

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  информационных технологий | **Кафедра**  информационных систем |

**Реферат № 2**

по дисциплине «**Основы программной инженерии**»

на тему: «Жизненный цикл систем и программных средств в программной инженерии»

|  |  |  |
| --- | --- | --- |
| **Студент** группа ИДБ–22–06 |  | **Мустафаева П.М.** |
|  | подпись |  |
| **Руководитель**  к.э.н., доцент |  | **Ахмедов Э.Р.** |
|  | подпись |  |

**1. Что понимают под понятиями ЖЦ ПП, модели и методологии ЖЦ? Охарактеризуйте основные проблемы практического применения модели ЖЦ.**

**Жизненный цикл ПП** - полная совокупность всех процессов и действий по созданию и применению программного обеспечения, непрерывный процесс, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.

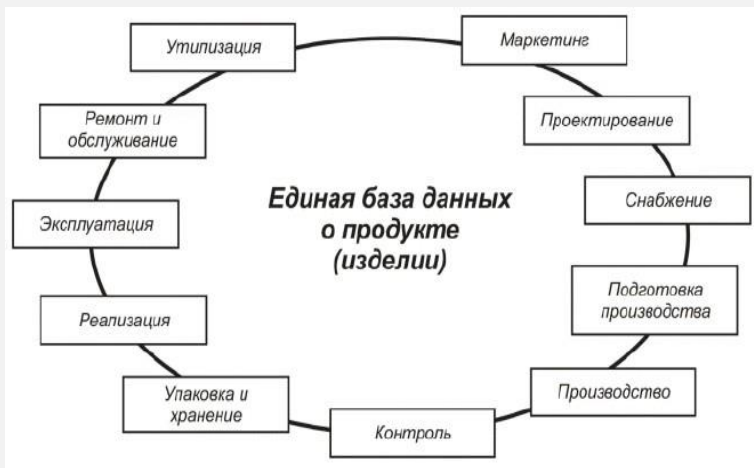


Рис. 1 Пример ЖЦ ПП

**Модель ЖЦ** - это схема выполнения работ и задач на процессах, обеспечивающих разработку, эксплуатацию и сопровождение программного продукта, отражающая жизнь ПП и включает в себя: разработку требований или технического задания; разработку системы или технического проекта; программирование или рабочее проектирование; пробную эксплуатацию; сопровождение и улучшение; снятие с эксплуатации.

**Методология ЖЦ** — это совокупность принципов, правил и методов, определяющих порядок организации и управления процессами разработки, тестирования, внедрения, сопровождения и вывода из эксплуатации программного обеспечения на каждом этапе его жизненного цикла. Методология регламентирует, как выполняются работы, взаимодействуют участники проекта и контролируется качество на всех стадиях разработки ПО.

**Основные проблемы практического применения модели ЖЦ:**

* Отсутствие единого мнения специалистов в вопросе определения объектов ЖЦ.
* Отсутствие точных определений начала и конца различных видов ЖЦ.
* Проблематичность определения длительности различных стадий ЖЦ и всего ЖЦ в целом.
* Невозможность однозначной идентификации конкретного этапа ЖЦ.
* Сложность определения момента наступления конкретного этапа ЖЦ.
* Сложность определения границ между стадиями ЖЦ.
* Вариация продолжительности жизненного цикла разных продуктов. Это затрудняет применение концепции ЖЦ для прогнозирования.

**2. Определите структуру и организацию стандарта ЖЦ ISO - 12207. Назовите основные результаты разработки этого стандарта.**

В стандарте ISO/IEC 12207 описаны пять основных процессов жизненного цикла программного обеспечения:

1. процесс приобретения определяет действия предприятия - покупателя информационной системы, программного продукта или службы программного обеспечения;
2. процесс поставки определяет действия предприятия-поставщика по снабжению покупателя информационной системой, программным продуктом или службы программного обеспечения;
3. процесс разработки определяет действия предприятия-разработчика, который разрабатывает принципы построения программного изделия и собственно программный продукт;
4. процесс функционирования определяет действия предприятия-оператора, обслуживающего систему в целом. Сюда входят консультация пользователей, получение обратной связи и т.д.;
5. процесс сопровождения определяет действия персонала, обеспечивающего сопровождение программного продукта, т.е. управление модификацией программного продукта, поддержку текущего состояния и функциональной пригодности, установку и удаление.

