Курганский государственный университет
Кафедра программного обеспечения автоматизированных систем

К У Р С Л Е К Ц И Й

по дисциплине

ВВЕДЕНИЕ В ПРОГРАММНУЮ ИНЖЕНЕРИЮ

для студентов направления 231000.62

«Программная инженерия»

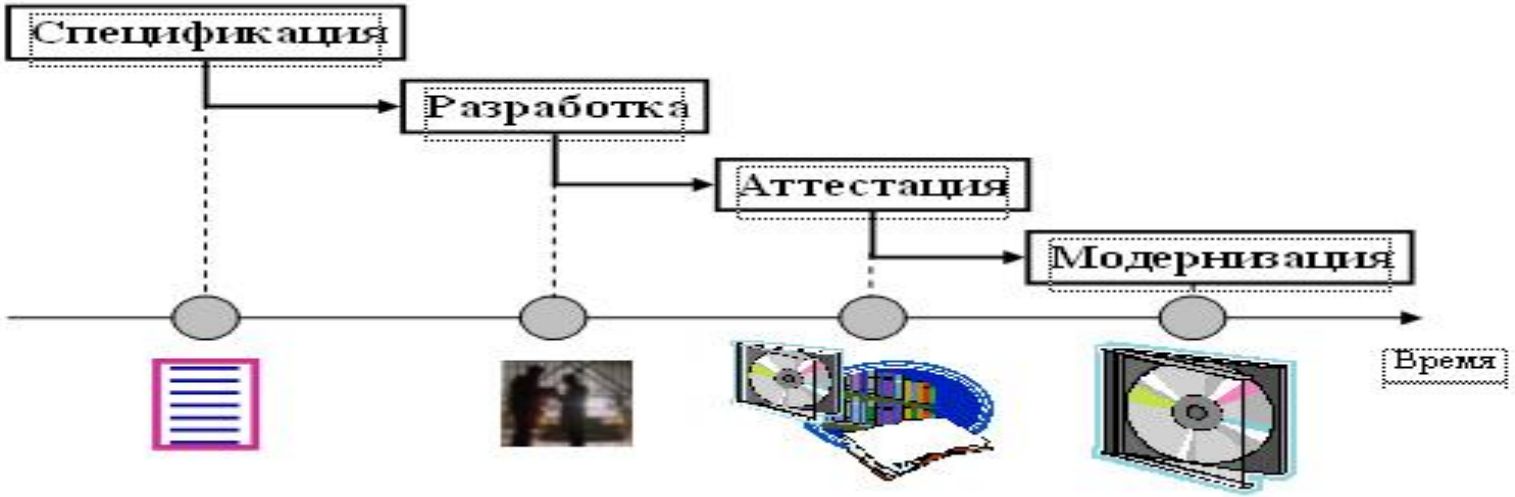
Лекция 2.2
Модели жизненного цикла ПО

План лекции

1. Понятие модели жизненного цикла ПО
2. Типовые модели процесса создания ПО
   - 2.1. Каскадная модель
   - 2.2. Модель формальной разработки
   - 2.3. Эволюционная модель
   - 2.4. Модель пошаговой разработки
   - 2.5. Спиральная модель
   - 2.6. Модель разработки ПО на основе ранее созданных компонентов
3. Контрольные вопросы и задания

2. *Стадии процесса создания ПО*

- ***Спецификация***: формулирование спецификаций определяет основные требования к ПО (что должна делать система).
- ***Разработка***: создание ПО в соответствии со спецификациями.
- ***Аттестация***: проверка ПО на соответствие потребностям заказчика.
- ***Модернизация***: развитие ПО в соответствии с изменившимися потребностями заказчика.



3. *Стадии процесса создания ПО*

(продолжение)

- Все стадии основаны на специальных технологиях. Например, на стадии Разработки могут использоваться модульная, структурная, объектно-ориентированная или компонентная технологии программирования.
- Каждая организация может организовать процесс создания ПО так, как ей это представляется разумным, этот процесс может иметь разную степень формализации. Так, например, возможен подход "вечером обсудили и обо всем договорились", но этот принцип работает только для команд из 2-3 человек в достаточно простых проектах.
- Возможна другая крайность - каждое действие жестко определено и прописано в описании процесса. В этом случае возникает необходимость длительного предварительного обучения сотрудников работе в рамках этого, безусловно, сложного описания. Подобный подход может бать рекомендован для очень больших проектов, выполняемых распределенными коллективами.
- Несмотря на естественные отличия в описаниях процессов, как правило, в нем присутствуют все рассмотренные выше стадии.
- Существуют известные, хорошо проработанные процессы и технологии: Microsoft Solutions Framework (MSF), Rational Unified Process (RUP), eXtreme Programming (XP) - и другие.

