**Жизненный цикл программного обеспечения.**

Понятие «жизненный цикл» предполагает нечто рождаю­щееся, развивающееся и умирающее. Подобно живому организму программные изделия создаются, эксплуатируются и развиваются во времени.

**Жизненный цикл**программного обеспечения включает в себя все этапы его развития: от возникновения потребности в нем до полного прекращения его использования вследствие морального старения или потери необходимости решения соответствующих задач.

Можно выделить несколько фаз существования программного изделия в течение его жизненного цикла. Общепринятых названий для этих фаз и их числа пока еще нет. Но и особых разногласий по этому вопросу нет. Поэтому существует несколько вариантов разбиения жизненного цикла программного обеспечения на этапы. Вопрос о том, лучше ли данное конкретное разбиение, чем другие, не является основным. Главное, необходимо пра­вильно организовать разработку программного обеспечения с их учетом.

По длительности жизненного цикла программные изделия можно разделить на два класса: с **малым**и **большим временем жизни***.* Этим классам программ соответствуют гибкий (мягкий) подход к их созданию и использованию и жесткий промыш­ленный подход регламентированного проектирования и эксплуа­тации программных изделий. В научных организациях и вузах, например, преобладают разработки программ первого класса, а в проектных и промышленных организациях — второго.

**Программные изделия с малой длительностью эксплуатации**создаются в основном для решения научных и инженерных задач, для получения конкретных результатов вычислений. Такие прог­раммы обычно относительно невелики. Они разрабатываются одним специалистом или маленькой группой. Главная идея программы обсуждается одним программистом и конечным пользователем. Некоторые детали заносятся на бумагу, и проект реализуется в течение нескольких дней или недель. Они не предназначены для тиражирования и передачи для последующего использования в другие коллективы. По существу, такие прог­раммы являются частью научно-исследовательской работы и не могут  рассматриваться как отчуждаемые программные изделия.

Их жизненный цикл состоит из длительного интервала сис­темного анализа и формализации проблемы, значительного этапа проектирования программ и относительно небольшого времени эксплуатации и получения результатов. Требования, предъяв­ляемые к функциональным и конструктивным характеристикам, как правило, не формализуются, отсутствуют оформленные испытания программ. Показатели их качества контролируются только разработчиками в соответствии с их неформальными представлениями.



**Программные изделия с малой длительностью эксплуатации**

Сопровождение  и  модификация  таких  программ  не  обязатель­ны,  и  их  жизненный  цикл  завершается  после  получения  резуль­татов  вычислений.  Основные  затраты  в  жизненном  цикле  таких программ  приходятся  на  этапы  системного  анализа  и  проектирования,  которые  продолжаются  от  месяца  до 1…2 лет, в результате

чего жизненный цикл программного изделия редко превышает 3 года.

**Программные изделия с большой длительностью эксплуатации** создаются для регулярной обработки информации и управления. Структура таких программ сложная. Их размеры могут изменяться в широких пределах (1...1000 тыс. команд), однако все они обладают свойствами познаваемости и возможности модифи­кации в процессе длительного сопровождения и использования различными специалистами. Программные изделия этого класса допускают тиражирование, они сопровождаются документацией как промышленные изделия и представляют собой отчуждаемые от разработчика программные продукты.



**Программные изделия с большой длительностью эксплуатации**

Их проектированием и эксплуатацией занимаются большие коллективы специалистов, для чего необходима формализация программной системы, а также формализованные испытания и определение достигнутых показателей качества конечного про­дукта. Их жизненный цикл составляет 10...20 лет. До 70...90 % этого времени приходится на эксплуатацию и сопровождение. Вследствие массового тиражирования и длительного сопровождения совокупные затраты в процессе эксплуатации и сопровождения   таких программных изделий значительно превышают затраты на системный анализ и проектирование.

**Жизненный цикл ПО в общих чертах включает в себя следующие этапы:**

1. Постановка и формулировка задачи:

Изучение предметной области, определение назначения программы, определение структур входных и выходных данных, формирование ограничений и т.д.

2. Этап проектирования:

Выбор метода реализации задачи и разработка алгоритма реализации задачи;

3. Этап кодирования:

Программирование задачи;

4. Этап отладки и тестирования:

Обнаружение ошибок, корректировка кода программы и ее описания.

5. Этап эксплуатации и сопровождения:

Внедрение в работу, обучение пользователей работе с программой;

Многое определяется особенностями конкретной задачи, ее назначением, объемами кода и обрабатываемых данных, другими характеристиками исходной задачи.

