Лекция 1 Программная инженерия: назначение, основные принципы и понятия

1.1 Предпосылки и история

В конце 60-х – начале 70-х годов прошлого века произошло событие, которое вошло в историю как первый кризис программирования. Событие состояло в том, что стоимость программного обеспечения стала приближаться к стоимости аппаратуры («железа»), а динамика роста этих стоимостей позволяла прогнозировать, что к середине 90-годов все человечество будет заниматься разработкой программ для компьютеров. Тогда и заговорили о программной инженерии (или технологии программирования, как это называлось в России) как о некоторой дисциплине, целью которой является сокращение стоимости программ.

С тех пор программная инженерия прошла достаточно бурное развитие. Этапы развития программной инженерии можно выделять по-разному. Каждый этап связан с появлением (или осознанием) очередной проблемы и нахождением путей и способов решения этой проблемы. На слайде представлены ряд фундаментальных проблем разработки программ и найденных фундаментальных методов их решения. Эти методы и по сей день составляют основу подходов к проектированию программных продуктов.

1.1.1. Повторное использование кода (модульное программирование)

Проблема. На первых этапах становления программной инженерии (даже когда она так еще не называлась) было отмечено, что высокая стоимость программ связана с разработкой одинаковых (или похожих) фрагментов кода в различных программах. Вызвано это было тем, что в различных программах как части этих программ решались одинаковые (или похожие) задачи: решение нелинейных уравнений, расчет заработной платы, … Использование при создании новых программ ранее написанных фрагментов сулило существенное снижение сроков и стоимости разработки.

Модульное программирование. Главный принцип модульного программирования состоял в выделении таких фрагментов и оформлении их в виде модулей. Каждый модуль снабжался описанием, в котором устанавливались правила его использования – интерфейс модуля. Интерфейс задавал связи модуля с основной программой – связи по данным и связи по управлению. При этом возможность повторного использования модулей определялась количеством и сложностью этих связей, или насколько эти связи удалось согласовывать с организацией данных и управления основной программы. Наиболее простыми в этом отношении оказались модули решения математических задач: решения уравнений, систем уравнений, задач оптимизации. К настоящему времени накоплены и успешно используются большие библиотеки таких модулей.

Для многих других типов модулей возможность их повторного использования оказалась проблематичной в виду сложности их связей с основной программой. Например, модуль расчета зарплаты, написанный для одной фирмы, может не подойти для другой, т.к. зарплата в этих фирмах рассчитывается не во всем одинаково. Повторное использование модулей со сложными интерфейсами является достаточно актуальной и по сей день. Для ее решения разрабатываются специальные формы (структуры) представления модулей и организации их интерфейсов.

1.1.2. Рост сложности программ (структурное программирование)

Проблема. Следующий этап возрастания стоимости ПО был связан с переходом от разработки относительно простых программ к разработке сложных программных ком-

плексов. К числу таких сложных программ относятся: системы управления космическими объектами, управления оборонным комплексом, автоматизации крупного финансового учреждения и т.д. Сложность таких комплексов оценивалась следующими показателями:

1. Большой объем кода (миллионы строк)
2. Большое количество связей между элементами кода
3. Большое количество разработчиков (сотни человек)
4. Большое количество пользователей (сотни и тысячи)
5. Длительное время использования

Для таких сложных программ оказалось, что основная часть их стоимости приходится не на создание программ, а на их внедрение и эксплуатацию. По аналогии с промышленной технологией стали говорить о жизненном цикле программного продукта, как о последовательности определенных этапов: этапа проектирования, разработки, тестирования, внедрения и сопровождения.

Структурное программирование. Этап сопровождения программного комплекса включал действия по исправлению ошибок в работе программы и внесению изменений в соответствии с изменившимися требованиями пользователей. Основная причина высокой стоимости (а порой и невозможности выполнения) этапа сопровождения состояла в том, что программы были плохо спроектированы – документация была не понятна и не соответствовала программному коду, а сам программный код был очень сложен и запутан. Нужна технология, которая обеспечит «правильное» проектирование и кодирование. Основные принципы технологии структурного проектирования и кодирования:

1. Нисходящее функциональное проектирование, при котором в системе выделяются основные функциональные подсистемы, которые потом разбиваются на подсистемы и т.д. (принцип «разделяй и властвуй»)
2. Применение специальных языков проектирования и средств автоматизации использования этих языков
3. Дисциплина проектирования и разработки: планирование и документирование проекта, поддержка соответствие кода проектной документации
4. Структурное кодирование без goto

1.1.3. Модификация программ (ООП)

Проблема. Следующая проблема роста стоимости программ была вызвана тем, что изменение требований к программе стали возникать не только на стадии сопровождения, но и на стадии проектирования – проблема заказчика, который не знает, что он хочет. Создание программного продукта превратилось в его перманентное перепроектирование. Возник вопрос, как проектировать и писать программы, чтобы обеспечить возможность внесений изменений в программу, не меняя ранее написанного кода.

Объектно-ориентированное программирование. Решением этой проблемы стало использование подхода или метода, который стали называть объектно-ориентированным проектированием и программированием. Суть подхода состоит в том, что вводится понятие класса как развитие понятия модуля с определенными свойствами и поведением, характеризующими обязанностями класса. Каждый класс может порождать объекты – экземпляры данного класса. При этом работают основные принципы (парадигмы) ООП:

1. Инкапсуляция – объединение в классе данных (свойств) и методов (процедур обработки).
2. Наследование – возможность вывода нового класса из старого с частичным изменением свойств и методов
3. Полиморфизм – определение свойств и методов объекта по контексту

Проиллюстрировать возможности принципов ООП можно на следующем примере. В организации, состоящей из трех отделов надо начислять заработную плату. В программе каждый отдел представлен своим модулем – объектом, а начисление зарплаты – объектом «Зарплата». При необходимости расчета зарплаты объекту «Отдел» передается экзем-

пляр объекта «Зарплата». Объект «Отдел» передает объекту «Зарплата» необходимые данные и затем с помощью методов объекта «Зарплата» выполняет необходимые расчеты.

В отделе 3 частично изменились правила начисления зарплаты. В этой ситуации при объектно-ориентированном подходе из класса «Зарплата» выводится класс «Зарплата 1», который наследует неизменившиеся правила начисления зарплаты и переопределяет изменившиеся. Здесь при расчете зарплаты объектам «Отдел 1» и «Отдел 2» будет передаваться объект «Зарплата», а объекту «Отдел 3» - объект «Зарплата 1». При таких изменениях:

1. Срабатывает принцип наследования: код «Зарплата», «Отдел 1» и «Отдел 2» остаются без изменения, а код «Зарплата 1» изменяется ровно настолько, насколько это необходимо.
2. Срабатывает принцип полиморфизма: код «Отдел 3» также не изменяется – он продолжает считать, что работает с объектом «Зарплата»

1.1.4. Некоторые итоги

Программная инженерия (или технология программирования) как некоторое направление возникло и формировалось под давлением роста стоимости создаваемого программного обеспечения. Главная цель этой области знаний - сокращение стоимости и сроков разработки программ.

Программная инженерия прошла несколько этапов развития, в процессе которых были сформулированы фундаментальные принципы и методы разработки программных продуктов. Основной принцип программной инженерии состоит в том, что программы создаются в результате выполнения нескольких взаимосвязанных этапов (анализ требований, проектирование, разработка, внедрение, сопровождение), составляющих жизненный цикл программного продукта. Фундаментальными методами проектирования и разработки являются модульное, структурное и объектно-ориентированное проектирование и программирование.

1.1.5. Продолжение кризиса программирования

Несмотря на то, что программная инженерия достигла определенных успехов, перманентный кризис программирования продолжается. Связано это с тем, рубеж 80–90-х годов отмечается как начало информационно-технологической революции, вызванной взрывным ростом использования информационных средств: персональный компьютер, локальные и глобальные вычислительные сети, мобильная связь, электронная почту, Internet и т.д.

