Технология программирования

Блок 1. Основы систем программирования

Тема 2. Проблемы разработки сложных программных систем. Жизненный цикл и процессы разработки ПО.

Цели изучения темы:

- ознакомить студентов с понятием сложных программных систем;
- сформировать базу знаний о процессах разработки программного обеспечения.

Задачи изучения темы:

- сформировать знания о особенностях и принципах жизненных циклов программ.

В результате изучения данной темы Вы будете
знать:

- понятие и особенности жизненного цикла программ;
- модели жизненных циклов;
- методологии разработки ПО;

уметь:

- отличать программы с малым временем жизни и с большим временем жизни;
- применять программы с малым и с большим временем жизни.

владеть:

- методами и критериями оценивания качества программного обеспечения.

Учебные вопросы темы:

1. Жизненный цикл программного обеспечения: понятие, стандарты, процессы.
2. Методологии разработки ПО.

Основные термины и понятия, которые Вам предстоит изучить:

жизненный цикл, каскадная модель, итерационная модель, Rational Unified Process, экстремальное программирование, Microsoft Solutions Framework.

Вопрос 1. Жизненный цикл программного обеспечения: понятие, стандарты, процессы

Первое поколения ЭВМ отличалось конструктивной простотой и небольшими возможностями. Поэтому человек, решавший задачу на такой ЭВМ, должен был быть и программистом, и специалистом в области решаемой задачи, и разбираться в устройстве ЭВМ.

С развитием ВТ увеличились ее возможности, вызвавшие развитие программирования, которое потребовало произвести разделение труда не только между программистами и представителями других профессий, но и в среде программистов. Следствием стало создание программ как отчуждаемых самостоятельных продуктов, которые обладают всеми свойствами любого промышленного изделия, а программирование превратилось в индустриальную отрасль.

Появилось понятие *жизненного цикла программы*, т.е. времени всех этапов развития – от возникновения, до полного прекращения ее эксплуатации вследствие морального старения или потери необходимости решения некоторой задачи.

Разработка ПО невозможна без понимания жизненного цикла программ. Рядовому пользователю это, может быть, и не нужно знать, но основные стандарты желательно усвоить.

Под жизненным циклом любого программного продукта принято понимать время его существования, начиная со стадии разработки и до момента полного отказа от использования в выбранной сфере применения вплоть до полного изъятия приложения из обихода.

Комплексы программ создаются, эксплуатируются и развиваются во времени. Жизненный цикл программных средств включает в себя все этапы развития: от возникновения потребности в программе, определением целевого назначения, до полного прекращения использования этого программного средства, вследствие его морального старения или потере необходимости решения соответствующих задач.

По длительности жизненного цикла все программы делятся на два класса – с *малым* и *большим* временем жизни.

Этим классам соответствуют 2 подхода в их создании:

- *гибкий* (как объект научного творчества);

- *жестко стандартизированный* (промышленный).

Программы с *малым* сроком жизни обычно создаются для решения научных или инженерных задач, когда главную ценность имеет результат.

Программы с *большим* сроком жизни создаются для регулярной обработки информации или управления в процессе функционирования сложных систем.

Программы *первого* типа допускают тиражирование, оформляются документацией, как любое промышленное изделие и представляют собой отчуждаемый программный продукт.

Программы с малой длительностью эксплуатации создаются в основном для решения научных и инженерных задач, для получения конкретных результатов вычислений. Такие программы относительно невелики от 1 до 10000 команд. Разрабатываются как правило одним специалистом или маленькой группой, не предназначены для тиражирования и передачи для последующего использования в другие коллективы. ЖЦ таких программ состоит из длительного интервала системного анализа и форматизации 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 и показатели их качества контролируются только разработчиками, в соответствии с неформальными представлениями.

Сопровождение и модификация таких программ не нужны и их ЖЦ завершается после получения результатов вычислений. Основные затраты в ЖЦ таких программ приходятся на этапы системного анализа и проектирования, которые продолжаются от месяца до одного или двух лет. В результате ЖЦ редко превышает три года, т.е. основное время идет на анализ и проектирование.