Некоторые из этих этапов могут либо выполняться одновременно с другими, либо вовсе отсутствовать. Главная цель формирования подобного списка работ — в том, чтобы изначально упорядочить процесс создания нового программного продукта с сохранением концептуальной целостности постановки задачи и исключением анархии в процессе разработки.

Наиболее подробно рассмотрим этапы жизненного цикла ПО по Боэму.

**Фазы и работы ЖЦПО по Боэму:**

1**. Каскадная модель**

Каскадная модель была введена в 70 – 80 гг. Она удобна для однородных ПП, когда каждое приложение представляло собой единое целое.

Основные характеристики модели:

- Жизненный цикл разбивается на этапы (фазы);

- Переход с этапа на этап – только после полного завершения текущего этапа;

- Этап завершается выпуском полного комплекта документации, достаточной для того, чтобы работа могла быть выполнена другой командой разработчиков.

Главные характерные черты каскадной модели следующие:

завершение каждой фазы верификацией и подтверждением, цель которых – устранить возможно большее число проблем, связанных с разработкой изделия;

В каскадной модели успешное окончание одной из фаз ЖЦПО означает достижение соответствующей цели инженерного программирования . К этим подцелям необходимо добавить еще две:

Детальная проектируемость – получение полных верифицированных спецификаций и структур управления и данных, интерфейсных связей, характеристик, основных алгоритмов и определение условий работы каждого программного компонента.

Кодируемость – получение полного, верифицированного набора компонентов программы.

**Основные достоинства:**

Формирование полного набора проектной документации в конце работы над этапом. Документация отвечает критериям полноты и завершенности;

Возможность планирования сроков и затрат. Для целого ряда ПП эта модель реализуема – это для систем, для которых на этапе анализа можно точно и полно сформировать все требования. Например, сложные вычислительные программы.

**Основные недостатки:**

- Большие сроки от анализа до завершения;

- Требования к ПО «заморожены» в виде ТЗ до конца разработки.

2. **Экономическое обоснование каскадной модели**

Не углубляясь в экономический анализ, которому Б.У. Боэм уделяет большое внимание в книге «Инженерное проектирование программного обеспечения», скажем лишь, что экономическое обоснование каскадной модели, ориентированной на последовательное достижение целей, базируется на двух главных предпосылках:

Для получения качественного программного изделия (т.е. такого, которое в полной мере удовлетворяет всем целям требуемого программного изделия) необходимо в любом случае осуществить все подцели на каждом этапе.

Любое другое упорядочение подцелей приводит к созданию менее качественного программного изделия.

3. **Усовершенствование каскадной модели**

Рассмотрим одно из усовершенствований идеальной каскадной модели – пошаговую разработку.

Пошаговая разработка является усовершенствованием метода повторной разработки с созданием прототипа и поуровневой разработкой сверху – вниз. Этот метод предполагает пошаговое увеличение функциональных возможностей ПО в процессе разработки.

В качестве усовершенствованной каскадной модели пошаговая разработка успешно применялась при создании как очень больших, так и небольших программных изделий.

Главными преимуществами пошаговой разработки перед абсолютно повторной разработкой и поуровневой разработкой сверху - вниз являются следующие:

использование последовательных расширений программы обеспечивает гораздо менее дорогой способ учета в усовершенствованном изделии опыта пользователей, чем при повторной разработке;

расширение функциональных возможностей намного упрощает проверку и полезнее, чем промежуточные изделия при поуровневой разработке.

Значение пошаговой разработки заключается главным образом в изменении распределения затрат труда на проект.

**4 Определение фаз жизненного цикла**

Ниже будут даны формулировки конечных целей каждой фазы для перехода к следующей фазе. Для пошаговой разработки приводимые формулировки относятся к границам фаз каждого шага расширения.

Начать фазу планирования и анализа требований. (Завершение концептуального обзора ЖЦПО.)

Получение одобренной и подтвержденной архитектуры системы с включением основных соглашений о распределении функций между аппаратурой и программами. Получение одобренного и подтвержденного общего представления об функционировании ПО с включением основных соглашений о распределении функций между человеком и системой.

Формирование общего плана ЖЦПО с определением основных этапов, ресурсов, обязанностей, сроков и главных работ.

Завершить фазу планирования и анализа требований. Начать фазу проектирования изделия. (Завершение обзора требований к ПО).

Формирование детального плана разработки: детальных показателей завершения этапов разработки, планов распределения ресурсов, схем организационной структуры, обязанностей, сроков, работ, методов и изделий.

Формирование детального плана использования: пунктов плана разработки, содержание которых ориентировано на обучение, перенос программ, внедрение, эксплуатацию и поддержание.

