Технологии программирования. Компонентный подход

В. В. Кулямин

Лекция 2. Жизненный цикл и процессы разработки ПО

Аннотация

Вводятся понятия жизненного цикла ПО и технологических процессов его разработки. Рассматриваются различные способы организации жизненного цикла ПО, каскадные и итеративные модели жизненного цикла, а также набор стандартов, регулирующих процессы разработки ПО в целом.

Ключевые слова

Жизненный цикл ПО, виды деятельности, роли заинтересованных лиц, процессы жизненного цикла, процесс разработки ПО, стандарты жизненного цикла ПО, модель зрелости возможностей, модели жизненного цикла ПО, каскадная модель жизненного цикла, итеративная модель жизненного цикла, спиральная модель жизненного цикла.

Текст лекции

Понятие жизненного цикла ПО

В первой лекции говорилось о том, что сложную программную систему построить «простыми» методами невозможно. Ее разработка с неизбежностью будет тоже сложной деятельностью.

Привести такое предприятие к успеху возможно на основе общих принципов работы со сложными системами: организовав его в виде набора модулей, используя разные уровни абстракции, переиспользуя отдельные элементы в разных местах так, чтобы изменения в таком элементе автоматически отражались всюду, где он используется.

Разработка ПО — разновидность человеческой деятельности. Выделить ее компоненты можно, определив набор задач, которые нужно решить для достижения конечной цели — построения достаточно качественной системы в рамках заданных сроков и ресурсов. Для решения каждой такой задачи организуется вспомогательная деятельность, к которой можно также применить декомпозицию на отдельные, более мелкие деятельности, и т.д. В итоге должно стать понятно, как решать каждую отдельную подзадачу и всю задачу целиком на основе имеющихся решений для подзадач.

В качестве примеров деятельностей, которые нужно проводить для построения программной системы, можно привести ***проектирование*** — выделение отдельных модулей и определение связей между ними с целью минимизации зависимостей между частями проекта и достижения лучшей его обозримости в целом, ***кодирование*** — разработку кода отдельных модулей, разработку пользовательской документации, которая необходима для достаточно сложной системы.

Однако для корректного с точки зрения инженерии и экономики рассмотрения вопросов создания сложных систем необходимо, чтобы были затронуты и вопросы эксплуатации системы, внесения в нее изменений, а также самые первые действия в ходе ее создания — анализ потребностей пользователей и выработка решений, «изобретение» функций, удовлетворяющих эти потребности. Без этого невозможно, с одной стороны, учесть реальную эффективность системы в виде отношения полученных результатов ко всем сделанным затратам и, с другой стороны, правильно оценивать в ходе разработки степень соответствия системы реальным нуждам пользователей и заказчиков.

Все эти факторы приводят к необходимости рассмотрения всей совокупности деятельностей, связанных с созданием и использованием ПО, начиная с возникновения идеи о новом продукте и заканчивая удалением его последней копии. Весь период существования ПО, связанный с подготовкой к его разработке, разработкой, использованием и модификациями, начиная с того момента, когда принимается решение разработать/приобрести/собрать из имеющихся

компонентов новую систему или приходит сама идея о необходимости программы определенного рода, до того момента, когда полностью прекращается всякое ее использование, называют ***жизненным циклом ПО***.

В ходе жизненного цикла ПО проходит через анализ предметной области, сбор требований, проектирование, кодирование, тестирование, сопровождение и др. ***виды деятельности***. Каждый вид представляет собой достаточно однородный набор действий, выполняемых для решения одной задачи или группы тесно связанных задач в рамках разработки и поддержки эксплуатации ПО.

При этом создаются и перерабатываются различного рода ***артефакты*** — создаваемые человеком информационные сущности, документы в достаточно общем смысле, участвующие в качестве входных данных и получающиеся в качестве результатов различных деятельностей. Примерами артефактов являются: модель предметной области, описание требований, техническое задание, архитектура системы, проектная документация на систему в целом и на ее компоненты, прототипы системы и компонентов, собственно, исходный код, пользовательская документация, документация администратора системы, руководство по развертыванию, база пользовательских запросов, план проекта, и пр.

На различных этапах в создание и эксплуатацию ПО вовлекаются люди, выполняющие различные ***роли***. Каждая роль может быть охарактеризована как абстрактная группа заинтересованных лиц, участвующих в деятельности по созданию и эксплуатации системы и решающих одни и те же задачи или имеющих одни и те же интересы по отношению к ней. Примерами ролей являются: бизнес-аналитик, инженер по требованиям, архитектор, проектировщик пользовательского интерфейса, программист-кодировщик, технический писатель, тестировщик, руководитель проекта по разработке, работник отдела продаж, конечный пользователь, администратор системы, инженер по поддержке и т.п.

Похоже, что общую структуру жизненного цикла любого ПО задать невозможно, поскольку она существенно зависит от целей, для которых это ПО разрабатывается или приобретается, и от решаемых им задач. Структура жизненного цикла будет существенно разной у программы для форматирования кода, которая сначала делалась программистом для себя, а впоследствии была признана перспективной в качестве продукта и переработана, и у комплексной системы автоматизации предприятия, которая с самого начала задумывалась как таковая. Тем не менее, часто определяют основные элементы структуры жизненного цикла в виде ***модели жизненного цикла ПО***. Модель жизненного цикла ПО выделяет конкретные наборы видов деятельности (обычно разбиваемых на еще более мелкие активности), артефактов, ролей и их взаимосвязи, а также дает рекомендации по организации процесса в целом. Эти рекомендации включают ответы на вопросы о том, какие артефакты являются входными данными у каких видов деятельности, а какие появляются в качестве результатов, какие роли вовлечены в различные деятельности, как различные деятельности связаны друг с другом, каковы критерии качества полученных результатов, как оценить степень соответствия различных артефактов общим задачам проекта и когда можно переходить от одной деятельности к другой.