4. *Модели процесса создания ПО*

- Каскадная модель
- Модель формальной разработки
- Эволюционная модель
- Модель пошаговой разработки
- Спиральная модель
- Повторное использование компонентов.

Каскадная и эволюционная модели разработки наиболее широко распространены и используются на практике.

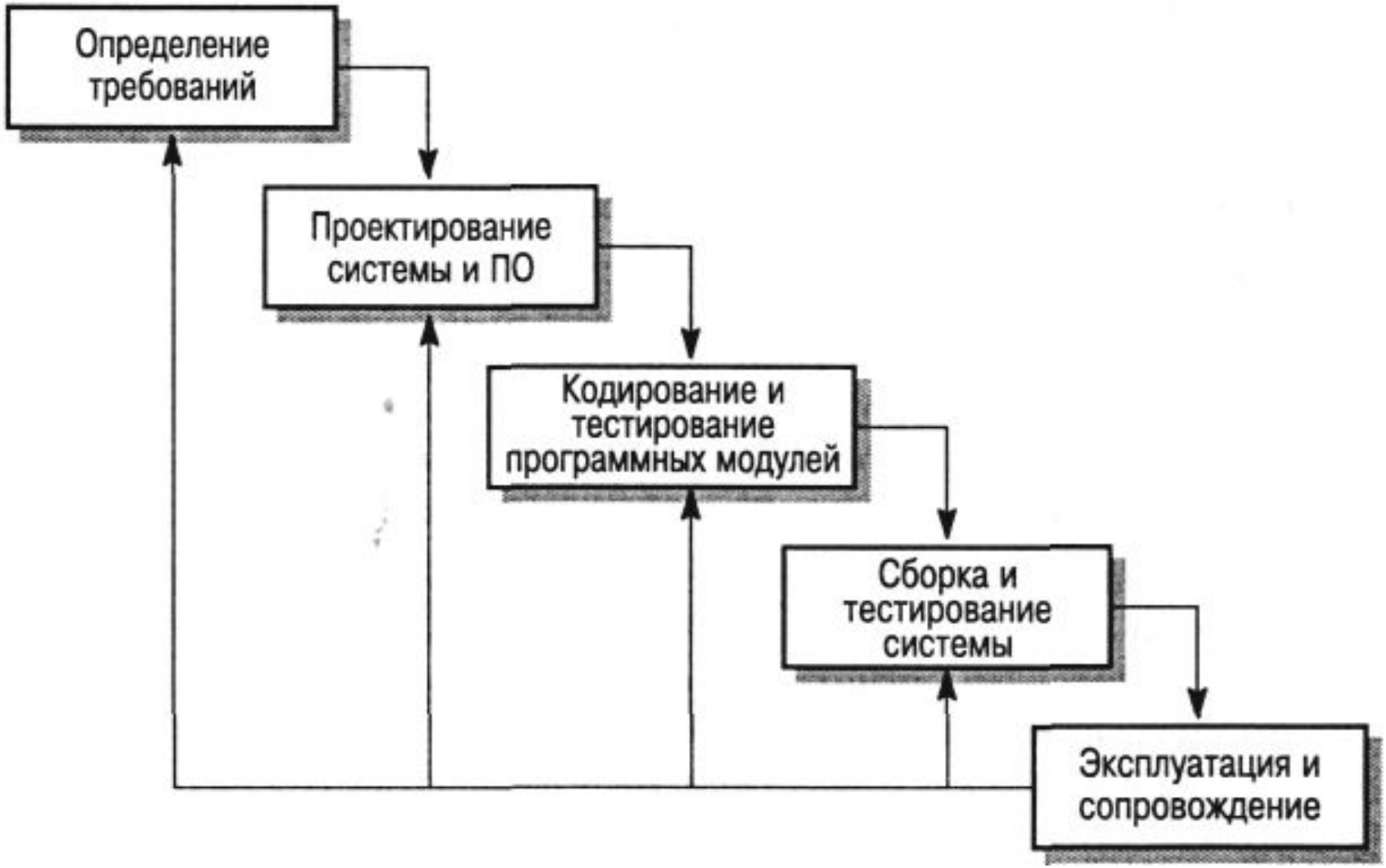
Модель формальной разработки систем успешно применялась во многих проектах, но количество организаций-разработчиков, постоянно использующих этот метод, невелико.

Отличительной чертой эволюционной модели является то, что процессы специфицирования, разработки и аттестации ПО выполняются параллельно при постоянном обмене информацией между ними.