Цена успеха – кризис программирования принимает хронические формы:

- США тратит ежегодно более $200 млрд. на более чем 170 тыс. проектов разработки ПО в сфере IT;
- 31,1% из них закрываются, так и не завершившись; 52,7% проектов завершаются с превышением первоначальных оценок бюджета/сроков и ограниченной функциональностью;
- потери от недополученного эффекта внедрения ПО измеряются триллионами.

Статистика по 30,000 проектам по разработке ПО в американских компаниях показывает следующее распределение между:

- *У*спешными – вовремя и в рамках бюджета был выполнен весь намеченный фронт работ
- Проблемными – нарушение сроков, перерасход бюджета и/или сделали не все, что требовалось
- Проваленными – не были доведены до конца из-за перерасхода средств, бюджета, качества.

Источник: The Standish Group International *The Standish Group International, Inc., Extreme Chaos, 2000 - http://www1.standishgroup.com//sample_research/PDFpages/extreme_chaos.pdf*

1.2 Программная инженерия – что это такое?

1.2.1. Начнем с определений

На сегодняшний день нет единого определения понятия «программная инженерия». На слайде приведено несколько таких определений, данных крупными специалистами в этой области, или зафиксированные в документах ведущих организаций.

Сам термин – software engineering (программная инженерия) - впервые был озвучен в октябре 1968 года на конференции подкомитета НАТО по науке и технике (г.Гармиш, Германия). Присутствовало 50 профессиональных разработчиков ПО из 11 стран. Рассматривались проблемы проектирования, разработки, распространения и поддержки программ. Там впервые и прозвучал термин «программная инженерия» как некоторая дисциплина, которую надо создавать и которой надо руководствоваться в решении перечисленных проблем.

Вскоре после этого в Лондоне состоялась встреча 22-х руководителей проектов по разработке ПО. На встрече анализировались проблемы и перспективы развития ПО. Отмечалась возрастающее воздействие ПО на жизнь людей. Впервые серьезно заговорили о надвигающемся кризисе ПО. Применяющиеся принципы и методы разработки ПО требовали постоянного усовершенствования. Именно на этой встрече была предложена концепция жизненного цикла ПО (SLC – Software Lifetime Cycle) как последовательности шагов-стадий, которые необходимо выполнить в процессе создания и эксплуатации ПО. Вокруг этой концепции было много споров. В 1970 г. У.У. Ройс (W.W. Royce) произвел идентификацию нескольких стадий в типичном цикле и было высказано предположение, что контроль выполнения стадий приведет к повышению качества ПО и сокращению стоимости разработки.

1.2.2. Разберемся в вопросах

Для того, чтобы получить представление о том, что такое программная инженерия, попробуем разобраться в следующих вопросах:

- Что такое программное обеспечение (software)?
- Что такое программная инженерия?
- В чем разница между программной инженерией (software engineering) и информатикой (computer science)?
- В чем разница между программной инженерией и системной инженерией (systems engineering)?
- В чем отличие программной инженерии от других инженерий?

1.2.3. Что такое программное обеспечение (software)?

Программное обеспечение это набор компьютерных программ, процедур и связанной с ними документации и данных (ISO/IEC 12207). Взгляд на ПО как только на программу, сидящую в компьютере слишком узок. Дело в том, что продается (поставляется) не только программа, но еще и документация, в которой можно прочитать как установить программу и как ей пользоваться и данные для установки программы в различных условиях (конфигурационные файлы). Поэтому ПО иногда называют программным продуктом. Т.е. программный продукт (программное обеспечение) – это не только программы, а также вся связанная с ними документация и конфигурационные данные, необходимые для корректной работы программы. А специалисты по программному обеспечению разрабатывают программные продукты, т.е. такое ПО, которое может быть продано потребителю.

В зависимости от того, для кого разрабатываются программные продукты (конкретного заказчика или рынка, программные продукты бывают двух типов:

- коробочные продукты (generic products – общие продукты или shrink-wrapped software – упакованное ПО)

- заказные продукты (bespoke – сделанный на заказ или customized products – настроенный продукт). Важная разница между ними заключается в том, кто ставит задачу (определяет, или специфицирует требования). В первом случае это делают сами разработчики на основе анализа рынка (маркетинга) – и при этом рискуют сами. Во втором – заказчик и при этом рискует, что разработчик не сможет реально выполнить все требования в срок и при выделенном бюджете.

1.2.4. Что такое программная инженерия?

Программная инженерия — это инженерная дисциплина, которая связана со всеми аспектами производства ПО от начальных стадий создания спецификации до поддержки системы после сдачи в эксплуатацию. В этом определении есть две ключевые фразы:

- Инженерная дисциплина
- Все аспекты производства ПО

Инженерная дисциплина. Инженеры – это те специалисты, которые выполняют практическую работу и добиваются практических результатов. Ученый может сказать: проблема неразрешима в рамках существующих теорий и это будет научный результат, достойный опубликования и защиты диссертации.

Для решения задачи инженеры применяют теории, методы и средства, пригодные для решения данной задачи, но они применяют их выборочно и всегда пытаются найти решения, даже в тех случаях, когда теорий или методов, соответствующих данной задаче, еще не существует. В этом случае инженер ищет метод или средство для решения задачи, применяет его и несет ответственность за результат – ведь метод или средство еще не проверены. Набор таких инженерных методов или способов, теоретически возможно не обоснованных, но получивших неоднократное подтверждение на практике, играет большую практическую роль. В программной инженерии они получили название лучших практик (best practices).

Инженеры работают в условиях ограниченных ресурсов: временных, финансовых и организационных (оборудование, техника, люди). Иными словами, продукт должен быть создан в установленные сроки, в рамках выделенных средств, оборудования и людей. Хотя это в первую очередь относится к созданию заказных продуктов (оговаривается в условиях контракта), но при создании коробочных продуктов эти ограничения имеют не меньшее значение, т.к. здесь они диктуются условиями рыночной конкуренции.

Все аспекты производства ПО. Программная инженерия занимается не только техническими вопросами производства ПО (специфицирование требований, проектирование, кодирование,…), но и управлением программными проектами, включая вопросы планирования, финансирования, управления коллективом и т.д. Кроме того, задачей программной инженерии является разработка средств, методов и теорий для поддержки процесса производства ПО.

Программные инженеры применяют систематичные и организованные подходы к работе для достижения максимальной эффективности и качества ПО. Их задача состоит в адаптации существующих методов и подходов к решению свой конкретной проблемы.

1.2.5. В чем отличия от информатики?

Информатика (computer science) занимается теорией и методами вычислительных и программных систем, в то время как программная инженерия занимается практическими проблемами создания ПО. Информатика составляет теоретические основы программной инженерии и инженер по программному обеспечению должен знать информатику. Так же, как инженер по электронике должен знать физику. В идеале, программная инженерия должна быть поддержана какими-то теориями информатики, но самом деле это не всегда так. Программные инженеры зачастую используют приемы, которые применимы только в конкретных условиях и не могут быть обобщены на другие случаи, а элегантные теории информатики не всегда могут быть применены к реальным большим системам.

И наконец, информатика – это не единственный теоретический фундамент программной инженерии, т.к. круг проблем, стоящих перед программным инженером значительно шире просто написания программ. Это еще управление финансами, организация работ в коллективе, взаимодействие с заказчиком и т.д. Решение этих проблем требуют фундаментальных знаний, выходящих за рамки информатики.

1.2.6. В чем отличие от других инженерий?

Отличие программной инженерии от других инженерий интересно прежде всего с точки зрения двух вопросов:

- Почему доля провальных проектов в программной инженерии так велика по сравнению с другими инженериями?
- Можно ли в программной инженерии применять опыт других инженерий?

Эти вопросы является фундаментальными для программной инженерии. По этому поводу высказывается много мнений (и часто противоположных). Остановимся на некоторых более или менее очевидных отличиях программной инженерии от других инженерий.

Прежде всего, отметим, что жизненный цикл продукта любой инженерии в упрощенном виде включает фазы: проектирование, создание образца, испытание, производство, эксплуатация.