Жизненный цикл ПО является составной частью жизненного цикла программно-аппаратной системы, в которую это ПО входит. Поэтому часто различные его аспекты рассматриваются в связи с элементами жизненного цикла системы в целом.

Существует набор стандартов, определяющих различные элементы в структуре жизненных циклов ПО и программно-аппаратных систем. В качестве основных таких элементов выделяют ***технологические процессы*** — структурированные наборы деятельностей, решающих некоторую общую задачу или связанную совокупность задач, такие, как процесс сопровождения ПО, процесс обеспечения качества, процесс разработки документации и пр. Процессы могут определять разные этапы жизненного цикла и увязывать их с различными видами деятельностей, артефактами и ролями заинтересованных лиц.

Стоит отметить, что процессом (или технологическим процессом) называют и набор процессов, увязанных для совместного решения более крупной задачи, например, всю совокупность деятельностей, входящих в жизненный цикл ПО. Таким образом, процессы могут

разбиваться на подпроцессы, решающие частные подзадачи той задачи, с которой работает общий процесс.

Стандарты жизненного цикла

Чтобы получить представление о возможной структуре жизненного цикла ПО, обратимся сначала к соответствующим стандартам, описывающим технологические процессы. Международными организациями, такими, как:

- IEEE — читается «ай-трипл-и», Institute of Electrical and Electronic Engineers, Институт инженеров по электротехнике и электронике;
- ISO — International Standards Organization, Международная организация по стандартизации;
- EIA — Electronic Industry Association, Ассоциация электронной промышленности;
- IEC — International Electrotechnical Commission, Международная комиссия по электротехнике;

а также некоторыми национальными и региональными институтами и организациями (в основном, американскими и европейскими, поскольку именно они оказывают наибольшее влияние на развитие технологий разработки ПО во всем мире):

- ANSI — American National Standards Institute, Американский национальный институт стандартов;
- SEI — Software Engineering Institute, Институт программной инженерии;
- ECMA — European Computer Manufactures Association, Европейская ассоциация производителей компьютерного оборудования;

разработан набор стандартов, регламентирующих различные аспекты жизненного цикла и вовлеченных в него процессов. Список и общее содержание этих стандартов представлены ниже.

Группа стандартов ISO

- **ISO/IEC 12207 Standard for Information Technology — Software Life Cycle Processes** [1] (процессы жизненного цикла ПО, есть его российский аналог **ГОСТ Р-1999** [2]). Определяет общую структуру жизненного цикла ПО в виде 3-х ступенчатой модели, состоящей из процессов, видов деятельности и задач. Стандарт описывает вводимые элементы в терминах их целей и результатов, тем самым задавая неявно возможные взаимосвязи между ними, но не определяя четко структуру этих связей, возможную организацию элементов в рамках проекта и метрики, по которым можно было бы отслеживать ход работ и их результативность.

  Самыми крупными элементами являются *процессы жизненного цикла ПО (lifecycle processes).* Всего выделено 18 процессов, которые объединены в 4 группы.

Основные процессы	**Поддерживающие процессы**	**Организационные процессы**	**Адаптация**
Приобретение ПО; Передача ПО (в использование); Разработка ПО; Эксплуатация ПО; Поддержка ПО	Документирование; Управление конфигурациями; Обеспечение качества; Верификация; Валидация; Совместные экспертизы; Аудит; Разрешение проблем	Управление проектом; Управление инфраструктурой; Усовершенствование процессов; Управление персоналом	Адаптация описываемых стандартом процессов под нужды конкретного проекта

Таблица 1. Процессы жизненного цикла ПО по ISO 12207.

  Процессы строятся из отдельных *видов деятельности (activities)*.
  Стандартом определены 74 вида деятельности, связанной с разработкой и поддержкой ПО. Здесь мы упомянем только некоторые из них.

- Приобретение ПО включает такие деятельности, как инициация приобретения, подготовка запроса предложений, подготовка контракта, анализ поставщиков, получение ПО и завершение приобретения.
- Разработка ПО включает развертывание процесса разработки, анализ системных требований, проектирование (программно-аппаратной) системы в целом, анализ требований к ПО, проектирование архитектуры ПО, детальное проектирование, кодирование и отладочное тестирование, интеграцию ПО, квалификационное тестирование ПО, системную интеграцию, квалификационное тестирование системы, развертывание (установку или инсталляцию) ПО, поддержку процесса получения ПО.
- Поддержка ПО включает развертывание процесса поддержки, анализ возникающих проблем и необходимых изменений, внесение изменений, экспертизу и передачу измененного ПО, перенос ПО с одной платформы на другую, изъятие ПО из эксплуатации.
- Управление проектом включает запуск проекта и определение его рамок, планирование, выполнение проекта и надзор за его выполнением, экспертизу и оценку проекта, свертывание проекта.

Каждый вид деятельности нацелен на решение одной или нескольких *задач (tasks)*. Всего определено 224 различные задачи. Например:

- Развертывание процесса разработки состоит из определения модели жизненного цикла, документирования и контроля результатов отдельных работ, выбора используемых стандартов, языков и инструментов и пр.
- Перенос ПО между платформами состоит из разработки плана переноса, оповещения пользователей, выполнения анализа произведенных действий и пр.

- **ISO/IEC 15288 Standard for Systems Engineering — System Life Cycle Processes** [3] (процессы жизненного цикла систем).
  Отличается от предыдущего нацеленностью на рассмотрение программно-аппаратных систем в целом.
  В данный момент продолжается работа по приведению этого стандарта в соответствие с предыдущим.
  ISO/IEC 15288 предлагает похожую схему рассмотрения жизненного цикла системы в виде набора процессов. Каждый процесс описывается набором его *результатов (outcomes)*, которые достигаются при помощи различных видов деятельности.

  Всего выделено 26 процессов, объединяемых в 5 групп.