Программы с большой длительностью эксплуатации создаются для регулярной обработки информации и управления в процессе функционирования сложных вычислительных систем. Размеры программных средств могут изменяться в широких пределах (от 1 до бесконечности команд), однако все они обладают свойством познаваемости и возможности модификации в процессе использования различными специалистами. Программы такого класса допускают тиражирование. Они сопровождаются документацией как промышленные изделия и представляют собой отчуждаемый программный продукт.

Проектированием и эксплуатацией программного средства могут заниматься большие коллективы специалистов, для чего необходима формализация требуемых технических характеристик комплекса программ и их компонент. А также формализированые испытания и определение достигнутых показателей качества программных средств.

Модели жизненного цикла

Существует несколько моделей жизненного цикла.

Каскадная модель жизненного цикла (англ. waterfall model) была предложена в 1970 г. Уинстоном Ройсом. Она предусматривает

последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе. Требования, 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.

Этапы проекта в соответствии с каскадной моделью:

1. формирование требований;
2. проектирование;
3. реализация;
4. тестирование;
5. внедрение;
6. эксплуатация и сопровождение.

В каскадной модели переход от одной фазы проекта к другой предполагает полную корректность результата предыдущей фазы. В больших проектах этого добиться практически невозможно. Поэтому такая модель пригодна только для разработки небольшого проекта. (Сам У. Ройс не придерживался данной модели и использовал модель итерационную).

Итерационная модель. Альтернативой каскадной модели является модель итеративной и инкрементальной разработки (англ. iterative and incremental development, IID), получившей от Т. Гилба в 70-е гг. название эволюционной модели. Модель IID предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «мини-проект», включая все процессы разработки в применении к созданию меньших фрагментов функциональности, по сравнению с проектом в целом. Цель каждой итерации — получение работающей версии программной системы, включающей функциональность, определённую интегрированным содержанием всех предыдущих и текущей итерации. Результат финальной итерации содержит всю требуемую функциональность продукта. Таким образом, с завершением каждой итерации продукт получает приращение — инкремент — к его возможностям, которые, следовательно, развиваются эволюционно.

Стандартизация жизненного цикла приложения

Среди систем, предопределяющих условия и требования, предъявляемые к таким процессам, сегодня можно назвать только три основных:

- ГОСТ 34.601-90;
- ISO/IEC 12207:2008;
- Oracle CDM.

Для второго международного стандарта имеется российский аналог. Это ГОСТ Р ИСО/МЭК 12207-2010, **«Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств»**. Но жизненный цикл программного обеспечения, описываемый в обоих правилах, является идентичным по сути. Данный стандарт, устанавливает общую структуру процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии. Стандарт не предлагает конкретную модель жизненного цикла. Его положения являются общими для любых моделей жизненного цикла, методов и технологий создания ПО. Он описывает структуру процессов жизненного цикла, не конкретизируя, как реализовать или выполнить действия и задачи, включенные в эти процессы.

Виды ПО и апдейты

Современные информационные системы таковы, что для них устанавливаются общепринятые понятия области применения.

Например, есть системные программы и утилиты, средства мультимедиа, драйверы устройств, офисные приложения и т. д. Для любого типа программных продуктов можно определить этапы жизненного цикла существования.

Для его продления чаще всего применяются средства обновления (как для операционных систем, так и для платформ и прикладного ПО). Наверное, не нужно объяснять, что любой пользователь компьютерной системы на основе Windows проходил этап обновления самой системы или компонентов вроде Microsoft .NET Framework или виртуальной машины Java.

Вопрос 2. Методологии разработки ПО

Различные варианты итерационного подхода реализованы в большинстве современных методологий разработки:

- Rational Unified Process
- Extreme Programming
- Microsoft Solutions Framework

Rational Unified Process (RUP) (рациональный унифицированный процесс) — методология разработки программного обеспечения, которая поддерживается компанией Rational Software (IBM). В методологии даются рекомендации по всем этапам разработки: от моделирования бизнеса до тестирования и сдачи в эксплуатацию

готовой программы. В качестве языка моделирования используется язык Unified Modelling Language (UML).