Компьютерная программа – это (в отличие от объектов других инженерий) не материальный объект (просьба не путать с носителем программы – устройством памяти любого типа). Отсюда следуют следующие отличия. Фаза производства состоит в копировании образца на другие носители. Стоимость фазы исчезающее мала. Если кодирование считать элементом проектирования (что очень близко к истине), то отсутствует также и фаза создания образца (строится компилятором и линковщиком)

Отсюда следуют следующие выводы:

- Стоимость программы – это стоимость только ее проектирования
- Стоимость проектирования коробочных продуктов «размазывается» по копиям
- Стоимость заказных продуктов (массово не копируемых) остается высокой

1.2.7. В чем еще отличие от других инженерий?

Второе существенное отличие состоит в том, что программа – искусственный объект. Т.е. для программы нет объективных законов, которым бы подчинялось ее поведение. Например, у инженера – строителя есть объективные законы строительной механики: равновесия моментов и сил, устойчивости механических систем и т.д. Инженер – строитель может проверить свои архитектурные решения на соответствие этим законам и тем самым обеспечить удачу проекта. Эти законы объективны, они будут действовать всегда. У программного инженера на первый взгляд также есть типовые, проверенные временем архитектурные решения (например, клиент-серверная архитектура). Но эти решения определяются уровнем развития вычислительной техники (и адекватным им уровнем требований). С появлением техники с принципиально новыми возможностями программному инженеру придется искать новые решения.

Прямым следствием отсутствия возможности «теоретического» контроля проекта является то, что тестирование продукта – это единственный способ убедиться в его качестве. Именно поэтому стоимость тестирования составляет существенную стоимость ПО. Кстати, строительный инженер, как правило, лишен возможности такого «тестирования» своего продукта перед сдачей его в эксплуатацию.

Ну и наконец, программная инженерия – молодая дисциплина, опыт которой насчитывает всего несколько десятков лет. По сравнению с опытом строительной инженерии (тысячелетия) это конечно очень мало. Программную инженерию иногда сравнивают с ранней строительной, когда законы строительной механики еще не были известны и

строительные инженеры действовали методом проб и ошибок, накапливая бесценный опыт. Несмотря на молодой возраст, программная инженерия также накопила определенный опыт, который позволяет (при разумном его применении) делать удачные проекты. Этот опыт выражен в основных принципах программной инженерии, которые мы с вами сейчас рассмотрим.

Подробнее о проблемах проектирования ПО можно посмотреть в неоднозначной статье Кони Бюрера «От ремесла к науке: поиск основных принципов разработки ПО» http://interface.ru/fset.asp?Url=/rational/science.htm

1.2.8. Из чего складывается стоимость ПО?

Структура стоимости ПО существенно зависит от типа ПО, применяемых методов его разработки и … метода оценки. Так, многие авторы отмечают высокую долю стоимости этапа сопровождения. Для некоторых типов ПО она может составлять 60 и более процентов от общей стоимости. Между тем, этап сопровождения включает выполнение двух видов работ: исправление ошибок в программе (несоответствий первоначальным требованиям) и внесение изменений в программу (добавление новых требований). При другом подходе к оценке можно считать, что этап сопровождения на стоит оценивать отдельно, т.к. исправление ошибок можно отнести к продолжению тестирования, а внесение изменений – к новому проекту.

Типовое распределение стоимости между основными этапами (без сопровождения) выглядит следующим образом:

- 15% - спецификация – формулировка требований и условий разработки
- 25% - проектирование – разработка и верификация проекта
- 20% - разработка – кодирование и тестирование компонент
- 40% - интеграция и тестирование – объединение и сборочное тестирование продукта

Отклонения от этой схемы в зависимости от типа ПО выглядят следующим образом:

Для коробочного ПО характерна более высокая доля тестирования за счет сокращения прежде всего доли спецификации (до 5%)

Распределение стоимости заказного ПО зависит от его сложности. При сложном ПО также возрастает доля интеграции и тестирования, но за счет сокращения доли проектирования и разработки Доля спецификаций может возрастать. Сокращение доли проектирования и разработки достигается за счет применения опробованных проектных решений и повторного использования готовых компонент.

Применение опробованных решений и готовых компонент при создании коробочных продуктов позволяет повысить качество и сократить сроки разработки.

1.2.9. Еще вопросы

Для того, чтобы получить представление о том, в чем состоит накопленный программной инженерией опыт, попробуем разобраться в следующих вопросах:

- Что такое программный процесс?
- Что такое модель программного процесса?
- Что такое методы программной инженерии?
- Что такое CASE (Computer-Aided Software Engineering)?
- Какими свойствами обладает хорошая программа?
- Какие основные трудности стоят перед программной инженерией?

1.2.10. Программный процесс?

Одним из основных понятий программной инженерии является понятие жизненного цикла программного продукта и программного процесса.

Жизненный цикл – непрерывный процесс, начинающийся с момента принятия решения о создании ПО и заканчивающийся снятием его с эксплуатации. Жизненный цикл разбивается на отдельные процессы.

Процесс – совокупность действий и задач, имеющих целью достижение значимого результата. Основными процессами (иногда называют этапами или фазами) жизненного цикла являются:

- Разработка спецификации требований (результат – описания требований к программе, которые обязательны для выполнения – описание того, что программа должна делать)
- Разработка проекта программы (результат – описание того, как программа будет работать)
- Кодирование (результат – исходный код и файлы конфигурации)
- Тестирование программы (результат - контроль соответствия программы требованиям)
- Документирование (результат – документация к программе)

Кроме основных, существует много дополнительных и вспомогательных процессов, связанных не созданием продукта, а с организацией работ (нефункциональные процессы): создание инфраструктуры, управление конфигурацией, управление качеством, обучение, разрешение противоречий, …

Процесс должен быть установлен. Полное установление процесса предполагает:

- Описание процесса – детальное описание действий и операций процесса
- Обучение процессу – проведение занятий с персоналом по освоению процесса
- Введение метрик – установление количественных показателей хода выполнения
- Контроль выполнения – измерение метрических показателей и оценка хода выполнения
- Усовершенствование – изменение процесса при меняющихся условиях применения

Применение полных (тяжелых) процессов требует дополнительных ресурсов (зачастую существенных) и далеко не всегда окупается полученным результатом. Поэтому, выбор состава процессов, степени их установленности (полноты установленности) в каждом конкретном случае может быть сделан по-разному в соответствии с выбранной моделью процесса.

1.2.11. Модель программного процесса?

Модель программного процесса — это упрощенное описание программного процесса, представленное с некоторой точки зрения. Модель задается в виде практических этапов, необходимых для создания ПО. В модели мы говорим, что и как мы будем делать. Т.е. какие процессы, с какой степенью конкретизации и в какой последовательности мы будем выполнять. Выбор модели процесса является первым шагом в создании ПО. Правильны выбор модели очень важен, т.к. во многом определяет успех проекта. Выбор тяжелых процессов может утопить проект, а слишком легкое отношение к процессам – к потере контроля над ходом выполнения.

В соответствии с двумя типами процессов – основных и дополнительных - можно говорить о двух типах моделей процесса: модели процесса разработки (модели жизненного цикла) и модели организации работ по выполнению разработки.

К первым типам моделей (модели жизненного цикла) относятся модели, в которых описывается порядок выполнения действий по созданию продукта. К наиболее известным моделям относятся:

- Водопадная (каскадная) модель – процесс разбивается но последовательное выполнение стадий; каждая стадия начинается после полного завершения предыдущей, продукт создается завершением последней стадии и должен полностью соответствовать изначально установленным требованиям.
- Спиральная (циклическая) модель – процесс также разбивается на стадии, но стадии выполняются циклическим повторением. На первом цикле создается прототип продукта, выполняющий часть требований. Дальнейшие циклы связаны с наращиванием прототипа до полного удовлетворения требований.

- Компонентная модель предполагает сборку продукта из заранее написанных частей – компонент. Основной упор делается на интеграцию и совместное тестирование компонент.
- Формальная модель основана на формальном описании требований с последующим преобразованием (трансляцией) требований в исходный код. Применение формальной модели гарантирует соответствие кода описанным требованиям.

Следует отметить, что различия между этими моделями существуют только в теории. На практике спиральная модель может быть дополнена элементами каскадной и компонентной. Задача программного инженера – подобрать правильную их комбинацию, ориентируясь только на конечный результат.

