Абрамова О. Ф.

Индустриальная разработка программных продуктов

Часть1

Учебное пособие

Волжский

2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВОЛЖСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО
УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Абрамова О.Ф.

Индустриальная разработка программных продуктов

Часть 1

Электронное учебное пособие

2020

УДК 004.45(07)
ББК 32.973я73
А 161

Рецензенты:
кандидат педагогических наук, доцент кафедры методики преподавания математики и физики, ИКТ ФГБОУ ВО «Волгоградский государственный социально-педагогический университет»
Филиппова Е.М.,
кандидат физико-математических наук, доцент, заведующий кафедрой математики, информатики и естественных наук Волжского филиала Волгоградского государственного университета
Полковников А.А.

Издается по решению редакционно-издательского совета
Волгоградского государственного технического университета

Абрамова, О.Ф.
Индустриальная разработка программных продуктов [Электронный ресурс] : учебное пособие/ О.Ф. Абрамова ; Министерство науки и высшего образования Российской Федерации, ВПИ (филиал) ФГБОУ ВО ВолгГТУ. – Электрон. текстовые дан. (1 файл: 626,70 КБ). – Волжский, 2020. – Режим доступа: http://lib.volpi.ru. – Загл. с титул. экрана.
ISBN 978-5-9948-3704-7

В пособии собран методический материал (часть 1) к лекционному курсу по дисциплине «Индустриальная разработка программных продуктов».

Предназначено для студентов, обучающихся по направлению подготовки бакалавров 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» и рекомендуется как основной источник для изучения дисциплины «Индустриальная разработка программных продуктов».

Ил. 1, табл. 1, библиограф.: 15 назв.

ISBN 978-5-9948-3704-7

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время производство и продажа программных продуктов приобрели черты бизнеса высокорентабельного вида. Это связано как с высокой долей интеллектуального труда создателей программ, так и с низкой материалоемкостью процессов производства. Процесс проектирования и разработки программного обеспечения представляет собой достаточно специфический набор из взаимосвязанных этапов, многие из которых автоматизированы и поддерживаются специальными программными инструментами. Он реализует преобразование исходных требований заказчика о предметной области в готовый программный продукт. В этом процессе участвуют как специалисты, выполняющие определенную работу с помощью специфических инструментальных средств и предметов производства, так и заказчики, оценивающие и, впоследствии, оплачивающие разработку. На вход этого процесса в виде требований поступают сведения о предметной области. На выходе процесса – разработанное программное обеспечение и техническая документация. Одной из важнейших особенностей процесса можно считать материальную заинтересованность участников, что существенно влияет как на сам процесс, так и на результат. Секрет успеха реализации проектов в области промышленного программирования – это выполнение работы в заданные сроки в рамках определенного предварительно бюджета с заданным качеством. Жесткая конкуренция разработчиков в сфере IT-технологий объективно требует от разработчиков обращать также максимум внимания и на экономику и маркетинг их разработки, и продвижения на рынок. Кроме того, индустриальные способы производства должны основываться на современных инструментальных средствах создания программных продуктов, поддерживающих все этапы жизненного цикла, начиная от выявления требований пользователей и заканчивая поставкой им готового

продукта, снабженного качественной документацией. Очевидно также, что реализация процесса индустриальной разработки программных продуктов невозможна без учета разнообразных стандартов и нормативных документов как корпоративных, так и отраслевых, государственных и международных. Стандарты в области информационных технологий не являются жесткими и требующими неукоснительного исполнения, однако они регламентируют требования к процессам жизненного цикла, качества, документирования программного продукта, а также требования к оценке зрелости организаций-разработчиков, позволяя достичь установленных целей как команде разработчиков, так и участвующим в процессах заказчикам.

Все эти вопросы входят в круг интересов программной инженерии как методологии промышленного создания и продвижения на рынок качественных и экономичных программных продуктов и систем. Студенты, обучающиеся по направлению 09.03.04 «Программная инженерия» должны ознакомиться, изучить и иметь практические навыки работы на каждом из этапов процесса проектирования и разработки программных систем. Это будущие профессионалы, занимающиеся решением конкретных практических задач, отвечающие за реализацию жизненного цикла программного продукта на всех этапах, знающие и осмысленно использующие существующие стандарты и нормативные документы в области индустриальной разработки программных продуктов.

Данное пособие преследует цель ознакомить студентов направления 09.03.04 «Программная инженерия» с этапами, методами и особенностями индустриальной разработки программных продуктов с учетом разного рода стандартов в этой области.

ЛЕКЦИЯ 1
ИНДУСТРИАЛЬНАЯ РАЗРАБОТКА ПРОГРАММНЫХ ПРОДУКТОВ

Ничто так не раскрывает недостатки проектирования как реализация.
Дж. Нортон (J. Horton)

Основные понятия, термины и определения

Проектирование программного обеспечения (ПО) – это инженерная деятельность в рамках жизненного цикла ПО, в которой надлежащим образом анализируются требования для создания описания его внутренней структуры, являющейся основой для конструирования ПО как такового. Это процесс определения архитектуры, компонентов, интерфейсов, а также других характеристик системы и конечного состава программного продукта. Быстрое увеличение сложности и размеров современных комплексов программ при одновременном повышении ответственности выполняемых функций резко повысило требования со стороны пользователей к их качеству, надежности функционирования и безопасности применения. Это также привело к принципиальному изменению методов в этой сфере: к переходу от технологии индивидуального программирования отдельных небольших программ к коллективному созданию крупных комплексов программ инженерными методами проектирования и разработки. При индустриальном подходе к разработке и сопровождению ПО особый вес приобретают технологические характеристики разрабатываемых программ.

Термин «технологические характеристики» подчеркивает аналогию между созданием программного продукта и промышленным производством. Он подчеркивает, что процесс программирования в современных условиях разработки сложных и объемных программных продуктов немыслим без регламентирования, организации и инструментирования. Такой процесс нуждается в наборе соглашений и правил, не говоря уже об инструментальном обеспечении, несмотря на интеллектуальность и творческий характер этой деятельности. Для получения качественных программных продуктов необходимо руководствоваться следующими принципами:

- эффективность – результаты должны отвечать заданным требованиям и стандартам в условиях ограниченных ресурсов;
- практичность – результаты должны иметь конкретных заказчиков;
- фундаментальность – результаты должны базироваться на знаниях фундаментальных наук;
- сопровождаемость – результаты, находясь в эксплуатации, обязательно должны обслуживаться.

Из истории вопроса

На начальном этапе развития программной инженерии основными задачами проектирования были проектирование алгоритмов и структур данных. Помимо этого, проектировалась модульная структура программных приложений. С появлением объектно-ориентированной парадигмы программирования проектируются взаимодействия между объектами классов, появляются объектно-ориентированные шаблоны микроархитектуры (шаблоны проектирования).

Более сложной организацией программного обеспечения является программная система, состоящая из компонентов. В настоящее время первоочередной задачей проектирования является проектирования архитектуры программных систем – макроархитектуры.

Проектирование интерфейса пользователя включает не только графический дизайн, но проектирование взаимодействия пользователя и программного приложения (User Experience design, UX design), где учитывается количество переходов и схемы взаимодействий между окнами, среднее время на выполнение определенного действия и другие параметры. Проектирование интерфейса пользователя существенно отличается для различных типов клиентских устройств и вида программного приложения (веб-приложение, полноэкранное игровое программное приложение, корпоративное приложение). Подход к проектированию программного приложения, при котором наибольшее значение имеет качество человеко-

машинного взаимодействие, именуется Look and Feed Driven Design [Chung C. Pro Objective-C Design Patternsfor iOS. - New York, USA: Appress, 2011.]. Данный подход имеет особую популярность для мобильных программных приложений, где эстетические предпочтения пользователя оказывают значительное влияние на решение о покупке и установке программного приложения.

Проектирование играет важную роль в процессах жизненного цикла создания программного обеспечения (Software Development Life Cycle). Например, в стандарте IEEE и ISO/IEC (ГОСТ Р ИСО.МЭК) 12207 по SWEBOK проектирование определяется как «процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или ее компонентов». Проектирование программных систем можно рассматривать как деятельность, результат которой состоит из двух составных частей:

– Архитектурный или высокоуровневый дизайн (software architectural design, toplevel design) – описание высокоуровневой структуры и организации компонентов системы;

– Детализированная архитектура (software detailed design) – описывающая каждый компонент в том объеме, который необходим для конструирования.

Проектирование ПС охватывает три основные области:

- проектирование объектов данных, которые будут реализованы в базе данных;
- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

Область знаний «Проектирование ПО (Software Design)», согласно стандартам программной инженерии, состоит из шести следующих разделов (см. рисунок 1):

1) базовые концепции проектирования ПО (Software Design Basic Concepts);

2) ключевые вопросы проектирования ПО (Key Issue in Software Design);

3) структура и архитектура ПО (Software Structure and Architecture);

4) анализ и оценка качества проектирования ПО (Software Design Quality Analysis and Evaluation);

5) нотации проектирования ПО (Software Design Notations);

6) стратегия и методы проектирования ПО (Software Design Strategies and Methods).

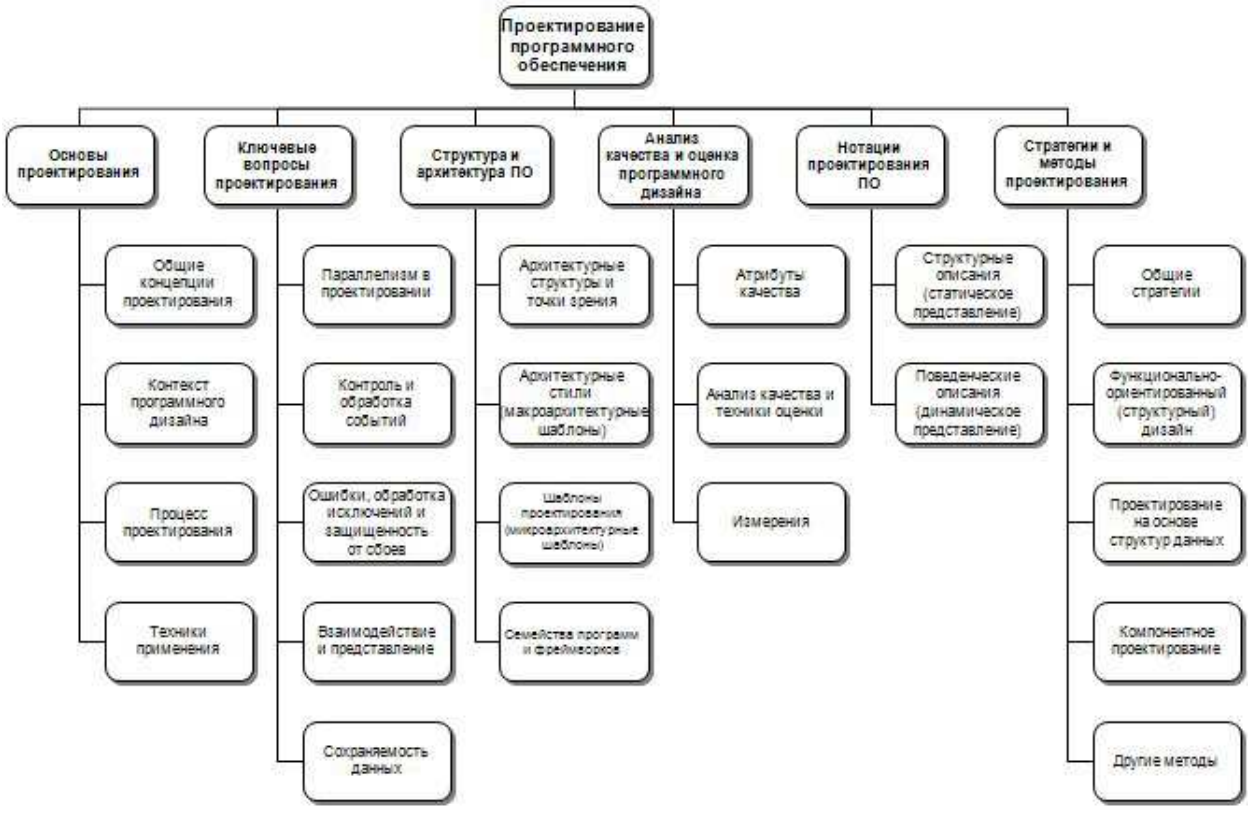

Рисунок 1. Область знаний «Проектирование программного обеспечения»

К ключевым вопросам проектирования ПО относятся: декомпозиция программ на функциональные компоненты для независимого и

параллельного их выполнения, принципы распределения компонентов в среде выполнения и их взаимодействия между собой, механизмы обеспечения качества и живучести системы и др.