Процессы выработки соглашений	**Процессы уровня организации**	**Процессы уровня проекта**	**Технические процессы**	**Специальные процессы**
Приобретение системы; Поставка системы	Управление окружением; Управление инвестициями; Управление процессами; Управление ресурсами; Управление качеством	Планирование; Оценивание; Мониторинг; Управление рисками; Управление конфигурацией; Управление информацией; Выработка решений	Определение требований; Анализ требований; Проектирование архитектуры; Реализация; Интеграция; Верификация; Валидация; Передача в использование; Эксплуатация; Поддержка; Изъятие из эксплуатации	Адаптация описываемых стандартом процессов под нужды конкретного проекта

Таблица 2. Процессы жизненного цикла систем по ISO 15288.

Помимо процессов, определено 123 различных результата и 208 видов деятельности, нацеленных на их достижение. Например, определение требований имеет следующие результаты.

- Должны быть поставлены технические задачи, которые предстоит решить.
- Должны быть сформулированы системные требования.

Деятельности в рамках этого процесса следующие.

- Определение границ функциональности системы.
- Определение функций, которые необходимо поддерживать.
- Определение критериев оценки качества при использовании системы.
- Анализ и выделение требований по безопасности.
- Анализ требований защищенности.
- Выделение критических для данной системы аспектов качества и требований к ним.
- Анализ целостности системных требований.
- Демонстрация прослеживаемости требований.
- Фиксация и поддержка набора системных требований.

- **ISO/IEC 15504 (SPICE) Standard for Information Technology — Software Process Assessment** [4] (оценка процессов разработки и поддержки ПО).
Определяет правила оценки процессов жизненного цикла ПО и их возможностей, опирается на модель CMMI (см. ниже) и больше ориентирован на оценку процессов и возможностей их улучшения.
В качестве основы для оценки процессов определяет некоторую базовую модель, аналогичную двум описанным выше. В ней выделены категории процессов, процессы и виды деятельности.

  Определяются 5 категорий, включающих 35 процессов и 201 вид деятельности.

Отношения заказчик-поставщик	**Процессы уровня организации**	**Процессы уровня проекта**	**Инженерные процессы**	**Процессы поддержки**
Приобретение ПО; Составление контракта; Определение нужд заказчика; Проведение совместных экспертиз и аудитов; Подготовка к передаче; Поставка и развертывание; Поддержка эксплуатации; Предоставление услуг; Оценка удовлетворенности заказчиков	Развитие бизнеса; Определение процессов; Усовершенствование процессов; Обучение; Обеспечение переиспользования; Обеспечение инструментами; Обеспечение среды для работы	Планирование жизненного цикла; Планирование проекта; Построение команды; Управление требованиями; Управление качеством; Управление рисками; Управление ресурсами и графиком работ; Управление подрядчиками	Выделение системных требований и проектирование системы в целом; Выделение требований к ПО; Проектирование ПО; Реализация, интеграция и тестирование ПО; Интеграция и тестирование системы; Сопровождение системы и ПО	Разработка документации; Управление конфигурацией; Обеспечение качества; Разрешение проблем; Проведение экспертиз

Таблица 3. Процессы жизненного цикла ПО и систем по ISO 15504.

Например, приобретение ПО включает такие виды деятельности, как определение потребности в ПО, определение требований, подготовку стратегии покупки, подготовку запроса предложений, выбор поставщика.

Группа стандартов IEEE

- **IEEE 1074-1997 — IEEE Standard for Developing Software Life Cycle Processes** [5] (стандарт на создание процессов жизненного цикла ПО).
  Нацелен на описание того, как создать специализированный процесс разработки в рамках конкретного проекта. Описывает ограничения, которым должен удовлетворять любой такой процесс, и, в частности, общую структуру процесса разработки. В рамках этой структуры определяет основные виды деятельностей, выполняемых в этих процессах и документы, требующиеся на входе и возникающие на выходе этих деятельностей. Всего рассматриваются 5 подпроцессов, 17 групп деятельностей и 65 видов деятельности. Например, подпроцесс разработки состоит из групп деятельностей по выделению требований, по проектированию и по реализации. Группа деятельностей по проектированию включает архитектурное проектирование, проектирование баз данных, проектирование интерфейсов, детальное проектирование компонентов.
- **IEEE/EIA 12207-1997 — IEEE/EIA Standard: Industry Implementation of International Standard ISO/IEC 12207:1995 Software Life Cycle Processes** [6-8] (промышленное использование стандарта ISO/IEC 12207 на процессы жизненного цикла ПО).
  Аналог ISO/IEC 12207, сменил ранее использовавшиеся стандарты J-Std-016-1995 EIA/IEEE Interim Standard for Information Technology — Software Life Cycle Processes — Software Development Acquirer-Supplier Agreement (промежуточный стандарт на процессы жизненного цикла ПО и соглашения между поставщиком и заказчикам ПО) и стандарт министерства обороны США MIL-STD-498.

Группа стандартов CMM, разработанных SEI

- **Модель зрелости возможностей CMM (Capability Maturity Model)** [9,10] предлагает унифицированный подход к оценке возможностей организации выполнять задачи различного уровня. Для этого определяются 3 уровня элементов: *уровни зрелости организации (maturity levels)*, *ключевые области процесса (key process areas)* и *ключевые практики (key practices)*. Чаще всего под моделью CMM имеют в виду модель уровней зрелости. В настоящий момент CMM считается устаревающей и сменяется моделью CMMI (см. ниже).
  - Уровни зрелости.
    CMM описывает различные степени зрелости процессов в организациях, определяя 5 уровней организаций.
    - Уровень 1, начальный (initial).
      Организации, разрабатывающие ПО, но не имеющие осознанного процесса разработки, не производящие планирования и оценок своих возможностей.
    - Уровень 2, повторяемый (repeatable).
      В таких организациях ведется учет затрат ресурсов и отслеживается ход проектов, установлены правила управления проектами, основанные на имеющемся опыте.
    - Уровень 3, определенный (defined).
      В таких организациях имеется принятый, полностью документированный, соответствующий реальному положению дел и доступный персоналу процесс разработки и сопровождения ПО. Он должен включать как управленческие, так и технические подпроцессы, а также обучение сотрудников работе с ним.
    - Уровень 4, управляемый (manageable).
      В этих организациях, помимо установленного и описанного процесса, используются измеримые показатели качества продуктов и результативности процессов, которые позволяют достаточно точно предсказывать объем ресурсов (времени, денег, персонала), необходимый для разработки продукта с определенным качеством.
    - Уровень 5, совершенствующийся (optimizing).
      В таких организациях, помимо процессов и методов их оценки, имеются методы определения слабых мест, определены процедуры поиска и оценки новых методов и

техник разработки, обучения персонала работе с ними и их включения в общий процесс организации в случае повышения эффективности производства.

