ИНТЕГРИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Лекция №11

Программное обеспечение ИСУ

Процесс создания

Процесс создания программного обеспечения — это множество взаимосвязанных процессов и результатов их выполнения, которые ведут к созданию программного продукта.

Фундаментальные базовые процессы, без реализации которых не может обойтись ни одна технология разработки программных продуктов.

1. ***Разработка спецификации ПО.*** Это фундамент любой программной системы. Спецификация определяет все функции и действия, которые будет выполнять разрабатываемая система.
2. ***Проектирование и реализация (производство) ПО.*** Это процесс непосредственного создания ПО на основе спецификации.
3. ***Аттестация ПО.*** Разработанное программное обеспечение должно быть аттестовано на соответствие требованиям заказчика.
4. ***Эволюция ПО.*** Любые программные системы должны модифицироваться в соответствии с изменениями требований заказчика.

Модели процесса создания

Модель процесса создания программного обеспечения - это общее абстрактное представление данного процесса. Каждая такая модель представляет процесс создания ПО в каком-то своем "разрезе", используя только определенную часть всей информации о процессе.

1. ***Каскадная модель.*** Основные базовые виды деятельности, выполняемые в процессе создания ПО (такие, как разработка спецификации, проектирование и производство, аттестация и модернизация ПО), представляются как отдельные этапы этого процесса.

2. ***Эволюционная модель разработки ПО.*** Здесь последовательно перемежаются этапы формирования требований, разработки ПО и его аттестации. Первоначально программная система быстро разрабатывается на основе некоторых абстрактных общих требований. Затем они уточняются и детализируются в соответствии с требованиями заказчика. Далее система дорабатывается и аттестуется в соответствии с новыми уточненными требованиями.

Модели процесса создания

3. ***Модель формальной разработки систем.*** Основана на разработке формальной математической спецификации программной системы и преобразовании этой спецификации посредством специальных математических методов в исполняемые программы. Проверка соответствия спецификации и системных компонентов также выполняется математическими методами.

4. ***Модель разработки ПО на основе ранее созданных компонентов.*** Предполагает, что отдельные составные части программной системы уже существуют, т.е. созданы ранее. В этом случае технологический процесс создания ПО основное внимание уделяет интеграции отдельных компонентов в общее целое, а не их созданию.

Модели процесса создания

Каскадная и эволюционная модели разработки широко используются на практике. Модель формальной разработки систем успешно применялась во многих проектах, но количество организаций-разработчиков, постоянно использующих этот метод, невелико. 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 21 столетии, поскольку сборка систем из готовых или ранее использованных компонентов значительно ускоряет разработку ПО.

Каскадная модель

Основные принципиальные этапы (стадии) этой модели отражают все базовые виды деятельности, необходимые для создания ПО.

1. ***Анализ и формирование требований.*** Путем консультаций с заказчиком ПО определяется функциональные возможности, ограничения и цели создаваемой программной системы.

2. ***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.

3. ***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.

Каскадная модель

4. ***Сборка и тестирование системы.*** Отдельные программы и программные модули интегрируются и тестируются в виде целостной системы. Проверяется, соответствует ли система своей спецификации.

5. ***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.