Внедрение новой методологии индустриальной разработки программных систем (ПС), цель которой является регламентация процесса проектирования ПС и обеспечении управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой ПС, так и к характеристикам процесса разработки, позволило решить основные задачи процесса проектирования:

- обеспечивать создание промышленных ПС, отвечающих целям и задачам организации, а также предъявляемым требованиям по автоматизации деловых процессов заказчика;
- гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта;
- поддерживать удобную дисциплину сопровождения, модификации и наращивания системы;
- обеспечивать преемственность разработки, т.е. использование в разрабатываемой ПС существующей информационной инфраструктуры организации (задела в области информационных технологий).

Внедрение методологии должно приводить к снижению сложности процесса создания ПС за счет полного и точного описания этого процесса, а также применения современных методов и технологий создания ПС на всем жизненном цикле ПС – от замысла до реализации.

Классификация программных продуктов

Любая компания-разработчик при выводе своего программного продукта (ПП) на рынок должна представлять, в каком сегменте рынка она предполагает работать, кто является ее основными конкурентами. В этом случае необходимо условно разбить весь рынок ПП на несколько сегментов. В настоящее время в литературе нет единого подхода к классификации рынка

ПП. Рассмотрим несколько классификаций, пользующихся популярностью в области ИТ – разработки.

В зависимости от того, для кого разрабатывается ПП – для конкретного заказчика или всего ИТ-рынка – выделяют:

- **тиражные (коробочные) программные продукты** – коммерческие программные продукты специального и широкого применения;
- **заказные (внутрифирменные) программные продукты** – разрабатываются под информационную поддержку управления конкретного бизнес-процесса, либо адаптируются под требования этих бизнес-процессов;
- **коммерческие продукты специального применения** предназначены для использования ограниченным кругом пользователей в определенных предметных областях (издательские системы, научные пакеты и пр.).

Процессы разработки тиражного и заказного программного продуктов соответствуют стандартному и индивидуальному подходам к созданию ПП. Безусловно, любой способ создания (приобретения) прикладного ПП таит в себе свои риски. Всегда есть риск купить типовой пакет прикладных программ, долго пытаться его освоить, но в конечном итоге не получить ожидаемого эффекта. С другой стороны, имеется риск разработать или заказать индивидуальный прикладной программный продукт, работающий с ошибками, непригодный для сопровождения и не соответствующий техническим и отраслевым стандартам. Кроме того, возможен риск затянуть проект или попасть в слишком опасную зависимость от разработчиков (как внешних, так и внутренних).

Внутрифирменные ПП разрабатываются, как правило, по специальным заказам собственными или сторонними программистами. Именно к данной группе программных продуктов относится часто используемый в последнее время термин «заказное ПО».

Принципиальным отличием между программными продуктами двух последних групп является способ их распространения: ПП широкого применения изначально ориентировано на использование разветвленной сетью потребителей, и в этом плане его можно охарактеризовать как «коробочное» ПО. Специальные ПП распространяются, прежде всего, самими разработчиками, и только самые лучшие его образцы – через фирм-посредников.

Классическим вариантом классификации программных продуктов является разделение на три самостоятельных класса рынка ПП:

1) **прикладное программное обеспечение** – программный продукт для индивидуальных пользователей, включая программы для развлечений, образования и обработки данных, автоматизации различных бизнес-процессов в экономике, коммерции, бизнесе, индустрии и т. д.

2) **общесистемное программное обеспечение** – это комплекс программ, которые обеспечивают эффективное управление компонентами вычислительной системы. В отличие от прикладного программного обеспечения, системное не решает конкретные прикладные задачи, а лишь обеспечивает работу других программ, управляет аппаратными ресурсами вычислительной системы и т.д. К общесистемному ПО обычно относят: операционные системы; сервисные программные средства, включая программные средства защиты.

3) **инструментальное программное обеспечение** (средства разработки и развертывания) – предназначено для профессионального использования специалистами по проектированию разработке различных программных продуктов. Это – языки и среды программирования и проектирования; проблемно-ориентированные оболочки; системы управления базами данных.

ЛЕКЦИЯ 2

ОСНОВНЫЕ ЭТАПЫ ПРОЕКТИРОВАНИЯ ПРОГРАММНЫХ ПРОДУКТОВ

«Начни с начала, – торжественно произнёс Король, – и продолжай, пока не дойдешь до конца. Тогда остановись!»

Льюис Кэррол

В 80-е и 90-е в области разработки программного обеспечения преобладали две тенденции. Одна – это быстрый рост приложений, в том числе создаваемых для Web. Другая – расцвет инструментальных средств и парадигм (подходов к проектированию). Порядок разработки проекта определяется в зависимости от вида, масштабов и потребностей проекта. Он будет несколько отличаться для разработки встроенного ПО, мобильных приложений, решений для автоматизации и БД, для программных продуктов, создаваемых по индивидуальному заказу и инвестиционных проектов, но общая последовательность действий для создания ПО универсальна:

Рассмотрим указанные этапы подробнее.

Подготовительный этап

Цель этапа для разработчиков: принять решение, стоит ли браться за реализацию программного продукта. Цель этапа для заказчиков: принять решения, можно ли доверять исполнителю. Результат (при успешном развитии событий): заключение договора (или иного соглашения) на создание или модификацию системы, требуемой заказчику.

Для достижения поставленных целей заказчику и исполнителю совместно нужно решить ряд вполне определённых задач:

1. На основе исходной идеи сформулировать цели и задачи будущего проекта.
2. Разработать некоторое исходное видение – концепцию проекта.
3. Провести анализ востребованности будущего продукта.
4. Провести предварительную оценку рисков будущего проекта.
5. На основе концепции и списка предварительных рисков подготовить предварительное техническое решение.
6. Выбрать методологию разработки и подготовить предварительный план работ.
7. Провести предварительную оценку трудозатрат и необходимых ресурсов.
8. Провести анализ реализуемости продукта.
9. Провести независимое рецензирование технического решения.
10. Принять решение о том, стоит ли продолжать работы.

Определение цели проекта

Хотелось бы подробнее остановиться на важнейшей деятельности – определении цели проекта. Проектирование информационных систем всегда начинается с определения цели проекта. Формулирование цели и задач проекта является одной из важнейших стадий проекта разработки программного обеспечения. Важность её заключается в том, что на этой стадии заказчик программного продукта знакомит исполнителя со своим замыслом, задаёт направление для развития будущего проекта и определяет

предельно допустимые рамки, за которые проект не должен выйти. Если заказчик не расскажет исполнителю о своей идее в доступной для исполнителя форме, то исполнитель не сможет выполнить работу. Если заказчик неверно или неточно определит цель будущего проекта, то исполнитель не будет иметь ориентира для своей деятельности и вряд ли сможет осуществить замысел заказчика.

В общем виде цель проекта можно определить как решение ряда взаимосвязанных задач, включающих в себя обеспечение на момент запуска системы и в течение всего времени ее эксплуатации:

- требуемой функциональности системы и уровня ее адаптивности к изменяющимся условиям функционирования;
- требуемой пропускной способности системы;
- требуемого времени реакции системы на запрос;
- безотказной работы системы;
- необходимого уровня безопасности;
- простоты эксплуатации и поддержки системы.

Решающую роль для любого проекта в рамках индустриальной разработки ПП имеют три составляющих: цель, бюджет и время. Отсутствие четкой целевой установки, в достижении которой должны принимать участие разные специалисты, сводит на нет все усилия команды по реализации проекта. Достижение цели является бесспорным критерием завершения проекта. В случае индустриальной разработки цель, конечно же, должна быть комплексной, многоуровневой, представляющей собой набор взаимосвязанных подцелей и задач. Но эта цель должна быть. И она должна быть сформулирована на подготовительном этапе и никак иначе.

Проектирование

Проектирование алгоритмов и программ – наиболее ответственный этап жизненного цикла программных продуктов, определяющий, насколько создаваемая программа соответствует спецификациям и требованиям со стороны конечных пользователей. Затраты на создание, сопровождение и

эксплуатацию программных продуктов, научно-технический уровень разработки, время морального устаревания и многое другое – все это также зависит от проектных решений.

Как правило, именно заказчик, как носитель идеи проекта, формулирует цель и задачи. Цель проекта задаёт направление развития проекта. Она должна коротко и ясно указать эффект, который окажет создаваемая или модернизируемая система на бизнес-процессы заказчика. Если цель будет указана неточно, проект станет развиваться в неправильном направлении. Развернуть проект в другом направлении очень сложно, и чем больше будет сделано работ в рамках проекта, тем сложнее будет корректировать направление его развития. Очень скоро проект достигает состояния, когда ошибку в постановке цели исправить будет невозможно, и тогда придётся всё начинать заново, либо совсем отказываться от проекта.

Проектирование алгоритмов и программ может основываться на различных подходах, среди которых наиболее распространены:

• структурное проектирование программных продуктов;

• информационное моделирование предметной области и связанных с ней приложений;

• объектно-ориентированное проектирование программных продуктов.

Функционально–ориентированные (структурные) методы ориентированы на идентификацию функций и их уточнение сверху–вниз, после чего проводится разработка диаграмм потоков данных и описание процессов.

В объектно-ориентированном проектировании базируется на концепции объектов, ключевую роль играет наследование, полиморфизм и инкапсуляция, а также абстрактные структуры данных и отображение объектов

Подходы, ориентированные на структуры данных, базируются на методе Джексона и используются для задания входных и выходных данных структурными диаграммами. В данном подходе фокус сконцентрирован в

большей степени на структурах данных, которыми управляет система, чем на функциях системы. Инженеры по программному обеспечению часто вначале описывают структуры данных входов и выходов, а затем разрабатывают структуру управления этими данными.

Компонентное проектирование ориентировано на использование и интеграцию компонентов (особенно компонентов повторного использования) и на их интерфейсов, обеспечивающих взаимодействие компонентов.

Конструирование

Конструирование программного обеспечения в описания области знаний SWEBOK заключается в создании рабочего программного обеспечения посредством комбинации кодирования, верификации (проверки), модульного тестирования, интеграционного тестирования и отладки.

Данная область знаний тесно связана с другими областями проектированием и тестированием, так как конструирование отталкивается от результатов проектирования, а тестирование (в любой своей форме) предполагает работу с результатами конструирования. Достаточно сложно определить границы между проектированием, конструированием и тестированием, так как все они связаны в единый комплекс процессов жизненного цикла и, в зависимости от выбранной модели жизненного цикла и применяемых методов (методологии), такое разделение может выглядеть по-разному.

Основы конструирования программного обеспечения определяются следующими положениями:

- Минимизация сложности при создании программного кода.
- Готовность (ожидание) периодических изменений требований.
- Конструирование с возможностью проверки программного кода.
- Обязательное использование стандартов в конструировании.

Уменьшение сложности в конструировании программного обеспечения достигается за счет особого внимания к созданию простого и легко читаемого кода (например, с точки зрения гибкости или следования тем или иным представлениям о красоте, утончённости кода, ловкости тех или иных приемов, позволяющих его сократить и т.п.), пусть и в ущерб стремлению сделать его идеальным.

Большинство программного обеспечения изменяются с течением времени, так как программное обеспечение не является изолированным от внешнего окружения. Более того, программное обеспечение является частью изменяющейся среды и должны меняться вместе с ней, а иногда и быть источником изменений самой среды.

«Конструирование для проверки» предполагает, что построение программного обеспечения должно вестись таким образом, чтобы само программное обеспечение помогала вести поиск причин сбоев, будучи прозрачной для применения различных методов проверки, как на стадии независимого тестирования (например, инженерами-тестировщиками), так и в процессе эксплуатации, когда особенно важна возможность быстрого обнаружения и исправления возникающих ошибок.

Конструирование зависит от внешних стандартов, связанных с языками программирования, используемым инструментальным обеспечением, техническими интерфейсами. Внешние стандарты создаются разными источниками, например, международными организациями по стандартизации, производителями платформ, операционных сред, производителями инструментов, систем управления базами данных и т.п. Определенные стандарты, соглашения и процедуры могут быть также созданы внутри организации или даже проектной команды. Эти стандарты поддерживают координацию между определенными видами деятельности, группами операций, минимизируют сложность, могут быть связаны с вопросами ожидания и обработки изменений, рисков и вопросами конструирования для проверки и дальнейшего тестирования.

Тестирование

В разделе **основы тестирования** приводятся терминология, методы и инструменты подготовки и проведения процесса тестирования, а также показатели оценки данных статистического анализа процесса тестирования. В соответствии с принятой терминологией при тестировании выявляются следующие недостатки: отказы и дефекты, сбои, ошибки. Важно чётко разделять *причину* нарушения работы прикладных программ, обычно описываемую терминами *недостаток* или *дефект*, и наблюдаемый нежелательный эффект, вызываемый этими причинами, – *сбой*. Термин *ошибка*, в зависимости от контекста, может описывать и как причину сбоя, и сам сбой. Тестирование позволяет обнаружить дефекты, приводящие к сбоям.

