**Вопрос 143**

**модели ЖЦ ПО**

Понятие жизненного цикла программного обеспечения (ЖЦ ПО) является одним из базовых в программной инженерии. *Жизнен­ный цикл программного обеспечения* определяет­ся как период времени, который начинается с момента принятия ре­шения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.

Под *моделью Ж Ц ПО* понимается структура, определя­ющая последовательность выполнения и взаимосвязи процессов, дей­ствий и задач на протяжении ЖЦ. Модель ЖЦ зависит от специфи­ки, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

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 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, вклю­ченные в эти процессы.

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

К настоящему времени наибольшее распространение получили следующие две основные модели ЖЦ ПО: каскадная модель (1970 -1985 гг.) и спиральная модель (I986 - 1990 гг.).

В однородных ЭИС 70-х и 80-х гг. прикладное ПО представляло собой единое целое. Для разработки такого типа ПО применялся *каскадный подход* (другое название - водопад (waterfall)) (рис. 1.3). Принципиальной особенностью каскадного подхода является следу­ющее: *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 некоторых результатов, которые слу­жат в качестве исходных данных для следующей стадии. Требования к разрабатываемому ПО, определенные на стадии формирования требований, строго документируются в виде технического задания и фиксируются на все время разработки проекта. Каждая стадия за­вершается выпуском полного 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.

|  |
| --- |
|  |
|  |  |

1.3 Каскадная схема разработки ПО                                                       1.4  Реальный процесс

|  |
| --- |
|  |
|  |  |

разработки ПО

При этом основное внимание разработчиков сосредоточивается на достижении оптимальных значений технических характеристик раз­рабатываемого ПО: производительности, объема занимаемой памя­ти и др.

*Преимущества* применения каскадного способа заключа­ются в следующем:

на каждой стадии формируется законченный набор проектной документации, отвечающий критериям полноты и согласованно­сти;

выполняемые в логичной последовательности стадии работ по­зволяют планировать сроки завершения всех работ и соответству­ющие затраты.

Каскадный подход хорошо зарекомендовал себя при построении ЭИС, 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 системы с большим количеством задач вычислительного характера, системы реального времени и др.

В то же время этот подход обладает рядом *недостатков,* вызванных прежде всего тем, что реальный процесс создания ПО никогда полностью не укладывался в такую жесткую схему. Процесс создания ПО носит, как правило, *итерационный* характер: результа­ты очередной стадии часто вызывают изменения в проектных реше­ниях, выработанных на более ранних стадиях. Таким образом, по­стоянно возникает потребность в возврате к предыдущим стадиям и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ПО принимает иной вид (рис. 1.4).

Изображенную на рис. 1.4 схему часто относят к отдельной модели, так называемой *модели с промежуточным контролем*, в которой межстадийные корректировки обеспечивают большую надежность по сравне­нию с каскадной моделью, хотя и увеличивают весь период разработки.

Основным недостатком каскадного подхода являются суще­ственное запаздывание с получением результатов и, как следствие, достаточно высокий риск создания системы, не удовлетворяющей изменившимся потребностям пользователей. Практика показыва­ет, что на начальной стадии проекта полностью и точно сформу­лировать все требования к будущей системе не удается. Это объяс­няется двумя причинами: 1) пользователи не в состоянии сразу изложить все свои требования и не могут предвидеть, как они изменятся в ходе разработки; 2) за время разработки могут про­изойти изменения во внешней среде, которые повлияют на требо­вания к системе. В рамках каскадного подхода требования к ЭИС фиксируются в виде технического задания на все время ее созда­ния, а согласование получаемых результатов с пользователями производится только в точках, планируемых после завершения каждой стадии (при этом возможна корректировка результатов по замечаниям пользователей, если они не затрагивают требования, изложенные в техническом задании). Таким образом, пользовате­ли могут внести существенные 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. В результате приходится начинать новый проект, который может постигнуть та же участь.

Для преодоления перечисленных проблем в середине 80-х гг. была предложена *спиральная модель ЖЦ* (рис. 1.5).

Ее принципиальной особенностью является следующее: *прикладное ПО 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 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 тщатель­ная 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. Спиральная модель из­бавляет пользователей и разработчиков ПО от необходимости 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. Неполное завершение работ на каждой стадии позволяет переходить на следующую стадию, не до­жидаясь полного завершения работы на текущей. 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.

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.