Существенное отличие спиральной модели заключается в точном определении и оценивании рисков принятия тех или иных проектных решений, что существенно повышает надежность.

Использование готовых системных компонентов практикуется повсеместно, но большинство организаций не придерживаются в точности модели разработки ПО на основе ранее созданных компонентов. Вместе с тем этот метод должен получить широкое распространение в XXI столетии, поскольку сборка систем из готовых или ранее использованных компонентов значительно ускоряет разработку ПО.

5. ***Каскадная модель***



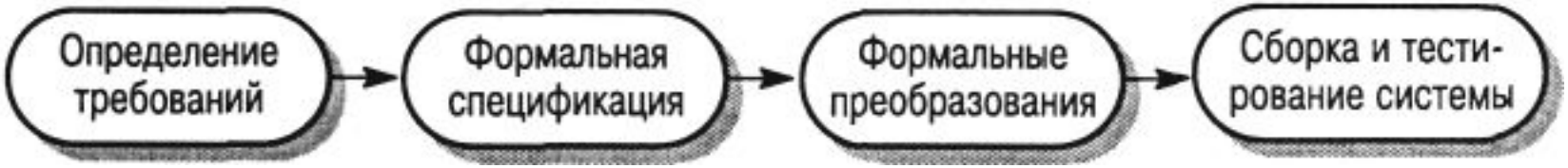
6. *Каскадная модель*

- ***Анализ и формирование требований***. Путем консультаций с заказчиком ПО определяются функциональные возможности, ограничения и цели создаваемой программной системы.
- ***Проектирование системы и программного обеспечения***. Процесс проектирования системы разбивает системные требования на требования, предъявляемые к аппаратным средствам, и требования к программному обеспечению системы. Разрабатывается общая архитектура системы. Проектирование ПО предполагает определение и описание основных программных компонентов и их взаимосвязей.
- ***Кодирование и тестирование программных модулей***. На этой стадии архитектура ПО реализуется в виде множества программ или программных модулей. Тестирование каждого модуля включает проверку его соответствия требованиям к данному модулю.
- ***Сборка и тестирование системы***. Отдельные программы и программные модули интегрируются и тестируются в виде целостной системы. Проверяется, соответствует ли система своей спецификации.
- ***Эксплуатация и сопровождение системы***. Обычно (хотя и не всегда) это самая длительная фаза жизненного цикла ПО. Система инсталлируется, и начинается период ее эксплуатации. Сопровождение системы включает исправление ошибок, которые не были обнаружены на более ранних этапах жизненного цикла, совершенствование системных компонентов и "подгонку" функциональных возможностей системы к новым требованиям.

7. *Каскадная модель*

- Требования к системе определяются на начальном этапе и далее не меняются.
- Каждый этап может начаться лишь тогда, когда закончился (будет утвержден и документирован) предыдущий.
- Достоинства каскадной модели в простоте и хорошей структурированности.
- Основные недостатки связаны с прямолинейностью и отсутствием гибкости. Дело в том, что неизменность требований является мифом. Требования менялись, меняются и будут меняться всегда в ходе разработки, а каскадная модель не учитывает этот факт.
- К недостаткам каскадной модели можно отнести негибкое разбиение процесса создания ПО на отдельные фиксированные этапы. В этой модели определяющие систему решения принимаются на ранних этапах и затем их трудно отменить или изменить, особенно это относится к формированию системных требований.
- Каскадная модель применяется тогда, когда требования формализованы достаточно четко и корректно. Вместе с тем каскадная модель хорошо отражает практику создания ПО. Технологии создания ПО, основанные на данной модели, используются повсеместно, в частности для разработки систем, входящих в состав больших инженерных проектов.

