Введение в программную инженерию

Предмет изучения

Отличие программирования от ПрИнж

Программирование	Программная инженерия
Программировать для удовольствия; Программировать для того, чтобы научиться; Программировать в рамках научных разработок	Программирование для заказчика; Программирование в определенные **сроки**; Программирование с результатом **нужного качества**. → **дополнительные виды деятельности**: Разработка требований; Планирование; Тестирование; Конфигурационное управление; Проектный менеджмент; Создание различной документации (проектной, пользовательской и пр.)

Последовательность разработки программного кода:

Анализ - создание **функциональной модели** будущей системы для осознания программистами требований и ожиданий заказчика.

Проектирование - предварит. макет, эскиз, план системы на бумаге.

Вид проектов и предпочтений разработчиков и заказчиков влияют на трудозатраты на анализ и проектирование.

Нужны специальные усилия по организации процесса разработки.
Разработку системы необходимо выполнять с учетом удобств ее:

- *дальнейшего сопровождения*,
- *повторного использования* и
- *интеграции с другими системами*.

Система разбивается на компоненты, удобные в разработке, годные для повторного использования и интеграции.
Компоненты имеют необходимые характеристики по быстродействию.
Для компонент тщательно прорабатываются интерфейсы.
Сама система документируется на многих уровнях, создаются правила оформления программного кода.

Программная инженерия (software engineering)

- дополнительные виды деятельности, выполняемые в процессе промышленного программирования и необходимые для успешного выполнения заказов,
- некоторая практическая деятельность,
- специальная **область знания (научная дисциплина)** когда
  - для облегчения выполнения каждого отдельного проекта,
  - для возможности использовать разнообразный положительный опыт, достигнутый другими командами и разработчиками,

этот опыт подвергается осмыслению, обобщению и надлежащему оформлению.

Так появляются различные методы и практики (best practices):
- тестирования,
- проектирования,
- работы над требованиями и пр.,
- архитектурных шаблонов и пр.

А также стандарты и методологии, касающиеся всего процесса в целом (например, MSF и др.).

конец 60-х годов прошлого века	Рождение программной инженерии, 1968 г. - конференция NATO Software Engineering, г. Гармиш (ФРГ)

В сферу ПрИнж попадают все вопросы и темы, связанные с:

- организацией и улучшением процесса разработки ПО,
- управлением коллективом разработчиков,
- разработкой и внедрением программных средств поддержки ЖЦ разработки ПО.

- **Информатика** (computer science) – свод теоретических наук:
  - мат. логика,
  - теория грамматик,
  - методы построения компиляторов,
  - мат. формальные методы, используемые в верификации и модельном тестировании и т.д.

ПрИнж нацелена на решение проблем производства,

информатика – на разработку формальных, математизированных подходов к программированию.

- **Системотехника** (system engineering) объединяет различные инженерные дисциплины по разработке всевозможных искусственных систем – энергоустановок, телекоммуникационных систем, встроенных систем реального времени и т.д.

- Часто ПО оказывается частью таких систем, выполняя задачу управления соответствующего оборудования.

- Такие системы наз. *программно-аппаратными*, и участвуя в их создании, программисты вынуждены глубоко разбираться в особенностях соответствующей аппаратуры.

- **Бизнес-реинжиниринг** (business reengineering) – в широком смысле обозначает
  - модернизацию бизнеса в определенной компании,
  - внедрение новых практик, поддерживаемых соответствующими, новыми ИС.

При этом акцент м.б. как на внутреннем переустройстве компании так и на разработке нового клиентского сервиса.

