**Лекция 2: Жизненный цикл программного обеспечения ИС**

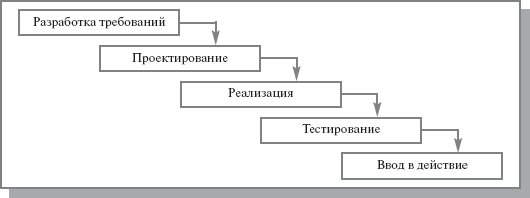
Методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде *жизненного цикла* (ЖЦ) ИС, представляя его как некоторую последовательность стадий и выполняемых на них процессов. Для каждого этапа определяются состав и последовательность выполняемых *работ*, получаемые результаты, методы и средства, необходимые для выполнения *работ*, роли и ответственность участников и т.д. Такое формальное описание ЖЦ ИС позволяет спланировать и организовать процесс коллективной разработки и обеспечить управление этим процессом.

***Жизненный цикл*** ИС можно представить как ряд событий, происходящих с системой в процессе ее создания и использования.

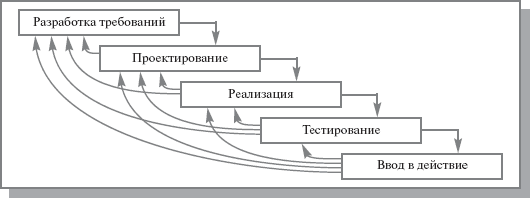
*Модель жизненного цикла* отражает различные состояния системы, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления. *Модель жизненного цикла* - структура, содержащая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение всей жизни системы, от определения требований до завершения ее использования.

В настоящее время известны и используются следующие *модели жизненного цикла*:

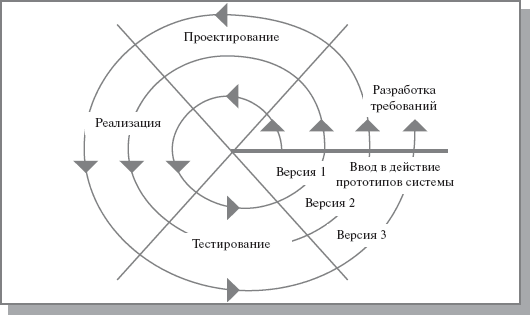
* ***Каскадная модель*** ([рис. 2.1](http://www.intuit.ru/studies/courses/2195/55/lecture/1620?page=1#image.2.1)) предусматривает последовательное выполнение всех *этапов проекта* в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе.
* ***Поэтапная модель с промежуточным контролем*** ([рис. 2.2](http://www.intuit.ru/studies/courses/2195/55/lecture/1620?page=1#image.2.2)). Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.
* ***Спиральная модель*** ([рис. 2.3](http://www.intuit.ru/studies/courses/2195/55/lecture/1620?page=1#image.2.3)). На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка.Особое внимание уделяется начальным этапам разработки - анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (*макетирования*).



**Рис. 2.1.**Каскадная модель ЖЦ ИС



**Рис. 2.2.**Поэтапная модель с промежуточным контролем



**Рис. 2.3.**Спиральная модель ЖЦ ИС

На практике наибольшее распространение получили две основные *модели жизненного цикла*:

* *каскадная модель* (характерна для периода 1970-1985 гг.);
* *спиральная модель* (характерна для периода после 1986.г.).

В ранних проектах достаточно простых ИС каждое *приложение* представляло собой единый, функционально и информационно независимый блок. Для разработки такого типа приложений эффективным оказался каскадный способ. Каждый этап завершался после полного выполнения и документального оформления всех предусмотренных *работ*.

Можно выделить следующие положительные стороны применения каскадного подхода:

* на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
* выполняемые в логической последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении относительно простых ИС, когда в самом начале разработки можно достаточно точно и полно сформулировать все требования к системе. Основным недостатком этого подхода является то, что реальный процесс создания системы никогда полностью не укладывается в такую жесткую схему, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ИС оказывается соответствующим *поэтапной модели с промежуточным контролем*.

Однако и эта схема не позволяет оперативно учитывать возникающие изменения и уточнения требований к системе. Согласование результатов разработки с пользователями производится только в точках, планируемых после завершения каждого этапа *работ*, а общие требования к ИС зафиксированы в виде технического задания на все время ее создания. Таким образом, пользователи зачастую получают систему, не удовлетворяющую их реальным потребностям.

*Спиральная модель* ЖЦ была предложена для преодоления перечисленных проблем. На этапах анализа и проектирования реализуемость технических решений и степень удовлетворения потребностей заказчика проверяется путем создания прототипов. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы. Это позволяет уточнить требования, цели и характеристики проекта, определить качество разработки, спланировать работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который удовлетворяет действительным *требованиям заказчика* и доводится до реализации.