- Ключевые области процесса.
  Согласно CMM, уровни зрелости организации можно определять по использованию четко определенных техник и процедур, относящихся к различным ключевым областям процесса. Каждая такая область представляет собой набор связанных видов деятельности, которые нацелены на достижение целей, существенных для общей оценки результативности технологического процесса. Всего выделяется 18 областей. Множество областей, которые должны поддерживаться организацией, расширяется при переходе к более высоким уровням зрелости.
  - К первому уровню не предъявляется никаких требований.
  - Организации второго уровня зрелости должны поддерживать управление требованиями, планирование проектов, надзор за ходом проекта, управление подрядчиками, обеспечение качества ПО, управление конфигурацией.
  - Организации третьего уровня должны, помимо деятельностей второго уровня, поддерживать проведение экспертиз, координацию деятельности отдельных групп, разработку программного продукта, интегрированное управление разработкой и сопровождением, обучение персонала, выработку и поддержку технологического процесса организации, контроль соблюдения технологического процесса организации.
  - К деятельностям организаций четвертого уровня добавляются: управление качеством ПО и управление процессом, основанное на измеримых показателях.
  - Организации пятого уровня зрелости должны дополнительно поддерживать управление изменениями процесса, управление изменениями используемых технологий и предотвращение дефектов.
- Ключевые практики.
  Ключевые области процесса описываются с помощью наборов ключевых практик. Ключевые практики классифицированы на несколько видов: обязательства (commitments to perform), возможности (abilities to perform), деятельности (activities performed), измерения (measurements and analysis) и проверки (verifying implementations).
  Например, управление требованиями связано со следующими практиками.
  - Обязательство.
    Проекты должны следовать определенной политике организации по управлению требованиями.
  - Возможности.
    В каждом проекте должен определяться ответственный за анализ системных требований и привязку их к аппаратному, программному обеспечению и другим компонентам системы.
    Требования должны быть документированы.
    Для управления требованиями должны быть выделены адекватные ресурсы и бюджет.
    Персонал должен проходить обучение в области управления требованиями.
  - Деятельности.
    Прежде чем быть включенными в проект, требования подвергаются анализу на полноту, адекватность, непротиворечивость и пр.
    Выделенные требования используются в качестве основы для планирования и выполнения других работ.
    Изменения в требованиях анализируются и включаются в проект.
  - Измерение.
    Производится периодическое определение статуса требований и статуса деятельности по управлению ими.

- Проверки.
  Деятельность по управлению требованиями периодически анализируется старшими менеджерами.
  Деятельность по управлению требованиями периодически и на основании значимых событий анализируется менеджером проекта.
  Группа обеспечения качества проводит анализ и аудит деятельности по управлению требованиями и отчитывается по результатам этого анализа.

Таблица 4 суммирует информацию о количестве практик различных видов, приписанных к разным ключевым областям процесса.

Уровни	Область процесса	Обязательства	Возможности	Деятельности	Измерения	Проверки
2	Управление требованиями	1	4	3	1	3
	Планирование проектов	2	4	15	1	3
	Надзор за ходом проекта	2	5	13	1	3
	Управление подрядчиками	2	3	13	1	3
	Обеспечение качества ПО	1	4	8	1	3
	Управление конфигурацией	1	5	10	1	4
3	Контроль соблюдения технологического процесса	3	4	7	1	1
	Выработка и поддержка технологического процесса	1	2	6	1	1
	Обучение персонала	1	4	6	2	3
	Интегрированное управление	1	3	11	1	3
	Разработка программного продукта	1	4	10	2	3
	Координация деятельности групп	1	5	7	1	3
	Проведение экспертиз	1	3	3	1	1
4	Управление процессом на основе метрик	2	5	7	1	3
	Управление качеством ПО	1	3	5	1	3
5	Предотвращение дефектов	2	4	8	1	3
	Управление изменениями технологий	3	5	8	1	2
	Управление изменениями процесса	2	4	10	1	2

Таблица 4. Количество ключевых практик в разных областях процесса по CMM версии 1.1.

- **Интегрированная модель зрелости возможностей CMMI (Capability Maturity Model Integration)** [11,12].
  Эта модель представляет собой результат интеграции моделей CMM для продуктов и процессов, а также для разработки ПО и разработки программно-аппаратных систем. Основные изменения по сравнению с CMM следующие.
  - Созданы два несколько отличающихся изложения модели — непрерывное и поэтапное. Первое предназначено для облегчения миграции от поддержки американского отраслевого стандарта EIA/AIS 713 и постепенного усовершенствования процессов за счет внедрения различных практик. Второе предназначено для облегчения миграции от поддержки CMM и поуровневого рассмотрения вводимых практик.
  - Элементы модели получили четкие пометки о том, являются ли они обязательными (required), рекомендуемыми (expected) или информативными (informative).