8. *Модель формальной разработки*

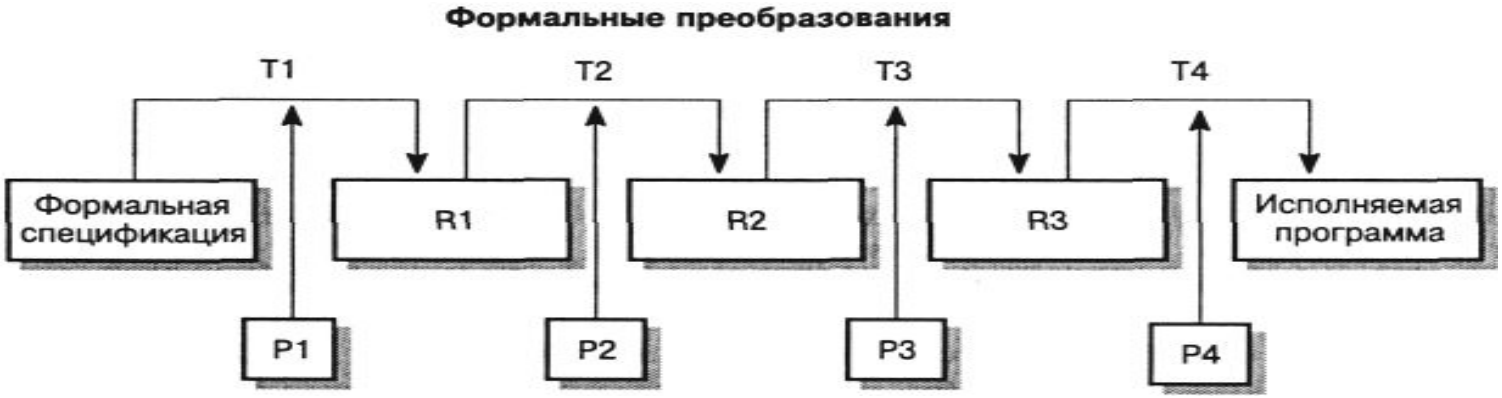


Эта модель может считаться разновидностью каскадной модели, но построена на основе ***формальных математических преобразований системной спецификации в исполняемую программу.***

Существенные отличия от каскадной модели:

- Здесь спецификация системных требований имеет вид детализированной формальной спецификации, записанной с помощью специальной математической нотации.
- Процессы проектирования, кодирования и тестирования программных модулей заменяются процессом, в котором формальная спецификация путем последовательных формальных преобразований трансформируется в исполняемую программу.

9. *Модель формальной разработки*



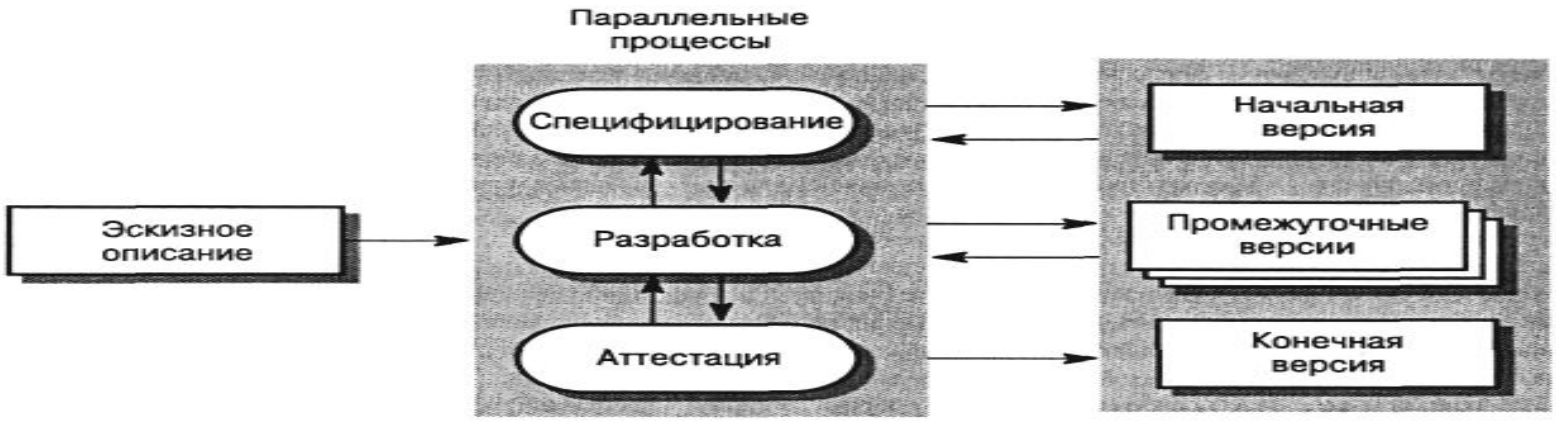
В процессе преобразования формальное математическое представление системы последовательно трансформируется в программный код, постепенно все более детализированный. Преобразования выполняются математически корректно и проверка соответствия спецификации и программы не требуется.

Наиболее известным примером метода формальных преобразований является метод "чистой комнаты" (Cleanroom), разработанный компанией IBM. Этот метод предполагает пошаговую разработку ПО, когда на каждом шаге применяется формальные преобразования, что позволяет отказаться от тестирования отдельных программных модулей, а тестирование всей системы происходит после ее сборки.

Методы формальных преобразований не нашли широкого применения - основная причина заключается в том, что функционирование большинства систем с трудом поддается описанию методом формальных спецификаций.

Методы формальных преобразований обычно применяют для разработки систем, которые должны удовлетворять строгим требованиям надежности, безотказности и безопасности, так как они гарантируют соответствие созданных систем их спецификациям.