Ко второму типу моделей – моделей организации работ относятся:

- Модель потока работ (workflow model) — показывает последовательность действий, выполняемых людьми на различных этапах разработки ПО. Для каждого действия указываются входы, выходы (результаты) и связи по входам и выходам.
- Модель потоков данных (data flow model) — представляет процесс в виде последовательного преобразования данных. Каждое преобразование может выполняться одним или несколькими действиями.
- Ролевая модель — показывает роли людей, участвующих в программном процессе, а также действия и результаты, за которые они отвечают.

1.2.12. Методы программной инженерии?

Метод программной инженерии — это структурный подход к созданию ПО, который способствует производству высококачественного продукта эффективным в экономическом аспекте способом. В этом определении есть две основные составляющие: (а) создание высококачественного продукта и (б) экономически эффективным способом. Иными словами, метод – это то, что обеспечивает решение основной задачи программной инженерии: создание качественного продукта при заданных ресурсах времени, бюджета, оборудования, людей.

Начиная с 70-х годов создано достаточно много методов разработки ПО. Наиболее известны:

- Метод структурного анализа и проектирования Том ДеМарко (1978),
- Метод сущность-связь проектирования информационных систем Чен (1976)
- Метод объектно-ориентированного анализа Буч (1994), Рамбо (1991).

Метод программной индустрии основан на идее создания моделей ПО с поэтапным преобразованием этих моделей в программу – окончательную модель решаемой задачи. Так, на этапе спецификаций создается модель – описание требований, которая далее преобразуется в модель проекта ПО, проект – в программный код. При этом важно, чтобы модели метода представлялись графически с помощью некоторого языка представления моделей.

Методы должны включать в себя следующие компоненты:

- Описание моделей системы и нотация, используемая для описания этих моделей (например, объектные модели, конечно-автоматные модели и т.д.)
- Правила и ограничения, которые надо выполнять при разработке моделей (например, каждай объект должен иметь одинаковое имя)
- Рекомендации — эвристики, характеризующие хорошие приемы проектирования в данном методе (скажем, рекомендация о том, что ни у одного объекта не должно быть больше семи подобъектов)
- Руководство по применению метода — описание последовательности работ (действий), которые надо выполнить для построения моделей (все атрибуты должны быть задокументированы до определения операций, связанных с этим объектом)

Нет идеальных методов, все они применимы только для тех или иных случаев. Нет абсолютных методов – применяемые на практике методы могут включать элементы различных подходов. Выбор метода составляет задачу специалиста по программной инженерии.

1.2.12.1. Модель прецедентов (требований)

На слайде представлена модель прецедентов, или вариантов использования (Use case) в нотации языка UML (Unified Modeling Language), поддерживающего объектно-ориентированный анализ и проектирование ПО. Модель описывает (специфицирует) требования к программе регистрации курсов в ВУЗе.

На диаграмме представлены действующие лица (акторы) и действия (прецеденты), которые они выполняют:

- Студент регистрируется на курсе
- Преподаватель: выбирает курс для преподавания и запрашивает расписание курсов

Для каждого действия – прецедента дается его содержательное описание. Далее на основе этой модели строится путем поэтапного ее уточнения и преобразования будут строиться другие модели программы.

1.2.12.2. Модель классов

На слайде представлена одна из таких моделей – диаграмма классов. На диаграмме показаны основные классы программы: ВУЗ, факультет, Студент, Курс, Преподаватель. Приведены основные атрибуты и методы классов.

Кроме того, отмечены отношения (зависимости) между классами:

- Так отмечено, что ВУЗ состоит из Факультетов (агрегация), причем, один ВУЗ может состоять из одного или нескольких факультетов.
- Каждый преподаватель работает на факультете, но при этом, каждый преподаватель может работать на нескольких факультетах и на каждом факультете могут работать несколько преподавателей.

1.2.12.3. Модель сущность-связь

На слайде представлена модель сущность-связь для задачи автоматизации продаж товаров со склада. Представлены сущности, участвующие в процессе: Покупатель, Накладная, Список товаров, Склад, … Для каждой сущности представлены ее атрибуты – для покупателя это код покупателя, имя, адрес, банк. Выделены ключевые атрибуты, однозначно идентифицирующие экземпляры сущностей (у покупателя это код). Установлены связи между сущностями: Покупатель получает Накладную. Отмечены свойства связей: один покупатель может получать несколько накладных.

Далее эта модель преобразуется в реляционную модель базы данных для хранения информации о выделенных сущностях.

Представленная на слайде модель записана в нотации IE (Information Engineering). В других нотациях она будет выглядеть иначе.

1.2.12.4. Нотации модели

На слайде представлен фрагмент модели сущность-связь задачи учета сотрудников, записанный в различных нотациях: Чена (автор метода сущность-связь), Мартина (соавтор), стандарта IDEF1X (Information Modeling Method) и Баркера.

1.2.13. Что такое CASE?

CASE - Computer Aided System Engineering - различного рода инструментальные программы, используемые для поддержки процесса создания программ

CASE средства могут быть классифицированы по нескольким признакам:

- По уровню применения:
  - Upper CASE -средства анализа требований
  - Middle CASE - средства проектирования

- Low CASE - ссредства разработки приложений
- Специализированные
  - Средства проектирования баз данных
  - Средства реинжиниринга (восстановления) модели (формирование ERD на основе анализа схем БД или формирования диаграмм на основе анализа программных кодов)
- Вспомогательные
  - Планирования и управления проектом
  - Конфигурационного управления
  - Тестирования
- Интегрированные CASE охватывают все этапы и процессы создания ПО от анализа требований до тестирования и выпуска документации. Интегрированные CASE выступают, как правило, в виде набора согласованных по интерфейсу средств, предназначенных для поддержки отдельных этапов процесса.

В настоящее время существует очень много CASE средств и фирм, специализирующихся на их разработке (Rational). При выборе CASE средств следует руководствоваться основным принципом: сначала метод создания ПО, а потом – CASE средства, применимые для этого метода. Выбор наоборот чреват нехорошими последствиями.

Computer Aided System Engineering - использование компьютеров для поддержки процесса создания программ.

Это широкий спектр программ – инструментальных средств, применяемых на разных этапах разработки ПО: спецификации требований, проектирования, кодирования, тестирования, документирования, …

1.2.14. Свойства хорошей программы?

Удовлетворять функцио-нальным требованиям. Прежде всего, хорошая программа должна делать то, что ожидает от нее заказчик – т.е. удовлетворять требованиям заказчика. Такие требования называют функциональными. Но кроме функциональных требований, существует еще ряд общих характеристик, которым в той или иной степени должна обладать каждая программа. Эти характеристики принято называть нефункциональными требованиями. К нефункциональным требованиям относят:

- Сопровождаемость (maintainability). Сопровождаемость означает, что программа должна быть написана с расчетом на дальнейшее развитие. Это критическое свойство системы, т.к. изменения ПО неизбежны вследствие изменения бизнеса. Сопровождение программы выполняют, как правило, не те люди, которые ее разрабатывали. Сопровождаемость включает такие элементы как наличие и понятность проектной документации, соответствие проектной документации исходному коду, понятность исходного кода, простота изменений исходного кода, простота добавления новых функций.
- Надежность (dependability). Надежность ПО включает такие элементы как:
  - Отказоустойчивость – возможность восстановления программы и данных в случае сбоев в работе
  - Безопасность – сбои в работе программы не должны приводить к опасным последствиям (авариям)
  - Защищенность от случайных или преднамеренных внешних воздействий (защита от дурака, вирусов, спама).
- Эффективность (efficiency). ПО не должно впустую тратить системные ресурсы, такие как память, процессорное время, каналы связи. Поэтому эффективность ПО оценивается следующими показателями: время выполнения кода, загруженность процессора, объем требуемой памяти, время отклика и т.п.
- Удобство использования (usability). ПО должно быть легким в использовании, причем именно тем типом пользователей, на которых рассчитано приложение. Это

включает в себя интерфейс пользователя и адекватную документацию. Причем, пользовательский интерфейс должен быть не интуитивно, а профессионально понятным пользователю.