- Используемые практики разделяются на общие (generic) и специфические (specific). Они дополняются набором общих и специфических целей, которые необходимы для достижения определенного уровня зрелости в определенных областях процесса.
- Некоторые уровни зрелости получили другие названия. Второй уровень назван управляемым (managed), а четвертый — управляемым на основе метрик (quantitatively managed).
- Набор выделяемых областей процесса и практик значительно изменился.
  Все области процесса делятся на 4 категории. В приводимом ниже списке области процесса помечены номером уровня, начиная с которого они должны поддерживаться согласно CMMI.
  - Управление процессом.
    Включает выработку и поддержку процесса (3), контроль соблюдения процесса (3), обучение (3), измерение показателей процесса (4), внедрение инноваций (5).
  - Управление проектом.
    Включает планирование проектов (2), контроль хода проекта (2), управление соглашениями с поставщиками (2), интегрированное управление проектами (3), управление рисками (3), построение команд (3), управление поставщиками (3) и измерение показателей результативности и хода проекта (4).
  - Технические.
    Включают выработку требований (3), управление требованиями (2), выработку технических решений (3), интеграцию продуктов (3), верификацию (3) и валидацию (3).
  - Поддерживающие.
    Включают управление конфигурацией (2), обеспечение качества продуктов и процессов (2), проведение измерений и анализ их результатов (2), управление окружением (3), анализ и принятие решений (3), анализ, разрешение и предотвращение проблем (5).

В целом перечисленные стандарты связаны так, как показано на Рис. 2 (сплошные стрелки указывают направления исторического развития, жирная стрелка обозначает идентичность, пунктирные стрелки показывают влияние одних стандартов на другие).

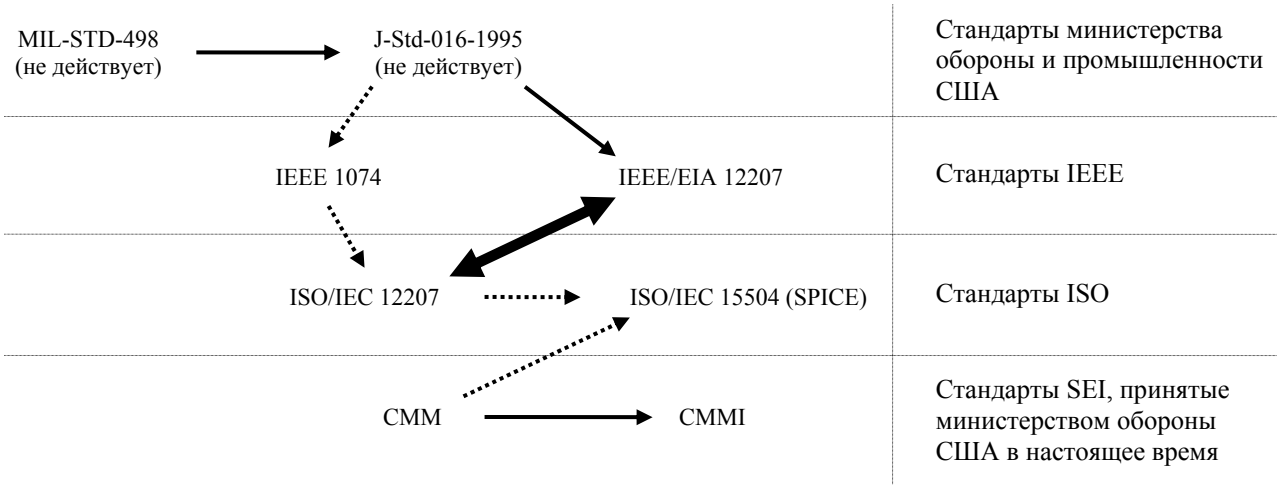

Рисунок 2. Стандарты, описывающие структуру жизненного цикла ПО.

Стандарты являются суммой опыта, который был накоплен экспертами в инженерии ПО на основе огромного количества проектов, проводившихся в рамках коммерческих структур США и Европы и в рамках военных контрактов. Б***о***льшая часть стандартов создавалась как набор критериев отбора поставщиков программного обеспечения для министерства обороны США, и эту задачу они решают достаточно успешно.

Все рассмотренные стандарты определяют некоторый набор видов деятельности, из которых должен состоять процесс разработки, и задают ту или иную структуру на этих видах деятельности,

выделяя их элементы. Вместе с тем, как можно заметить, они не могут быть сведены без существенных изменений в единую модель жизненного цикла ПО. В целом имеющиеся стандарты слабо согласованы между собой. Так что на сегодняшний день (2005 год) нет согласованного комплекта стандартов, покрывающего всю данную область, и в ближайшие несколько лет он вряд ли появится, хотя работа по согласованию различных стандартов ведется.

Кроме того, данные стандарты не предписывают четких и однозначных схем построения жизненного цикла ПО, в частности, связей между отдельными деятельностями. Это сделано намеренно, поскольку ранее действовавшие стандарты типа DoD-Std-2167, были достаточно жестко привязаны к каскадной модели жизненного цикла (см. ниже) и тем самым препятствовали использованию более прогрессивных технологий разработки. Современные стандарты стараются максимально общим образом определить набор видов деятельности, которые должны быть представлены в рамках жизненного цикла (с учетом целей отдельных проектов — т.е. проект, не стремящийся достичь каких-то целей, может не включать деятельностей, связанных с их достижением), и описать их при помощи наборов входных документов и результатов.

Стоит заметить, что стандарты могут достаточно сильно разойтись с реальной разработкой, если в ней используются новейшие методы автоматизации разработки и сопровождения ПО. Стандарты организаций ISO и IEEE построены на основе имеющегося эмпирического опыта разработки, полученного в рамках распространенных некоторое время назад парадигм и инструментальных средств. Это не значит, что они устарели, поскольку их авторы имеют достаточно хорошее представление о новых методах и технологиях разработки и пытались смотреть вперед. Но при использовании новаторских технологий в создании ПО часть требований стандарта может обеспечиваться совершенно иными способами, чем это предусмотрено в нем, а часть артефактов может отсутствовать в рамках данной технологии, исчезнув внутри автоматизированных процессов.

Модели жизненного цикла

Все обсуждаемые стандарты так или иначе пытаются описать, как должен выглядеть *любой* процесс разработки ПО. При этом они вынуждены вводить слишком общие модели жизненного цикла ПО, которые тяжело использовать при организации конкретного проекта.

