Учебный курс

Модели жизненного цикла и методологии разработки корпоративных систем

Лекция 2

Общая схема жизненного цикла корпоративных систем

Лекции читает

кандидат технических наук, доцент
Зыков Сергей Викторович

Понятие программной системы

Программная система – совокупность взаимодействующих программ под общим управлением, предназначенная для решения задачи или ряда взаимосвязанных задач

Приложение (прикладная программа) предназначена для решения функциональных задач по обработке информации той или иной предметной области

Программная инженерия – комплекс задач, методов, средств и технологий создания (проектирования и реализации) сложных, расширяемых, тиражируемых, высококачественных программных систем (возможно, включающих БД)
В.В.Липаев. Программная инженерия, 2006

Виды разработки ПО

- частная разработка: некоммерческое использование;
- разработка продукта:
  - заказчика нет, а есть идея.
  - Необходимы начальные инвестиции.
  - Требования к продукту определяет отдел маркетинга;
- проектная разработка:
  - Есть потребность и цели проекта разработки.
  - Есть бюджет и временные рамки.
  - Есть заказчик и др. заинтересованные лица.
- смешанная.

- Вид разработки может измениться в ходе развития ПО (частная/продуктовая разработка → смешанная)

Программный продукт

- Характеристики:
  - коммерческая ценность
  - может быть предложен рынку для удовлетворения потребности
- Виды/примеры:
  - физический объект (носитель информации, скажем, DVD);
  - нематериальный объект (код, ПО, лицензия, соглашение о партнерстве и т.п.);
  - услуга (внедрение, сопровождение, обучение, консультации).
- **Классификации**:
  - масштаб использования (личное, некоммерческое, коммерческое)
  - мель использования (специальное ПО или общего назначения: ОС, MS Office и т.п.).
  - степень открытости (компоненты: ActiveX, API, библиотеки или готовые продукты)

ЖЦ разработки ПО

«Любая разработка ПО происходит по «жизненному циклу», состоящему из всех видов деятельности, которые осуществляются с того момента, как версия 1.0 системы начинает свое существование лишь как идея, и до того момента, когда версия 6.74b сделает свой последний вздох на компьютере последнего заказчика.»

Стив Мак Коннелл, Быстрая разработка, 1996
(Steve McConnell, Rapid Development)

ЖЦ разработки ПО - особенности

- Программная система разрабатывается постепенно и развивается, начиная от зарождения идеи ПО до реализации и сдачи пользователю, и далее.
- Каждый этап завершается разработкой части системы или связанной с ней документации (план тестирования, руководство пользователя и т.д).
- Теоретически, для каждого этапа четко определены начальные и конечные точки, а также известно, что он должен передать следующему этапу.
- На практике все сложнее.

Цели изучения ЖЦ

- организация и управление разработкой ПО.
- основа для анализа разработки ПО.
- основа для планирования разработки ПО.
- корректная постановка процессов разработки ПО

Анализ ЖЦ обязателен для сложных проектов.

Важные предварительные замечания:

- В процессах создания ПО участвует много сторон:
  - заказчики,
  - разработчики,
  - руководство

- У сторон различные цели, ожидания и ограничения

  (часто даже согласование разумных подходов приводит к значительному росту сроков и стоимости проекта)

Участники проекта

- Заказчик
- Руководитель портфеля проектов
- Менеджер проекта
- Руководитель команды
- Эксперт предметной области
- Аналитик
- Архитектор
- Проектировщик подсистем
- Специалист по пользовательскому интерфейсу
- Разработчик
- Тест-менеджер
- Тестировщик
- Технический писатель

Цель и основные факторы разработки:

Разработка ПО – многофакторная оптимизация

Пути создания ПО с желаемым выходом по заданному входу множественны

Цель разработки ПО: выбор методологии путем многомерной оптимизации с учетом, прежде всего, следующих факторов:

- сроки;
- стоимость;
- качество;
- сопровождаемость

Приоритетность факторов определяется характером и масштабом проекта

Масштабы программных систем

- Малые: до 10 человеко-лет.
- Средние: 10-100 человеко-лет.
- Большие: 100-1000 человеко-лет
- Огромные: от 1000 человеко-лет

Корпоративные приложения: от 100 человеко-лет

Стадии ЖЦ ПО, не зависящие от методологий:

- Анализ требований
- Подготовка спецификаций
- Проектирование (эскизное, детальное, рабочее)
- Реализация
- Тестирование
- Интеграция (сборка – ср. assembly в MS .NET)
- Сопровождение
- Снятие с эксплуатации

Все стадии, кроме последней, включают **документирование!**

Обратите внимание на взаимосвязи документов!

Анализ требований:

- Встреча разработчика и заказчика
- Достижение общего понимания задачи, для решения которой будет разработано ПО
- Выявление и обсуждение требований и ограничений заказчика к ПО (посредством собеседования)
- Результат:
  - формализованное описание требований (statement of scope?)