Кроме пяти основных процессов, ISO/IEC 12207 оговаривает восемь вспомогательных процессов, которые являются неотъемлемой частью всего жизненного цикла системы: процесс решения проблем; процесс документирования; процесс управления конфигурацией; процесс обеспечения качества; процесс верификации; процесс аттестации; процесс совместной оценки; процесс аудита.

В стандарте ISO/IEC 12207 также определяются четыре организационных процесса: процесс управления; процесс создания инфраструктуры; процесс усовершенствования; процесс обучения.

**Основные результаты разработки стандарта ISO 12207:** Унификация процессов жизненного цикла программного обеспечения; Повышение качества программного продукта; Стандартизация подходов к разработке и сопровождению ПО; Улучшение взаимодействия между заказчиками и поставщиками программного обеспечения; Создание основы для сертификации процессов разработки ПО; Возможность адаптации стандартов под конкретные проекты и организации.

**3. Охарактеризуйте стандарт ЖЦ ISO 15504 и его отличие от ISO 12207.**

Стандарт ISO/IEC 15504 предназначен для оценки зрелости и способности процессов жизненного цикла программного обеспечения. Он представляет собой модель, которая позволяет определить, насколько эффективно выполняются процессы разработки, сопровождения и управления программными продуктами в организации. Основная цель стандарта — не только оценка текущего состояния процессов, но и их постоянное улучшение.

В отличие от ISO 12207, который описывает, какие процессы должны выполняться на каждом этапе жизненного цикла программного обеспечения, стандарт ISO 15504 фокусируется на качестве их выполнения и степени зрелости. Если ISO 12207 предоставляет структуру и последовательность процессов, то ISO 15504 оценивает, насколько эффективно эти процессы выполняются в организации и насколько они могут быть улучшены. Оценка осуществляется по многоуровневой шкале, где каждый уровень характеризует определенную степень зрелости — от хаотичного и непредсказуемого выполнения процессов до их оптимизации и постоянного совершенствования. Таким образом, ISO 12207 определяет, что нужно делать, а ISO 15504 показывает, как хорошо это делается и какие меры можно принять для повышения качества процессов.

**4. Что подразумевается под адаптацией стандарта?**

Адаптация стандарта подразумевает применение требований стандарта к конкретному проекту или проектам, например, в рамках создания внутрикорпоративных регламентов ведения проектов ПО. Адаптация включает в себя следующие виды работ: определение исходной информации для адаптации стандарта; определение условий выполнения проекта; отбор процессов, работ и задач, используемых в проекте или соответствующих регламентах; документирование полученных требований, решений и процессов.

Адаптация также подразумевает выбор модели (или комбинации моделей) жизненного цикла, а также применение соответствующих методологий, детализирующих процедуры выполнения процессов, работ и задач в рамках заданных границ (содержания) жизненного цикла программного обеспечения и организационной структуры и ролевой ответственности в конкретной организации (ее подразделении) и/или в проектной группе.

Главная цель адаптации заключается в том, чтобы стандарт не был формальностью, а стал частью повседневной деятельности, помогая компании достигать поставленных целей оптимальным способом.