Каскадная модель

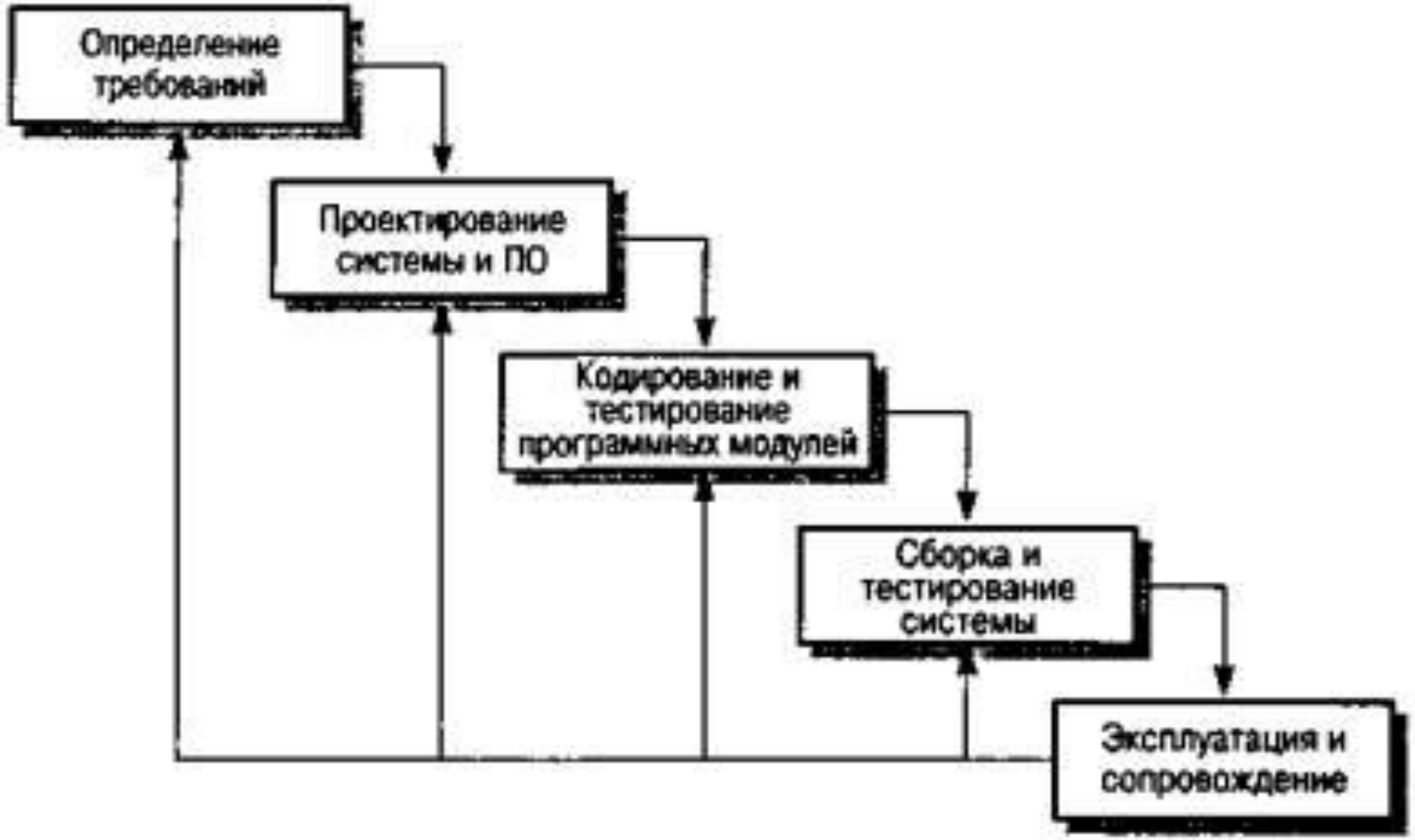


Рис. 3.1. Жизненный цикл программного обеспечения

Каскадная модель

Поскольку на каждом этапе проводятся определенные работы и оформляется сопутствующая документация, повторение этапов приводит к повторным работам и значительным расходам. Поэтому после небольшого числа повторений обычно ***"замораживается"*** часть этапов создания ПО, например этап определения требований, но продолжается выполнение последующих этапов. Возникающие проблемы, решение которых требует возврата к "замороженным" этапам, игнорируются либо делаются попытки решить их программно. "Замораживание" этапа определения требований может привести к тому, что разработанная система не будет удовлетворять всем требованиям заказчика. Это также может привести к появлению плохо структурированной системы, если упущения этапа проектирования исправляются только с помощью программистских хитростей.