Концепции, стратегии, техники и измерения тестирования должны быть объединены в единый процесс тестирования (от планирования тестов до оценки их результатов) как деятельности по обеспечению качества ПО, поддерживающей «правила игры» для членов команды тестирования. Только в том случае, если тестирование рассматривать как один из важных процессов всей деятельности по созданию и поддержке программного обеспечения, можно добиться оценки стоимости соответствующих работ и, в конце концов, выполнить те ограничения, которые определены для проекта.

Работы по управлению процессом тестирования, ведущиеся на разных уровнях, должны быть организованы в единый процесс, на основе учета четырех элементов и связанных с ними факторов: *участников процесса* (в том числе, в контексте организационной структуры и культуры), *инструментов, регламентов и количественных оценок измерения.*

В состав этого процесса должны входить:

- планирование работ по тестированию (составление планов, тестов, наборов данных) и измерению показателей качества ПО;
- генерация необходимых тестовых сценариев, соответствующих среде выполнения ПО;

- проведение тестирования с учетом следующих положений;
- все работы и результаты процесса тестирования должны обязательно фиксироваться;
- форма журналирования работ и их результатов должна быть такой, чтобы соответствующее содержание было понятно, однозначно интерпретируемо и повторяемо другими лицами;
- тестирование должно проводиться в соответствии с заданными и документированными процедурами;
- тестирование должно производиться над однозначно идентифицируемой версией и конфигурацией ПО;
- сбор данных об отказах, ошибках и др. непредвиденных ситуациях при выполнении программного продукта;
- подготовка отчетов по результатам тестирования и оценки характеристик ПО.

Создание

Процесс разработки программного обеспечения представляет собой специфический технологический процесс преобразования исходных требований заказчика о предметной области в готовый программный продукт и состоит из совокупности взаимосвязанных этапов (технологических операций). В процессе преобразования участвуют специалисты, выполняющие определенную работу с помощью специфических инструментальных средств и предметов производства. На вход этого процесса в виде требований поступают сведения о предметной области. На выходе процесса – разработанное программное обеспечение и техническая документация. Механизмами, обеспечивающими выполнение процесса, являются специалисты и используемые ими инструментальные средства. Весь процесс протекает в соответствии с принятыми на предприятии стандартами и нормативными документами, регламентирующими

требования к процессам жизненного цикла, качества, документирования программного продукта и оценке технологической зрелости организаций-разработчиков. К таким документам, в частности, относятся:

1. Международный стандарт ISO/IEC 12207 «Информационная технология. Жизненный цикл процессов разработки программного обеспечения».
2. ГОСТ Р ИСО/МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств».

- ГОСТ Р ИСО/МЭК 15288-2005. Системная инженерия. Процессы жизненного цикла систем.
- ГОСТ Р ИСО/МЭК 15910-2002. Процесс создания документации пользователя программного средства.
- ГОСТ Р ИСО 9127-94. Документация пользователя и информация на упаковке для потребительских программных пакетов.
- ГОСТ Р ИСО/МЭК 9126-93. Оценка программного продукта. Характеристики качества и руководящие указания по их применению.
- CMM — Capability Maturity Model (Модель зрелости процесса конструирования ПО).

Исходя из вышеизложенного, можно констатировать, что технологический процесс (схема) создания программного продукта сводится к преобразованию сведений о какой-либо предметной области в комплекс исполняемых программных модулей, способных обеспечить решение определенных задач пользователей, такие схемы получили название моделей жизненного цикла.

Сопровождение

Фаза сопровождения в жизненном цикле обычно начинается сразу после приемки/передачи продукта и действует в течение периода гарантии или технической поддержки. Объективная потребность в сопровождении связана:

- с обнаружением при эксплуатации ПП скрытых дефектов и необходимости устранение сбоев;
- улучшение дизайна;
- реализации новых функциональных возможностей;
- созданием интерфейсов взаимодействия с внешними системами;
- адаптации для обеспечения возможности работы ПП на другой программно-аппаратной платформе;
- изменением бизнес-процессов в предметной области;
- вывода программного обеспечения из эксплуатации.

Процесс сопровождения выполняется как перед вводом системы в эксплуатацию, так и после этого и состоит из планирования деятельности по сопровождению системы, организации перехода к ее полнофункциональному использованию, обучению пользователей и их ежедневной поддержки при работе с текущей версией продукта. Если новая система должна заменить старую систему, важно обеспечить плавный переход со старой системы на новую, максимально естественно для пользователей.

Сопровождение программного обеспечения определяется стандартами как:

- совокупность деятельности, необходимой для обеспечения эффективной (с точки зрения затрат) поддержки ПО;
- модификация программного продукта после передачи в эксплуатацию для устранения сбоев, улучшения показателей производительности и/или других характеристик (атрибутов) продукта, или адаптации продукта для использования в модифицированном окружении;
- процесс модификации программного продукта в части его кода и документации для решения возникающих проблем при эксплуатации или реализации новых потребностей пользователей, в улучшения тех или иных характеристик продукта;
- модификацией программного продукта в процессе эксплуатации при условии сохранения целостности продукта.

Деятельность персонала сопровождения включает четыре ключевых аспекта: поддержка контроля программного обеспечения в течение всего цикла эксплуатации; поддержка модификаций ПО; совершенствование существующих функций; предотвращение снижения производительности ПО до критического уровня.

Техники сопровождения

Для реализации изменений программисты тратят значительную часть времени на чтение и формирование понимания в большинстве случаев чужого программного кода. Средства работы с кодом являются ключевым инструментом для решения этой задачи. Четкая, однозначная и лаконичная документация обеспечивает адекватное понимание ПО.

Общепринятыми техниками, используемыми в процессе сопровождения ПО, являются: реинжиниринг, обратный инжиниринг и рефакторинг.

Реинжиниринг – это улучшение возможностей, функций в устаревшем ПО, путем его реорганизации и реструктуризации, перепрограммирования или настройка на другую платформу или среду с обеспечением удобства его сопровождения.

Обратный инжиниринг состоит в анализе программного обеспечения с целью идентификации программных компонент и связей между ними, восстановлении спецификации по полученному коду ПО (особенно, когда в нее внесено много изменений) для наблюдения за ней на более высоком уровне. Конечными целями обратного инжиниринга могут быть: создание новой документации на существующее ПО; восстановление дизайна и т.д.

Рефакторинг – трансформация программного обеспечения, в процессе которого ПО реорганизуется (не переписываясь) с целью улучшения структуры, без изменения ее функционала. Рефакторинг ориентирован на улучшение структурных характеристик и качественных показателей объектно-ориентированных программ без изменения их поведения. Этот процесс реализуется путем изменения отдельных операций над текстами,

интерфейсами, средой программирования и выполнения ПО, а также настройки или внесения изменений в инструментальные средства поддержки ПО.

Выводы

Получив набор технических решений, нужно для каждого из них выбрать методологию разработки. Выбор методологии должен учитывать специфику проекта. При работе с государственными структурами или крупными коммерческими заказчиками часто проявляются особенности, препятствующие применению гибких методологий. В одних случаях исполнитель может обойти ограничения, например, абстрагируя свой внутренний процесс разработки от процессуальных ограничений заказчика. В других случаях, возможно, следует отказаться от гибкой методологии, помня, что у любой методологии есть свои ограничения.

Согласно современной методологии, процесс создания ИС представляет собой процесс построения и последовательного преобразования ряда согласованных моделей на всех этапах жизненного цикла (ЖЦ) ИС. На каждом этапе ЖЦ создаются специфичные для него модели – организации, требований к ИС, проекта ИС, требований к приложениям и т.д. Модели формируются рабочими группами команды проекта, сохраняются и накапливаются в репозитории проекта. Создание моделей, их контроль, преобразование и предоставление в коллективное пользование осуществляется с использованием специальных программных инструментов – CASE-средств.

ЛЕКЦИЯ 3
МЕТОДОЛОГИИ ПРОЕКТИРОВАНИЯ ПРОГРАММНЫХ ПРОДУКТОВ

Если руководствоваться эффективными принципами, то вы всегда будете знать, как поступить, и сможете быстрее принимать решения
Рей Далио

В основе *метода проектирования* (также употребляются термины *методика* или *методология*) лежит алгоритм, который определяет проектные действия, их последовательность, состав исполнителей, средства и ресурсы, требуемые для выполнения этих действий.

Методология – это система принципов, а также совокупность идей, понятий, методов, способов и средств, определяющих стиль разработки программного обеспечения.

Методология – это реализация стандарта. Сами стандарты лишь говорят о том, что должно быть, оставляя свободу выбора и адаптации.

Конкретные вещи реализуется через выбранную методологию. Именно она определяет, как будет выполняться разработка. Существует много успешных методологий создания программного обеспечения. Выбор конкретной методологии зависит от размера команды, от специфики и сложности проекта, от стабильности и зрелости процессов в компании и от личных качеств сотрудников.

Процесс проектирования ИС делится на совокупность взаимосвязанных действий, каждое из которых может иметь свой объект.

Действия могут быть:

- *проектировочными*, формирующими или изменяющими текущий проект;
- *оценочными*, вырабатывающими по установленным критериям оценку результатов проектирования.

Совокупность состояний, которые проходит ИС в своем развитии, от момента принятия решения о создании системы до момента прекращения ее

функционирования, называется *жизненным циклом* (ЖЦ) программной системы. Жизненный цикл – это своего рода «карта-путеводитель» для всех участников проекта, которая помогает им понять, не выходят ли они за определенные для них границы. Для управления программным проектом возникает необходимость в заранее определенной некой карте для планирования действий и хронологий их выполнения. Сегодня существует огромное количество различных процессов для создания ПО. Тем не менее, именно технологий, рассматривающих полный жизненный цикл проекта разработки ПО, сочетающих в себе научный подход, серьезную базу исследований и имеющих историю реального использования и адаптации, относительно немного.

К основным требованиям, предъявляемым к выбираемой методологии проектирования, относятся следующие:

- созданный с помощью этой методологии проект должен максимально соответствовать требованиям заказчика, причем требования могут меняться уже в ходе создания ПС;
- методологии должна максимально отражать все этапы жизненного цикла проекта и служить основой связи между проектированием и сопровождением системы в процессе ее эксплуатации;
- методологии должна обеспечивать минимальные затраты времени и средств на проектирование и сопровождение системы при условии обеспечения должного качества конечного продукта.

Системный подход к проектированию программных продуктов

Методологическую основу проектирования программных продуктов (ПП) составляет системный подход. Весь мир можно рассматривать как сложную взаимосвязанную совокупность естественных и искусственных систем. Это могут быть достаточно сложные системы (например, планеты в составе Солнечной системы), системы средней сложности (космический корабль) или сверхсложные системы (системы молекулярных

взаимодействий в живых организмах). Искусственные системы, к которым относится ПП, как правило, по своей сложности занимают среднее положение. *Системный подход* – это методология специального научного познания и социальной практики, а также объяснительный принцип, в основе которого лежит исследование объектов как систем. Он ориентирует исследование на:

- раскрытие целостности объекта и обеспечивающих его механизмов;
- выявление многообразных типов связей сложного объекта;
- сведение этих связей в единую теоретическую картину.

Системный подход реализует представление сложного объекта в виде иерархической системы взаимосвязанных моделей, позволяющих фиксировать целостные свойства объекта, его структуру и динамику.

Системный анализ – это совокупность методологических средств, используемых для подготовки и обоснования решений по сложным проблемам социального, технического и экономического характера. Он основывается на системном подходе, а также на ряде математических дисциплин и современных методов управления. Основной процедурой системного анализа является построение обобщенной модели, адекватно отображающей интересующие исследователя свойства реальной системы и ее взаимосвязи. Основополагающая концепция системного анализа состоит в построении при помощи специализированных графических методов совокупности моделей деятельности организации, которые дают возможность заинтересованным лицам получить понятную и детализированную общую картину бизнес-процессов.

Принципы системного анализа:

- *оптимальность* – в результате анализа необходимо найти оптимальное решение задачи;
- *эмерджентность* (эмергентность) – появление у системы свойств, не присущих её элементам в отдельности; несводимость свойств системы к сумме свойств её компонентов (Wikipedia). Чем больше

система и чем больше различие между частью и целым, тем выше вероятность того, что свойства целого могут сильно отличаться от свойств его частей. Этот фактор играет достаточно важную роль и его необходимо учитывать при проведении анализа;

- *системность* – исследование объекта, с одной стороны, как единого целого, а с другой, как части более крупной системы, с которой объект находится в определенных отношениях;
- *иерархичность* – определение в системе структурных отношений, характеризуемых упорядоченностью, организованностью взаимодействий между отдельными ее уровнями по вертикали.
- *интеграция* – изучение интеграционных свойств и закономерностей системы;
- *формализация* – получение комплексных количественных характеристик.

Классификация методов проектирования программных продуктов

Методы проектирования ИС можно классифицировать по степени использования средств автоматизации, типовых проектных решений, адаптивности к предполагаемым изменениям.