Экстремальное программирование – одна из гибких методологий разработки программного обеспечения. Название методологии исходит из идеи применить полезные традиционные методы и практики разработки программного обеспечения, подняв их на новый «экстремальный» уровень. Так, например, практика выполнения ревизии кода, заключающаяся в проверке одним программистом кода, написанного другим программистом, в «экстремальном» варианте представляет собой «парное программирование», когда один программист занимается написанием кода, а его напарник в это же время непрерывно просматривает только что написанный код.

Microsoft Solutions Framework (**MSF**) — методология разработки программного обеспечения, предложенная корпорацией Microsoft. MSF опирается на практический опыт Microsoft и описывает управление людьми и рабочими процессами в процессе разработки решения. MSF представляет собой согласованный набор концепций, моделей и правил.

Полный жизненный цикл разработки продукта состоит из четырех фаз, каждая из которых включает в себя одну или несколько итераций.

- **Начальная стадия (Inception)**

  Определение масштабов проекта и объема необходимых ресурсов. Определяются основные требования, ограничения и ключевая функциональность продукта. Оцениваются риски. Планирование действий. При завершении начальной фазы оценивается достижение этапа жизненного цикла цели (англ. Lifecycle Objective Milestone), которое предполагает соглашение заинтересованных сторон о продолжении проекта.

- **Уточнение (Elaboration)**

  Документирование требований. Проектирование, реализация и тестирование исполняемой архитектуры. Уточнение сроков и стоимости. Снижение основных рисков. Успешное выполнение фазы разработки означает достижение этапа жизненного цикла архитектуры (англ. Lifecycle Architecture Milestone).

- **Построение (Construction)**

  В фазе «Построение» происходит реализация большей части функциональности продукта: дизайн приложения завершен, исходный код написан. Фаза Построение завершается первым внешним релизом системы и вехой начальной функциональной готовности (Initial Operational Capability).

- **Внедрение (Transition)**

  В фазе «Внедрение» создается финальная версия продукта и передается от разработчика к заказчику. Это включает в себя

программу бета-тестирования, обучение пользователей, а также определение качества продукта. В случае, если качество не соответствует ожиданиям пользователей или критериям, установленным в фазе Начало, фаза Внедрение повторяется снова. Выполнение всех целей означает достижение вехи готового продукта (Product Release) и завершение полного цикла разработки.

Стадия проектирования

Теперь несколько слов непосредственно о стадиях разработки. Жизненный цикл ПО изначально включает в себя планирование проекта, анализ системных и целевых требований, возможности предварительного или детального проектирования, кодирование и тестирование, возможность применения программ в специализированных системах и т. д.

Модели жизненного цикла программного обеспечения предполагают, что изначально поставленная задача по созданию программного обеспечения должна сводиться к разработке универсальных приложений или программных продуктов, использующих определенную среду запуска.

Разработка

Системы разработки представляют собой языки программирования. Проектирование программного обеспечения на первой стадии может сводиться именно к этому.

Будет ли это C+/C++, Java, Delphi или тот же Pascal – не столь важно. Вопрос состоит в том, насколько созданное приложение сможет интегрироваться в операционную систему и работать без сбоев.

В этом смысле 1 жизненный цикл программного обеспечения является временем его тестирования от начальной инсталляции продукта до его полного удаления вследствие несоответствия требованиям системы, неработоспособности или невозможности выполнения изначально поставленных задач.

Последующие этапы

Дальнейшее сопровождение, определяющие жизненный цикл программного обеспечения, сводится к тому, чтобы произвести кодирование и получение исходного кода приложения.

