**Конспект лекции ЛК4 Жизненный цикл процесса**

**Жизненным циклом программного обеспечения (ПО)** называют период от момента появления идеи создания некоторого программного обеспечения до момента завершения его поддержки фирмой-разработчиком или фирмой, выполняющей сопровождение [4]. Состав процессов жизненного цикла регламентируется международным стандартом ISO/IEC 12207: 1995 «Information Technologe - Software Life Cycle Processes» («Информационные технологии - Процессы жизненного цикла программного обеспечения»). Стандарт описывает структуру жизненного цикла ПО, называет и определяет процессы жизненного цикла ПО, не конкретизируя в деталях, как реализовывать или выполнять действия и задачи, включенные в эти процес­сы. Процесс жизненного цикла – это совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные. На рисунке A.5 представлены процессы жизненного цикла по указанному стандарту, каждый из которых характеризуется определенными задачами и методами их решения, а также исходными данными и результатами.

**Процесс разработки (development process)** в соответствии со стандартом предусматривает действия: подготовительную работу, анализ требований к системе, проектирование архитектуры системы, анализ требований к ПО, проектирование архитектуры ПО, детальное проектирование ПО, кодирование и тестирование ПО, интеграцию ПО, квалификационное тестирование ПО, интеграцию системы, квалификационное тестирование системы, установку и приемку ПО. Сгруппировав указанные действия, условно выделяют следующие основные этапы разработки программного обеспечения (в скобках указаны соответствующие стадии разработки по ГОСТ 19.102-77 «Стадии разработки»):

а) постановка задачи (стадия «Техническое задание») - формулируется назначение программного обеспечения, а также определяются основные требования к нему (функциональные и эксплуатационные). Результатом является разработка техничес­кого задания, фиксирующего принципиальные требования, и принятие ос­новных проектных решений;

б) анализ требований и разработка спецификаций (стадия «Эскизный проект») - выполняется анализ требований технического задания, формулируется содержательная постановка задачи, выбирается математический аппарат формализации, строится модель предметной области, определяются подзадачи и выбираются или разрабатываются методы их решения, формируются тесты для поиска ошибок в проектируемом программном обеспечении с указанием ожидаемых результатов. Точное формализованное описание функций и ограничений разрабатываемого ПО называется спецификациями, часть из которых может быть определена в процессе предпроектных ис­следований и зафиксирована в техническом задании; различают функ­циональные и эксплуатационные спецификации. Совокупность спецификаций представляет собой общую логическую модель проектируемого про­граммного обеспечения;

в) проектирование (стадия «Технический проект») - определяются подробные спецификации разрабатываемого программного продукта, выполняется проектирование общей структуры, декомпозиция компонентов и построение структурных иерархий в соответствии блочно-иерархическим подхода, проектирование компонентов. Результат - детальная модель разрабатываемого ПО со спецификациями компонен­тов всех уровней; тип модели зависит от выбранного подхода и конкретной технологии проектирования. Различают логическое и физическое проектирование;

г) реализация (стадия «Рабочий проект») - процесс поэтапного напи­сания кодов программы на выбранном языке программирования (кодирование), их тестирование и отладка.

**Сопровождение** - это процесс создания и внедрения новых версий программного обеспечения, причинами выпуска которых могут служить необходимость исправления ошибок, выявленных в процессе эксплуа­тации предыдущих версий; необходимость совершенствования предыдущих версий (улучшение интерфейса, расширение состава выполняемых функций или повышение его производительности); изменение среды функционирования, (появление новых тех­нических средств и/или программных продуктов, с которыми взаимодейст­вует сопровождаемое программное обеспечение). В программный продукт вносят необходимые изменения, которые могут потребовать пересмотра уже принятых проектных решений. С изменением модели жизненного цикла ПО роль этого этапа существенно возросла, так как продукты теперь создаются итерацион­но: сначала выпускается сравнительно простая версия, затем следующая с большими возможностями, затем следующая и т. д. Поэтому этап сопровождения был выделен в отдельный процесс жизненного цикла в соответствии со стандартом ISO/IEC 12207.

На протяжении последних тридцати лет в программировании сменились три модели жизненного цикла программного обеспечения: каскадная, модель с промежуточным контролем и спиральная.

Первоначально (1970-1985) была предложена и использовалась каскадная схема разработки ПО, которая предполагала, что переход на следующую стадию осуществляется только после того, как полностью будут завершены проектные операции предыдущей стадии и получены все исходные данные для следующей ста­дии. Достоинствами такой схемы являются получение в конце каждой стадии законченного набора проектной документации, отвечающего требованиям полноты и согласованности, а также простота планирования процесса разработки. Именно эту схему используют обычно при блочно-иерархическом подходе к разработке сложных технических объектов, обеспечивая очень высокие параметры эффективности разработки. Однако схема оказалась применимой только к созданию систем, для которых в самом начале разра­ботки удавалось точно и полно сформулировать все требования, что уменьшало вероятность возникновения в процессе разработки проблем, связанных с принятием неудачного решения на предыдущих стадиях. На практике такие разработки встречается крайне редко. В целом необходимость возвратов на предыдущие стадии обусловлена следующими причинами: неточные спецификации; изменение требований заказчика непосредственно в процессе разработки; быстрое моральное устаревание используемых технических и программных средств; отсутствие удовлетворительных средств описания разработки на стадиях постановки задачи, анализа и проектирования. Реальный же процесс носит итерацион­ный характер.