Последний этап жизненного цикла ПО (эксплуатация и сопровождение) — это "полноценное" использование программной системы. На этом этапе могут обнаружиться ошибки, допущенные, например, на первом этапе формирования требований.

Каскадная модель

Могут также проявиться ошибки проектирования и кодирования, что может потребовать определения новых функциональных возможностей системы. С другой стороны, система постоянно должна оставаться работоспособной. Внесение необходимых изменений в программную систему может потребовать повторения некоторых или даже всех этапов процесса создания ПО.

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. Поэтому 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.

Эволюционная модель разработки

Эта модель основана на следующей идее: разрабатывается первоначальная версия программного продукта, которая передается на испытание пользователям, затем она дорабатывается с учетом мнения пользователей, получается промежуточная версия продукта, которая также проходит "испытание пользователем", снова дорабатывается и гак несколько раз, пока не будет получен необходимый программный *продукт*. Отличительной чертой данной модели является то, что ***процессы специфицирования, разработки и аттестации ПО выполняются параллельно*** при постоянном обмене информацией между ними. Различают два подхода к реализации эволюционного метода разработки.

Эволюционная модель разработки

1. ***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.

2. ***Прототипирование.*** Здесь целью процесса эволюционной разработки ПО является поэтапное уточнение требований заказчика и, следовательно, получение законченной спецификации, определяющей разрабатываемую систему. Прототип обычно строится для экспериментирования с той частью требований заказчика, которые сформированы нечетко или с внутренними противоречиями.