В случае его бесплатного (открытого) распространения применяется так называемый сертификат на основе лицензии GNU, что предусматривает возможность изменения самого программного продукта по желанию стороннего пользователя, знакомого с языком программирования, с помощью которого приложение создавалось.

Если речь идет о закрытом коде, можно воспользоваться утилитами вроде Disassembler. Но в этом случае можно добиться только

раскодирования исполняемого EXE-файла, а уж никак не привязанных динамических библиотек DLL.

На практике стадии жизненного цикла ПО включают в свой список куда больше элементов. Даже самая простая моделируемая ситуация состоит из рассмотрения стандартов и формулирования замечаний (высокоуровневые требования к архитектуре, соответствие исполняемого кода, средства и методика верификации). Это и есть процесс жизненного цикла программного обеспечения.

Основы управления

Управление жизненным циклом программного обеспечения осуществляется на основе разбиения программ на составляющие, что дает достаточно широкий выбор средств для их создания.

Есть и обратная сторона медали. Выбор модулей ограничивается разработчиком изначальной платформы, на основе которой производится программирование. Конечно, если взять в расчет унификацию и типизацию применяемых средств разработки (особенно многократно используемых модулей), тут вопросов нет.

А вот этапы жизненного цикла программного обеспечения в обязательном порядке содержат создание протоколов обработки данных, подпрограмм, стандартных библиотек и многого другого.

Используемые модули

И ни один процесс жизненного цикла программного обеспечения не обходится без использования весьма специфичных компонентов. Среди них приоритетными считаются следующие:

- главный (головной) модуль, отвечающий за запуск программного продукта;
- управляющий модуль, ответственный за вызов присоединяемых компонентов или динамических библиотек;
- функциональные и сервисные средства обработки данных и дополнительные утилиты.

Исполняемый файл, как правило, для Windows-систем представлен в виде «экзэшника». Управляющие компоненты имеют расширение конфигураторов (config.sys применительно к операционным системам), дополнительно подключаемые библиотеки имеют расширение DLL. Средства контроля и обработки функций и настроек некоторых приложений могут выглядеть в виде файлов XML.

Перспективы развития

Что собой представляют этапы жизненного цикла программного обеспечения, уже понятно. Но вот о развитии таких технологий стоит сказать отдельно.

Не нужно говорить, что любой разработчик программного обеспечения не заинтересован в создании мимолетного продукта, который едва ли удержится на рынке в течение нескольких лет. В перспективе все

смотрят на долгосрочное его использование. Достигаться это может разными способами. Но, как правило, практически все они сводятся к выпуску обновлений или новых версий программ.

Даже в случае с ОС Windows такие тенденции можно заметить невооруженным взглядом. Вряд ли сегодня найдется хоть один юзер, использующий системы вроде модификаций 3.1, 95, 98 или Millennium. Их жизненный цикл закончился после выхода версии XP. Но вот серверные версии на основе технологий NT все еще актуальны. Даже Windows 2000 на сегодняшний день является не только весьма актуальной, но и по некоторым параметрам установки или безопасности, даже превосходящей самые новые разработки. То же самое касается системы NT 4.0, а также специализированной модификации Windows Server 2012.

Но по отношению именно к этим системам все равно заявлена поддержка на самом высоком уровне. А вот нашумевшая в свое время Vista явно испытывает закат цикла. Мало того, что она оказалась недоработанной, так еще и ошибок в ней самой и прорех в ее системе безопасности было столько, что остается только догадываться о том, как можно было выпустить на рынок программных продуктов такое несостоятельное решение.

Но если говорить о том, что развитие ПО любого типа (управляющего или прикладного) не стоит на месте, можно только констатировать факты. Ведь сегодня дело касается не только компьютерных систем, а и мобильных устройств, в которых применяемые технологии зачастую опережают компьютерный сектор. Появление процессорных чипов на основе восьми ядер – чем не самый лучший пример? А ведь еще далеко не каждый ноутбук может похвастаться наличием такого «железа».