**5. Каковы основные особенности каскадной модели ЖЦ?**

Каскадная (водопадная — waterfall) модель включает в себя выполнение следующих фаз:

1) исследование концепции: происходит исследование требований, разрабатывается видение продукта и оценивается возможность его реализации;

2) выработка требований: определяются программные требования для информационной предметной области системы, а также предназначение, линия поведения, производительность и интерфейсы;

3) проектирование: разрабатывается и формулируется логически последовательная техническая характеристика программной системы, включая структуру данных, архитектуру ПО, интерфейсные представления и процессуальную (алгоритмическую) детализацию;

4) реализация: эскизное описание ПС превращается в полноценный программный продукт, результатом является исходный код, база данных и документация; в реализации обычно выделяют два этапа: реализацию компонентов ПО и интеграцию компонент в готовый продукт; на обоих этапах выполняется кодирование и тестирование,

5) эксплуатация и поддержка: подразумевает запуск и текущее обеспечение, предоставление технической помощи, обсуждение возникших вопросов с пользователем, регистрацию запросов пользователя на модернизацию и внесение изменений, а также устранение ошибок;

6) сопровождение: устранение программных ошибок, неисправностей, модернизация и внесение изменений, что обычно приводит к повторению или итерации отдельных этапов разработки.



Рис. 2 Каскадная модель ЖЦ ПП

**Основные особенности:**

* Последовательность этапов — каждый этап разработки начинается только после завершения предыдущего (анализа, проектирования, разработки, тестирования, эксплуатации).
* Жесткая структура — невозможность возврата к предыдущим этапам без полной переработки проекта.
* Документирование — подробная документация создается на каждом этапе, что упрощает контроль и взаимодействие с заказчиком.
* Предсказуемость — сроки, бюджет и конечный результат определяются на начальном этапе.
* Маленькая гибкость — сложность внесения изменений на поздних этапах разработки.
* Позднее тестирование — ошибки могут быть обнаружены только после завершения программирования.
* Применимость — используется для проектов с четко определенными требованиями, которые вряд ли изменятся в процессе разработки.
* Простота управления — модель удобна для контроля, особенно при работе с большими командами.

Недостатки каскадной модели для «неподходящего» проекта следующие: попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо проблему или недостаток, приведет к значительному увеличению затрат и сбою в графике; интеграция компонентов, на которой обычно выявляется большая часть ошибок, выполняется в конце разработки, что сильно увеличивает стоимость устранения ошибок; запаздывание с получением результатов (если в процессе выполнения проекта требования изменились, то получится устаревший результат).

**6. В чем состоит главная особенность спиральной модели? Каковы сходства и различия спиральной модели и классического ЖЦ?**

Спиральная модель жизненного цикла программного обеспечения основана на итеративном подходе, при котором процесс разработки проходит через несколько циклов (итераций). Главная особенность модели заключается в сочетании элементов каскадной модели с управлением рисками. Каждый виток спирали представляет собой отдельный этап разработки, включающий планирование, анализ рисков, проектирование, реализацию и тестирование.



Рис. 3 Спиральная модель ЖЦ ПП

Спиральная и классическая модели жизненного цикла программного обеспечения схожи в том, что обе предполагают поэтапную разработку, где каждый этап имеет свои цели и результаты. Также в обеих моделях важна документация на каждом шаге, что помогает отслеживать прогресс и управлять проектом. Планирование, анализ требований и тестирование присутствуют в обоих подходах.

Основное различие между ними заключается в гибкости. Спиральная модель итеративная: каждый цикл включает анализ рисков, планирование и корректировку проекта. Это позволяет вносить изменения по ходу работы. Каскадная модель более жесткая — этапы выполняются последовательно, и изменения на поздних стадиях могут быть сложными и затратными. В спиральной модели риски управляются на каждом цикле, в то время как в каскадной модели этот процесс ограничен начальными этапами. Также тестирование в спиральной модели может происходить на каждом витке, а в каскадной — только после завершения разработки.

**7. Каковы основные особенности итерационной модели ЖЦ?**

Итерационная модель жизненного цикла является развитием классической каскадной модели, но предполагает возможность возврата на ранее выполненные этапы. Причинами возврата в классической итерационной модели являются выявленные ошибки, устранение которых и требует возврата на предыдущие этапы в зависимости от типа ошибки (ошибки кодирования, проектирования, спецификации или определения требований). Итерационная модель является более жизненной, чем классическая каскадная модель, так как создание ПО всегда связано с устранением ошибок.