Эволюционная модель разработки

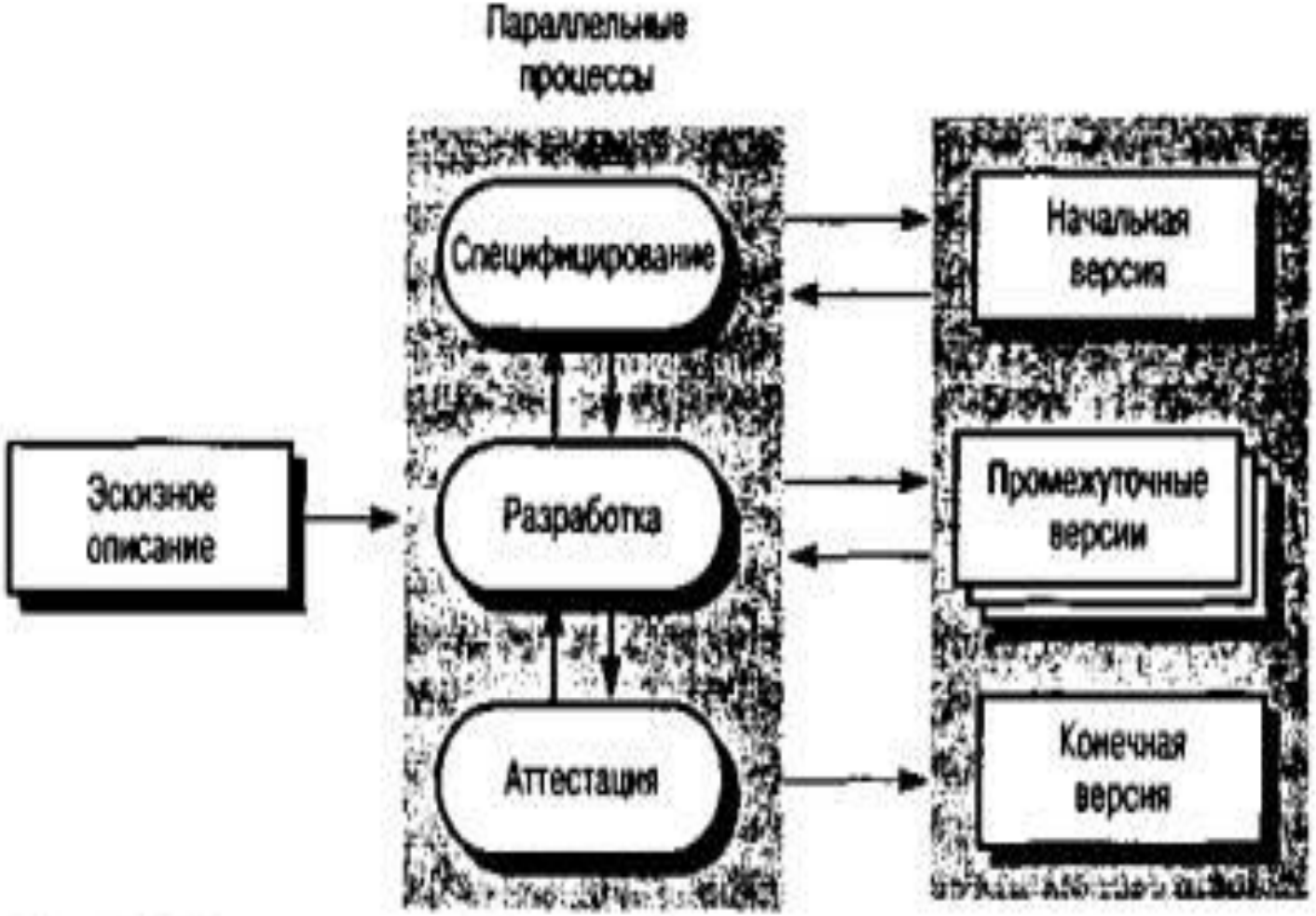


Рис. 3.2. Эволюционная модель разработки

Эволюционная модель разработки

Эволюционный подход часто более эффективен, чем подход, построенный на основе каскадной модели, особенно если требования заказчика могут меняться в процессе разработки системы.

Достоинством процесса создания ПО, построенного на основе эволюционного подхода, 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 данный подход имеет и некоторые ***недостатки.***

1. 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.
2. 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.

Эволюционная модель разработки

3. 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.

Эволюционный подход наиболее приемлем для разработки небольших программных систем (до 100 000 строк кода) и систем среднего размера (до 500 000 строк кода) с относительно коротким сроком жизни. На больших долгоживущих системах слишком заметно проявляются недостатки этого подхода. Для создания таких систем используется смешанный подход к созданию ПО, который вобрал бы в себя лучшие черты каскадной и эволюционной моделей разработки.

Формальная разработка систем

Этот подход к созданию ПО имеет много черт, сходных с каскадной моделью, но построен на основе формальных математических преобразований системной спецификации в исполняемую программу. Здесь для упрощения не указаны обратные связи между делами процесса.

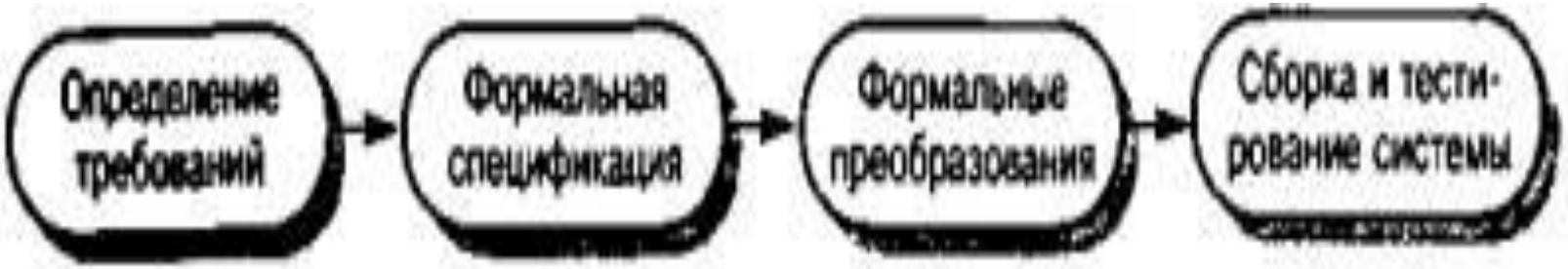


Рис. 3.3. Модель формальной разработки ПО

Формальная разработка систем

Между данным подходом и каскадной моделью существуют следующие кардинальные отличия.