  или

  - список требований (requirements checklist)

Подготовка проектных спецификаций:

- На основе описания требований
- Готовится разработчиком
- Содержит:
  - Описание всей функциональности проекта
  - Выбранную методологию/модель разработки ПО (следует определить как можно раньше!)
  - Оценку сроков проекта
  - Оценку стоимости проекта

Детальное проектирование:

- Производится на основе проектных спецификаций
- Выполняется разработчиком
- Содержит:
  - Описания всех программных модулей
  - Описание программной архитектуры:

    интеграция компонент проекта (при ООП - модулей и интерфейсов) с программной средой заказчика

Реализация:

- Производится на основе:
  - документов детального проектирования,
  - общего плана проекта (глобальные ограничения сроков и стоимости, важнейшие функциональные параметры и ограничения)
- Готовится программистами разработчика
- Содержит:
  - Отдельные программные модули
- Результат:
  - каждый программный модуль

  реализован и протестирован (**пока по отдельности!**)

Интеграция:

- Разработчик
  - Сборка модулей в общую архитектурную схему
- Разработчик и заказчик
  - Тестирование
- Результаты:
  - ПО разворачивается у заказчика
  - **все** приемочные тесты (проведенные заказчиком на реальном АО и ПО согласно функциональным требованиям) **успешны**
  - ПО передается заказчику

Наступает фаза эксплуатации ПО

Сопровождение:

- По окончании приемочного тестирования программного продукта
- Включает следующие типы:
  - ***Корректирующее*** – устранение остаточных сбоев без изменения спецификаций
  - ***Совершенствующее/ обновляющее*** – внесение изменений в проектные спецификации и новая итерация стадии разработки
  - ***Улучшающее*** – рост производительности с сохранением функциональности
  - ***Адаптивное*** - при миграции в новую среду
- Экономика: **Max (~70%) затрат по времени и средствам!**
- **Необходимо для любого ПО!**

Вывод из эксплуатации:

- **После полного вывода** ПО из использования
- Если функции ПО все еще необходимы, включает экспорт данных в новые приложения
- Стоимость замены ПО складывается, в частности, из:
  - стоимости смены технологий (цены нового ПО);
  - стоимости разработки и поддержки приложений на основе нового ПО;
  - затрат на обучение персонала нового ПО;
  - краткосрочной потери производительности в переходный период
- **Решение принимается на основе оценки стоимости**
- Иногда вынужденный – напр., при несовместимости (Y2K)

План проекта

- Важнейший, «глобальный» проектный документ
- Объединяет перечисленные фазы ЖЦ ПО
- Включает:
  - общее расписание проекта
    (в т.ч. «работы» - activities и «вехи» - milestones,
    в MSF – deliveries/milestones);
  - план управления рисками;
  - план тестирования;
  - план интеграции;
  - другие «глобальные» документы

Особенности ЖЦ:

- неполная определенность проектных спецификаций, описаний компонент / особенностей архитектуры и т.п.;
- циклическое/итеративное повторение ряда фаз ЖЦ;
- «классическая» разработка ПО – SADT – не учитывает:
  - неструктурное проектирование;
  - специфику компонент (модулей) кода и выбор ЯП до завершения построения проектных спецификаций;
  - повторное использование кода
    (как в OOAD и OO-модели)

Границы фаз могут изменяться (в т.ч. динамически) в зависимости от модели ЖЦ и подхода к реализации

Вклад фаз ЖЦ в сроки и стоимость проекта:

Фаза	Стоимость	Сроки
Требования (со спецификациями)	2%	21% / 18%
Спецификации	5%	см. «Требования»
Проектирование	6%	18% / 19%
Кодирование (с тестированием)	5%	36% / 34%
Тестирование	7%	см. «Кодирование»
Интеграция	8%	24% / 29%
Сопровождение	**67%**	**---**

Выводы (1):

- Основные затраты – на **сопровождение** (особенно для долгосрочных, многокомпонентных проектов)
- Средства, увеличивающие расширяемость ПО

(и/или сроки обновления и тестирования) более эффективны, чем все ухищрения при кодировании

- Фазы, предшествующие кодированию и следующие за ним, составляют 30% затрат (кодирование – 5%):
  - «обрамляющие» стадии улучшают качество ПО;
  - «обрамляющие» стадии ускоряют кодирование
- Большинство ошибок – в ходе проектирования и

спецификаций (нужны формальные методы анализа)

Выводы (2):

- Цена поиска ошибок **экспоненциально** растет по мере продвижения проекта к завершению
- **Ошибки нужно обнаруживать как можно раньше!**
  (иначе придется изменять версию ПО, документации, …)
- Существуют специальные методы поиска ошибок
- Каждая фаза ЖЦ ПО включает:
  - **Процессы** – различные и независимые задачи;
  - **Методы** – описания каждой из задач процесса
  - **Средства** – (полу)автоматические инструменты для поддержки процессов и методов

Виды моделей ЖЦ:

- Модель Build-and-Fix
- Водопадная модель
- Модель быстрого прототипирования
- Инкрементная модель
- Модель синхронизации и стабилизации
- Спиральная модель
- ОО-модель

Модели ЖЦ ПО: Общие черты – как правило:

- включают все стадии ЖЦ ПО (кроме Build-and-Fix)
- предполагают несколько итераций по разработке проекта
- стадии ЖЦ ПО четко различимы (кроме ОО)
- связаны с методологиями проектирования (синхронизации и стабилизации - MSF, каскадная и спиральная - RUP и т.д.)
- требуют высокой организационной зрелости команды разработчиков и дисциплины проекта (при их недостатке ОО-модель может выродиться в САВТАВ)
- **Нет универсальной модели!**
- **Модели можно комбинировать!**
- **Все модели имеют преимущества и недостатки!**

Модели ЖЦ ПО: Определяют:

- Характер и масштаб проекта (объем, сроки, риски, …)
- Экономику проекта (в т.ч. ROI)
- Степень сопровождаемости
- Перспективы развития (прогноз запросов клиента)
- Архитектуру проекта (стабильная эволюция, революционные усовершенствования)
- Скорость поиска и устранения ошибок
- Управление рисками проекта
- Степень полноты реализации (прототип, промежуточное решение, готовый продукт)

Модели ЖЦ ПО: Особенности

- **Build-and-Fix** – неполный ЖЦ ПО, малые проекты
- **Водопадная** – обратная связь с ранними стадиями ЖЦ ПО
- **Быстрое прототипирование** – не самостоятельная
- **Инкрементная** – всегда имеется готовый продукт
- **Синхростабилизации** – ранее выявление ошибок
- **Спиральная** – несколько итераций с анализом рисков
- **ОО** – перекрытие фаз ЖЦ ПО с итеративными возвратами

Модели ЖЦ ПО: сравнительный анализ и выводы

- Цена поиска ошибок **экспоненциально** растет по мере продвижения проекта к завершению
- **Ошибки нужно обнаруживать как можно раньше!**
  (иначе придется изменять версию ПО, документации, …)
- Существуют специальные методы поиска ошибок
- Каждая фаза ЖЦ ПО включает:
  - **Процессы** – различные и независимые задачи;
  - **Методы** – описания каждой из задач процесса
  - **Средства** – (полу)автоматические инструменты для поддержки процессов и методов

Модели ЖЦ ПО: сравнительный анализ и выводы

Модель ЖЦ	Преимущества	Недостатки
Build-and-Fix	Хороша для небольших, не требующих сопровождения проектов	Абсолютно непригодна для нетривиальных проектов
Водопадная	Четкая дисциплина проекта, документно-управляемая	ПО может не соответствовать требованиям клиента
Быстрого прототипирования	Обеспечивает соответствие ПО требованиям клиента	Вызывает соблазн повторного использования кода, который следует заново реализовать
Инкрементная	Максимально ранний возврат инвестиций; способствует сопровождаемости	Требует открытой архитектуры; может выродиться в Build-and-fix
Синхронизации и стабилизации	Удовлетворяет будущим потребностям клиента; обеспечивает интеграцию компонент	Не получила широкого применения вне Microsoft Пригодна лишь для крупных внутренних проектов; разработчики должны владеть управлением рисками
Спиральная	Объединяет хар-ки всех перечисленных выше моделей	
ОО-модель	Обеспечивает итерацию внутри фаз и параллелизм между фазами	Может выродиться в CAB TAB

На что влияет выбор ЖЦ?

- скорость разработки (время выхода на рынок);
- качество продукта;
- стоимость продукта;
- стратегию управления изменениями;
- стратегию управления рисками;
- отношения с заказчиками
- и т.д.

Модели ЖЦ ПО: Выводы:

- **Выбор модели определяет успех проекта!**
- **Модель определяет архитектуру проекта!**
- **Модель определяет экономику проекта!**
- **Модель д.б. адекватна опыту проектной команды!**
- **Серьезные модели требуют дисциплины и зрелости!**
- **Нет универсальной модели!**
- **Модели можно комбинировать!**
- **Все модели имеют преимущества и недостатки!**
- **Преимущества и недостатки имеют смысл только в контексте проекта!**

CASE-технологии: основы

- ПО «в малом» - кодирование модулей
- ПО «в большом» = SE
- ПО «в массе» - командная работа
- **CASE-технологии помогают во всех трех аспектах**

Типы CASE-средств:

- Верхнего уровня (**front-end**) – для требований, спецификаций и проектирования
- Нижнего уровня (**back-end**) – для реализации, внедрения и сопровождения
- Конвейеры (workbench) и среды CASE – наборы небольшого кол-ва инструментов для связанных операций (компиляция, редактирование, сборка, отладка)

CASE-технологии: Первые выводы:

Необходимые условия применения:

- Организационная зрелость
- Серьезный масштаб проекта

Результаты успешного применения:

- Рост производительности труда
- Снижение времени и стоимости проекта

Метрики ЖЦ ПО:

- Проект в целом: сроки – стоимость – функциональность, Cost-Benefit анализ
- Тестирование – сложность модуля/кода (кол-во строк KLOC, кол-во (различных) операторов/операндов, относительная ошибка – кол-во ошибок на KLOC)
- Сопровождение – отслеж./испр.ошибок (общее кол-во сбоев, классификация и состояние сбоев/отчетов, метрики предыдущих стадий)

Выводы:

- решение – за PM;
- простых метрик на базе KLOC, как правило, достаточно