Итеративная разработка отражает объективно существующий спиральный цикл создания сложных систем. Она позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем и решить главную задачу - как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального *цикла* - *определение* момента перехода на следующий этап. Для ее решения вводятся временные ограничения на каждый из этапов *жизненного цикла*, и переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. Планирование производится на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Несмотря на настойчивые рекомендации компаний - вендоров и экспертов в области проектирования и разработки ИС, многие компании продолжают использовать *каскадную модель* вместо какого-либо варианта итерационной модели. Основные причины, *по*которым *каскадная модель* сохраняет свою популярность, следующие [[ 2.1 ]](http://www.intuit.ru/studies/courses/2195/55/literature#literature.2.1):

1. **Привычка** - многие ИТ-специалисты получали образование в то время, когда изучалась только *каскадная модель*, поэтому она используется ими и в наши дни.
2. **Иллюзия снижения рисков** участников проекта (заказчика и исполнителя). *Каскадная модель* предполагает разработку законченных продуктов на каждом этапе: технического задания, *технического проекта*, программного продукта и пользовательской документации. Разработанная документация позволяет не только определить требования к продукту следующего этапа, но и определить обязанности сторон, объем работ и сроки, при этом окончательная оценка сроков и стоимости проекта производится на начальных этапах, после завершения обследования. Очевидно, что если требования к информационной системе меняются в ходе реализации проекта, а качество документов оказывается невысоким (требования неполны и/или противоречивы), то в действительности использование *каскадной модели* создает лишь иллюзию определенности и на деле увеличивает риски, уменьшая лишь ответственность участников проекта. При формальном подходе*менеджер проекта* реализует только те требования, которые содержатся в спецификации, опирается на документ, а не на реальные потребности бизнеса. Есть два основных типа контрактов на разработку ПО. Первый тип предполагает выполнение определенного объема работ за определенную сумму в определенные сроки (fixed price). Второй тип предполагает повременную оплату работы (time work). Выбор того или иного типа контракта зависит от степени определенности задачи.*Каскадная модель* с определенными этапами и их результатами лучше приспособлена для заключения контракта с оплатой по результатам работы, а именно этот тип контрактов позволяет получить полную *оценку стоимости* проекта до его завершения. Более вероятно заключение контракта с повременной оплатой на небольшую систему, с относительно небольшим весом в структуре затрат предприятия. Разработка и внедрение интегрированной информационной системы требует существенных финансовых затрат, поэтому используются контракты с фиксированной ценой, и, следовательно, *каскадная модель* разработки и внедрения. *Спиральная модель* чаще применяется при разработке информационной системы силами собственного отдела ИТ предприятия.
3. **Проблемы внедрения** при использовании итерационной модели. В некоторых областях *спиральная модель* не может применяться, поскольку невозможно использование/тестирование продукта, обладающего неполной функциональностью (например, военные разработки, атомная энергетика и т.д.). Поэтапное итерационное внедрение информационной системы для бизнеса возможно, но сопряжено с организационными сложностями (перенос данных, интеграция систем, изменение бизнес-процессов, *учетной политики*, обучение пользователей). Трудозатраты при поэтапном итерационном внедрении оказываются значительно выше, а управление проектом требует настоящего искусства. Предвидя указанные сложности, заказчики выбирают *каскадную модель*, чтобы "внедрять систему один раз".

Каждая из стадий создания системы предусматривает выполнение определенного объема *работ*, которые представляются в виде*процессов ЖЦ*. *Процесс* определяется как совокупность взаимосвязанных действий, преобразующих входные данные в выходные. Описание каждого процесса включает в себя перечень решаемых задач, исходных данных и результатов.

Существует *целый* ряд стандартов, регламентирующих ЖЦ *ПО*, а в некоторых случаях и процессы разработки.

Значительный вклад в теорию проектирования и *разработки информационных систем* внесла компания *IBM*, предложив еще в середине 1970-х годов методологию *BSP* (Business *System* *Planning* - методология организационного планирования). Метод структурирования информации с использованием матриц пересечения бизнес-процессов, функциональных подразделений, функций систем обработки данных (информационных систем), информационных объектов, документов и баз данных, предложенный в *BSP*, используется сегодня не только в ИТ-проектах, но и проектах *по* *реинжинирингу бизнес-процессов*, изменению организационной структуры. Важнейшие шаги процесса *BSP*, их последовательность (получить поддержку высшего руководства, определить процессы предприятия, определить классы данных, провести интервью, обработать и организовать данные интервью) можно встретить практически во всех формальных методиках, а также в проектах, реализуемых на практике.