1. Здесь спецификация системных требований имеет вид детализированной формальной спецификации, записанной с помощью специальной математической нотации.

2. Процессы проектирования, написания программного кода и тестирования системных модулей заменяются процессом, в котором формальная спецификация путем последовательных формальных преобразований трансформируется в исполняемую программу.

Разработка 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.

Разработка 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 осуществляется поиск компонентов, которые могли бы удовлетворить сформулированным требованиям. Обычно невозможно точно сопоставить функции, реализуемые готовыми компонентами, и функции, определенные спецификацией требований.

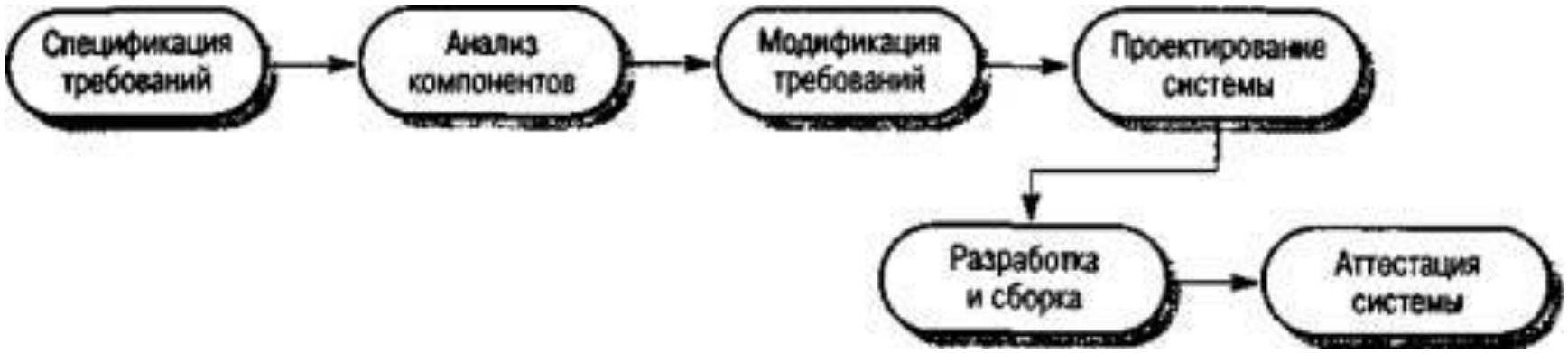
2. ***Модификация требований.*** На этой стадии анализируются требования с учетом информации о компонентах, полученной на предыдущем этапе. Требования модифицируются таким образом, чтобы максимально использовать возможности отобранных компонентов. Если изменение требований невозможно, повторно выполняется анализ компонентов для того, чтобы найти какое-либо альтернативное решение.

Разработка на основе ранее созданных компонентов

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.

Разработка на основе ранее созданных компонентов



Разработка ПО с повторным использованием ранее созданных компонентов

Разработка на основе ранее созданных компонентов

Основные достоинства описываемой модели процесса разработки ПО с повторным использованием ранее созданных компонентов заключаются в том, что сокращается количество непосредственно разрабатываемых компонентов и уменьшается общая стоимость создаваемой системы.

Вместе с тем при использовании этого подхода неизбежны компромиссы при определении требований; это может привести к тому, что законченная система не будет удовлетворять всем требованиям заказчика. Кроме того, при проведении модернизации системы (т.е. при создании ее новой версии) отсутствует возможность влиять на появление новых версий компонентов, используемых в системе, что значительно затрудняет сам процесс модернизации.