В рамках специфических моделей жизненного цикла, которые предписывают правила организации разработки ПО в рамках данной отрасли или организации, определяются более конкретные процессы разработки. Отличаются они от стандартов, прежде всего, большей детальностью и четким описанием связей между отдельными видами деятельности, определением потоков данных (документов и артефактов) в ходе жизненного цикла. Таких моделей довольно много, ведь фактически каждый раз, когда некоторая организация определяет собственный процесс разработки, в качестве основы этого процесса разрабатывается некоторая модель жизненного цикла ПО. В рамках данной лекции мы рассмотрим лишь несколько моделей. К сожалению, очень тяжело выбрать критерии, по которым можно было бы дать хоть сколько-нибудь полезную классификацию известных моделей жизненного цикла.

Наиболее широко известной и применяемой долгое время оставалась так называемая ***каскадная*** или ***водопадная (waterfall)*** модель жизненного цикла, которая, как считается, была впервые четко сформулирована в работе [13] и впоследствии запечатлена в стандартах министерства обороны США в семидесятых-восьмидесятых годах XX века. Эта модель предполагает последовательное выполнение различных видов деятельности, начиная с выработки требований и заканчивая сопровождением, с четким определением границ между этапами, на которых набор документов, созданный на предыдущей стадии, передается в качестве входных данных для следующей. Таким образом, каждый вид деятельности выполняется на какой-то одной фазе жизненного цикла. Предлагаемая в статье [13] последовательность шагов разработки показана на Рис. 3. «Классическая» каскадная модель предполагает только движение вперед по этой схеме: все необходимое для проведения очередной деятельности должно быть подготовлено в ходе предшествующих работ.

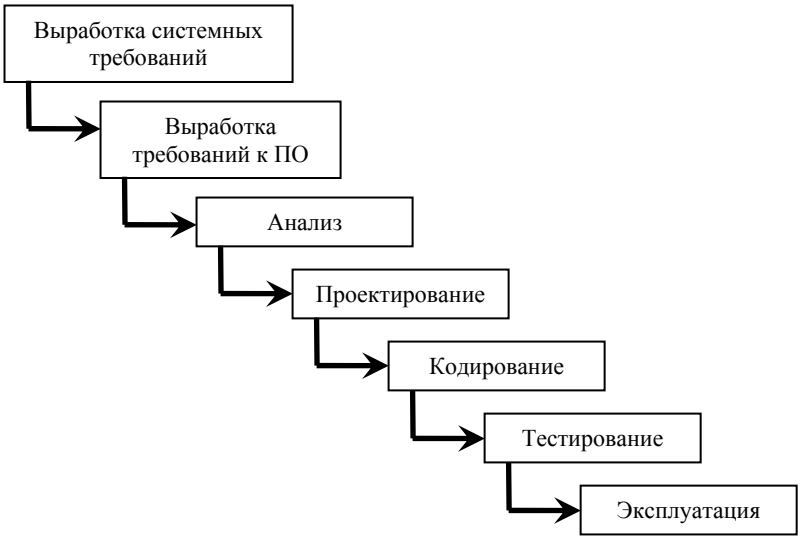

Рисунок 3. Последовательность разработки согласно «классической» каскадной модели.

Однако, если внимательно прочитать статью [13], оказывается, что она не предписывает следование именно этому порядку работ, а, скорее, представляет модель *итеративного процесса* (см. Рис. 4) — в ее последовательном виде эта модель закрепилась, по-видимому, в представлении чиновников из министерств и управленцев компаний, работающих с этими министерствами по контрактам. При реальной работе в соответствии с моделью, допускающей движение только в одну сторону, обычно возникают проблемы при обнаружении недоработок и ошибок, сделанных на ранних этапах. Но еще более тяжело иметь дело с изменениями окружения, в котором разрабатывается ПО (это могут быть изменения требований, смена подрядчиков, изменения политик разрабатывающей или эксплуатирующей организации, изменения отраслевых стандартов, появление конкурирующих продуктов и пр.).

Работать в соответствии с этой моделью можно, только если удается предвидеть заранее возможные перипетии хода проекта и тщательно собирать и интегрировать информацию на первых этапах, с тем, чтобы впоследствии можно было пользоваться их результатами без оглядки на возможные изменения.

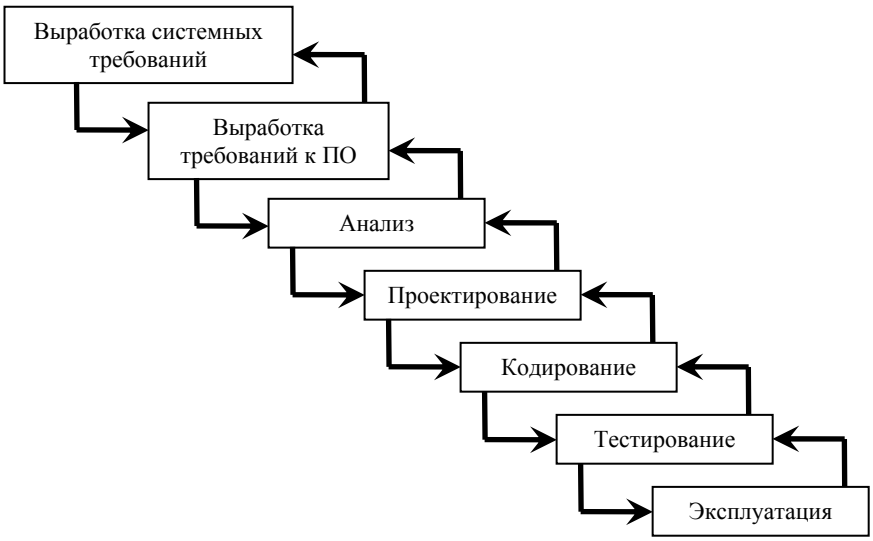

Рисунок 4. Ход разработки, предлагаемый в статье [13].