- Бизнес-реинжиниринг часто предваряет разработку и внедрение ИС на предприятии, так как требуется сначала навести определенный порядок в делопроизводстве, а лишь потом закрепить его ИС.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

- множество развивающихся во времени логических предписаний, с помощью кот. коллектив управляет и использует **многопроцессорную** и **распределенную** систему выч. устройств (Харальд Миллсом, IBM):

- Логические предписания – программы + *документация* (по эксплуатации программ) – *определенная система отношений между людьми, использующих эти программы в рамках процесса деятельности*.
- Современ. ПО предназначено для одновременной работы со *многими пользователями*, кот. м. б. значительно удалены друг от друга в физич. пространстве. Вычислит. среда (ПК, серверы и т. д.), в кот. функционирует ПО, оказывается *распределенной*.
- Задачи, решаемые современ. ПО, требуют различных вычислит. ресурсов в силу различной специализации этих задач, из-за большого объема выполняемой работы, а также из соображений безопасности (появляется сервер БД, сервер приложений и пр., т. е. вычислит. среда, в кот. функционирует ПО, оказывается *многопроцессорной*).
- ПО развивается во времени – исправляются ошибки, добавляются нов. функции, выпускаются нов. версии, меняется его аппаратная база (версии).

СВОЙСТВА ПО

ПО – сложн. динамич. система, включающая технические, психологические и социальные аспекты.

ПО отличается от др. видов систем, создаваемых человеком – механических, социальных и пр.

Особенности ПО (Фредерик Брукс):

- Сложность программных объектов существенно ~ от их размеров.

- Согласованность – ПО д. б. согласовано с большим количеством интерфейсов, с кот. впоследствии оно д. взаимодействовать.

Интерфейсы плохо поддаются стандартизации, поскольку основываются на многочисленных и плохо формализуемых человеческих соглашениях.

- Изменяемость – ПО легко изменить, требования к нему постоянно меняются в процессе разработки. Это создает много дополнит. трудностей при его разработке и эволюции.

- Нематериальность – ПО виртуально.

Процесс разработки ПО

Центральный объект изучения ПрИнж - процесс создания ПО - множество различных:

- видов деятельности,
- методов,
- методик и
- шагов,

используемых для разработки и эволюции ПО и связанных с ним продуктов (*проектных планов*, *документации*, *программного кода*, *тестов*, *пользовательской документации* и пр.).

На сегодняшний день не существует универсального процесса разработки ПО

– набора методик, правил и предписаний, подходящих **для ПО любого вида**, **для любых компаний**, **для любых команд разработчиков**.

Перед началом проекта планируют процесс работы:

- определив роли и обязанности в команде,
- рабочие продукты (промежуточные и финальные),
- порядок участия в их разработке членов команды и т.д.

– это предварительное описание конкретного процесса, отличается от плана работ, проектных спецификаций и пр.

В рамках компании возможна и полезна стандартизация всех текущих процессов - **<u>стандартный процесс</u>**.

Т.о., создается некот. **БД**, содержащая:

- *информацию*, *правила использования*, *документацию* и *инсталляционные пакеты средств разработки*, используемые в проектах компании (систем версионного контроля, средств контроля ошибок, средств программирования – различных СУБД и т.д.);
- *описание практик разработки* – проектного менеджмента, правил работы с заказчиком и т.д.;
- *шаблоны проектных документов* – технич. заданий, проект. спецификаций, планов тестирования и т.д. и пр.

Совершенствование процесса (software process improvement) – деятельность по изменению существующего процесса с целью улучшения **качества** создаваемых продуктов и/или снижения **цены** и **времени** их разработки.

Причины актуальности:

- Происходит быстрая смена технологий разработки ПО, требуются изучение и внедрение новых средств разработки.
- Наблюдается быстрый рост компаний и их выход на новые рынки, что требует новой организации работ.
- Имеет место высокая конкуренция, кот. требует поиска более эффективных, более экономичных способов разработки.

Элементы совершенствования

- Переход на новые
  - средства разработки,
  - языки программирования и т.д.
- Улучшение отдельных управленческих и инженерных практик:
  - тестирования,
  - управления требованиями и пр.
- Полная, комплексная перестройка всех процессов в проекте, департаменте, компании (в соответствии, напр., со стандартами CMMI).
- Сертификация компании (CMM/CMMI, ISO 9000 и пр.).