Итерационные модели разработки

Описанные модели процесса создания ПО имеют свои достоинства и недостатки. При создании больших систем, как правило, приходится использовать различные подходы к разработке разных частей системы, т.е. в целом к разработке системы применяются гибридные (смешанные) модели. Поэтому важную роль играет возможность выполнять процесс создания ПО итерационно, когда в ответ на изменения требований повторно выполняются определенные этапы создания системы.

1 ***Модель пошаговой разработки,*** где процессы спецификировании требований, проектирования и написания кода разбиваются на последовательность небольших шагов, которые ведут к созданию ПО.

2 ***Спиральная модель разработки,*** в которой весь процесс создания ПО, от начального эскиза системы до ее конечной реализации, разворачивается по спирали.

Итерационные модели разработки

Существенным отличием итерационных моделей является то, что здесь процесс работки спецификации протекает параллельно с разработкой самой программной системы. Более того, в модели пошаговой разработки полную системную спецификацию можно получить только после завершения самого последнего шага процесса создания ПО. Очевидно, что такой подход входит в противоречие с моделью приобретения ПО, когда полная системная спецификация является составной частью контракта на разработку системы. Поэтому, чтобы применять итерационные модели для разработки больших систем, которые заказываются "серьезными" организациями, например государственными агентствами, необходимо менять форму контракта, на что такие организации идут с большим трудом.

Модель пошаговой разработки

Модель пошаговой разработки

Процесс пошаговой разработки имеет целый ряд ***достоинств***.

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.

Модель пошаговой разработки

Вместе с тем при реализации пошаговой разработки могут возникнуть определенные проблемы. Компоненты, получаемые на каждом шаге разработки, имеют относительно небольшой размер (обычно не более 20 000 строк кода), но должны реализовывать какую-либо системную функцию. Отобразить множество системных требований к компонентам нужного размера довольно сложно. Более того, многие системы должны обладать набором базовых системных свойств, которые реализуются совместно различными частями системы. Поскольку требования детально не определены до тех пор, пока не будут разработаны все компоненты, бывает весьма сложно распределить общесистемные функции по компонентам.

Спиральная модель разработки

Каждый виток спирали разбит на четыре сектора.

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). В зависимости от "проявленных" рисков, могут планироваться альтернативные стратегии разработки ПО.

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 определен неверно, планируется разработать прототип системы.

Спиральная модель разработки

3. ***Разработка и тестирование.*** После оценки рисков выбирается модель процесса создания системы. Например, сети доминируют риски, связанные с разработкой интерфейсов, наиболее подходящей будет эволюционная модель разработки прототипированием. Если основные риски связаны с соответствием системы и спецификации, 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.

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.

Спиральная модель разработки

Существенное отличие спиральной модели от других моделей процесса создания ПО заключается в ***точном определении и оценивании рисков***. Если говорить неформально, то риск — это те неприятности, которые могут случиться в процессе разработки системы. 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. Таким образом, уменьшение (разрешение) рисков — важный элемент управления системным проектом.