Следует отметить, что реализация нефункциональных требований часто требует больших затрат, чем функциональных. Так, сопровождаемость требует значительных усилий по поддержанию соответствия проекта исходному коду и применения специальных методов создания модифицируемых программ. Надежность – дополнительных средств восстановления системы при сбоях. Эффективность – поиска специальных архитектурных решений и оптимизации кода. А удобство – проектирования не «интуитивно» понятного, а профессионально понятного интерфейса пользователя.

1.2.15. Основные трудности

Трудностей достаточно много. Все они в той или иной степенью связаны с главной проблемой программной инженерии: поиском универсального метода и процесса, пригодного для создания программного продукта любого типа в любых условиях. Здесь главная проблема – меняющиеся условия. В этой связи разработчики ПО сталкиваются со следующими трудностями:

- Наследование ранее созданного ПО (legacy systems). Существует достаточно много систем, созданных много лет назад, морально устаревших, но продолжающих работать. Проблема наследования таких систем состоит в их сопровождении – поддержке и развитии.
- Разнородность программных систем. ПО должно работать в распределенных сетях, на разнородном оборудовании, в разных средах, под управлением различных операционных систем.
- Сокращение времени на разработку. Запросы рынка требования к программным системам меняются очень быстро. Суть проблемы состоит в том, чтобы сократить время разработки ПО без снижения его качества.

Эти трудности часто оказываются связанными между собой. Задача разработки сетевого варианта старой локальной базы данных в ограниченные сроки является достаточно типичной.

1.2.16. Профессинальные и этические требования

Развитие средств вычислительной техники, коммуникаций и программных систем (Internet, телекоммуникации, распределенные системы, IP телефония, компьютерные игры и обучающие программы) оказывает все большее воздействие на общество. Роль специалистов по программному обеспечению при этом все время возрастает. Они работают в определенном правовом и социальном окружении, находятся под действием, международных, национальных и местных законодательств.

Ясно, что программисты, как и специалисты других профессий, должны быть честными и порядочными людьми. Но вместе с тем, программисты не могут руководствоваться только моральными нормами или юридическими ограничениями, т.к. они обычно бывают связаны более тонкими профессиональными обязательствами:

- Конфиденциальность – программные специалисты должны уважать конфиденциальность в отношении своих работодателей или заказчиков независимо от того, подписывалось ли ими соответствующее соглашение.
- Компетентность – программный специалист не должен завышать свой истинный уровень компетентности и не должен сознательно браться за работу, которая этому уровню не соответствует.
- Защита интеллектуальной собственности – специалист должен соблюдать законодательство и принципы защиты интеллектуальной собственности при использовании чужой интеллектуальной собственности. Кроме того, он должен защищать интеллектуальную собственность работодателя и клиента. Внимание: создаваемая им

интеллектуальная собственность является собственностью работодателя или клиента.

- Злоупотребление компьютером – программный специалист не должны злоупотреблять компьютерными ресурсами работодателя или заказчика; под злоупотреблениями мы здесь понимаем широкий спектр — от игр в компьютерные игрушки на рабочем месте до распространения вирусов и т.п.

1.2.16.1. Кодекс этики IEEE-CS/ACM

В разработке таких этических обязательств ведущую роль играют профессиональные сообщества. Такие общества, как

- ACM – Association for Computing Machinery - Ассоцтация по вычислительной технике,
- IEEE – Institute of Electrical and Electronic Engineers – Институт инженеров по электротехнике и электронике
- CS - British Computer Society – Британское компьютерное общество

совместно разработали и опубликовали IEEE-CS/ACM Software Engineering Code of Ethics and Professional Practices – Кодекс этики и профессиональной практики программной инженерии..

Члены этих организация принимают обязательство следовать этому кодексу в момент вступления в организацию

Кодекс содержит восемь Принципов, связанных с поведением и решениями, принимаемыми профессиональными программистами, включая практиков, преподавателей, менеджеров и руководителей высшего звена

Кодекс распространяется также на студентов и «подмастерьев», изучающих данную профессию

Кодекс имеет краткую и полную версии

1.2.16.2. Кодекс этики - Преамбула

Краткая версия кодекса

- суммирует стремления кодекса на высоком уровне абстракции.
- полная версия показывает как эти стремления отражаются на деятельности профессиональных программистов.
- без высших принципов детали кодекса станут казуистическими и нудными;
- без деталей стремления останутся возвышенными, но пустыми;
- вместе же они образуют целостный кодекс.

Программные инженеры должны добиваться, чтобы анализ, спецификация, проектирование, разработка, тестирование и сопровождение программного обеспечения стали полезной и уважаемой профессией. В соответствии с их приверженностью к процветанию, безопасности и благополучию общества, программные инженеры будут руководствоваться следующими Восемью Принципами

1.2.16.3. Кодекс этики: 8 принципов

1. ОБЩЕСТВО
   - Программные инженеры будут действовать соответственно общественным интересам.
2. КЛИЕНТ И РАБОТОДАТЕЛЬ
   - Программные инженеры будут действовать в интересах клиентов и работодателя, соответственно общественным интересам.
3. ПРОДУКТ
   - Программные инженеры будут добиваться, чтобы произведенные ими продукты и их модификации соответствовал высочайшим профессиональным стандартам.

4. СУЖДЕНИЕ
   - Программные инженеры будут добиваться честности и независимости в своих профессиональных суждениях

5. МЕНЕДЖМЕНТ
   - Менеджеры и лидеры программных инженеров будут руководствоваться этическим подходом к руководству разработкой и сопровождением ПО, а также будут продвигать и развивать этот подход

6. ПРОФЕССИЯ
   - Программные инженеры будут улучшать целостность и репутацию своей профессии соответственно с интересами общества

4. КОЛЛЕГИ
   - Программные инженеры будут честными по отношению к своим коллегам и будут всячески их поддерживать

8. ЛИЧНОСТЬ
   - Программные инженеры в течение всей своей жизни будут учиться практике своей профессии и будут продвигать этический подход к практике своей профессии

Полная версия кодекса: IEEE-CS/ACM Software Engineering Ethics and Professional Practices. http://www.computer.org/tab/seprof/code.htm#Public

1.2.17. Стандартизация и стандарты

Как отмечалось, по происхождению программные продукты бывают двух типов: заказные (под заказ конкретного потребителя) и коробочные (для массовой продажи на рынке). Для заключения контракта заказчик должен быть уверен, что разработчик справится и не завалит проект. Вопрос: как его в этом убедить? Варианты ответов: «Мы умные люди с научными степенями» или «У нас есть опыт разработки подобных программ» звучат либо наивно, либо не вполне убедительно. В мировой практике промышленного производства ответы на эти вопросы дают стандарты на производство продуктов и услуг и сертификация производителей на соответствие этим стандартам. Вопрос заказчика в этом случае звучит так: Какими стандартами вы владеете и есть ли у вас сертификаты на соответствие этим стандартам?

Процесс стандартизации и сертификации давно вошел и в программную инженерию, где он составляет основу промышленного производства программных продуктов. При изготовлении коробочных продуктов стандартизация имеет не меньшее значение, т.к. она обеспечивает качество продуктов и продвижение их на рынок.

1.3.1. Стандарты и сертификация

Организация производит товары или услуги. При этом она применяет некоторую технологию производства. Эта технология должна соответствовать стандартам на товары или услуги. Применяемая организацией технология проходит сертификацию на соответствие этим стандартам.

1.3.1.1. Что такое технология

Происходит от греческого téchne (искусство, мастерство) и логия (знание, умение). Определяется как:

- совокупность приёмов и способов получения, обработки или переработки сырья, материалов, полуфабрикатов или изделий, осуществляемых в различных отраслях промышленности, в строительстве и т. д.;
- научная дисциплина, разрабатывающая и совершенствующая такие приёмы и способы;
- сами операции добычи, обработки, переработки, …, которые являются основной составной частью производственного процесса;
- описание производственных процессов;
- инструкции по их выполнению;
- технологические правила, требования, карты, графики и др.