10. *Эволюционная модель*



Разрабатывается первоначальная версия программного продукта, которая передается на испытание пользователям, затем она дорабатывается с учетом мнения пользователей.

Полученная промежуточная версия продукта также передается для испытаний пользователям, затем снова дорабатывается и так несколько раз, пока не будет получен необходимый программный продукт.

Отличительной чертой данной модели является то, что процессы спецификирования, разработки и аттестации ПО выполняются параллельно при постоянном обмене информацией между ними.

11. Два подхода к реализации эволюционной модели

- *Подход пробных разработок.* Здесь большую роль играет постоянная работа с заказчиком (или пользователями) для того, чтобы определить полную систему требований к ПО, необходимую для разработки конечной версии продукта. В рамках этого подхода вначале разрабатываются те части системы, которые очевидны или хорошо специфицированы. Система эволюционирует (дорабатывается) путем добавления новых средств по мере их предложения заказчиком.
- *Прототипирование.* Здесь целью процесса эволюционной разработки ПО является поэтапное уточнение требований заказчика и, следовательно, получение законченной спецификации, определяющей разрабатываемую систему. Прототип (*действующий программный модуль, реализующий отдельные функции создаваемого ПО*) обычно создается для экспериментирования с той частью требований заказчика, которые сформированы нечетко или с внутренними противоречиями.

12. Достоинства и недостатки эволюционной модели

Эволюционная модель часто более эффективна каскадной модели, особенно если требования заказчика могут меняться в процессе разработки системы. Достоинством эволюционной модели процесса создания ПО является то, что здесь спецификация может разрабатываться постепенно, по мере того как заказчик (или пользователи) осознает и сформулирует те задачи, которые должно решать программное обеспечение.

Вместе с тем эволюционные модели имеют и недостатки:

- *Многие этапы процесса создания ПО не документированы.* Менеджерам проекта создания ПО необходимо регулярно документально отслеживать выполнение работ. Но если система разрабатывается быстро, то экономически не выгодно документировать каждую версию системы.
- *Система часто получается плохо структурированной.* Постоянные изменения в требованиях приводят к ошибкам и упущениям в структуре ПО. Со временем внесение изменений в систему становится все более сложным и затратным.
- *Часто требуются специальные средства и технологии разработки ПО.* Это вызвано необходимостью быстрой разработки версий программного продукта. Но, с другой стороны, это может привести к несовместимости некоторых применяемых средств и технологий, что, в свою очередь, требует наличия в команде разработчиков специалистов высокого уровня.
- Эволюционные модели наиболее приемлем для разработки небольших (до 100 000 строк кода) и средних (до 500 000 строк кода) программных систем с относительно коротким сроком жизни. На больших долгоживущих системах слишком заметно проявляются недостатки этого подхода.

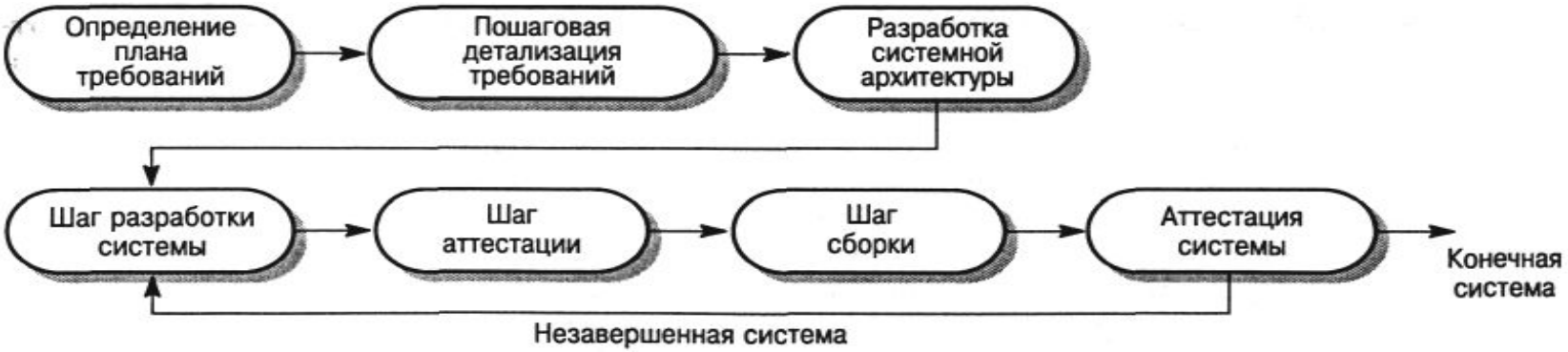
13. *Модель пошаговой разработки*

Модель пошаговой разработки относится к категории гибридных ***итерационных моделей***, позволяющих выполнять процессы создания ПО итерационно, когда в ответ на изменения требований повторно выполняются определенные этапы разработки (чаще всего на стадиях проектирования и кодирования).