Основная особенность итерационной модели жизненного цикла заключается в том, что процесс разработки разбивается на несколько итераций (повторов). Каждая итерация включает в себя все этапы разработки: анализ требований, проектирование, кодирование, тестирование и внедрение, но на каждом цикле выполняются только части функционала. После завершения каждой итерации продукт может быть частично использован и протестирован, что позволяет получать обратную связь от пользователей на более ранних стадиях.

Каждая итерация основывается на результатах предыдущей, что позволяет постепенно улучшать продукт, уточнять требования и вносить необходимые изменения. Этот подход особенно полезен, когда требования не полностью определены с самого начала и могут изменяться в процессе работы. Итерационная модель обеспечивает гибкость в разработке, позволяя адаптировать продукт под изменяющиеся потребности.

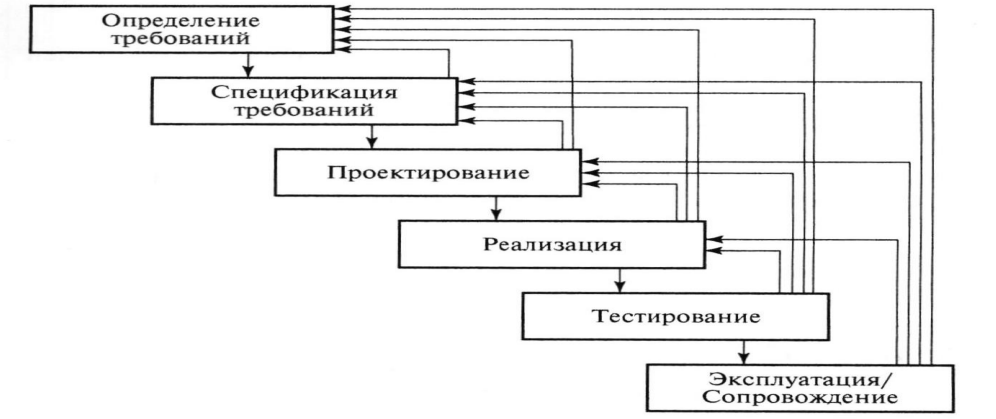
****

Рис. 4 Итерационная модель ЖЦ ПП

**8. Каковы достоинства и недостатки инкрементной модели ЖЦ?**

Достоинства инкрементной модели жизненного цикла (ЖЦ):

* получение функционального продукта после реализации каждого инкремента;
* предотвращение формирования громоздких перечней требований;
* стабилизация требований во время создания определённого инкремента;
* улучшение понимания требований для более поздних инкрементов;
* упрощение тестирования инкрементов;
* возможность пересмотра рисков, связанных с затратами и соблюдением установленного графика;
* снижение риска неудачи и изменения требований;
* снижение затрат на первоначальную поставку программного продукта;
* ускорение начального графика поставки и графика всего проекта в целом.

Недостатки инкрементной модели ЖЦ:

* непредусмотренность итераций в рамках каждого инкремента модели;
* необходимость полного функционального определения системы в начале жизненного цикла, чтобы обеспечить определение инкрементов и управление проектом;
* недостаточно чёткое определение требований;
* необходимость в чётко определённых интерфейсах между модулями, связанная с различными сроками их создания;
* сложность формального анализа и проверки отдельных инкрементов;
* наличие тенденции к оттягиванию решения трудных проблем на поздние инкременты, что может нарушить график работ;
* отсутствие снижения общих затрат на выполнение проекта;
* возможность изменений в технологиях работ, что может нарушить график работ;
* нежелательность для руководства использования на этапе анализа общих целей вместо полностью сформулированных требований;
* возможность текущего изменения требований к системе, которые уже реализованы в предыдущих инкрементах;
* необходимость хорошего планирования и проектирования, грамотного распределения работы;
* ограниченность привлечения ресурсов на длительный срок.