Спиральная модель разработки

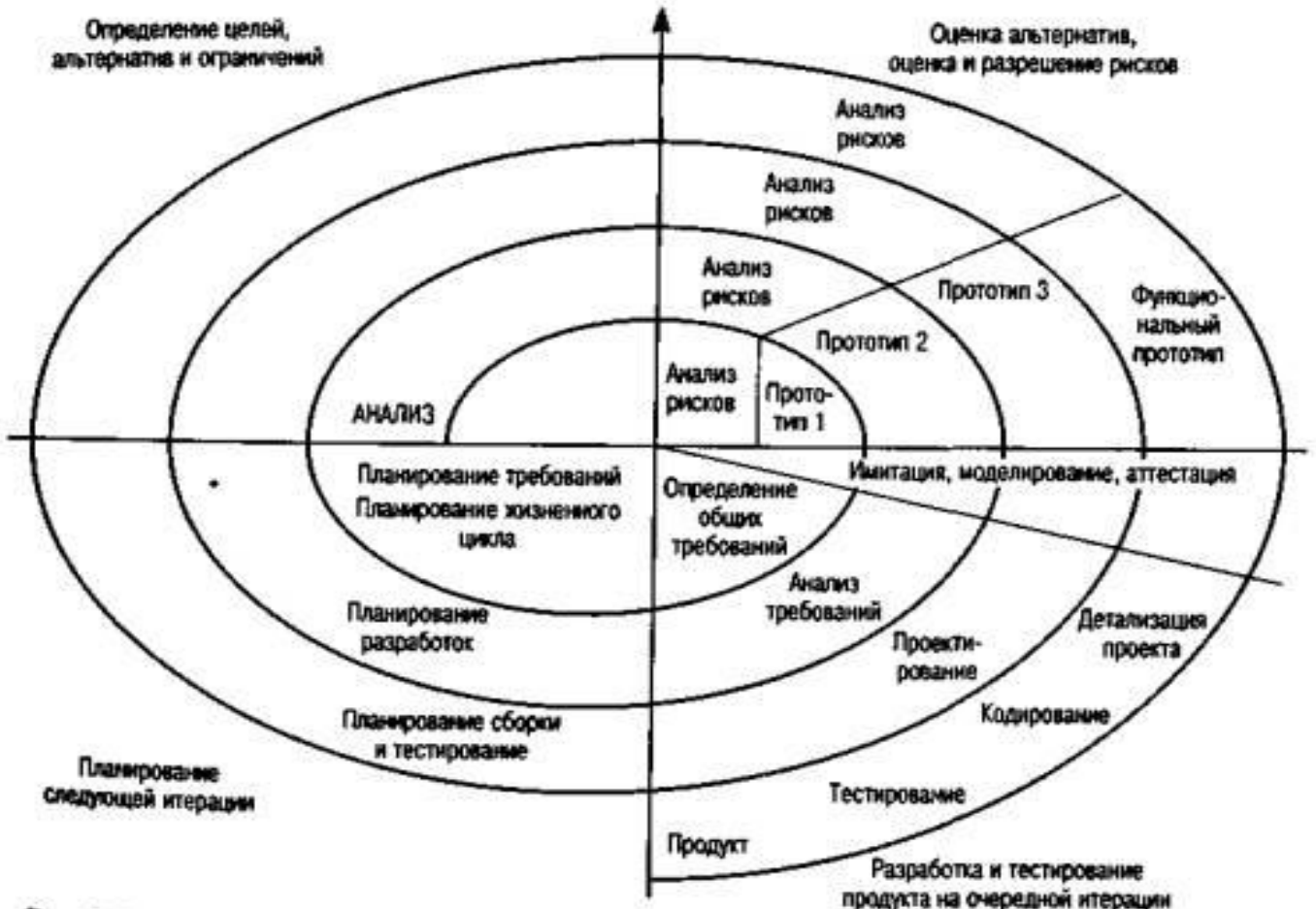


Рис. 3.7. Спиральная модель создания ПО (© 1988 IEEE)

Спецификация программного обеспечения

Рис. 3.8. Процесс разработки требований

Спецификация программного обеспечения

Предварительные исследования. Оценивается степень удовлетворенности пользователей существующими программными продуктами и аппаратными средствами, а также экономическая эффективность будущей системы и возможность уложиться в существующие бюджетные ограничения при ее разработке.

Формирование и анализ требований. Формируются системные требования путем изучения существующих аналогичных систем, обсуждения будущей системы с потенциальными пользователями и заказчиками, анализа задач, которые должна решать система, и т.п. Этот этап может включать разработку нескольких моделей системы и ее прототипов, что помогает сформировать функциональные требования к системе.

Специфицирование требований. Осуществляется перевод всей совокупности информации, собранной на предыдущем этапе, в документ, определяющий множество требований. Этот документ обычно содержит два типа требований: пользовательские — обобщенные представления заказчиков и конечных пользователей о системе; системные — детальное описание функциональных показателей системы.