Существенным отличием итерационных моделей является то, что здесь процесс разработки спецификации протекает параллельно с разработкой самой программной системы.

Модель пошаговой разработки содержит элементы каскадной и эволюционной моделей и была предложена как попытка уменьшить количество повторно выполняемых работ в процессе создания ПО и увеличить для заказчика временной период окончательного принятия решения обо всех деталях требований.

14. Модель пошаговой разработки



Определяются (в общих чертах) те сервисы (функциональные возможности), которые должны присутствовать у создаваемой системы, и устанавливаются их приоритеты.

Определяется количество и последовательность выполнения шагов разработки, причем ***на каждом шаге должен быть получен компонент, реализующий определенное подмножество сервисов***. Распределение реализации системных сервисов по шагам разработки определяется их приоритетностью.

На первых шагах детализируются требования к компонентам, реализующих сервисы высших приоритетов, и выбираются средства их реализации.

В ходе их реализации (на последующих шагах) анализируются и детализируются требования для компонентов, которые будут разрабатываться на более поздних шагах, причем изменение требований для тех компонентов, которые уже находятся в процессе разработки, не допускается.

По завершении очередного шага разработки полученный компонент интегрируется с ранее произведенными компонентами; таким образом, после каждого шага разработки система приобретает все большую функциональную завершенность..

Заказчик может экспериментировать с готовыми подсистемами и компонентами для того, чтобы уточнить требования, предъявляемые к следующим версиям уже готовых компонентов или к компонентам, разрабатываемым на последующих шагах.

15. Достоинства модели пошаговой разработки

1. Заказчику нет необходимости ждать полного завершения разработки системы, чтобы получить о ней представление. Компоненты, полученные на первых шагах разработки, удовлетворяют наиболее критическим требованиям (так как имеют наибольший приоритет) и их можно оценить на самой ранней стадии создания системы.
2. Заказчик может использовать компоненты, полученные на первых шагах разработки, как прототипы и провести с ними эксперименты для уточнения требований к тем компонентам, которые будут разрабатываться позднее.
3. Данный подход уменьшает риск общесистемных ошибок. Хотя в разработке отдельных компонентов возможны ошибки, но эти компоненты должны пройти соответствующее тестирование и аттестацию, прежде чем их передадут заказчику.
4. Поскольку системные сервисы с высоким приоритетом разрабатываются первыми, а все последующие компоненты интегрируются с ними, неизбежно получается так, что наиболее важные подсистемы подвергаются более тщательному всестороннему тестированию и проверке. Это значительно снижает вероятность программных ошибок в особо важных частях системы.

16. Недостатки модели пошаговой разработки

Компоненты, получаемые на каждом шаге разработки, имеют относительно небольшой размер (обычно не более 20 000 строк кода), но должны реализовать какую-либо системную функцию. Отобразить множество системных требований к компонентам нужного размера довольно сложно.

Базовые свойства системы могут обеспечиваться различными ее частями совместно. Поскольку требования детально не определены до тех пор, пока не будут разработаны все компоненты, бывает весьма сложно распределить общесистемные функции по компонентам.

В настоящее время предложен метод экстремального программирования (XP - eXtreme Programming), который устраняет некоторые недостатки метода пошаговой разработки. Этот метод основан на пошаговой разработке малых программных компонентов, реализующих небольшие функциональные требования, постоянном вовлечении заказчика в процесс разработки и обезличенном программировании