**9. Каковы основные промышленные модели процесса разработки ПП?**

В настоящее время широкое применение получают промышленные технологии создания ПП. Это разработки фирм, накопивших большой опыт создания ПО. Такие технологии представлены описаниями принципов, методов, применяемых процессов и операций и, как правило, поддерживаются набором CASE-средств (Computer-Aided Software Engineering), охватывают все этапы ЖЦ ПП и успешно применяются для решения практических задач. Рассмотрим особенности моделей жизненного цикла трех наиболее известных промышленных технологий.

1. Microsoft Solution Framework (MSF)-методология разработки программного обеспечения фирмы «Microsoft», предназначенная для решения широкого круга задач. Технология масштабируема, т.е. настраиваема на решение задач любой сложности коллективом любой численности.

2. Rational Unified Process (RUP) - разработка фирмы «Rational», долгое время успешно занимавшейся созданием CASE-средств, применяемых на различных этапах жизненного цикла продукта от анализа до тестирования и документирования. Аналогично MSF технология RUP универсальна, масштабируема и настраиваема на применение в конкретных условиях.

3. Extreme Programming (ХР) - методология экстремального программирования, активно развивающаяся в последнее время. и предназначенная для решения относительно небольших задач относительно небольшими коллективами профессиональных разработчиков в условиях жестко ограниченного времени.

**10. Каковы основные фазы модели MSF?**

Модель жизненного цикла MSF является некоторым гибридом каскадной и спиральной моделей, сочетая простоту управления каскадной модели с гибкостью спиральной.

Основные фазы модели MSF:

1. Создание общей картины приложения (Envisioning). На этом этапе решаются следующие задачи: оценка существующей ситуации; определение состава команды, структуры проекта, бизнес-целей, требований и профилей пользователей; разработка концепции решения и оценка риска.

2. Планирование (Panning). Включает планирование и проектирование продукта. На основе анализа требований разрабатывается проект и основные архитектурные решения, функциональные спецификации системы, планы и календарные графики; выбираются среды разработки, тестирования и пилотной эксплуатации. Состоит из концептуального, логического и физического проектирования.

3. Разработка (Developing). Создается вариант решения проблемы в виде кода и документации очередного прототипа, включая спецификации и сценарии тестирования. Основная веха - этапа «Окончательное утверждение области действия проекта».

4. Стабилизация (Stabilizing). Подготовка к выпуску окончательной версии продукта, доводка его до заданного уровня качества. Здесь выполняется комплекс работ по тестированию (обнаружение и устранение дефектов).

5. Развертывание (Deploying). Выполняется установка продукта и необходимых компонентов окружения, проводится его стабилизация в промышленных условиях и передача проекта в группу сопровождения, которая анализирует проект в целом на предмет уровня удовлетворенности заказчика.

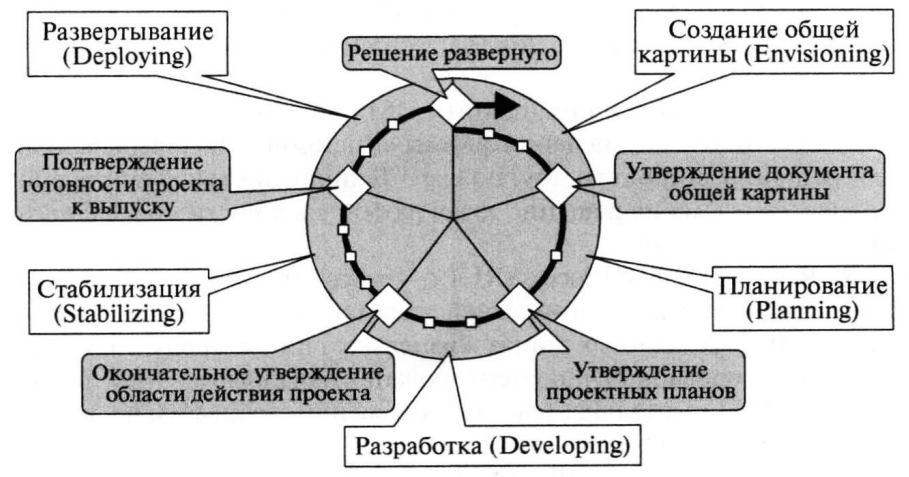
****

Рис. 5 Модель жизненного цикла MSF

**11. Дайте характеристику модели RUP и ее основное отличие от модели MSF.**

RUP (Rational Unified Process) — это итеративная методология, которая делит процесс разработки на четыре фазы: инициация, проектирование, разработка и внедрение. В каждой фазе проходят несколько итераций, с акцентом на управление рисками и обеспечение качества. RUP строгий и структурированный, с четким разделением этапов и постоянным улучшением продукта.