Среди разработчиков и исследователей, имевших дело с разработкой сложного ПО, практически с самого зарождения индустрии производства программ (см., например, [14]) б***о***льшую популярность имели модели *эволюционных* или *итеративных* процессов, поскольку они обладают большей гибкостью и способностью работать в меняющемся окружении.

Итеративные или ***инкрементальные модели*** (известно несколько таких моделей) предполагают разбиение создаваемой системы на набор кусков, которые разрабатываются с помощью нескольких последовательных проходов всех работ или их части.

На первой итерации разрабатывается кусок системы, не зависящий от других. При этом большая часть или даже полный цикл работ проходится на нем, затем оцениваются результаты и на следующей итерации либо первый кусок переделывается, либо разрабатывается следующий, который может зависеть от первого, либо как-то совмещается доработка первого куска с добавлением новых функций. В результате на каждой итерации можно анализировать промежуточные результаты работ и реакцию на них всех заинтересованных лиц, включая пользователей, и вносить корректирующие изменения на следующих итерациях. Каждая итерация может содержать полный набор видов деятельности от анализа требований, до ввода в эксплуатацию очередной части ПО.

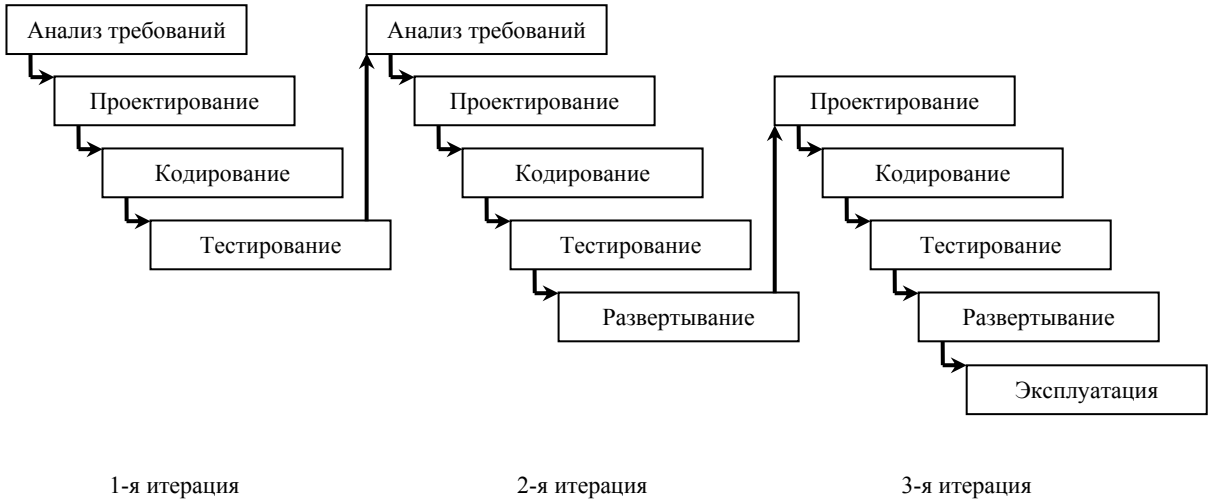

Рисунок 5. Возможный ход работ по итеративной модели.

Каскадная модель с возможностью возвращения на предшествующий шаг при необходимости пересмотреть его результаты, становится итеративной.

Итеративный процесс предполагает, что разные виды деятельности не привязаны намертво к определенным этапам разработки, а выполняются по мере необходимости, иногда повторяются, до тех пор, пока не будет получен нужный результат.

Вместе с гибкостью и возможностью быстро реагировать на изменения, итеративные модели привносят дополнительные сложности в управление проектом и отслеживание его хода. При использовании итеративного подхода значительно сложнее становится адекватно оценить текущее состояние проекта и спланировать долгосрочное развитие событий, а также предсказать сроки и ресурсы, необходимые для обеспечения определенного качества результата.

Развитием идеи итераций является ***спиральная*** модель жизненного цикла ПО, предложенная Боэмом (Boehm) в [15]. Она предлагает каждую итерацию начинать с выделения целей и планирования очередной итерации, определения основных альтернатив и ограничений при ее выполнении, их оценки, а также оценки возникающих рисков и определения способов избавления от них, а заканчивать итерацию оценкой результатов проведенных в ее рамках работ.

Основным ее новым элементом является общая структура действий на каждой итерации — планирование, определение задач, ограничений и вариантов решений, оценка предложенных решений и рисков, выполнение основных работ итерации и оценка их результатов.

Название спиральной эта модель получила из-за изображения хода работ в «полярных координатах», в которых угол соответствует выполняемому этапу в рамках общей структуры итераций, а удаление от начала координат — затраченным ресурсам.