Фазы и виды деятельности

- Фаза – определенный этап процесса: имеет **начало, конец и выходной результат** (напр., фаза сдачи проекта)

Фазы

- следуют друг за другом,
- характеризуются предоставлением отчетности заказчику,
- часто, выплатой денег за выполненную часть работы.

- Вид деятельности – определенный тип работы, выполняемый в процессе разработки ПО.

Разные виды деятельности:

- требуют разные профессиональные навыки и
- выполняются разными специалистами (напр., управление проектом выполняется менеджером проекта, кодирование – программистом, тестирование – тестировщиком).

Классические модели процесса

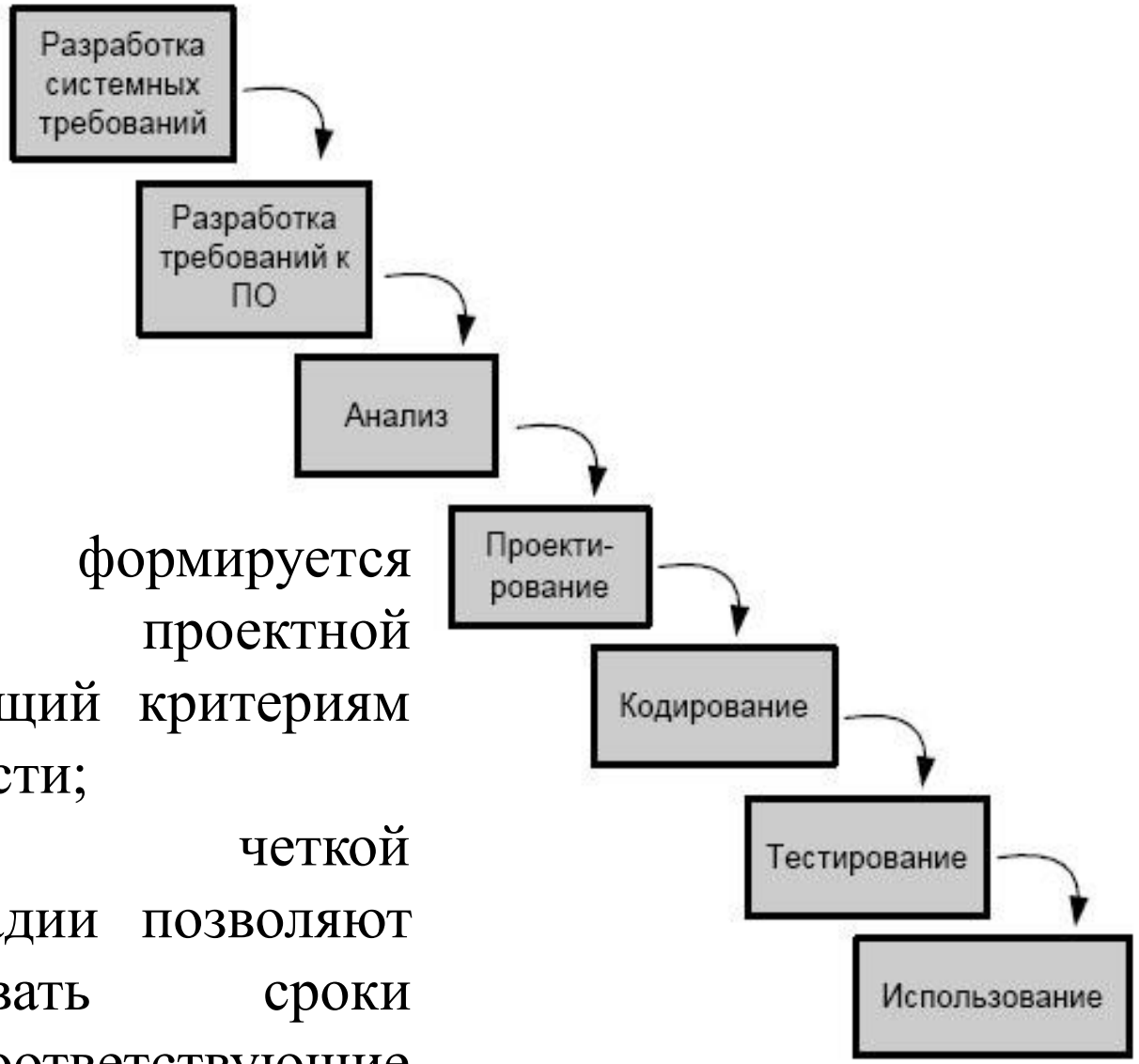
Процесс создания ПО не является однородным.