Иными словами – это подробное описание того, как надо изготовлять то или иное изделие и наука о составлении таких описаний.

1.3.1.2. Что такое стандарт?

Происходит от английского standard - норма, образец, мерило. Это:

- утверждаемый компетентным органом нормативно-технический документ, устанавливающий комплекс норм, правил по отношению к предмету стандартизации;
- типовой образец, эталон, модель, принимаемые за исходные для сопоставления с ними других предметов.

Например: ГОСТ ЕСПД – единая система программной документации – документы, описывающие состав и структуру документации на разработку программ для ЭВМ (общее описание, техническое задание, эскизный проект, технический проект, описание применения). Типовые образцы – эталоны мер и весов (эталон метра, хранящийся в Париже в палате мер и весов).

Стандарт может быть разработан на:

- материально-технические предметы (продукцию, эталоны, образцы веществ);
- нормы, правила, требования организационно-методического и общетехнического характера.

Пример: Вузы работают в соответствии с государственными образовательными стандартами, представленными в виде паспортов специальностей.

Стандартизация распространяется на все сферы человеческой деятельности: науку, технику, промышленное и с.-х. производство, строительство, здравоохранение, транспорт и т.д. Шкаф проходит в дверь потому, что есть согласованные стандарты на размеры мебели и дверных проемов. Электрическая вилка втыкается в розетку по той же причине. Но можно вспомнить евро вилку и евро розетку.

Из истории стандартов: длина крепостной стены нижегородского кремля равна длине крепостной стены московского кремля. Также совпадают размеры Красной площади и площади Минина.

1.3.1.3. Что такое сертификация?

Сертификация в переводе с латыни означает "сделано верно". Для того чтобы убедиться в том, что продукт "сделан верно", надо знать:

- каким требованиям он должен соответствовать
- каким образом возможно получить достоверные доказательства этого соответствия

Общепризнанным способом такого доказательства служит сертификация соответствия и заявление о соответствии.

Заявление поставщика о соответствии:

- означает, что поставщик (изготовитель) под свою личную ответственность сообщает о том, что его продукция отвечает требованиям конкретного нормативного документа
- содержит следующие сведения:
    - адрес изготовителя, представляющего заявление-декларацию,
    - обозначение изделия и дополнительную информацию о нем;
    - наименование, номер и дату публикации стандарта, на который ссылается изготовитель;
    - указание о личной ответственности изготовителя за содержание заявления и др.

Заявление не является гарантией на соответствие стандарту. Заявление отражает готовность нести ответственность.

Сертификация соответствия:

- предполагает обязательное участие третьей стороны
- осуществляется по правилам определенной процедуры, включающей обязательные испытания на соответствие стандарту

Сертификация считается основным достоверным способом доказательства соответствия продукции (процесса, услуги) заданным требованиям (стандартам). Систему сертификации (в общем виде) составляют:

- центральный орган который управляет системой, проводит надзор за ее деятельностью и может передавать право на проведение сертификации другим органам; правила и порядок проведения сертификации;
- нормативные документы, на соответствие которым осуществляется сертификация;
- процедуры (схемы) сертификации;
- порядок инспекционного контроля.

Системы сертификации могут действовать на национальном, региональном и международном уровнях.

1.3.2. Какие бывают стандарты?

Среди всего многообразия стандартов принято выделять следующие основные типы стандартов:

Корпоративные стандарты разрабатываются крупными фирмами (корпорациями) с целью повышения качества своей продукции. Такие стандарты разрабатываются на основе собственного опыта и с учетом требований мировых стандартов. Корпоративные стандарты не сертифицируются, но являются обязательными для применения внутри корпорации. В условиях рыночной конкуренции могут иметь закрытый характер.

В IT сфере известны стандарты, разработанные Microsoft, Intel, IBM.

Отраслевые стандарты действуют в пределах организаций некоторой отрасли (министерства). Например, СНИП – строительные нормы и правила. Разрабатываются с уче-

том требований мирового опыта и специфики отрасли. Являются, как правило, обязательными для отрасли. Подлежат сертификации.

Государственные стандарты (ГОСТы) принимаются государственными органами, имеют силу закона. Разрабатываются с учетом мирового опыта или на основе отраслевых стандартов. Могут иметь как рекомендательный, так и обязательный характер (стандарты безопасности). Для сертификации создаются государственные или лицензированные органы сертификации.

Международные стандарты. Разрабатываются, как правило, специальными международными организациями на основе мирового опыта и лучших корпоративных стандартов. Имеют сугубо рекомендательный характер. Право сертификации получают организации (государственные и частные), прошедшие лицензирование в международных организациях.

1.3.3. Кто разрабатывает стандарты SE?

Основными разработчиками международных стандартов являются следующие организации:

ISO - International Organization for Standardization – Международная организация по стандартизации. Наиболее представительная и влиятельная организация, разрабатывающая стандарты почти во всех областях деятельности, в том числе и в IT.

ACM - Association for Computing Machinery –Ассоциация по вычислительной технике. Всемирная научная и образовательная организация в области вычислительной технике. Известна также и разработкой образовательных стандартов.

SEI - Software Engineering Institute - Институт Программной Инженерии. Исследования в области программной инженерии с упором на разработку методов оценки и повышения качества ПО. Стандарты по качеству ПО и зрелости организаций, разрабатывающих ПО.

PMI - Project Management Institute - Международный Институт Проектного Менеджмента (Управления Проектами). Некоммерческая организация, целью которой является продвижение, пропаганда, развитие проектного менеджмента в разных странах. PMI разрабатывает стандарты проектного менеджмента, занимается повышением квалификации специалистов.

IEEE - Институт инженеров по электронике. Поддержка научных и практических разработок в области электроники и вычислительной техники. Большие вложения в разработку стандартов в этой области.

См. также: Васютович В.В. Стандартизация в области информационных технологий. http://inform.alee.ru/item_541.html

1.3.3.1. ISO - International Organization for Standardization

Неправительственная организация с консультативным статусом ООН. Главная цель - развитие стандартизации и родственных направлений деятельности во всем мире.

Официальное название -**International Organization for Standardization**, и сокращенным - **ISO**. Выходит, что аббревиатурой ISO должен быть IOS. Слово ISO образовано от греческого слова isos , что означает «равный» и служит приставкой iso- в таких терминах как «изономия», «изометрия».

Во избежание многочисленных вариаций аббревиатуры, получающейся от перевода **International Organization for Standardization** на родные языки членов организации решили использовать аббревиатуру ISO, как сокращенную форму обозначения Международной организации по стандартизации (ISO) во всем мире.

Международная организация по стандартизации (ISO):

- Является всемирной федерацией национальных организаций по стандартизации (*комитетов-членов* ISO)

- Разработка международных стандартов обычно осуществляется техническими комитетами ISO. Каждый *комитет-член*, заинтересованный в деятельности, для которой создан технический комитет, имеет право быть представленным в этом комитете.

Международные правительственные и неправительственные организации, имеющие связи с ISO, также принимают участие в работах.

Стандарты ISO являются рекомендательными; в то же время некоторые международные стандарты (например по проблемам здравоохранения, безопасности, охраны окружающей среды) приняты рядом стран в качестве обязательных на территории данной страны.

Как правило, никакого контроля за выполнением стандартов, никакой сертификации на соответствие своим стандартам ISO не ведет - это также считается суверенным правом стран. Обычно эти процедуры поручаются либо специально назначенному государственному органу регистрации, либо так называемой третьей стороне -- лаборатории или аудиторскому институту, в том числе и частному аудитору, действующему на коммерческой основе.

Использование логотипа ISO на каких-либо продуктах или в предприятиях является также незаконным, поскольку дает ощущение "одобрения" данного продукта; а как мы уже говорили, ISO не занимается сертификацией.

Проекты международных стандартов, принятые техническими комитетами, рассылаются комитетам-членам на голосование. Их опубликование в качестве международных стандартов требует одобрения, по меньшей мере, 75% комитетов-членов, принимающих участие в голосовании.

