, то разработка ПС считается законченной. Это обычно оформляется в виде некоторого документа, фиксирующего решение комиссии, проводящей аттестацию ПС.

*Программное изделие (ПИ) −* экземпляр или копия разработанного ПС. *Изготовление* ПИ − это процесс генерации и/или воспроизведения (снятия копии) программ и программных документов ПС с целью их поставки пользователю для применения по назначению.

*Производство* ПИ − это совокупность работ по обеспечению изготовления требуемого количества ПИ в установленные сроки. Стадия производства ПИ в жизненном цикле ПС является, по существу, вырожденной (не существенной), так как представляет рутинную работу, которая может быть выполнена автоматически и без ошибок. Этим она принципиально отличается от стадии производства различной техники. В связи с этим в литературе эту стадию, как правило, не включают в жизненный цикл ПС.

Стадия *эксплуатации* ПС охватывает процессы хранения, внедрения ПС, а также транспортировки и применения ПИ по своему назначению. Она состоит из двух параллельно проходящих фаз: фазы применения ПС и фазы сопровождения ПС.

*Применение* (*operation*) ПС − это использование ПС для решения практических задач на компьютере путем выполнения ее программ.

*Сопровождение* (*maintenance*) ПС − это процесс сбора информации качестве ПС в эксплуатации, устранения обнаруженных в нем ошибок, его доработки и модификации, а также извещения пользователей о внесенных в него изменениях.

Каскадная модель широко использовалась в 70-80 годах.

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 (рисунок 1.2). 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:

* 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 ПС, 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 (рисунок 1.3):

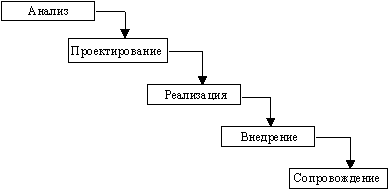


Рисунок. 1.2 - Каскадная схема разработки ПО

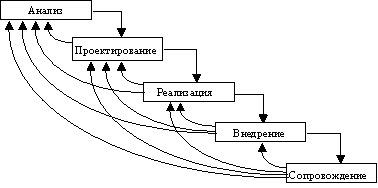


Рисунок 1.3 - Реальный процесс разработки ПО по каскадной схеме

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 ПС «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.

Спиральная модель жизненного цикла нашла свое широкое применение в 86-90 годах.

Для преодоления проблем, которые возникали при каскадном подходе была предложена спиральная модель жизненого цикла(ЖЦ) (рисунок 1.4), 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.

Основным нормативным документом, регламентирующим ЖЦ ПО, 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 ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

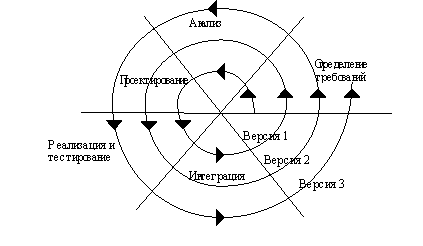


Рисунок 1.4 - Спиральная модель ЖЦ

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).