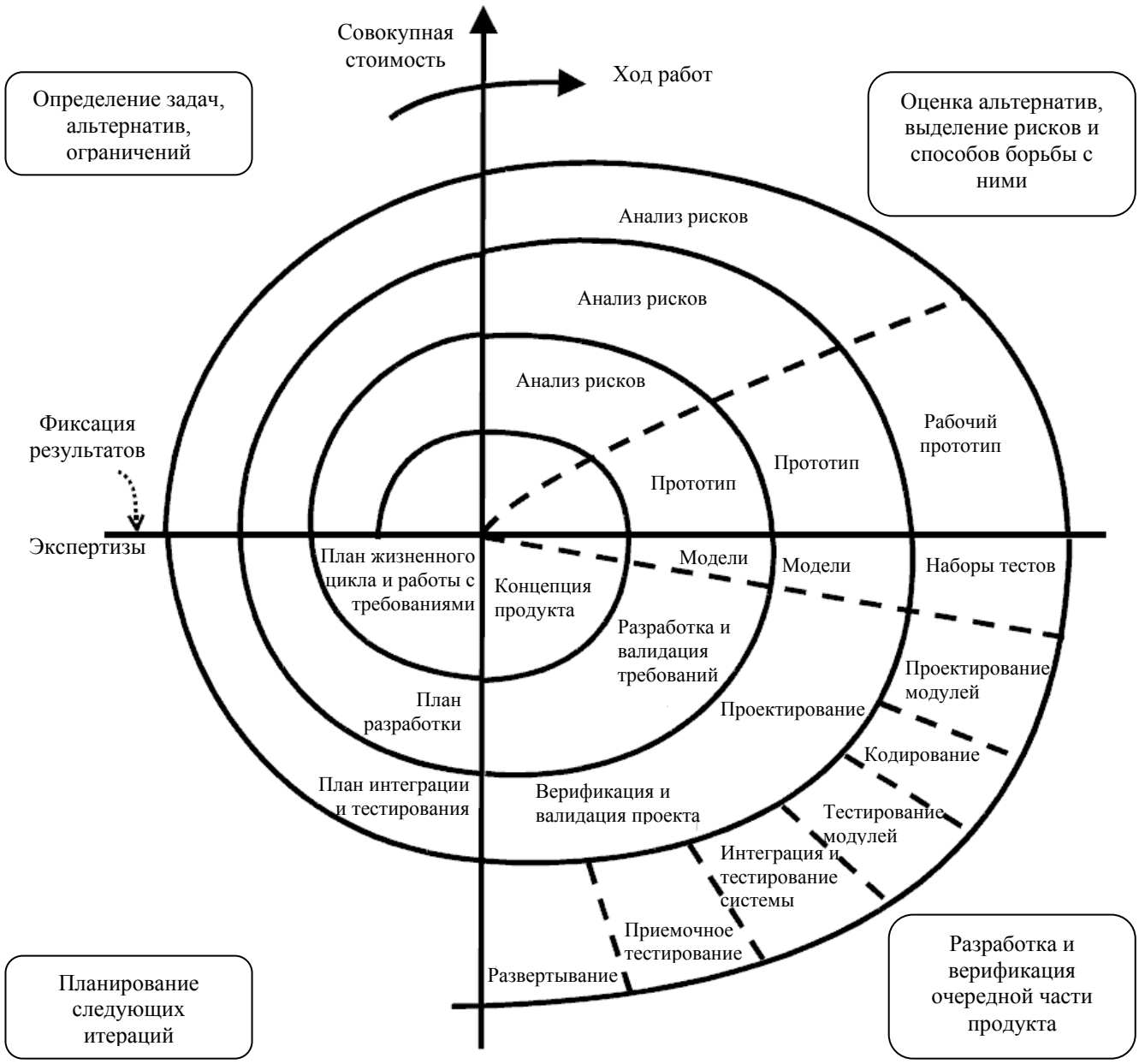

Рисунок 6. Изображение хода работ по спиральной модели согласно [15].

Рис. 6 показывает возможное развитие проекта по спиральной модели. Количество витков, а также расположение и набор видов деятельности в правом нижнем квадранте могут изменяться в зависимости от результатов планирования и анализа рисков, проводимых на предыдущих этапах.

На следующей лекции мы рассмотрим в деталях два современных итеративных процесса разработки — унифицированный процесс разработки Rational и экстремальное программирование.

Литература к Лекции 2

[1] ISO/IEC 12207:1995, Information Technology — Software life cycle processes, 1995. Amendments 2002, 2004.

[2] ГОСТ Р-1999. ИТ. Процессы жизненного цикла программных средств.

[3] ISO/IEC 15288:2002, Systems engineering — System life cycle processes, 2002.

[4] ISO/IEC 15504-1-9, Information technology — Process assessment, Parts 1-9. 15504-1,3,4:2004, 15504-2:2003/Cor 1:2004, TR 15504-5:2004.

[5] IEEE 1074-1997 IEEE Standard for Developing Software Life Cycle Processes, 1997.

[6] IEEE/EIA 12207.0-1996 Industry Implementation of International Standard ISO/IEC 12207:1995, New York, Mar. 1998.

[7] IEEE/EIA 12207.1-1997 Industry Implementation of International Standard ISO/IEC 12207:1995 Software Life Cycle Processes — Life Cycle Data, New York, Apr. 1998.

[8] IEEE/EIA 12207.2-1997 Industry Implementation of Int'l Standard ISO/IEC 12207:1995 Software Life Cycle Processes — Implementation Considerations, New York, Apr. 1998.
[9] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber. Capability Maturity Model for Software, Version 1.1, SEI Technical Report CMU/SEI-93-TR-024, Software Engineering Institute, Pittsburgh, Feb. 1993.
http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf
[10] M. C. Paulk, C. V. Weber, S. M. Garcia, M. B. Chrissis, and M. Bush. Key Practices of the Capability Maturity Model, Version 1.1, SEI Technical Report CMU/SEI-93-TR-025, Software Engineering Institute, Pittsburgh, Feb. 1993.
http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr25.93.pdf
[11] Capability Maturity Model Integration (CMMI), Version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1). Continuous Representation. SEI Technical Report CMU/SEI-2002-TR-011, Software Engineering Institute, Pittsburgh, March 2002.
http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr011.pdf
[12] Capability Maturity Model Integration (CMMI), Version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1). Staged Representation. SEI Technical Report CMU/SEI-2002-TR-012, Software Engineering Institute, Pittsburgh, March 2002.
http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr012.pdf
[13] W. W. Royce. Managing the Development of Large Software Systems. Proceedings of IEEE WESCON, pp. 1–9, August 1970.
Переиздана: Proceedings of the 9th International Software Engineering Conference, Computer Society Press, pp. 328–338, 1987.
[14] B. Randell, F. W. Zurcher. Iterative Multi-Level Modeling: A Methodology for Computer System Design. Proc. IFIP, IEEE CS Press, 1968.
[15] B. Boehm. A Spiral Model of Software Development and Enhancement. Computer, May 1988, pp. 61-72.
[16] И. Соммервилл. Инженерия программного обеспечения. М.: Вильямс, 2002.
[17] У. Ройс. Управление проектами по созданию программного обеспечения. М.: Лори, 2002.
[18] Э. Дж. Брауде. Технология разработки программного обеспечения. СПб.: Питер, 2004.