1.3.3.2. ACM - Association for Computing Machinery

ACM - Association for Computing Machinery – название почти никогда не переводится. Можно перевести как Ассоциация для вычислительной техники, что звучит весьма коряво. ACM является крупнейшей всемирной научной и образовательной организацией, объединяющей более 75000 профессионалов компьютерной науки. Основанная в 1947 г, ACM ежегодно проводит до 100 международных (научных и практических) конференций, издает несколько десятков научных журналов и присуждает большое количество авторитетных наград за достижения в области компьютерной науки, в т.ч. A.M. Turing Award, известную как "нобелевская премия информатики". Под эгидой ACM проводятся ежегодные международные студенческие олимпиады по программированию.

Подробнее об ACM можно прочесть на Internet-сайте ассоциации: http://www.acm.org/.

1.3.3.3. SEI - Software Engineering Institute

Software Engineering Institute, Carnegie Mellon University - Институт Программной Инженерии в университете Карнеги-Меллона – это центр исследования и разработки, находящийся на федеральном финансировании и спонсируемый министерством обороны США. SEI ставит своей основной задачей создание методик для оценки уровня развития внутренних процессов в организации. В качестве подразделения широко известного благодаря разработкам в области вычислительной техники и программного инжиниринга, SEI имеет доступ к самым передовым техническим инновациям. С 1984 года SEI развивает и пропагандирует методики для разработки высококачественного ПО. Первая версия Модели Технологической Зрелости Компании-Разработчика ПО (Capability Maturity Model for Software, SW-CMM) была создана в SEI в 1991 году.

Подробнее: www.sei.cmu.edu

1.3.3.4. PMI - Project Management Institute

Международный Институт Проектного Менеджмента - Project Management Institute (PMI), основан в 1969 г. в США. Штаб-квартира в Филадельфии (Пенсильвания). Международная общественная организация, объединяющая профессионалов в области проектного менеджмента. PMI объединяет от 100000 до 135000 членов (данные различных источников расходятся) в 125 странах мира.

PMI - некоммерческая организация, целью которой является продвижение, пропаганда, развитие проектного менеджмента в разных странах. PMI разрабатывает стандарты проектного менеджмента, занимается повышением квалификации специалистов.

PMI являлся ведущей профессиональной организацией по управлению проектами в таких областях, как авиакосмическая и автомобильная промышленность, управление коммерческими

предприятиями, машиностроение, финансовые операции, информационные технологии, фармацевтика, телекоммуникации и многие другие.

PMI предоставляет всеобъемлющее руководство по разработке стандартов для проектного менеджмента (стандарт по управлению проектами PMBOK). PMI стал первой организацией в мире, имеющей программу сертификации специалистов по управлению проектами - Project Management Professional (PMP).

Для обучения проектному менеджменту и подготовки к экзамену PMP созданы Registered Education Provider (R.E.P) - Сертифицированный провайдер по образованию - во многих странах мира.

Исследования в области проектного менеджмента поддерживаются за счет проведения конференций, предоставления грантов, выпуска научных трудов, создания исследовательской базы данных и т.д. Кроме того, собирается и сортируется информация о текущем состоянии дел, потребностях, накопленных знаниях по проектному менеджменту, и на этой основе оценивается будущее профессии и путь ее развития.

PMI выпускает три вида периодических изданий для индивидуальных лиц, занимающихся проектным менеджментом: ежемесячный журнал PM Network, ежеквартальный журнал Project Management Journal и ежемесячный информационный бюллетень PMI Today. PMI является ведущим мировым издателем литературы и учебных материалов по проектному менеджменту. В онлайновом магазине PMI в настоящее время доступно более 1000 наименований.

Подробнее: http://www.pmi.org и http://www.pmi.ru

1.3.3.5. IEEE – Institute of Electrical and Electronics Engineers

IEEE - Институт инженеров по электронике объединяет почти 400000 технических специалистов из более чем 150 стран. IEEE состоит из ряда профессиональных сообществ, в самое крупное из которых - IEEE Computer Society - входят более 100000 человек. Компьютерное сообщество IEEE ежегодно спонсирует около ста пятидесяти научных конференций и симпозиумов, публикует более 20 периодических изданий. IEEE Computer Society также широко известно своей деятельностью по стандартизации, которую на сегодняшний день в рамках сообщества осуществляют порядка 200 рабочих групп.

Подробнее: http://www.ieee.org и http://www.computer.org.ru/

1.3.4. Основные стандарты SE

Наиболее известными стандартами программной инженерии являются:

- ISO/IEC 12207 - Information Technology - Software Life Cycle Processes - Процессы жизненного цикла программных средств. Стандарт содержит определения основных понятий программной инженерии (в частности программного продукта и жизненного цикла программного продукта), структуры жизненного цикла как совокупности процессов, детальное описание процессов жизненного цикла.
- SEI CMM - Capability Maturity Model (for Software) - модель зрелости процессов разработки программного обеспечения. Стандарт отвечает на вопрос: «Какими признаками должна обладать профессиональная организация по разработке ПО?». Профессионализм организации определяется через зрелость процесса, применяемого этой организацией. Выделяются пять уровней зрелости процесса.
- ISO/IEC 15504 - Software Process Assessment - Оценка и аттестация зрелости процессов создания и сопровождения ПО. Является развитием и уточнением ISO 12207 и SEI CMM. Содержит расширенное по отношению ISO 12207 количество процессов жизненного цикла и 6 уровней зрелости процессов. Дается подробное описание схемы аттестации процессов, на основе результатов которой может быть выполнена оценка зрелости процессов и даны рекомендации по их усовершенствованию.
- PMBOK - Project Management Body of Knowledge - Свод знаний по управлению проектами. Содержит описания состава знаний по следующим 9 разделам (областям знаний) управления проектами

- SWBOK - Software Engineering Body of Knowledge - Свод знаний по программной инженерии - содержит описания состава знаний по 10 разделам (областям знаний) программной инженерии.
- ACM/IEEE CC2001 - Computing Curricula 2001 – Академический образовательный стандарт в области компьютерных наук. Выделены 4 основных раздела компьютерных наук: Computer science, Computer engineering, Software engineering и Information systems, по каждому из которых описаны области знаний соответствующего раздела, состав и планы рекомендуемых курсов

1.3.4.1. ISO/IEC 12207-95

Понятие жизненного цикла ПО как некоторой последовательности этапов, которые надо выполнить для создания ПО: проектирование, разработка, тестирования, … составляет одно из фундаментальных понятий программной инженерии. Понятие возникло в конце 60-х годов, когда разработкой ПО занималось много фирм, и различными разработчиками трактовалось по-разному. Путаница в терминологии порождала много проблем во взаимопонимании между разработчиками, между разработчиками и заказчиками. Необходим был стандарт на представление терминов, понятий и составных элементов жизненного цикла ПО. Этот стандарт был принят ISO в 1995 году. Следует отметить, что работы по нему были начаты в 1987 году и стандарт формировался взаимными усилиями 125 стран – участниц ISO. В России этот стандарт принят в 2000 году под названием «ГОСТ Р ИСО/МЭК 12207 Процессы жизненного цикла программных средств».

В стандарте ISO 12207 дается определение программного продукта (ПО) как набора компьютерных программ, процедур и связанной с ними документации и данных и определение жизненного цикла программного продукта как непрерывного процесса, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

Стандарт определяет:

- организацию ЖЦ ПО
- структуру ЖЦ ПО

По организации жизненный цикл ПО описывается в виде совокупности взаимодействующих процессов, каждый из которых состоит из действий, которые в свою очередь могут быть разбиты на отдельные задачи.

Структура жизненного цикла дается в виде перечня его процессов: процесс управления, процесс разработки процесс тестирования, процесс документирования, … Для каждого процесса дается описание действий этого процесса. Так, действиями процесса разработки являются: подготовка процесса, анализ требований, проектирование системной архитектуры, анализ требований к программным средствам и т.д. Для каждого действия дается подробное описание его задач.

Примечание.