Так, по степени автоматизации методы проектирования разделяются на:

- ручное, при котором проектирование компонентов ИС осуществляется без использования специальных инструментальных программных средств, а программирование – на алгоритмических языках;
- компьютерное, при котором производится генерация или конфигурирование (настройка) проектных решений на основе использования специальных инструментальных программных средств.

По степени использования типовых проектных решений различают следующие методы проектирования:

- *оригинальное* (индивидуальное), когда проектные решения разрабатываются «с нуля» в соответствии с требованиями к АИС. Характеризуется тем, что все виды проектных работ ориентированы на создание индивидуальных для каждого объекта проектов, которые в максимальной степени отражают все его особенности;
- *типовое*, предполагающее конфигурирование ИС из готовых типовых проектных решений (программных модулей). Выполняется на основе опыта, полученного при разработке индивидуальных проектов. Типовые проекты, как обобщение опыта для некоторых групп организационно-экономических систем или видов работ, в каждом конкретном случае связаны со множеством специфических особенностей и различаются по степени охвата функций управления, выполняемым работам и разрабатываемой проектной документации.

По степени адаптивности проектных решений выделяют методы:

- *реконструкции*, когда адаптация проектных решений выполняется путем переработки соответствующих компонентов (перепрограммирования программных модулей);
- *параметризации*, когда проектные решения настраиваются (генерируются) в соответствии с изменяемыми параметрами;
- *реструктуризации* модели, когда изменяется модель проблемной области, на основе которой автоматически заново генерируются проектные решения.

К существующей классификации в зависимости от используемой в ней модели жизненного цикла (водопадные и итерационные методологии) добавилась более общая классификация на прогнозируемые и адаптивные методологии.

Прогнозируемые (промышленные) методологии фокусируются на детальном планировании будущего. Известны запланированные задачи и ресурсы на весь срок проекта. Команда с трудом реагирует на возможные

изменения. План оптимизирован исходя из состава работ и существующих требований. Изменение требований может привести к существенному изменению плана, а также дизайна проекта. Часто создается специальный комитет по «управлению изменениями», чтобы в проекте учитывались только самые важные требования.

Адаптивные методологии нацелены на преодоление ожидаемой неполноты требований и их постоянного изменения. Когда меняются требования, команда разработчиков тоже меняется. Команда, участвующая в адаптивной разработке, с трудом может предсказать будущее проекта. Существует точный план лишь на ближайшее время. Более удаленные во времени планы существуют лишь как декларации о целях проекта, ожидаемых затратах и результатах.

Среди адаптивных методологий: (Scrum, Crystal, Extreme Programming, Adaptive Software Development, DSDM, Feature Driven Development, Lean software development).

Еще один вариант классификации, подобный предыдущей, это ранжирование методологий по «весу», т. е. по количеству формализованных процессов и детальности их регламентации: чем больше процессов документировано (т.е. чем более детально описана методология), тем больше будет ее «вес». Придерживаясь такой классификации все методологии делят на тяжеловесные (промышленные) и легковесные (гибкие).

Тяжеловесные методологии рассчитаны на среднюю квалификацию исполнителей сравнительно большое количество специализаций внутри команды, предъявляют более низкие требования к стабильности команды, но при этом не имеют ограничений по сложности и размерам реализуемых проектов. Однако в рамках таких методологий предъявляются существенные требования управлению, формализации и документации процессов, а также значительное время уделяется предпроектному анализу и сбору требований. К таким методологиям можно отнести:

− ГОСТ 19 «Единая система программной документации» и ГОСТ 34 «Стандарты на разработку и сопровождение автоматизированных систем» ориентированы на последовательный подход к разработке программного обеспечения;

− Capability Maturity Model for Software (SW-CMM) определяет пять уровней «зрелости проекта»;

− Rational Unified Process (RUP) – итеративная модель разработки;

− Microsoft Solutions Framework (MSF) – база знаний компании Microsoft по разработки программ;

− Personal Software Process – модель определяет требования к компетенциям разработчика.

- Team Software Process – модель ориентирует на самоуправляемые команды от 3 до 20 разработчиков.

− и др.

Легковесные или ***agile***-методики, в отличие от тяжеловесных, не требуют значительных расходов, связанных с управлением проектом, просчету рисков и организацией стабильных коммуникаций. В рамках таких подходов основной упор делается на разработку функциональности, а не на предпроектный анализ, допускается совмещение ролей внутри команды. Однако эффективность такого подхода сильно зависит от квалификации членов команды и ее стабильности, а также от сложности и объемов реализуемых проектов. Примерами таких методологий могут служить:

− eXtreme Programming или XP – экстремальное программирование, предлагающее 12 инженерных практик;

− Crystal Clear – семейство методологий, определяют необходимую степень формализации процесса разработки в зависимости от количества участников и критичности задач [13];

− Feature Driven Development (FDD) – функционально-ориентированная разработка;

− OpenUP – итеративно-инкрементальный метод разработки программ, позиционируется как легкий и гибкий вариант тяжеловесной методологии RUP;

− Scrum – управление разработкой информационных систем с высокой степенью неопределенности;

− Kanban – методология «бережливого производства»;

− и др.

Сочетание различных признаков классификации методов обусловливает характер используемых технологий проектирования ИС, среди которых выделяют два основных класса: каноническую и индустриальную технологии. Индустриальная технология проектирования, в свою очередь, разбивается на два подкласса: автоматизированное (использование CASE-технологий) и типовое (параметрически-ориентированное или модельно-ориентированное) проектирование (табл.1). Использование индустриальных технологий не исключает использования в отдельных случаях канонических.

Табл. 1. Характеристики классов технологий проектирования

Класс технологии проектирования	**Степень автоматизации**	**Степень типизации**	**Степень адаптивности**
Каноническое проектирование	Ручное проектирование	Оригинальное проектирование	Реконструкция
Индустриальное автоматизированное проектирование	Компьютерное проектирование	Оригинальное проектирование	Реструктуризац модели (генерац ЭИС)
Индустриальное типовое проекти-рование	Компьютерное проектирование	Типовое сборочное проектирование	Параметризация и реструктуризац модели (конфигурац ЭИС)

Методы детализированного проектирования

Главной задачей системного анализа является поиск путей по разложению труднопонимаемой задачи на ряд задач, имеющих достаточно простое для понимания решение. Единственный эффективный подход к решению этой проблемы заключается в построении сложной системы из небольшого количества крупных частей, каждая из которых, в свою очередь, строится из частей меньшего размера, и т.д. Декомпозиция выполняется до тех пор, пока самые небольшие части можно будет строить из имеющегося материала. Этот подход известен под самыми разными названиями, среди них такие, как «разделяй и властвуй», иерархическая декомпозиция и др. По отношению к проектированию сложной программной системы это означает, что ее необходимо разделить (декомпозировать) на небольшие подсистемы, каждую из которых можно разрабатывать независимо от других. Это позволяет при разработке подсистемы любого уровня иметь дело только с отдельной, сравнительно простой подсистемой, а не со все ПП в целом. Правильная декомпозиция является главным способом преодоления сложности разработки больших систем ПО и подразумевает следующее:

- количество связей между отдельными подсистемами должно быть минимальным (принцип «слабой связанности»);
- связность отдельных частей внутри каждой подсистемы должна быть максимальной (принцип «сильного сцепления»).

Методы детализированного проектирования могут быть разделены на несколько категорий, соответствующих различным подходам проектирования:

1) Структурный подход – это классический подход, для которого характерно раздельное моделирование структур данных и алгоритмов работы с ними. Основной задачей является идентификация функций системы, детальная проработка и создание спецификаций для реализации отдельных функций. В основном применяется для нижних уровней декомпозиций.

2) Объектно-ориентированный подход. Для него характерно представление системы в виде взаимодействия объектов. Под объектом понимается сущность, представляющая собой совокупность данных, характеризующих состояние объекта и процедуры их обработки. Объекты объединяются в классы, представляющие собой абстрактные описания атрибутов и методов объекта. Основой для проектирования является модель предметной области, построенная в терминах взаимодействующих объектов. Метод эффективен при разработке ПО средней сложности, однако не обладает необходимыми возможностями для моделирования сложных структур данных.

3) Функционально-ориентированная разработка (FDD – Feature Driven Development). В основе проектирования модель предметной области, отображающая взаимодействие пользователя с системой, на основе которой строится сценарий взаимодействия пользователя с системой и выявляются ключевые функции. Формируются мини-команды для реализации отдельных функций. Данный подход основан на гибкой технологии разработки, однако предполагает широкое использование формальных средств моделирования и проектирования.

4) Проектирование на основе данных (Data Driven Design/Data Structure Centered Design). Основа проектирования: модель потока данных. Выявляются источники и потребители информации в системе, описываются структуры данных, участвующие в процессах обмена и хранения информации. На основании этого строится модель метаданных системы, описывающая структуры, форматы и характеристики данных. После этого описываются процессы обработки данных. Для каждого процесса определяются его входы и выходы и процедуры преобразования данных, а также процессы в управлении потоками данных.

5) Проектирование на основе тестов. Основан на упреждающей разработке тестов. Каждое из требований специфицируется набором тестов, описывающих как корректные, так и некорректные сценарии использования

системы. Задача разработчиков создать функционал, обеспечивающий корректное прохождения теста. Тесты выполняются в автоматическом режиме. При этом трудоемкость разработки возрастает, однако ускоряется отладка и облегчается дальнейшее сопровождение системы. Сложно обеспечить полное покрытие тестами всех сценариев использования системы, включая ошибочные.

6) Компонентное проектирование. Компоненты являются относительно независимыми программными единицами, обладающими определенным набором функций и однозначно определенным интерфейсом, которые могут реализовываться, собираться и развертываться независимо друг от друга. Основная задача: эффективное повторное использование компонентов при разработке различных систем. При этом компонент нередко рассматривается в виде черного ящика, интерфейс которого определен, но внутренняя структура неизвестна. Трудоемкость создания отдельных компонентов увеличивается, однако снижается трудоемкость создания систем, используя заранее созданные компоненты.

7) Сервис-проектированный подход является дальнейшим развитием компонентного подхода и применяется для разработки систем в соответствии с сервис-ориентированной архитектурой (Service Oriented Architecture). Система представляет собой систему компонентов или сервисов, каждый из которой обладает стандартизованным интерфейсом и выполняет единственную функцию. Сервисы слабо связаны между собой и взаимодействуют через общий коммуникационный механизм – шину сервиса (Enterprise Service Bus). Преимущество: гибкая архитектура системы, высокая степень повторного использования сервисов. Недостаток: недостаточная эффективность существующих решений и технологий.

Рассмотренные методологические подходы к проектированию не являются универсальными и самодостаточными. Они могут применяться совместно, на основе синтетических технологий проектирования, не обладающих концептуальным единством. На настоящий день отсутствует

универсальная комплексная методология проектирования, обеспечивающая совместное решение всех задач на разных стадиях проектирования ИС. Рассмотрим подробнее наиболее популярные подходы к детализированному проектированию ПП.

Структурные методы анализа и проектирования ИС

Методология структурного анализа и проектирования ПО определяет руководящие указания для оценки и выбора проекта разрабатываемого ПО, шаги работы, которые должны быть выполнены, их последовательность, правила распределения и назначения операций и методов. В основе структурного подхода лежит *последовательная функциональная декомпозиция*, при которой структура системы описывается в терминах *иерархии ее функций* и передачи информации между отдельными функциональными элементами. Структурные методологии жестко регламентируют фазы анализа требований и проектирования спецификаций.

Идеи структурного подхода:

- преодолеть сложность больших систем путем разделения их на части;
- представить части в виде иерархических структур;
- использовать графические нотации, для облегчения понимаемости сложных систем.

Рассмотрим их подробнее. Структурным анализом принято называть метод исследования системы, начинающий с ее общего обзора, который затем детализируется, приобретая иерархическую структуру со все большим числом уровней. Для таких методов характерно: разбиение системы на уровни абстракции с ограничением числа элементов на каждом из уровней (обычно от 3 до 6); ограниченный контекст, включающий лишь существенные на каждом уровне детали; использование строгих формальных правил записи; последовательное приближение к конечному результату. Методы структурного анализа и проектирования стремятся преодолеть сложность больших систем путем разделения их на части («черные ящики») и иерархической организации этих «черных ящиков».

Выгода в использовании «черных ящиков» заключается в том, что их пользователю не требуется знать, как они работают, необходимо знать лишь их входы и выходы, а также назначение (т.е. функции, которые они выполняет). При разбиении систем необходимо руководствоваться следующими критериями:

- каждый «черный ящик» должен реализовывать единственную функцию системы;
- функция каждого «черного ящика» должна быть легко понимаема независимо от сложности ее реализации;
- связь между «черными ящиками» должна вводиться только при наличии связи между соответствующими функциями системы
- связи между «черными ящиками» должны быть простыми, насколько это возможно, для обеспечения независимости между ними.