Что же касается понимания жизненного цикла программного обеспечения, сказать, что он закончился в некоторый определенный момент времени, можно весьма условно, ведь программные продукты все равно имеют поддержку со стороны разработчиков, их создававших. Скорее окончание относится к устаревшим приложениям, которые не отвечают требованиям современных систем и не могут работать в их среде.

Но даже с учетом технического прогресса многие из них уже в ближайшее время могут оказаться несостоятельными. Вот тогда и придется принимать решение либо о выпуске обновлений, либо о полном пересмотре всей концепции, изначально заложенной в программный продукт. Отсюда – и новый цикл, предусматривающий изменение начальных условий, среды разработки, тестирования и возможного долгосрочного применения в определенной сфере.

Но в компьютерных технологиях сегодня отдается предпочтение развитию автоматизированных систем управления (АСУ), которые

применяются на производстве. Даже операционные системы, в сравнении со специализированными программами, проигрывают. Те же среды на основе Visual Basic остаются намного более популярными, нежели Windows-системы. А о прикладном ПО под UNIX-системы речь не идет вообще. Что говорить, если практически все коммуникационные сети тех же Соединенных Штатов работают исключительно на них. Кстати, системы вроде Linux и Android тоже изначально создавались именно на этой платформе. Поэтому, скорее всего, у UNIX перспектив намного больше, чем у остальных продуктов вместе взятых.

В данном случае приведены только общие принципы и этапы жизненного цикла программного обеспечения. На самом деле даже начально поставленные задачи могут разниться очень существенно. Соответственно, различия могут наблюдаться и на остальных стадиях.

Но основные технологии разработки программных продуктов с их последующим сопровождением должны быть понятны. В остальном же следует учитывать и специфику создаваемого ПО, и среды, в которых оно предположительно должно работать, и возможности программ, предоставляемые конечному пользователю или производству, и многое другое.

К тому же, иногда жизненные циклы могут зависеть от актуальности средств разработки. Если, допустим, какой-то язык программирования устаревает, никто же не будет писать программы на его основе, и уж тем более – внедрять их в автоматизированные системы управления на производстве. Тут уже на первый план выходят даже не программисты, а маркетологи, которые должны своевременно реагировать на изменения компьютерного рынка. И таких специалистов в мире найдется не так уж и много. Высококвалифицированные кадры, способные держать руку на пульсе рынка, становятся наиболее востребованными. И именно они зачастую являются так называемыми «серыми кардиналами», от которых зависит успех или проигрыш определенного программного продукта в сфере IT.

Пусть они не всегда понимают суть программирования, зато четко способны определить модели жизненного цикла программного обеспечения и продолжительности времени их применения, исходя из мировых тенденций в этой области. Эффективный менеджмент зачастую дает более ощутимые результаты. А также PR-технологии, реклама и т. д. Может какое-то приложение пользователю и не нужно, зато при условии его активного афиширования юзер установит его.

Стоит добавить, что некоторые программы при запуске используют звуковой сигнал, привлекающий внимание юзера. И, как показывают исследования, такие приложения оказываются более жизнеспособными, в сравнении с другими программами. Естественно, увеличивается и жизненный цикл ПО, без разницы, какая функция на

него возложена изначально. И этим, к сожалению, пользуются многие разработчики, что вызывает сомнения в законности таких методов. Возможно, в ближайшее время будут разработаны средства, определяющие такие угрозы. Пока это только теория, но, как считают некоторые аналитики и эксперты, до практического применения осталось совсем немного.

Контрольные вопросы:

1. Что такое жизненный цикл ПО?
2. Каковы условия возникновения понятия жизненного цикла?
3. Какие отличия жизненных циклов у программ с малым временем жизни и с большим временем жизни?
4. В каких случаях применяются программы с малым и с большим временем жизни?
5. Назовите основные модели жизненного цикла и их отличия.
6. Какие есть стандарты жизненного цикла?
7. Какие существуют методологии разработки ПО?
8. Какие основные стадии (фазы) жизненного цикла существуют?
9. Какие бывают виды ПО?