- ISO - International Organization of Standardization - Международная организация по стандартизации
- IEC - International Electrotechnical Commission - Международная электротехническая комиссия

Тексты стандарта:

- ГОСТ ИСО 12207: http://www.staratel.com/iso/InfTech/DesignPO/ISO12207/ISO12207-99/ISO12207.htm
- ISO/IEC 12207: ftp://172.16.100.100/Soft/ntd/12207cpt.pdf

1.3.4.2. SEI CMM

Capability Maturity Model (for Software) - модель зрелости процессов разработки программного обеспечения – известный отчет SEI, который формально стандартом не является, но приобрел характер международного стандарта в силу его интересности и практической полезности. Отчет появился в 1993 году как результат исследования на тему: «Как выбирать организацию, которой можно доверить выполнение крупного IT проекта?». Это исследование проводилось SEI по заказу министерства обороны США, которое было очень озадачено этой проблемой. В отчете изложена модель зрелости организаций, которая определялась через зрелость процесса разработки ПО, применяемого в этой организации. В этой модели выделяется пять уровней зрелости процесса, которые и устанавливают степень готовности организации выполнить крупный проект:

1. Начальный (Initial)

Технология полностью импровизированная, в некоторых случаях — даже хаотическая. Успех всецело зависит от усилий отдельных сотрудников.

2. Повторяемый (Repeatable)

Базовые процессы управления проектом ПО установлены. Есть дисциплина соблюдения, что обеспечивает возможность повторения успеха предыдущих проектов в той же прикладной области

3. Определенный (Defined)

Процессы задокументированы, стандартизованы и интегрированы в единую для всей организации технологию создания ПО. Для каждого проекта используется адаптивный вариант этой технологии.

4. Управляемый (Managed)

Собираются и накапливаются метрики (объективные данные) о качестве исполнения процессов и выходной продукции. Управление процессами и выходной продукцией осуществляется по количественным оценкам.

5. Оптимизируемый (Optimized)

Совершенствование технологии создания ПО осуществляется непрерывно на основе количественной обратной связи от процессов и пилотного внедрения инновационных идей.

Подробнее:

- Capability Maturity Model for Software, Version 1.1, Paulk, Mark C.; Curtis, Bill; Chrissis, Mary Beth Chrissis, and Weber, Charles, Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, February 1993
- http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf
- Билл Куртис, Марк Паулк, Мэри Бет Хриссис. Модель зрелости процессов разработки программного обеспечения. Интерфейс-Пресс. 2003 г. · 400 стр. (: http://www.ryabikin.com/sw-cmm/index.htm)

А также:

- Терехов А.А., Туньон В. Современные модели качества программного обеспечения (обзор ISO9000, CMM SPICE) http://www.interface.ru/fset.asp?Url=/misc/qs.htm
- Назаренко Ю.А. Технологическая зрелость IT организаций. http://www.noumen.ru/go/company/obj1041600305/obj1043060989

1.3.4.3. ISO/IEC TR 15504

ISO/IEC 15504 TR Software Process Assessment - международный стандарт, опубликованный в виде отчета, известный также как SPICE: Software Process Improvement and Capability dEtermination - Оценка и аттестация зрелости процессов создания и сопровождения ПО.

Разработан на основе обобщения опыта 9 стандартов (международных и корпоративных), в том числе ISO 12207 и SEI CMM. Является развитием и уточнением этих стандартов. Содержит расширенное по отношению ISO 12207 количество процессов жизненного цикла и 6 уровней зрелости процессов по отношению к CMM.

Дается подробное описание схемы аттестации процессов, на основе результатов которой может быть выполнена оценка зрелости процессов и даны рекомендации по их усовершенствованию.

В стандарте установлены регламенты аттестации, оценки и усовершенствования процессов, дается подробное описание требований к аттестаторам.

Подробнее:

- ISO/IEC TR 15504. Information Technology - Software Process Assessment http://www.sqi.gu.edu.au/spice/docs/baseline.zip
- Оценка и аттестация зрелости процессов создания и сопровождения программных средств и информационных систем (ISO/IEC TR 15504) ISBN: 5-212-00884-0/ Изд: АйТи, Книга и бизнес. http://www.ntrlab.ru/rus/method/iso15504/
- Мельникова Н. Не так сложен SPICE, как его написали. Открытые системы. #12, 2001 год. http://www.osp.ru/os/2001/12/030.htm

1.3.4.4. PMI PMBOK

PMBOK - аббревиатура от Project Management Body of Knowledge, Свода знаний по управлению проектами. PMBOK представляет собой стандарт, развиваемый PMI. Известны версии 1996

и 2000 гг. Последняя версия стандарта вышла в 2004 году. Содержит описания состава знаний по следующим 9 разделам (областям знаний) управления проектами:

1. Управление интеграцией - Project Integration Management
2. Управление ограничениями - Project Scope Management
3. Управление временем - Project Time Management
4. Управление затратами - Project Cost Management
5. Управление рисками - Project Risk Management
6. Управление персоналом - Project Personnel Management
7. Управление коммуникациями - Project Communication Management
8. Управление закупками - Project Procurement Management
9. Управление качеством - Project Quality Management

Подробнее:

- A Guide to the Project Management Body of Knowledge 2000
  http://www.tline.ru/library/pmbok2000.pdf
- Руководство к своду знаний по управлению проектами. (PMBOK Guide) Редакция 2000 г. Изд-тва: Институт Управления Проектами, Project Management Institute, 2004 г.

1.3.4.5. IEEE SWEBOK

IEEE Computer Society Software Engineering Body of Knowledge – Свод знаний по программной инженерии - проект IEEE Computer Society. Официальная версия вышла 18 мая 2004 г. Основная идея проекта аналогична PMBOK и заключается в создании некоторого базового набора общепринятых знаний, необходимых любому профессиональному программисту.

Содержит описания состава знаний по следующим 10 разделам (областям знаний) программной инженерии:

1. Software Requirements – требования к ПО
2. Software Design – проектирование ПО
3. Software Construction – конструирование ПО
4. Software Testing – тестирование ПО
5. Software Maintenance – сопровождение ПО
6. Software Configuration Management – управление конфигурациями
7. Software Engineering Management – управление IT проектом
8. Software Engineering Process – процесс программной инженерии
9. Software Engineerting Tools and Methods – методы и инструменты
10. Software Quality – качество ПО

Подробнее: Guide to the Software Engineering Body of Knowledge - http://www.swebok.org/

1.3.4.6. ACM/IEEE Computing Curricula

ACM/IEEE Computing Curricula 2001 – Академический образовательный стандарт в области компьютерных наук - совместный проект международных профессиональных обществ ACM и IEEE Computer Society. Вышло несколько версий 1968, 78, 83, 91, 2001. Основная идея проекта состоит в разработке стандартов на учебные курсы по компьютерным наукам. В стандарте 2001 года выделены 4 основных раздела компьютерных наук:

- Computer science – Информатика (2001г); http://se.math.spbu.ru/cc2001
- Computer engineering – Компьютерная инженерия;
- Software engineering – Программная инженерия (2004г.)
- Information systems – Информационные системы.

Окончательный вариант стандарта ACM/IEEE Computing Curricula 2001: Computer Science был опубликован в декабре 2001, а Software engineering – в мае 2004г. По разделу Computer Science есть перевод (http://se.math.spbu.ru/cc2001). Работа над остальными разделами продолжается, но рабочие материалы этих разделов можно посмотреть на сайте: http://www.computer.org/education/cc2001

По содержанию образовательные стандарты состоят из описания областей знаний соответствующего раздела, состава и планов рекомендуемых курсов

Областями знаний раздела Software engineering являются:

- Computing Essentials - Основы применения ЭВМ
- Mathematical & Engineering Fundamentals - Математические и инженерные основы
- Professional Practice - Профессиональная практика

- Software Modeling & Analysis - Моделирование и анализ ПО
- Software Design - Проектирование ПО
- Software V & V –Верификация и валидация ПО
- Software Evolution - Эволюция ПО
- Software Process - Процесс ПО
- Software Quality - Качество ПО
- Software Management -Управление проектом

  Подробнее:
- ACM/IEEE Computing Curricula 2001 - http://www.computer.org/education/cc2001
- Computing Curricula 2001: Computer Science (рус.) http://se.math.spbu.ru/cc2001