Второй важной идеей, лежащей в основе структурных методов, является идея иерархии. Для понимания сложной системы недостаточно разбиения ее на части, необходимо эти части организовать определенным образом, а именно в виде иерархических структур. Строгая иерархическая декомпозиция подчиняется нескольким правилам:

1) на каждом уровне иерархии план или проект должен иметь законченный вид на данном уровне детализации;
2) на любом уровне иерархии каждое разбиение полностью охватывает отдельную функцию или проблему, соответствующую данному уровню детализации. Например, если имеются две функции, определяемые на четвертом уровне, но только одна из них может быть явно разложена дальше на пятом (нижнем) уровне, то другая функция четвертого уровня должна быть описана со степенью детализации, которая соответствует пятому уровню.

В иерархической декомпозиции, в какой-то степени, сохраняется однотипность отдельных элементов, как и в случае линейной декомпозиции, но каждому элементу дополнительно присваивается иерархический уровень или устанавливается родительский элемент. В силу визуального совпадения иерархическую декомпозицию называют иногда деревом. При этом выделенные функции допускают внутреннюю иерархическую декомпозицию.

В рамках системного подхода существует следующее разделение на иерархические уровни (блочно-иерархический подход):

1) системный уровень – на нем собирается информация в целом;
2) макроуровень – на нем собирается информация о конкретных агрегатах, установках; вся информация предоставляется в виде схем, таблиц;
3) микроуровень – детали машин, приборы, устройства; информация хранится в виде кинематических схем, принципиальных схем.

В целом разделение на уровни происходит по следующему принципу: вся информация на одном уровне должна быть обозрима и воспринимаема одним человеком. Основной задачей проектировщика является выделение в процессе проектирования основной и второстепенной задачи. Основная задача проектирования может быть прямой и обратной. Прямая задача отвечает на вопрос «Как сделать чтобы…». Обратная задача – «Что будет если…». Таким образом, можно четко сформулировать, что прямая задача имеет множество решений, а обратная – только одно.

Кроме того, структурные методы широко используют визуальное моделирование, служащее для облегчения понимания сложных систем. Здесь используются различные модели, описывающие: функции, которые система должна выполнять; процессы, обеспечивающие выполнение указанных функций; данные, необходимые при выполнении функций, и отношения между этими данными; организационные структуры, обеспечивающие

выполнение функций; материальные и информационные потоки, возникающие в ходе выполнения функций.

Среди многообразия средств, предусмотренных для проведения структурного анализа, наиболее часто и эффективно применяются:

- DFD (Data Flow Diagrams) – диаграммы потоков данных в нотациях Гейна-Сарсона, Йордона-Де Марко и других, обеспечивающие требования анализа и функционального проектирования информационных систем;
- STD (State Transition Diagrams) – диаграммы перехода состояний, основанные на расширениях Хартли и Уорда-Меллора для проектирования систем реального времени;
- ERD (Entity-Relationship Diagrams) – диаграммы «сущность-связь» в нотациях Чена и Баркера;
- структурные карты Джексона и/или Константайна для проектирования межмодульных взаимодействий и внутренней структуры объектов;
- FDD (Functional Decomposition Diagrams) – диаграммы функциональной декомпозиции;
- SADT (Structured Analysis and Design Technique) – технология структурного анализа и проектирования;
- семейство IDEF (Integration Definition for Function Modeling).

Все наиболее распространенные методы структурного подхода базируются на ряде общих принципов.

***Принципами структурного подхода являются*:**

принцип «разделяй и властвуй» – принцип решения трудных проблем путем разбиения их на множество меньших независимых задач, легких для понимания и решения;

принцип иерархического упорядочения – принцип организации составных частей системы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне;

принцип абстрагирования – выделение существенных аспектов системы и отвлечение от несущественных;

принцип формализации – применение строго методического подхода к решению проблемы;

принцип упрятывания – свертывание несущественной на конкретном этапе информации: каждая часть «знает» только необходимую ей информацию;

принцип концептуальной общности – следование единой философии на всех этапах ЖЦ (структурный анализ – структурное проектирование – структурное программирование – структурное тестирование);

принцип полноты – контроль за присутствием лишних элементов;

принцип непротиворечивости – обоснованность и согласованность элементов системы;

принцип логической независимости – концентрация внимания на логическом проектировании для обеспечения независимости от физического проектирования;

принцип независимости данных – модели должны быть проанализированы и спроектированы независимо от процессов их логической обработки, а также от их физической структуры и распределения;

принцип структурирования данных – данные должны быть структурированы и иерархически организованы;

принцип доступа конечного пользователя – пользователь должен иметь средства доступа к базе данных, которые он может использовать непосредственно (без программирования).

При разработке системы «снизу вверх», от отдельных задач ко всей системе целостность теряется, возникают проблемы при описании информационного взаимодействия отдельных компонентов.

Современные структурные методологии анализа и проектирования классифицируются по следующим признакам.

- *По отношению к школам:*
- Software Engineering (SE) – является нисходящим поэтапным подходом к разработке ПО, начинающимся с общего взгляда на его функционирование, последующей декомпозиции на подфункции до тех пор, пока они не станут достаточно малы для их реализации кодированием. В результате получается иерархическая, структурированная, модульная программа. SE является универсальной дисциплиной разработки ПО, успешно применяющейся как при разработке систем реального времени, так и при разработке информационных систем.
- Information Engineering (IE) – более новая дисциплина, которая имеет более широкую область применения, чем SE. IE является дисциплиной построения систем вообще, а не только систем ПО, и включает этапы более высокого уровня (например, стратегическое планирование), однако на этапе проектирования систем ПО эти дисциплины аналогичны. С другой стороны, IE более узкая дисциплина, чем SE, т.к. IE используется только для построения информационных систем, а SE - для всех типов систем.

- *По порядку построения модели*.

  Разработка ПО основана на модели ВХОД-ОБРАБОТКА-ВЫХОД: данные входят в систему, обрабатываются или преобразуются и выходят из системы. Такая модель используется во всех структурных методологиях. При этом важен порядок построения модели.

  Различают:

- процедурно-ориентированный подход – традиционный, регламентирует первичность проектирования функциональных компонент по отношению к проектированию структур данных. Основная идея: требования к данным раскрываются через функциональные требования;
- подход, ориентированный на данные – специфицируются данные, а затем определяются процессы, использующие эти данные. Идея: наиболее важными являются вход и выход модели. Структуры данных (но не потоки данных) определяются первыми, а процедурные компоненты строятся как производные от структур данных
- Информационно-ориентированный подход (как часть IE-дисциплины) – отличается от подхода, ориентированного на данные, только тем, что позволяет работать с неиерархическими структурами данных.

3. *По типу целевых систем*:
   - для систем реального времени (СРВ) – основная особенность систем реального времени заключается в том, что они контролируют и контролируются внешними событиями; реагирование на эти события во времени – основная и первоочередная функция таких систем;
   - для информационных систем (ИС).

Объектно-ориентированная методика проектирования ИС

Объектно-ориентированный анализ и проектирование – это технология разработки программных систем, в основу которых положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов. Основными понятиями объектно-ориентированного подхода являются объект и класс:

- объект – предмет или явление, имеющее четко определенное поведение и обладающие состоянием, поведением и индивидуальностью. Структура и поведение схожих объектов определяют общий для них класс;
- класс – это множество объектов, связанных общностью структуры и поведения.

Объектно-ориентированный подход (ООП) использует объектную декомпозицию, при этом статическая структура описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. Целью применения данной методики является выделение объектов, составляющих организацию, и распределение между ними ответственностей за выполняемые действия. В качестве объектов предметной области могут рассматриваться конкретные предметы, а также абстрактные или реальные сущности. Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой информационной системы от стадии формирования требований до стадии реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.

Объекты и классы организуются с использованием следующих принципов:

- *Принцип инкапсуляции* декларирует запрещение любого доступа к атрибутам объекта, кроме как через его операции. В соответствии с этим внутренняя структура объекта скрыта от пользователя, а любое его действие инициируется внешним сообщением, вызывающим выполнение соответствующей операции. Инкапсуляция – сокрытие отдельных деталей внутреннего устройства классов от внешних по отношению к нему объектов и пользователей.

- *Принцип наследования* декларирует создание новых классов от общего к частному. Такие новые классы сохраняют все свойства классов-родителей и при этом содержат дополнительные атрибуты и операции, характеризующие их специфику.
- *Принцип полиморфизма* декларирует возможность работы с объектом без информации о конкретном классе, экземпляром которого он является. Действия, выполняемые одноименными методами, могут отличаться в зависимости от того, к какому классу относится тот или иной метод, т.е. каждый объект может реагировать индивидуально на это (одно и то же для различных объектов) сообщение.
- *Принцип абстрагирования* – выделение значимых характеристик объекта.
- *Принцип прототипирования* – создание объекта-образца, по образу и подобию которого создаются другие объекты. Объекты-копии могут сохранять связь с родительским объектом, автоматически наследуя изменения в прототипе

Базовыми составляющими ООП можно определить:

- унифицированный процесс RUP – упорядоченный подход к распределению задач и обязанностей в организации
- унифицированный язык моделирования UML – стандартизированный подход к визуализации моделей систем
- шаблоны (паттерны) проектирования – обобщенное решение типичной проблемы.

Объектно-ориентированные методологии разработки ПО являются развитием структурных методологий. Данный подход не является противопоставлением структурному подходу, более того, методы и нотации структурных методологий, а именно, базовые диаграммы потоков данных, диаграммы "сущность-связь", диаграммы переходов состояний, используются при объектно-ориентированных анализе и проектировании для

моделирования структуры и поведения самих объектов. Фактически **структурный подход поглощается объектно-ориентированным подходом.**

Сравнение объектно-ориентированного и структурного подхода

Структурный подход	Объектно-ориентированный подход
Функциональные методики рассматривают организацию как набор функций, преобразующий поступающий поток информации в выходной поток.	Объектные методики рассматривают моделируемую организацию как набор взаимодействующих объектов – производственных единиц. Объект определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение.
	Четкое разделение функций (методов обработки данных) от самих данных — помимо функциональной декомпозиции существует структура данных, находящаяся на втором плане. Кроме того, не ясны условия выполнения процессов обработки информации, которые динамически могут изменяться.
Проектирование ПП по принципу "сверху – вниз", когда каждый функциональный блок может быть декомпозирован на множество подфункций и т.д., выполняя, таким	Проектирование ПП по принципу «снизу – вверх»: сначала выделяются классы объектов, а далее в зависимости от возможных состояний объектов (жизненного

образом, модульное проектирование систем	цикла объектов) определяются методы обработки (функциональные процедуры), что обеспечивает наилучшую реализацию динамического поведения информационной системы
Подходит для более регламентированных задач	Подходит для более адаптивных бизнес-процессов (управления рабочими потоками, реализации динамических запросов к информационным хранилищам)
Функциональная модель лучше систематизирована, т.к. ей присуща процедурная строгость декомпозиции ИС.	Объектная модель естественна, поскольку ориентирована на человеческое восприятие мира, но менее наглядна.
Объектный подход позволяет хорошо описать систему на уровне общего описания системы, т.к. он основан на понятии сценария использования. Ключевым является понятие о сценарии использования как о сеансе взаимодействия действующего лица с системой, в результате которого действующее лицо получает нечто, имеющее для него ценность.	Функциональные методики в целом лучше дают представление о существующих функциях в организации, о методах их реализации, причем чем выше степень детализации исследуемого процесса, тем лучше они позволяют описать систему. Под лучшим описанием в данном случае понимается наименьшая ошибка при попытке по полученной модели предсказать поведение реальной системы. На

	уровне отдельных рабочих процедур их описание практически однозначно совпадает с фактической реализацией в потоке работ. На уровне общего описания системы функциональные методики допускают значительную степень произвола в выборе общих интерфейсов системы, ее механизмов и т.д., то есть в определении границ системы.
При объектно-ориентированном подходе проектировщик сосредоточивается на тех функциях системы, которые оправдывают ее существование. Однако и в этом случае задача определения границ системы, выделения внешних пользователей является сложной.	Технология потоков данных легко решает проблему границ системы, поскольку позволяет за счет анализа информационных потоков выделить внешние сущности и определить основной внутренний процесс. Однако отсутствие выделенных управляющих процессов, потоков и событийной ориентированности не позволяет предложить эту методику в качестве единственной.
Объектная декомпозиция дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств.	В функциональных моделях при большом количестве уровней декомпозиции модель может быть сравнительно большой.

Функциональные модели, как правило, не пригодны для повторного использования.	Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей. В случае наследования функций можно абстрагироваться от конкретной реализации процедур (абстрактные типы данных), которые отличаются для определенных подклассов ситуаций. Это дает возможность обращаться к подобным программным модулям по общим именам (полиморфизм) и осуществлять повторное использование программного кода при модификации программного обеспечения.
Не требует высоких начальных затрат	Высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки двух–трех проектов и накопления повторно используемых компонентов.
Подход от выполняемых функций интуитивно лучше понимается исполнителями при получении от них	Понимание объектно-ориентированных моделей требуют предварительной подготовки и

информации об их текущей работе	специализированных методов сбора информации

Объектная технология открывает возможность построения гораздо более сложных систем и программных комплексов, чем допускала технология структурного программирования.