17. *Спиральная модель*

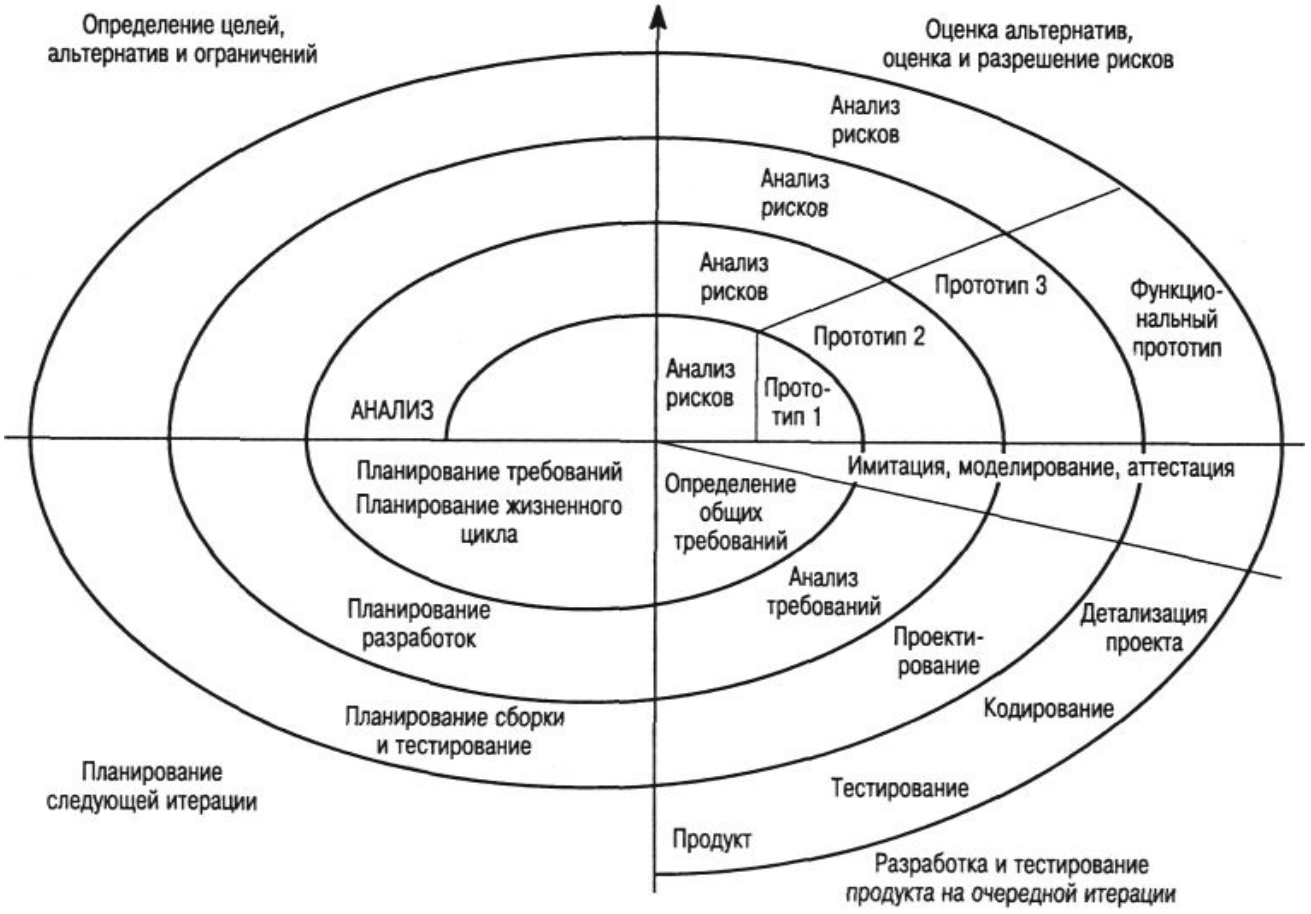
Спиральная модель также относится к категории гибридных ***итерационных моделей***, однако, в отличие от рассмотренных ранее моделей, где процесс создания ПО представлен в виде последовательности отдельных процессов с возможной обратной связью между ними, здесь ***процесс разработки представлен в виде спирали.***

Каждый ***виток спирали соответствует одной стадии (итерации) процесса создания ПО***: самый внутренний виток соответствует стадии принятия решения о создании ПО, на следующем витке определяются системные требования, далее следует стадия (виток спирали) проектирования системы и т.д.

В спиральной модели нет фиксированных этапов на каждом ее витке.

Спиральная модель может включать в себя любые другие модели разработки систем. Например, на одном витке спирали может использоваться протипирование для более четкого определения требований (и, следовательно, для уменьшения соответствующих рисков), но на следующем витке может применяться каскадная модель или метод формальных преобразований (если требования четко сформулированы).

18. **Спиральная модель**



19. Спиральная модель

Каждый виток спирали разбит на четыре сектора:

Сектор 1. ***Определение целей***. Определяются цели каждой итерации проекта. Кроме того, устанавливаются ограничения на процесс создания ПО и на сам программный продукт, уточняются планы производства компонентов. Определяются проектные риски (например, риск превышения сроков или риск превышения стоимости проекта) и планируются альтернативные стратегии разработки ПО.

Сектор 2. ***Оценка и разрешение рисков***. Для каждого определенного проектного риска проводится его детальный анализ. Планируются мероприятия для уменьшения (разрешения) рисков. Например, если существует риск, что системные требования определены неверно, планируется разработать прототип системы.