Метод разработки ПО определяет модель процесса.

Модель явл. хорошей абстракцией различных методов разработки ПО, позволяя лаконично, сжато и информативно их представить.

Существует несколько классических моделей процесса.

Водопадная модель была предложена в 1970 году Винстоном Ройсом



Достоинства модели:

- на каждой стадии формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в четкой последовательности стадии позволяют уверенно планировать сроки выполнения работ и соответствующие ресурсы (денежные, материальные и людские).

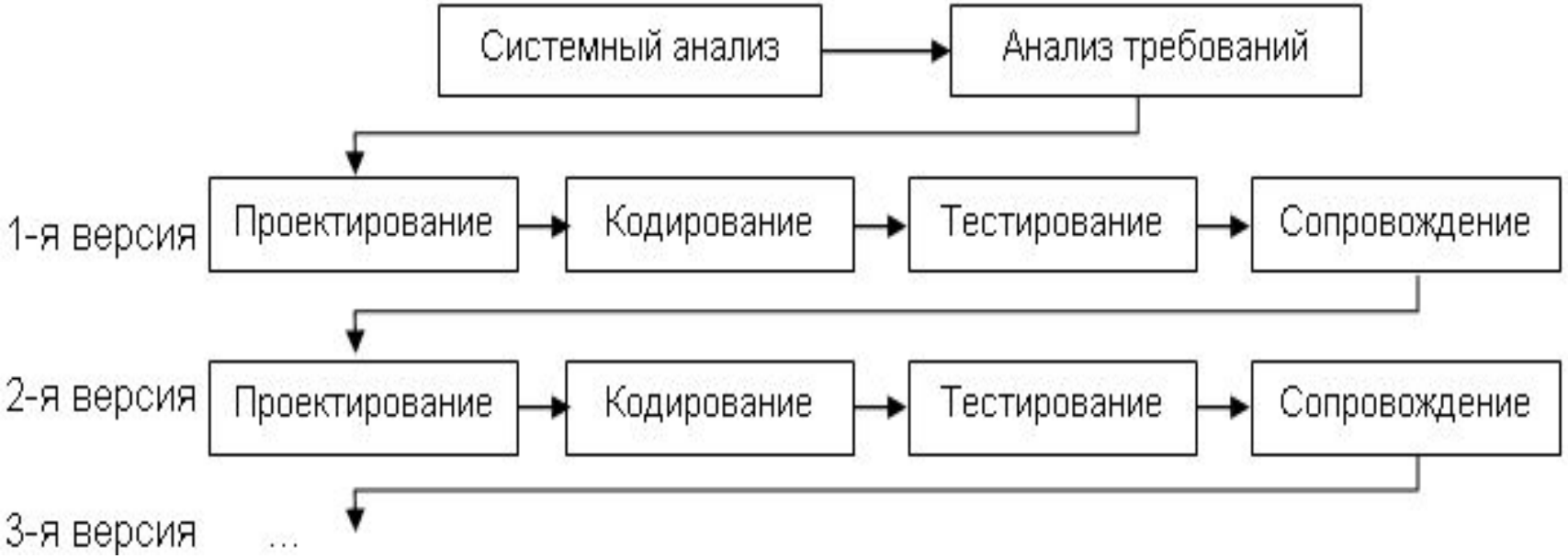
Недостатки водопадной модели являются:

- отождествление фаз и видов деятельности, что влечет потерю гибкости разработки, в частности, трудности поддержки итеративного процесса разработки;
- требование полного окончания фазы-деятельности, закрепление результатов в виде подробного исходного документа (технического задания, …); опыт разработки ПО показывает, что невозможно полностью завершить разработку требований, дизайн системы и т.д. – все это подвержено изменениям; не только потому, что подвижно окружение проекта, но и потому, что заранее не удается точно определить и сформулировать многие решения, они проясняются и уточняются лишь впоследствии;
- интеграция результатов разработки происходит в конце, интеграционные проблемы дают о себе знать слишком поздно;
- пользователи и заказчик видят результат только в самом конце; не могут повлиять на процесс создания системы, увеличиваются риски непонимания между разработчиками и пользователями/заказчиком;
- модель неустойчива к сбоям в финансировании проекта или перераспределению денежных средств, начатая разработка, фактически, не имеет альтернатив "по ходу дела".

Водопадная модель вошла в качестве составной части в другие модели и методологии, например, в MSF.

Инкрементная стратегия

- подразумевает разработку ИС с линейной последовательностью стадий, но в несколько инкрементов (версий), т. е. с запланированным улучшением продукта.



- определяются основные требования к системе,
- выполняется ее разработка в виде последовательности версий,
- каждая версия явл. законченным и работоспособным продуктом.
- 1 версия реализует часть запланированных возможностей,
- следующая версия реализует дополнительные возможности и т. д., пока не будет получена полная система.

Достоинства и **недостатки** этой стратегии такие же, как и у классической.

Но в отличие от классической стратегии заказчик м. раньше увидеть результаты.

Уже по результатам разработки и внедрения первой версии он м. незначительно изменить требования к разработке, отказаться от нее или предложить разработку более совершенного продукта с заключением нового договора.

Спиральная модель была предложена Бэри Боемом в 1988 году для преодоления недостатков водопадной модели.

Структура витка (секторы):

- определение целей, ограничений и альтернатив проекта;
- оценка альтернатив, оценка и разрешение рисков; возможно использование прототипирования (создание серии прототипов), симуляция системы, визуальное моделирование и анализ спецификаций; фокусировка на самых рисковых частях проекта;
- разработка и тестирование – здесь возможна водопадная модель или использование иных моделей и методов разработки ПО;
- планирование следующих итераций – анализируются результаты, планы и ресурсы на последующую разработку, принимается (или не принимается) решение о новом витке; анализируется, имеет ли смысл продолжать разрабатывать систему или нет; разработку можно и приостановить, например, из-за сбоев в финансировании; спиральная модель позволяет сделать это корректно.

Спиральная модель является первой итеративной моделью, имеет красивую метафору – спираль, – и, подобно водопадной модели, использовалась в дальнейшем при создании других моделей процесса и методологий разработки ПО.

Спиральная стратегия

Спиральная стратегия (эволюционная или итерационная модель) подразумевает разработку в виде последовательности версий, но в начале проекта определены не все требования.

Достоинства модели:

- позволяет быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований;
- допускает изменение требований при разработке ИС, что характерно для большинства разработок, в том числе и типовых;
- обеспечивает большую гибкость в управлении проектом;
- позволяет получить более надежную и устойчивую систему. По мере развития системы ошибки и слабые места обнаруживаются и исправляются на каждой итерации;
- позволяет совершенствовать процесс разработки – анализ, проводимый в каждой итерации, позволяет проводить оценку того, что должно быть изменено в организации разработки, и улучшить ее на следующей итерации;
- уменьшаются риски заказчика. Заказчик м. с минимальными для себя финансовыми потерями завершить развитие неперспективного проекта.

Недостатки модели:

- увеличивается неопределенность у разработчика в перспективах развития проекта. Этот недостаток вытекает из предыдущего достоинства модели;
- затруднены операции временного и ресурсного планирования всего проекта в целом. Для решения этой проблемы необходимо ввести временные ограничения на каждую из стадий ЖЦ. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа выполнена. План составляется на основе статистических данных, полученных в предыдущих проектах и личного опыта разработчиков.