Разница в подходах к разработке базируется именно на высокой сложности современного программного обеспечения.

Как результат применение структурной технологии ведет к слишком продолжительному циклу разработки, сложности в модернизации и управлении, сокращенному жизненному циклу и повышенной стоимости.

Основное различие между традиционными структурными методологиями проектирования и более новыми объектно-ориентированными методологиями находится в их первичном фокусировании: Структурные методы проектирования фокусируются на функциях системы: "**Что она делает**". Объектно-ориентированные методы фокусируются на данных (объектах) системы: "**Что делается с...**".

ЛЕКЦИЯ 4

СТАНДАРТИЗАЦИЯ В ОБЛАСТИ ПРОЕКТИРОВАНИЯ ПРОГРАММНЫХ ПРОДУКТОВ

Основу отечественной нормативной базы в области документирования программных систем (ПС) составляет комплекс стандартов Единой системы программной документации (ЕСПД). Основная и большая часть комплекса ЕСПД была разработана в 70-е и 80-е годы. Сейчас этот комплекс представляет собой систему межгосударственных стандартов стран СНГ (ГОСТ), действующих на территории Российской Федерации на основе межгосударственного соглашения по стандартизации.

Стандарты ЕСПД в основном охватывают ту часть документации, которая создается в процессе разработки ПС, и связаны, по большей части, с документированием функциональных характеристик ПС. Следует отметить, что стандарты ЕСПД (ГОСТ 19) носят рекомендательный характер. Впрочем, это относится и ко всем другим стандартам в области ПС (ГОСТ 34, Международному стандарту ISO/IEC, и др.). Дело в том, что в соответствии с Законом РФ "О стандартизации" эти стандарты становятся обязательными на контрактной основе – то есть при ссылке на них в договоре на разработку (поставку) ПС.

Говоря о состоянии ЕСПД в целом, можно констатировать, что большая часть стандартов ЕСПД морально устарела.

К числу основных недостатков **ЕСПД** можно отнести:

- ориентацию на единственную, "каскадную" модель жизненного цикла (ЖЦ) ПС;
- отсутствие четких рекомендаций по документированию характеристик качества ПС;

- отсутствие системной увязки с другими действующими отечественными системами стандартов по ЖЦ и документированию продукции в целом, например, ЕСКД;
- нечетко выраженный подход к документированию ПС как товарной продукции;
- отсутствие рекомендаций по самодокументированию ПС, например, в виде экранных меню и средств оперативной помощи пользователю ("хелпов");
- отсутствие рекомендаций по составу, содержанию и оформлению перспективных документов на ПС, согласованных с рекомендациями международных и региональных стандартов.

Тем не менее, до пересмотра всего комплекса, многие стандарты ЕСПД могут с пользой применяться в практике документирования ПС. Эта позиция основана на следующем:

- стандарты ЕСПД вносят элемент упорядочения в процесс документирования ПС;
- предусмотренный стандартами ЕСПД состав программных документов вовсе не такой "жесткий", как некоторым кажется: стандарты позволяют вносить в комплект документации на ПС дополнительные виды;
- стандарты ЕСПД позволяют вдобавок мобильно изменять структуры и содержание установленных видов ПД исходя из требований заказчика и пользователя.

При этом стиль применения стандартов может соответствовать современному общему стилю адаптации стандартов к специфике проекта: заказчик и руководитель проекта выбирают уместное в проекте подмножество стандартов и ПД, дополняют выбранные ПД нужными разделами и исключают ненужные, привязывают создание этих документов к той схеме ЖЦ, которая используется в проекте.

Стандарты ЕСПД (как и другие ГОСТы) подразделяют на группы, приведенные в таблице:

Код группы	Наименование группы
0	Общие положения
1	Основополагающие стандарты
2	Правила выполнения документации разработки
3	Правила выполнения документации изготовления
4	Правила выполнения документации сопровождения
5	Правила выполнения эксплуатационной документации
6	Правила обращения программной документации
7	Резервные группы
8	
9	Прочие стандарты

Обозначение стандарта ЕСПД строят по классификационному признаку:

Обозначение стандарта ЕСПД должно состоять из:

- числа 19 (присвоенных классу стандартов ЕСПД);
- одной цифры (после точки), обозначающей код классификационной группы стандартов, указанной таблице;
- двузначного числа (после тире), указывающего год регистрации стандарта.

Перечень документов ЕСПД

1. ГОСТ 19.001-77 ЕСПД. Общие положения.
2. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
3. ГОСТ 19.102-77 ЕСПД. Стадии разработки.
4. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
5. ГОСТ 19.104-78 ЕСПД. Основные надписи.

6. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.
7. ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.
8. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.
9. ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению.
10. ГОСТ 19.301-79 ЕСПД. Порядок и методика испытаний.
11. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.
12. ГОСТ 19.402-78 ЕСПД. Описание программы.
13. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
14. ГОСТ 19.501-78 ЕСПД. Формуляр. Требования к содержанию и оформлению.
15. ГОСТ 19.502-78 ЕСПД. Описание применения. Требования к содержанию и оформлению.
16. ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.
17. ГОСТ 19.504-79 ЕСПД. Руководство программиста.
18. ГОСТ 19.505-79 ЕСПД. Руководство оператора.
19. ГОСТ 19.506-79 ЕСПД. Описание языка.
20. ГОСТ 19.508-79 ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.
21. ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполняемые печатным способом.
22. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

23. ГОСТ 19.781-90. Обеспечение систем обработки информации программное.

Из всех стандартов ЕСПД остановимся только на тех, которые могут чаще использоваться на практике.

1. **ГОСТ (СТ СЭВ) 19.201-78 (1626-79). ЕСПД.** Техническое задание. Требование к содержанию и оформлению. (Переиздан в ноябре 1987г с изм.1).

Техническое задание (ТЗ) содержит совокупность требований к ПС и может использоваться как критерий проверки и приемки разработанной программы. Поэтому достаточно полно составленное (с учетом возможности внесения дополнительных разделов) и принятое заказчиком и разработчиком, ТЗ является одним из основополагающих документов проекта ПС.

Техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;
- в техническое задание допускается включать приложения.

В зависимости от особенностей программного продукта допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

2. **ГОСТ (СТ СЭВ) 19.101-77 (1626-79).** ЕСПД. Виды программ и программных документов (Переиздан в ноябре 1987г с изм.). Устанавливает виды программ и программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

Виды программ

Вид программы	Определение
Компонент	Программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса
Комплекс	Программа, состоящая из двух или более компонентов и (или) комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса

Виды программных документов

Вид программного документа	Содержание программного документа
Спецификация	Состав программы и документации на нее
Ведомость держателей подлинников	Перечень предприятий, на которых хранят подлинники программных документов
Текст программы	Запись программы с необходимыми комментариями
Описание программы	Сведения о логической структуре и функционировании программы
Программа и методика испытаний	Требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля
Техническое задание	Назначение и область применения программы, технические, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний
Пояснительная записка	Схема алгоритма, общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений
Эксплуатационные документы	Сведения для обеспечения функционирования и эксплуатации программы

Виды эксплуатационных документов

Вид эксплуатационного	Содержание эксплуатационного документа

документа	
Ведомость эксплуатационных документов	Перечень эксплуатационных документов на программу
Формуляр	Основные характеристики программы, комплектность и сведения об эксплуатации программы
Описание применения	Сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств
Руководство системного программиста	Сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения
Руководство программиста	Сведения для эксплуатации программы
Руководство оператора	Сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программы
Описание языка	Описание синтаксиса и семантики языка
Руководство по техническому обслуживанию	Сведения для применения тестовых и диагностических программ при обслуживании технических средств

Допускается объединять отдельные виды эксплуатационных документов (за исключением ведомости эксплуатационных документов и формуляра). Необходимость объединения этих документов указывается в техническом задании. Объединенному документу присваивают наименование и обозначение одного из объединяемых документов. В объединенных документах должны быть приведены сведения, которые необходимо включать в каждый объединяемый документ.

3. **ГОСТ 19.102-77. ЕСПД. Стадии разработки.** Устанавливает стадии разработки программ и программной документации для вычислительных машин, комплексов и систем независимо от их назначения и области применения

Стадии разработки, этапы и содержание работ

Стадии	Этапы работ	Содержание работ

разработки		
Техническое задание	Обоснование необходимости разработки программы	Постановка задачи. Сбор исходных материалов. Выбор и обоснование критериев эффективности и качества разрабатываемой программы. Обоснование необходимости проведения научно-исследовательских работ.
	Научно-исследовательские работы	Определение структуры входных и выходных данных. Предварительный выбор методов решения задач. Обоснование целесообразности применения ранее разработанных программ. Определение требований к техническим средствам. Обоснование принципиальной возможности решения поставленной задачи.
	Разработка и утверждение технического задания	Определение требований к программе. Разработка технико-экономического обоснования разработки программы. Определение стадий, этапов и сроков разработки программы и документации на нее. Выбор языков программирования. Определение необходимости проведения научно-исследовательских работ на последующих стадиях. Согласование и утверждение технического задания.
Эскизный проект	Разработка эскизного проекта	Предварительная разработка структуры входных и выходных данных. Уточнение методов решения задачи. Разработка общего описания алгоритма решения задачи. Разработка технико-экономического обоснования.

	Утверждение эскизного проекта	Разработка пояснительной записки. Согласование и утверждение эскизного проекта
Технический проект	Разработка технического проекта	Уточнение структуры входных и выходных данных. Разработка алгоритма решения задачи. Определение формы представления входных и выходных данных. Определение семантики и синтаксиса языка. Разработка структуры программы. Окончательное определение конфигурации технических средств.
	Утверждение технического проекта	Разработка плана мероприятий по разработке и внедрению программ. Разработка пояснительной записки. Согласование и утверждение технического проекта.
Рабочий проект	Разработка программы	Программирование и отладка программы
	Разработка программной документации	Разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.
	Испытания программы	Разработка, согласование и утверждение программы и методики испытаний. Проведение предварительных государственных, межведомственных, приемо-сдаточных и других видов испытаний. Корректировка программы и программной документации по результатам испытаний.
Внедрение	Подготовка и передача программы	Подготовка и передача программы и программной документации для сопровождения и (или) изготовления. Оформление и утверждение акта о передаче программы на сопровождение и (или) изготовление. Передача программы в фонд

		алгоритмов и программ.

Примечания:

1. Допускается исключать вторую стадию разработки, а в технически обоснованных случаях – вторую и третью стадии. Необходимость проведения этих стадий указывается в техническом задании.
2. Допускается объединять, исключать этапы работ и (или) их содержание, а также вводить другие этапы работ по согласованию с заказчиком.

4. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.

Код страны-разработчика и код организации-разработчика присваивают в установленном порядке.

- Регистрационный номер присваивается в порядке возрастания, начиная с 00001 до 99999, для каждой организации-разработчика.
- Номер издания программы или номер редакции, номер документа данного вида, номер части документа присваиваются в порядке возрастания с 01 до 99. (Если документ состоит из одной части, то дефис и порядковый номер части не указывают).
- Номер редакции спецификации и ведомости эксплуатационных документов на программу должны совпадать с номером издания этой же программы.

5. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам

Настоящий стандарт устанавливает общие требования к оформлению программных документов для вычислительных машин, комплексов и систем, независимо от их назначения и области применения и предусмотренных стандартами Единой системы программной документации (ЕСПД) для любого способа выполнения документов на различных носителях данных.

Программный документ может быть представлен на различных типах носителей данных и состоит из следующих условных частей: титульной, информационной, основной.

Правила оформления документа и его частей на каждом носителе данных устанавливаются стандартами ЕСПД на правила оформления документов на соответствующих носителях данных.

5.ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом

Программные документы оформляют:

- на листах формата А4 (ГОСТ 2.301-68) при изготовлении документа машинописным или рукописным способом;
- допускается оформление на листах формата А3;
- при машинном способе выполнения документа допускаются отклонения размеров листов, соответствующих форматам А4 и А3, определяемые возможностями применяемых технических средств; на листах форматов А4 и А3, предусматриваемых выходными характеристиками устройств вывода данных, при изготовлении документа машинным способом;
- на листах типографических форматов при изготовлении документа типографским способом.

Расположение материалов программного документа осуществляется в следующей последовательности:

титульная часть:

- лист утверждения (не входит в общее количество листов документа);
- титульный лист (первый лист документа);

информационная часть:

- аннотация;
- лист содержания;

основная часть:

- текст документа (с рисунками, таблицами и т.п.)

- перечень терминов и их определений;
- перечень сокращений;
- приложения;
- предметный указатель;
- перечень ссылочных документов;

часть регистрации изменений:

- лист регистрации изменений.