Сектор 3. ***Разработка и тестирование***. После оценки рисков выбирается модель процесса создания системы. Например, если доминируют риски, связанные с разработкой интерфейсов, наиболее подходящей будет эволюционная модель разработки ПО с прототипированием. Если основные риски связаны с соответствием системы её спецификациям, скорее всего, следует применить модель формальных преобразований. Каскадная модель может быть применена в том случае, если основные риски определены как ошибки, которые могут проявиться на этапе сборки системы.

Сектор 4. ***Планирование***. Здесь пересматривается проект и принимается решение о том, начинать ли следующий виток спирали. Если принимается решение о продолжении проекта, разрабатывается план на следующую стадию проекта.

20. Спиральная модель

Первая итерация создания ПО в спиральной модели начинается с тщательной проработки системных показателей (целей системы), таких как эксплуатационные показатели и функциональные возможности системы. Конечно, альтернативных путей достижения этих показателей или целей можно сформировать бесконечно много. Но каждая альтернатива должна оценивать стоимость достижения каждой сформулированной цели.

Результаты анализа возможных альтернатив служат источником оценки проектного риска. Это происходит на следующей стадии спиральной модели, где для оценки рисков используются более детальный анализ альтернатив, прототипирование, имитационное моделирование и т.п. С учетом полученных оценок рисков выбирается тот или иной подход к разработке системных компонентов, далее он реализуется, затем осуществляется планирование следующего этапа процесса создания ПО.

Существенное отличие спиральной модели от других моделей процесса создания ПО заключается в точном определении и оценивании рисков. Например, если при написании программного кода используется новый язык программирования, то риск может заключаться в том, что компилятор этого языка может быть ненадежным или что результирующий код может быть не достаточно эффективным. Риски могут также заключаться в превышении сроков или стоимости проекта.

Уменьшение (разрешение) рисков – важный элемент управления системным проектом. Возможные риски и способы их разрешения рассматриваются более детально в отдельной дисциплине «Управление программными проектами».

21. Разработка ПО на основе ранее созданных компонентов

В большинстве программных проектов применяется повторное использование некоторых программных модулей. Это обычно случается там, где разработчики проекта знают о ранее созданных программных продуктах, в составе которых есть компоненты, приблизительно удовлетворяющие требованиям разрабатываемых компонентов. Эти компоненты модифицируются в соответствии с новыми требованиями и затем включается в состав новой системы. В эволюционной модели разработки для ускорения процесса создания ПО повторное использование ранее созданных компонентов применяется достаточно часто.

Этот подход основан на наличии большой базы существующих программных компонентов, которые можно интегрировать в создаваемую новую систему. Часто такими компонентами являются свободно продаваемые на рынке программные продукты, которые можно использовать для выполнения определенных специальных функций, таких как форматирование текста, числовые вычисления и т.п.

22. Разработка ПО на основе ранее созданных компонентов

В этом подходе начальный этап специфицирования требований и этап аттестации такие же, как и в других моделях процесса создания ПО. А этапы, расположенные между ними, имеют следующий смысл.

1. *Анализ компонентов.* Имея спецификацию требований, на этом этапе осуществляют поиск компонентов, которые могли бы удовлетворить сформулированным требованиям. Обычно невозможно точно сопоставить функции, реализуемые готовыми компонентами, и функции, определенные спецификацией требований.
2. *Модификация требований.* На этой стадии анализируются требования с учетом информации о компонентах, полученной на предыдущем этапе. Требования модифицируются таким образом, чтобы максимально использовать возможности отобранных компонентов. Если изменение требований невозможно, повторно выполняется анализ компонентов для того, чтобы найти какое-либо альтернативное решение.
3. *Проектирование системы.* На данном этапе проектируется структура системы либо модифицируется существующая структура повторно используемой системы. Проектирование должно учитывать отобранные программные компоненты и строить структуру в соответствии с их функциональными возможностями. Если некоторые готовые программные компоненты недоступны, проектируется новое ПО.
4. *Разработка и сборка системы.* Это этап непосредственного создания системы. В рамках рассматриваемого подхода сборка системы является скорее частью разработки системы, чем отдельным этапом.

23. Контрольные вопросы и задания

1. Перечислите стадии разработки ПО и приведите основное содержание каждой из них.
2. Прокомментируйте понятия "*стадия разработки*", "*жизненный цикл ПО*" и "*модель жизненного цикла ПО*". Перечислите известные Вам модели ЖЦ ПО.
3. Опишите основные черты и области эффективного применения
   - каскадной модели.
   - модели формальной разработки
   - эволюционной модели.
   - модели пошаговой разработки.
   - спиральной модели.