Среди наиболее известных стандартов можно выделить следующие:

* ГОСТ 34.601-90 - распространяется на автоматизированные системы и устанавливает стадии и этапы их создания. Кроме того, в стандарте содержится описание содержания работ на каждом этапе. Стадии и этапы работы, закрепленные в стандарте, в большей степени соответствуют *каскадной модели* жизненного цикла [[ 2.2 ]](http://www.intuit.ru/studies/courses/2195/55/literature#literature.2.2).
* ISO/*IEC* 12207:1995 - стандарт на процессы и организацию *жизненного цикла*. Распространяется на все виды заказного ПО. Стандарт не содержит описания фаз, стадий и этапов [[ 2.3 ]](http://www.intuit.ru/studies/courses/2195/55/literature#literature.2.3).
* Custom Development Method (методика Oracle) по разработке прикладных информационных систем - технологический материал, детализированный до уровня заготовок проектных документов, рассчитанных на использование в проектах с применением Oracle. Применяется *CDM* для классической модели ЖЦ (предусмотрены все работы/задачи и этапы), а также для технологий "быстрой разработки" (Fast Track) или "облегченного подхода", рекомендуемых в случае малых проектов.
* Rational *Unified* Process (*RUP*) предлагает итеративную модель разработки, включающую четыре фазы: начало, исследование, построение и внедрение. Каждая фаза может быть разбита на этапы (итерации), в результате которых выпускается версия для внутреннего или внешнего использования. Прохождение через четыре основные фазы называется *циклом разработки*, каждый цикл завершается генерацией версии системы. Если после этого работа над проектом не прекращается, то полученный продукт продолжает развиваться и снова минует те же фазы. Суть работы в рамках *RUP* - это создание и сопровождение моделей на базе UML [[ 2.4 ]](http://www.intuit.ru/studies/courses/2195/55/literature#literature.2.4).
* *Microsoft Solution Framework* (*MSF*) сходна с *RUP*, так же включает четыре фазы: анализ, проектирование, разработка, стабилизация, является итерационной, предполагает использование *объектно-ориентированного моделирования*. *MSF* в сравнении с *RUP* в большей степени ориентирована на разработку бизнес-приложений.
* *Extreme Programming* (XP). *Экстремальное программирование* (самая новая среди рассматриваемых методологий) сформировалось в 1996 году. В основе методологии командная работа, эффективная коммуникация между заказчиком и исполнителем в течение всего проекта по разработке ИС, а разработка ведется с использованием последовательно дорабатываемых прототипов.

В соответствии с базовым международным стандартом ISO/*IEC* 12207 все *процессы ЖЦ ПО* делятся на три группы:

1. **Основные процессы**:
   * приобретение;
   * поставка;
   * разработка;
   * эксплуатация;
   * сопровождение.
2. **Вспомогательные процессы**:
   * документирование;
   * *управление конфигурацией*;
   * обеспечение качества;
   * разрешение проблем;
   * аудит;
   * аттестация;
   * совместная оценка;
   * верификация.
3. **Организационные процессы**:
   * создание инфраструктуры;
   * управление;
   * обучение;
   * усовершенствование.

В [таблице 2.1](http://www.intuit.ru/studies/courses/2195/55/lecture/1620?page=3#table.2.1) приведены ориентировочные описания основных процессов ЖЦ. Вспомогательные процессы предназначены для поддержки выполнения основных *процессов, обеспечения* качества проекта, организации верификации, проверки и тестирования*ПО*. Организационные процессы определяют действия и задачи, выполняемые как заказчиком, так и разработчиком проекта для управления своими процессами.

Для поддержки практического применения стандарта ISO/*IEC* 12207 разработан ряд технологических документов: Руководство для ISO/*IEC* 12207 (ISO/*IEC* TR 15271:1998 *Information* *technology* - *Guide* for ISO/*IEC* 12207) и Руководство *по* применению ISO/*IEC*12207 к управлению проектами (ISO/*IEC* TR 16326:1999 *Software engineering* - *Guide* for the *application* of ISO/*IEC* 12207 to *project management*).