Перечень терминов и их определений, перечень сокращений, приложения, предметных указатель, перечень ссылочных документов выполняются при необходимости.

6. ОСТ 19.402-78 ЕСПД. Описание программы

Следующий стандарт ориентирован на документирование результирующего продукта разработки. Состав документа "Описание программы" в своей содержательной части может дополняться разделами и пунктами, почерпнутыми из стандартов для других описательных документов и руководств: ГОСТ 19.404-79 ЕСПД. Пояснительная записка, ГОСТ 19.502-78 ЕСПД. Описание применения, ГОСТ 19.503-79 ЕСПД. Руководство системного программиста, ГОСТ 19.504-79 ЕСПД. Руководство программиста, ГОСТ 19.505-79 ЕСПД. Руководство оператора.

Есть также группа стандартов, определяющая требования к фиксации всего набора программ и ПД, которые оформляются для передачи ПС. Они порождают лаконичные документы учетного характера и могут быть полезны для упорядочения всего хозяйства программ и ПД (ведь очень часто требуется просто навести элементарный порядок!). Есть и стандарты, определяющие правила ведения документов в "хозяйстве" ПС.

7. **ГОСТ 19.301-79 ЕСПД.** Программа и методика испытаний, который (в адаптированном виде) может использоваться для разработки документов планирования и проведения испытательных работ по оценке готовности и качества ПС.

8. **ГОСТ 19.701-90 ЕСПД**. Схемы алгоритмов, программ, данных и систем. Обозначения условные графические и правила выполнения.

Он устанавливает правила выполнения схем, используемых для отображения различных видов задач обработки данных и средств их решения, и полностью соответствует стандарту ИСО 5807:1985.

Наряду с ЕСПД на межгосударственном уровне действуют еще два стандарта, также относящихся к документированию ПС и принятых не так давно, как большая часть ГОСТ ЕСПД.

ГОСТ 19781-90 Обеспечение систем обработки информации программное. Термины и определения. Разработан взамен ГОСТ 19.781-83 и ГОСТ 19.004-80 и устанавливает термины и определения понятий в области программного обеспечения (ПО) систем обработки данных (СОД), применяемые во всех видах документации и литературы, входящих в сферу работ по стандартизации или использующих результаты этих работ.

ГОСТ 28388-89 Системы обработки информации. Документы на магнитных носителях данных. Порядок выполнения и обращения. Распространяется не только на программные, но и на конструкторские, технологические и другие проектные документы, выполняемые на магнитных носителях.

Стандарты комплекса ГОСТ 34

ГОСТ 34 задумывался в конце 80-х годов как всеобъемлющий комплекс взаимоувязанных межотраслевых документов. Главный мотив: разрешить проблему "вавилонской башни". В 80-х годах сложилось положение, при котором в различных отраслях и областях деятельности использовалась плохо согласованная или несогласованная НТД – "нормативно-техническая документация". Это затрудняло интеграцию систем, обеспечение их эффективного совместного функционирования. Действовали различные комплексы и системы стандартов, устанавливающие требования к различным видам АС.

Практика применения стандартов показала, что в них применяется, по существу (но не по строгим определениям), единая система понятий, есть много общих объектов стандартизации, однако требования стандартов не согласованы между собой, имеются различия по составу и содержанию работ, различия по обозначению, составу, содержанию и оформлению документов и пр.

Разработчики комплекса стандартов 34 выбрали способ, близкий к первому из указанных выше, то есть пошли по пути, более близкому к схемам конкретных методик, чем к стандартам типа ISO12207. Тем не менее, благодаря общности понятийной базы стандарты остаются применимыми в весьма широком диапазоне случаев.

Объектами стандартизации для комплекса ГОСТ 34 являются АС различных (любых!) видов и все виды их компонентов, а не только ПО и БД. Комплекс рассчитан на взаимодействие заказчика и разработчика. Аналогично ISO12207 предусмотрено, что заказчик может разрабатывать АС для себя сам (если создаст для этого специализированное подразделение). Однако формулировки ГОСТ 34 не ориентированы на столь явное и, в известном смысле, симметричное отражение действий обеих сторон, как ISO12207. Поскольку ГОСТ 34 в основном уделяет внимание содержанию проектных документов, распределение действий между сторонами обычно делается, отталкиваясь от этого содержания.

Из всех существующих и нереализованных групп документов будем основываться только на Группе 0 "Общие положения" и Группе 6 "Создание, функционирование и развитие АС". Наиболее популярными можно считать стандарты ГОСТ 34.601-90 (Стадии создания АС), ГОСТ 34.602-89 (ТЗ на создание АС) и методические указания РД 50-34.698-90 (Требования к содержанию документов). Стандарты предусматривают стадии и этапы выполнения работ по созданию АС, но не предусматривают сквозных процессов в явном виде.

Для общего случая разработки АС стадии и этапы ГОСТ 34 приведены в таблице:

1. ФТ - Формирование требований к АС.	1.1. Обследование объекта и обоснование необходимости создания АС; 1.2. Формирование требований пользователя к АС; 1.3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания);
2. РК - Разработка концепции АС.	2.1. Изучение объекта; 2.2. Проведение необходимых научно-исследовательских работ; 2.3. Разработка вариантов концепции АС, удовлетворяющей требованиям пользователя 2.4. Оформление отчета о выполненной работе
3. ТЗ - Техническое создание АС.	3.1. Разработка и утверждение технического задания на задание.
4. ЭП - Эскизный проект.	4.1. Разработка предварительных проектных решений по системе и ее частям; 4.2. Разработка документации на АС и ее части.
5. ТП - Технический проект.	5.1. Разработка проектных решений по системе и ее частям; 5.2. Разработка документации на АС и ее части; 5.3. Разработка и оформление документации на поставку изделий для комплектования АС и/или технических требований (технических заданий) на их разработку; 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации.
6. РД - Рабочая документация.	6.1. Разработка рабочей документации на систему и ее части; 6.2. Разработка или адаптация программ.
7. ВД - Ввод в действие.	7.1. Подготовка объекта автоматизации к вводу АС в действие; 7.2. Подготовка персонала; 7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями); 7.4. Строительно-монтажные работы; 7.5. Пуско-наладочные работы; 7.6. Проведение предварительных испытаний; 7.7. Проведение опытной эксплуатации;

	7.8. Проведение приемочных испытаний.
8. Сп - Сопровождение АС.	8.1. Выполнение работ в соответствии с гарантийными обязательствами; 8.2. Послегарантийное обслуживание.

Описано содержание документов, разрабатываемых на каждом этапе. Это определяет потенциальные возможности выделения на содержательном уровне сквозных работ, выполняемых параллельно или последовательно (то есть по сути – процессов), и составляющих их задач. Такой прием может использоваться при построении профиля стандартов ЖЦ проекта, включающего согласованные подмножества стандартов ГОСТ 34 и ISO12207.

Степень адаптивности формально определяется возможностями:

- опускать стадию эскизного проектирования и объединять стадии "Технический проект" и "Рабочая документация";
- опускать этапы, объединять и опускать большинство документов и их разделов;
- вводить дополнительные документы, разделы документов и работы;
- динамически создавая т. н. ЧТЗ – частные технические задания – достаточно гибко формировать ЖЦ АС; как правило, этот прием используется на уровне крупных единиц (подсистем, комплексов), ради которых считается оправданным создавать ЧТЗ, однако нет никаких существенных оснований сильно ограничивать этот способ управления ЖЦ.

Стадии и этапы, выполняемые организациями – участниками работ по созданию АС, устанавливаются в договорах и техническом задании, что близко к подходу ISO.

Введение единой, достаточно качественно определенной терминологии, наличие достаточно разумной классификации работ, документов, видов обеспечения и др., безусловно, полезно. ГОСТ 34 способствует более полной и качественной стыковке действительно разных

систем, что особенно важно в условиях, когда разрабатывается все больше сложных комплексных АС, например, типа CAD-CAM, которые включают в свой состав АСУТП, АСУП, САПР-конструктора, САПР-технолога, АСНИ и др. системы.

Определено несколько важных положений, отражающих особенности АС как объекта стандартизации, например: "в общем случае АС состоит из программно-технических (ПТК), программно-методических (ПМК) комплексов и отдельных компонентов организационного, технического, программного и информационного обеспечения".

Разделение понятий ПТК и АС закрепляло принцип, по которому АС есть не "ИС с БД", но:

- "организационно-техническая система, обеспечивающая выработку решений на основе автоматизации информационных процессов в различных сферах деятельности (управление, проектирование, производство и т. д.) или их сочетаниях" (по РД 50-680-88), что особенно актуально в аспектах бизнес-реинжиниринга;
- "система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций" (по ГОСТ 34.003-90).

Эти определения указывают на то, что АС – это в первую очередь персонал, принимающий решения и выполняющий другие управляющие действия, поддержанный организационно-техническими средствами.

Степень обязательности:

прежняя полная обязательность отсутствует, материалы ГОСТ34, по сути, стали методической поддержкой, причем чаще для заказчиков, имеющих в стандарте набор требований к содержанию ТЗ и проведению испытаний АС. При этом польза ГОСТ34 может многократно возрасти в случае их более гибкого использования при формировании профиля ЖЦ АС.

Ключевым документом взаимодействия сторон является ТЗ – техническое задание на создание АС. ТЗ является основным исходным

документом для создания АС и его приемки, ТЗ определяет важнейшие точки взаимодействия заказчика и разработчика. При этом ТЗ разрабатывает организация-разработчик (по ГОСТ 34.602-89), но формально выдает ТЗ разработчику заказчик (по РД 50-680-88).

Государственные стандарты РФ (ГОСТ Р)

В РФ действует ряд стандартов в части документирования ПС, разработанных на основе прямого применения международных стандартов ИСО. Это самые "свежие" по времени принятия стандарты. Некоторые из них впрямую адресованы руководителям проекта или директорам информационных служб. Вместе с тем они неоправданно мало известны в среде профессионалов. Вот их представление.

ГОСТ Р ИСО/МЭК 9294-93 Информационная технология. Руководство по управлению документированием программного обеспечения. Стандарт полностью соответствует международному стандарту ИСО/МЭК ТО 9294:1990 и устанавливает рекомендации по эффективному управлению документированием ПС для руководителей, отвечающих за их создание. Целью стандарта является оказание помощи в определении стратегии документирования ПС; выборе стандартов по документированию; выборе процедур документирования; определении необходимых ресурсов; составлении планов документирования.

ГОСТ Р ИСО/МЭК 9126-93 Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению. Стандарт полностью соответствует международному стандарту ИСО/МЭК 9126:1991. В его контексте под характеристикой качества понимается "набор свойств (атрибутов) программной продукции, по которым ее качество описывается и оценивается". Стандарт определяет шесть комплексных характеристик, которые с минимальным дублированием описывают качество ПС (ПО, программной продукции): функциональные возможности; надежность; практичность; эффективность; сопровождаемость;

мобильность. Эти характеристики образуют основу для дальнейшего уточнения и описания качества ПС.

ГОСТ Р ИСО 9127-94 Системы обработки информации. Документация пользователя и информация на упаковке для потребительских программных пакетов. Стандарт полностью соответствует международному стандарту ИСО 9127:1989. В контексте настоящего стандарта под потребительским программным пакетом (ПП) понимается "программная продукция, спроектированная и продаваемая для выполнения определенных функций; программа и соответствующая ей документация, упакованные для продажи как единое целое". Под документацией пользователя понимается документация, которая обеспечивает конечного пользователя информацией по установке и эксплуатации ПП. Под информацией на упаковке понимают информацию, воспроизводимую на внешней упаковке ПП. Ее целью является предоставление потенциальным покупателям первичных сведений о ПП.

ГОСТ Р ИСО/МЭК 8631-94 Информационная технология. Программные конструктивы и условные обозначения для их представления. Описывает представление процедурных алгоритмов.

Международные стандарты

ISO/IEC 12207: 1995-08-01

Первая редакция ISO 12207 подготовлена в 1995 году объединенным техническим комитетом ISO/IEC JTC1 "Информационные технологии, подкомитет SC7, проектирование программного обеспечения".

По определению, ISO 12207 – базовый стандарт процессов ЖЦ ПО, ориентированный на различные (любые!) виды ПО и типы проектов АС, куда ПО входит как часть. Стандарт определяет стратегию и общий порядок в создании и эксплуатации ПО, он охватывает ЖЦ ПО от концептуализации идей до завершения ЖЦ.