Схема, поддерживающая итерационный характер процесса разработки, была названа моделью с промежу­точным контролем. Контроль, выполняемый по данной схеме после завершения каждого этапа, позволяет вер­нуться на любой уровень и внести изменения.

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

Для преодоления перечисленных проблем в середине 80-х годов XX века была предложена спиральная схема. В со­ответствии с данной схемой ПО создается не сразу, а итерационно с использованием метода прототипирования, базирующегося на создании прототипов - действующих программных продуктов, реализую­щих отдельные функции и внешние интерфейсы разрабатываемого программного обеспечения. На первой итерации специфицируют, проектируют, реали­зуют и тестируют интерфейс пользователя; на второй - добавляют некото­рый ограниченный набор функций; на последующих этапах набор рас­ширяют, наращивая возможности продукта. Достоинство схемы: начиная 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. Основной проблемой использования схемы является определение моментов перехода на следующие стадии, для чего ограничивают сроки прохождения каждой стадии, основываясь на эксперт­ных оценках.

Изменение жизненного цикла программного обеспечения стало возможным в результате использования при разработке ПО CASE-технологий. CASE-технологии представляют собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных программных систем, основанных как на структурном, так и на объектном подходах, которые поддерживаются комплексом взаимосвязан­ных средств автоматизации. В основу любой CASE-технологии положены методология, метод, нотация и средства .

Среди средств различают CASE-средства анализа требований, проектирования спецификаций и структуры, редактирования интерфейсов (первое поколение CASE-I, в основном включают средства для поддержки графических моделей, проектирования спецификаций, экранных редакторов и словарей данных); CASE-средства генерации исходных текстов и реализации интегрированного окружения поддержки полного жизненного цикла разработки ПО (второе поколение CASE-II, существенно 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, современные CASE-средства существенно повышают производительность труда программистов и улучша­ют качество создаваемого программного обеспечения, поскольку обеспечивают автоматизированный контроль совместимости спецификаций проекта; уменьшают время создания прототипа системы; ускоряют процесс проектирования и разработки; автоматизируют формирование проектной документации для всех этапов жизненного цикла в соответствии с современными стандартами; частично генерируют коды программ для различных платформ разработки; поддерживают технологии повторного использования компонентов си­стемы; обеспечивают возможность восстановления проектной документации по имеющимся исходным кодам.

Появление CASE-технологий изменило все этапы жизненного цикла программного обеспечения, при этом наибольшие изменения касаются анализа и проектирования, которые предполагают строгое и наглядное описание разрабатываемого программного обеспечения. Од­нако современные CASE-средства дороги, а их ис­пользование требует более высокой квалификации разработчиков, следова­тельно, имеет смысл использовать их в сложных проектах, причем, чем сложнее разрабатываемое программное обеспечение, тем больше выигрыш от использования CASE-технологий. На сегодняшний день практически все промышленно производимое сложное программное обеспечение разрабаты­вается с использованием CASE-средств.

Разработка спиральной модели жизненного цикла программного обеспечения и CASE-технологии позволили сформулировать условия, выполне­ние которых сокращает сроки создания программного обеспечения.

**Готовый к выполнению (ready)** – Состояние процесса, при котором он ожидает получения ресурсов процессора для его исполнения. Диспетчеризация – поочередное выделение процессам планировщиком операционной системы квантов процессорного времени.

**Блокировка или, что то же самое, сон (sleeping, waiting**) – состояние, в котором процесс не может продолжать выполнение, пока не произойдет некоторое внешнее по отношению к процессу событие.

**Работа (running)** – в этом состоянии находится процесс, программу которого в данный момент выполняет процессор. Работающий процесс иногда удобно называть также текущим процессом.

**Зомби** — это процесс, который завершился, но не был удален. Удаление зомби возлагается на родительский процесс. Функция wait() тоже это делает, поэтому перед ее вызовом не нужно проверять, продолжает ли выполняться требуемый дочерний процесс. Предположим, к примеру, что программа создает дочерний процесс, выполняет нужные вычисления и затем вызывает функцию wait(). Если к тому времени дочерний процесс еще не завершился, функция wait() заблокирует программу. В противном случае процесс на некоторое время превратится в зомби. Тогда функция wait() извлечет код его завершения, система удалит процесс и функция немедленно завершится.