Формирование детального плана отладки изделия – план управления конфигурацией технического обеспечения, план контроля качества, общий план верификации и подтверждения.

Одобренная и подтвержденная спецификация требований к ПО: функциональные, технические и интерфейсные спецификации, для которых подтверждены их полнота, непротиворечивость, проверяемость и осуществимость.

Одобренный (формально или неформально) договор на разработку, основанный на приведенных выше пунктах.

Закончить фазу проектирования изделия. Начать фазу детального проектирования. (Завершение анализа результатов проектирования изделия.)

Разработка верифицированной спецификации проекта программного изделия:

формирование иерархии программных компонентов, межблочных интерфейсов по данным и управлению;

формирование физической и логической структур данных до уровня отдельных полей;

разработка плана распределения вычислительных ресурсов (времени, памяти, точности);

верификация полноты, непротиворечивости, осуществимости и обоснованности требований.

Установление и разрешение всех противоречий разработки, которые повышают степень риска.

Разработка предварительного этапа комплексирования и отладки, плана руководства для пользователей и приемных испытаний.

Закончить фазу детального проектирования. Начать фазу кодирования и автономной отладки. (Завершение сквозного контроля проекта или критического поблочного анализа проекта.)

Верифицированная детальная спецификация каждого блока:

спецификация каждой подпрограммы, имени, назначения, предположений, размеров, последовательности вызовов, ошибочных выходов, входных и выходных данных, алгоритмов и логической схемы;

описание базы данных до уровня отдельных параметров, символов и битов;

верификация полноты, непротиворечивости и соответствия требованиям проектных спецификаций системы и планам распределения ресурсов.

Одобренный план приемных испытаний.

Руководства пользователю, а также завершенный предварительный план комплексирования и отладки.

Закончить фазу копирования и отладки. Начать фазу комплексирования и отладки. (Удовлетворение критериев автономной отладки.)

Проверка работы всех блоков не только для номинальных, но также для исключительных и предельных значений.

Проверка всех вариантов ввода и вывода, включая сообщения об ошибках.

Выполнение всех операторов и всех ветвей передачи управления.

Проверка выполнения стандартов программирования.

Завершение поблочного документирования внутренней структуры.

Закончить фазу комплексирования и испытаний. Начать фазу внедрения. (Завершение анализа результатов приемных испытаний.)

Проверка удовлетворения тесту приемных испытаний программ:

проверка удовлетворения требованиям к ПО;

демонстрация приемлемости указанных в спецификациях характеристик работы в нештатных условиях.

Приёмка поставляемых программных изделий, отчетов, руководств, баз данных, спецификаций внутренней структуры.

Закончить фазу внедрения. Начать фазу эксплуатации и сопровождения. (Завершение анализа приемки системы.)

Проверка удовлетворительности результатов приемных испытаний системы.

Проверка удовлетворительности системных требований.

Проверка производственной готовности ПО, аппаратуры, средств обслуживания и персонала.

Приёмка поставляемых и входящих в систему изделий: аппаратуры, ПО, документации, средств обучения и обслуживания.

Завершение всех специфицированных работ и ввод системы в действие.

Закончить фазу эксплуатации и сопровождения (путем снятия с производства).

Выполнение всех пунктов плана снятия с производства: перенос программ, документирование, создание архива, переход к новой системе.

**5. Основные работы над проектом**

Анализ требований.

Проектирование изделия.

Программирование.

Планирование отладки.

Верификация и подтверждение.

Управление проектом.

Управление конфигурацией и контроль качества.

Документирование.

**Вывод:**

Определение фаз и работ ЖЦПО Боэма наиболее обоснованно, т.к. опирается на более ориентированный подход в инженерном программировании (направленный на получение качественного программного изделия и реализацию эффективного процесса разработки и сопровождения ПО) и обосновывается экономически.

Важно так же, при составлении программы, учитывать то, что программа должна быть точной; полной по своему содержанию и пригодной для работы как с маленькими, так и с большими проблемами в соответствии со своим предназначением; ясной - для того чтобы пользователь мог спокойно, без затруднений работать с ней. А так же чтобы программу в любой момент можно было бы легко исправить или дополнить в соответствии с изменившимися требованиями в современном мире.

Следует помнить, что хорошее программирование – это не кодирование быстро найденного решения с помощью любой подходящей методики, а тщательно инструментированная инженерная процедура, позволяющая создать полное, точное и легко понимаемое (ясное) программное обеспечение.

**Источники:**

http://text.tr200.biz/referat\_informatika/?referat=596460&page=1

http://allrefs.net/c20/45yet/p9/

http://tehprog.ru/index.php\_page=lecture15.html