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица 2.1. Содержание основных процессов ЖЦ ПО ИС (ISO/*****IEC* 12207)** | | | |
| **Процесс (исполнитель процесса)** | **Действия** | **Вход** | **Результат** |
| Приобретение (заказчик) | * Инициирование * Подготовка заявочных предложений * Подготовка договора * Контроль деятельности поставщика * Приемка ИС | * Решение о начале работ по внедрению ИС * Результаты обследования деятельности заказчика * Результаты анализа рынка ИС/ тендера * План поставки/ разработки * Комплексный тест ИС | * Технико-экономическое обоснование внедрения ИС * Техническое задание на ИС * Договор на поставку/ разработку * Акты приемки этапов работы * Акт приемно-сдаточных испытаний |
| Поставка (разработчик ИС) | * Инициирование * Ответ на заявочные предложения * Подготовка договора * Планирование исполнения * Поставка ИС | * Техническое задание на ИС * Решение руководства об участии в разработке * Результаты тендера * Техническое задание на ИС * *План управления проектом* * Разработанная ИС и документация | * Решение об участии в разработке * *Коммерческие предложения*/ конкурсная заявка * Договор на поставку/ разработку * *План управления проектом* * Реализация/ корректировка * Акт приемно-сдаточных испытаний |
| Разработка (разработчик ИС) | * Подготовка * *Анализ требований* к ИС * Проектирование архитектуры ИС * Разработка требований к ПО * Проектирование архитектуры ПО * Детальное проектирование ПО * Кодирование и тестирование ПО * Интеграция ПО и квалификационное тестирование ПО * Интеграция ИС и квалификационное тестирование ИС | * Техническое задание на ИС * Техническое задание на ИС, модель ЖЦ * Подсистемы ИС * Спецификации требования к компонентам ПО * Архитектура ПО * Материалы детального проектирования ПО * План интеграции ПО, тесты * Архитектура ИС, ПО, документация на ИС, тесты | * Используемая модель ЖЦ, стандарты разработки * План работ * Состав подсистем, компоненты оборудования * Спецификации требования к компонентам ПО * Состав компонентов ПО, интерфейсы с БД, план интеграции ПО * Проект БД, спецификации интерфейсов между компонентами ПО, требования к тестам * Тексты модулей ПО, акты автономного тестирования * Оценка соответствия комплекса ПО требованиям ТЗ * Оценка соответствия ПО, БД, технического комплекса и комплекта документации требованиям ТЗ |

Позднее был разработан и в 2002 г. опубликован стандарт на процессы *жизненного цикла* систем (ISO/*IEC* 15288 *System* *life cycle*processes). К разработке стандарта были привлечены специалисты различных областей: системной инженерии, программирования, управления качеством, человеческими ресурсами, безопасностью и пр. Был учтен практический *опыт* создания систем в правительственных, коммерческих, военных и академических организациях. Стандарт применим для широкого класса систем, но его основное предназначение - *поддержка* создания компьютеризированных систем.

Согласно стандарту ISO/*IEC* серии 15288 [[ 2.5 ]](http://www.intuit.ru/studies/courses/2195/55/literature#literature.2.5)в структуру ЖЦ следует включать следующие *группы процессов*:

1. **Договорные процессы**:
   * приобретение (внутренние решения или решения внешнего поставщика);
   * поставка (внутренние решения или решения внешнего поставщика).
2. **Процессы предприятия**:
   * управление окружающей средой предприятия;
   * инвестиционное управление;
   * управление ЖЦ ИС;
   * управление ресурсами;
   * управление качеством.
3. **Проектные процессы**:
   * планирование проекта;
   * оценка проекта;
   * контроль проекта;
   * управление рисками;
   * *управление конфигурацией*;
   * управление информационными потоками;
   * принятие решений.
4. **Технические процессы**:
   * определение требований;
   * *анализ требований*;
   * разработка архитектуры;
   * внедрение;
   * интеграция;
   * верификация;
   * переход;
   * аттестация;
   * эксплуатация;
   * сопровождение;
   * утилизация.
5. **Специальные процессы**:
   * определение и установка взаимосвязей исходя из задач и целей.

Стадии создания системы, предусмотренные в стандарте ISO/*IEC* 15288, несколько отличаются от рассмотренных выше. Перечень стадий и основные результаты, которые должны быть достигнуты к моменту их завершения, приведены в [таблице 2.2](http://www.intuit.ru/studies/courses/2195/55/lecture/1620?page=3#table.2.2).

|  |  |  |
| --- | --- | --- |
| Таблица 2.2. Стадии создания систем (ISO/*IEC* 15288) | | |
| **№ п/п** | **Стадия** | **Описание** |
| 1 | Формирование концепции | Анализ потребностей, выбор концепции и проектных решений |
| 2 | Разработка | Проектирование системы |
| 3 | Реализация | Изготовление системы |
| 4 | Эксплуатация | Ввод в эксплуатацию и использование системы |
| 5 | Поддержка | Обеспечение функционирования системы |
| 6 | Снятие с эксплуатации | Прекращение использования, демонтаж, архивирование системы |