MSF (Microsoft Solutions Framework) также включает фазы, такие как инициация, проектирование, разработка, тестирование и внедрение, но отличается большей гибкостью и ориентированностью на практическое взаимодействие внутри команды и с заказчиком. MSF подходит для проектов с быстро меняющимися требованиями, а структура процесса менее формализована по сравнению с RUP.

Основное отличие RUP от MSF: RUP описывает строгую методологию, в которой существуют определённые критерии выхода из каждой фазы и итерации. MSF основана на гибкой процессной модели и включает в себя командную разработку. Важнейшим фокусом этой методологии является ориентация на бизнес, требования заказчика.

**12. Каковы особенности модели ЖЦ ХР экстремального программирования?**

Модель Extreme Programming является итерационно-инкрементной моделью быстрого создания и модификации протопопов программного продукта, которые должны удовлетворять очередному требованию.

Особенности модели жизненного цикла (ЖЦ) XP экстремального программирования:

* **Начальная фаза**. Включает в себя некоторое число итераций без выпуска релиза. В этот период изучают применяемые инструменты, ставят эксперименты, чтобы определить и построить стартовый вариант архитектурного скелета системы, а также осваивают методики экстремального программирования.
* **Сбор требований**. Их записывают на специальных карточках в виде сценариев выполнения отдельных функций.
* **Планирование версии (релиза)**. Проводится на собрании с участием заказчика путём выбора требований, которые войдут в следующую версию. Одновременно принимают решения, связанные с реализацией версии.
* **Разработка**. Проводится в соответствии с планом и включает только те функции, которые были отобраны на этапе планирования.
* **Тестирование**. Проводится с участием заказчика, который участвует в составлении тестов.
* **Выпуск релиза**. Разработанная версия передаётся заказчику для использования или бета-тестирования. По завершению цикла делается переход на следующую итерацию разработки.

Также в модели XP реализуется принцип «коллективного владения кодом**»**: любой член группы может изменить не только свой код, но и код другого программиста

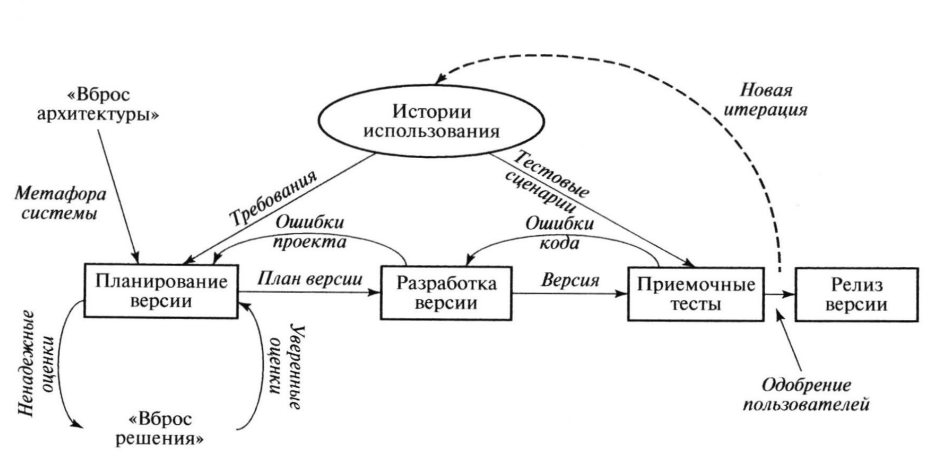


Рис. 6 Модель жизненного цикла ХР