Утверждение требований. Проверяется выполнимость, согласованность и полнота множества требований. В процессе формирования ограничений неизбежно возникновение каких-либо ошибок. На этом этапе они должны быть но возможности выявлены и устранены.

Проектирование и реализация ПО

Ниже перечислены отдельные этапы процесса проектирования.

1. ***Архитектурное проектирование.*** Определяются и документируются подсистемы и взаимосвязи между ними.
2. ***Обобщенная спецификация.*** Для каждой подсистемы разрабатывается обобщенная спецификация на ее сервисы и ограничения.
3. ***Проектирование интерфейсов.*** Для каждой подсистемы определяется и документируется ее интерфейс. Спецификации на эти интерфейсы должны быть точно выраженными и однозначными, чтобы использование подсистем не требовало знаний о том, как они реализуют свои функции.
4. ***Компонентное проектирование.*** Проводится распределение системных функций (сервисов) по различным компонентам и их интерфейсам.
5. ***Проектирование структур данных.*** Детально разрабатываются структуры данных, необходимые для реализации программной системы.
6. ***Проектирование алгоритмов.*** Детально разрабатываются алгоритмы, предназначенные для реализации системных сервисов.

Проектирование и реализация ПО

Рис. 3.9. Обобщенная схема процесса проектирования

Аттестация программных систем

Аттестация ПО, или более обобщенно — верификация и аттестация, предназначена показать ***соответствие системы ее спецификации***, а также ожиданиям и требованиям заказчика и пользователей.

За исключением небольших программ, программные системы невозможно протестировать как единый цельный программный элемент. Большие системы строятся на основе подсистем, которые, в свою очередь, строятся из модулей, модули же компонуются из программ-процедур и программ-функций. Для таких систем процесс тестирования выполняется постепенно, по мере реализации системы.

Процесс тестирования состоит из нескольких этапов.

1. ***Тестирование компонентов.*** Тестируются отдельные компоненты для проверки правильности их функционирования. Каждый компонент тестируется независимо от других.
2. ***Тестирование модулей.*** Программный модуль — это совокупность зависимых компонентов, таких как описание класса объектов, декларирование абстрактных типов данных и набор процедур и функций. Каждый модуль тестируется независимо от других системных модулей.

Аттестация программных систем

3. ***Тестирование подсистем.*** Тестируются наборы модулей, которые составляют отдельные подсистемы. Основная проблема, которая часто проявляется на этом этапе - несогласованность модульных интерфейсов. Поэтому при тестировании подсистем основное внимание уделяется обнаружению ошибок в модульных интерфейсах путем прогона их через все возможные режимы.
4. ***Тестирование системы.*** Из подсистем собирается конечная система. На этом этапе основное внимание уделяется совместимости интерфейсов подсистем и обнаружению программных ошибок, которые проявляются в виде непредсказуемого взаимодействия между подсистемами, а также оцениваются интеграционные характеристики системы.
5. ***Приемочные испытания.*** Это конечный этап процесса тестирования, после которого система принимается к эксплуатации. Здесь система тестируется с привлечением данных, предоставляемых заказчиком системы, а не на основе тестовых данных, как было на предыдущем этапе. На этом этапе могут проявиться ошибки, допущенные еще на этапе определения системных требований, поскольку испытания с реальными данными могут дать иной результат, чем тестирование со специально подобранными тестовыми данными.

Эволюция программных систем

Одна из основных причин того, что в настоящее время в большие сложные системы все шире внедряется программное обеспечение, заключается в ***гибкости программных систем.*** После принятия решения о разработке и производстве аппаратных компонентов системы внесение в них изменений становится весьма дорогостоящим. С другой стороны, ***в программное обеспечение можно вносить изменения в течение всего процесса разработки системы.*** Эти изменения также могут быть крайне дорогостоящими, но все-таки они значительно ***дешевле изменений в аппаратном оборудовании системы.***

Рис. 3.13. Эволюция систем