Очень важные замечания стандарта:

1. Процессы, используемые во время ЖЦ ПО, должны быть совместимы с процессами, используемыми во время ЖЦ АС. (Отсюда понятна целесообразность совместного использования стандартов на АС и на ПО.)
2. Добавление уникальных или специфических процессов, действий и задач должно быть оговорено в контракте между сторонами. Контракт понимается в широком смысле: от юридически оформленного контракта до неформального соглашения, соглашение может быть определено и единственной стороной как задача, поставленная самому себе.
3. Стандарт принципиально не содержит конкретные методы действий, тем более – заготовки решений или документации. Он описывает архитектуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить услуги и задачи, включенные в процессы, не предназначен для предписывания имени, формата или точного содержимого получаемой документации. Решения такого типа принимаются использующим стандарт.

Определения стандарта:

1. **Система** – это объединение одного или более процессов, аппаратных средств, программного обеспечения, оборудования и людей для обеспечения возможности удовлетворения определенных потребностей или целей.
2. **Модель жизненного цикла** – структура, содержащая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение всей жизни системы, от определения требований до завершения ее использования.

   Множество процессов и задач сконструировано так, что возможна их адаптация в соответствии с проектами ПО. Процесс адаптации является процессом исключения процессов, видов деятельности и задач, не

применимых в конкретном проекте. Степень адаптивности: максимальная.

3. **Требование квалификации** – набор критериев или условий (квалификационные требования), которые должны быть удовлетворены для того, чтобы квалифицировать программный продукт как подчиняющийся (удовлетворяющий условиям) его спецификациям и готовый для использования в целевой окружающей среде.

Стандарт не предписывает конкретную модель ЖЦ или метод разработки ПО, но определяет, что стороны-участники использования стандарта ответственны за выбор модели ЖЦ для проекта ПО, за адаптацию процессов и задач стандарта к этой модели, за выбор и применение методов разработки ПО, за выполнение действий и задач, подходящих для проекта ПО.

Стандарт ISO 12207 равносильно ориентирован на организацию действий каждой из двух сторон: поставщик (разработчик) и покупатель (пользователь); может быть в равной степени применен, когда обе стороны – из одной организации.

Каждый процесс ЖЦ разделен на набор действий, каждое действие – на набор задач. Очень важное отличие ISO: каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем нет заранее определенных последовательностей (естественно, при сохранении логики связей по исходным сведениям задач и т. п.).

В стандарте ISO 12207 описаны:

1. ***5 основных процессов ЖЦ ПО:***
    - Процесс приобретения. Определяет действия предприятия-покупателя, которое приобретает АС, программный продукт или сервис ПО.

- Процесс поставки. Определяет действия предприятия-поставщика, которое снабжает покупателя системой, программным продуктом или сервисом ПО.
- Процесс разработки. Определяет действия предприятия-разработчика, которое разрабатывает принцип построения программного изделия и программный продукт.
- Процесс функционирования. Определяет действия предприятия-оператора, которое обеспечивает обслуживание системы (а не только ПО) в процессе ее функционирования в интересах пользователей. В отличие от действий, которые определяются разработчиком в инструкциях по эксплуатации (эта деятельность разработчика предусмотрена во всех трех рассматриваемых стандартах), определяются действия оператора по консультированию пользователей, получению обратной связи и др., которые он планирует сам и берет на себя соответствующе обязанности.
- Процесс сопровождения. Определяет действия персонала сопровождения, который обеспечивает сопровождение программного продукта, что представляет собой управление модификациями программного продукта, поддержку его текущего состояния и функциональной пригодности, включает в себя инсталляцию и удаление программного изделия на вычислительной системе.

2. ***8 вспомогательных процессов***, которые поддерживают реализацию другого процесса, будучи неотъемлемой частью всего ЖЦ программного изделия, и обеспечивают должное качество проекта ПО:
   - решения проблем;
   - документирования;
   - управления конфигурацией;

- гарантирования качества, используются результаты остальных процессов группы обеспечения качества, в которую входят:
    - Процесс верификации;
    - Процесс аттестации;
    - Процесс совместной оценки;
    - Процесс аудита.

3. ***4 организационных процесса:***
    - Процесс управления;
    - Процесс создания инфраструктуры;
    - Процесс усовершенствования;
    - Процесс обучения.

К ним примыкает особый Процесс адаптации, который определяет основные действия, необходимые для адаптации стандарта к условиям конкретного проекта.

Под процессом усовершенствования здесь понимается не усовершенствование АС или ПО, а улучшение самих процессов приобретения, разработки, гарантирования качества и т.п., реально осуществляемых в организации.

Каких-либо этапов, фаз, стадий не предусмотрено, что дает описываемую ниже степень адаптивности.

"Динамический" характер стандарта определяется способом определения последовательности выполнения процессов и задач, при котором один процесс при необходимости вызывает другой или его часть.

Примеры:

- выполнение Процесса приобретения в части анализа и фиксации требований к системе или ПО может вызывать исполнение соответствующих задач Процесса разработки;

- в Процессе поставки поставщик должен управлять субподрядчиками согласно Процессу приобретения и выполнять верификацию и аттестацию по соответствующим процессам;
- сопровождение может требовать развития системы и ПО, что выполняется по Процессу разработки.

Такой характер позволяет реализовывать любую модель ЖЦ.

При выполнении анализа требований к ПО предусмотрено 11 классов характеристик качества, которые используются позже при гарантировании качества.

При этом разработчик должен установить и документировать как требования к программному обеспечению:

1. Функциональные и возможные спецификации, включая исполнение, физические характеристики и условия среды эксплуатации, при которых единица программного обеспечения должна быть выполнена;
2. Внешние связи (интерфейсы) с единицей программного обеспечения;
3. Требования квалификации;
4. Спецификации надежности, включая спецификации, связанные с методами функционирования и сопровождения, воздействия окружающей среды и вероятностью травмы персонала;
5. Спецификации защищенности,
6. Человеческие факторы спецификаций по инженерной психологии (эргономике), включая связанные с ручным управлением, взаимодействием человека и оборудования, ограничениями на персонал и областями, нуждающимися в концентрированном человеческом внимании, которые являются чувствительными к ошибкам человека и обучению;
7. Определение данных и требований базы данных;
8. Установочные и приемочные требования поставляемого программного продукта в местах функционирования и сопровождения (эксплуатации);

9. Документация пользователя;
10. Работа пользователя и требования выполнения;
11. Требования сервиса пользователя.

Интересно и важно, что эти и аналогичные характеристики хорошо корреспондируются с характеристиками АС, предусматриваемыми в ГОСТ 34 по видам обеспечения системы.

Стандарт содержит предельно мало описаний, направленных на проектирование БД. Это можно считать оправданным, так как разные системы и разные прикладные комплексы ПО могут не только использовать весьма специфические типы БД, но и не использовать.

Итак, ISO12207 имеет набор процессов, действий и задач, охватывающий наиболее широкий спектр возможных ситуаций при максимальной адаптируемости.

Он показывает пример того, как должен строиться хорошо организованный стандарт, содержащий минимум ограничений (принцип "нет одинаковых проектов"). При этом детальные определения процессов, форм документов и т.п. целесообразно выносить в различные функциональные стандарты, ведомственные нормативные документы или фирменные методики, которые могут быть использованы или не использованы в конкретном проекте.

По этой причине центральным стандартом, положения которого берутся за начальный "стержневой" набор положений в процессе построения профиля стандартов ЖЦ для конкретного проекта, полезно рассматривать именно ISO 12207. Этот "стержень" может задавать модель ЖЦ ПО и АС, принципиальную схему гарантирования качества, модель управления проектом

ISO/IEC 15288 Systems engineering. System life cycle processes
(Системная инженерия. Процессы жизненного цикла систем)

Принят в качестве российского стандарта ГОСТ Р ИСО/МЭК 15288-2005 – Информационная технология. Системная инженерия. Процессы

жизненного цикла систем. Стандарт применим для любого рода систем класса систем, но его основное предназначение – поддержка создания компьютеризированных систем.

В стандарте ISO/IEC 15288 предусмотрены следующие стадии создания систем:

№ п/п	Стадия	Описание
	Формирование концепции	Анализ потребностей, выбор концепции и проектных решений
	Разработка	Проектирование системы
	Реализация	Изготовление системы
	Эксплуатация	Ввод в эксплуатацию и использование системы
	Поддержка	Обеспечение функционирования системы
	Снятие с эксплуатации	Прекращение использования, демонтаж, архивирование системы

Согласно стандарту ISO/IEC серии 15288 в структуру ЖЦ следует включать следующие группы процессов:

1. Договорные процессы:

- приобретение (внутреннее или у внешнего поставщика решения);
- поставка (внутренняя или у внешнего поставщика решения).

2. Процессы предприятия:

- управление средой предприятия;
- управление инвестициями;
- управление жизненным циклом систем;
- управление ресурсами;
- управление качеством.

3. Проектные процессы:

- планирование проекта;
- оценка проекта;
- контроль проекта;
- управление рисками;

- управление конфигурацией;
- управление информационными потоками;
- принятие решений.

4. Технические процессы:

- определение требований заказчика;
- анализ требований;
- разработка архитектуры;
- реализация;
- внедрение;
- интеграция;
- верификация;
- поставки;
- аттестация;
- эксплуатация;
- сопровождение;
- выведения из эксплуатации.

В стандарте описываются процессы, составляющие жизненный цикл любой искусственной системы, создаваемой людьми. Стандарт применим для систем единичного и массового производства и систем, адаптируемых по требованиям заказчика. Стандарт может использоваться организациями, выступающими в роли как поставщиков, так и приобретающих сторон. Он может применяться одной из сторон в индивидуальном порядке или в порядке, согласованном несколькими участниками.

Rational Unified Process (RUP – Унифицированный процесс Rational)

Предлагает итеративную модель разработки, включающую четыре фазы:

– начало,

– исследование,

– построение,

– внедрение.

Каждая фаза может быть разбита на этапы (итерации), в результате которых выпускается версия для внутреннего или внешнего использования. Прохождение через четыре основные фазы называется циклом разработки, каждый цикл завершается генерацией версии системы. Если после этого работа над проектом не прекращается, то полученный продукт продолжает развиваться и снова минует те же фазы. Суть работы в рамках RUP – это создание и сопровождение моделей на базе UML.

Методика Oracle CDM

(Custom Development Method) – технологический материал, рассчитанный на использование в проектах с применением продуктов фирмы Oracle.

Основу CASE-технологии и инструментальной среды фирмы Oracle составляют:

– методология структурного нисходящего проектирования, при которой разработка прикладной системы представляется в виде последовательности четко определенных этапов;

– поддержка всех этапов жизненного цикла прикладной системы, начиная с самых общих описаний предметной области до получения и сопровождения готового программного продукта;

– ориентация на реализацию приложений в архитектуре клиент-сервер с использованием всех особенностей современных серверов баз данных;

– наличие централизованной базы данных, репозитория, для хранения спецификаций проекта прикладной системы на всех этапах ее разработки. Такой репозиторий представляет собой базу данных специальной структуры, работающую под управлением СУБД Oracle;

– возможность одновременной работы с репозиторием многих пользователей;

– автоматизация последовательного перехода от одного этапа разработки к следующему. Для этого предусмотрены специальные утилиты, с помощью которых можно по спецификациям концептуального уровня

(модели предметной области) автоматически получать первоначальный вариант спецификации уровня проектирования (описание структуры базы данных и состава программных модулей), чтобы на его основе после всех необходимых уточнений и дополнений автоматически генерировать готовые к выполнению программы;

– автоматизация различных стандартных действий по проектированию и реализации приложения: предусматривается генерация многочисленных отчетов по содержимому репозитория, обеспечивающих полное документирование текущей версии системы на всех этапах ее разработки; с помощью специальных процедур предоставляется возможность проверки спецификаций на полноту и непротиворечивость.

Методика Oracle CDM определяет следующие *фазы жизненного цикла* информационной системы:

– стратегия – необязательный этап, связанный с анализом и моделированием бизнес-процессов организации;

– анализ (формулирование детальных требований к прикладной системе);

– проектирование (преобразование требований в детальные спецификации системы);

– реализация (написание и тестирование приложений);

– внедрение (установка новой прикладной системы, подготовка к началу эксплуатации);

– эксплуатация (поддержка приложения и слежение за ним, планирование будущих функциональных расширений).

Методика Oracle CDM выделяет следующие *процессы*, протекающие на протяжении жизненного цикла информационной системы:

– определение производственных требований;

– исследование существующих систем;

– определение технической архитектуры;

– проектирование и построение базы данных;

– проектирование и реализация модулей;

– конвертирование данных;

– документирование;

– тестирование;

– обучение;

– переход к новой системе;

– поддержка и сопровождение.

Процессы состоят из последовательностей задач, задачи разных процессов взаимосвязаны с помощью явных ссылок.

Особенностью Oracle CDM является возможность применения трех моделей жизненного цикла:

– *классическая* – предусматривает все этапы;

– *быстрая разработка* – ориентирована на использование инструментов моделирования и программирования Oracle;

облегченный подход – рекомендуется в случае малых проектов и возможности быстро прототипировать приложения.

Практики используют еще один путь: сами переводят и используют в своих проектах современные стандарты на организацию ЖЦ ПС и их документирование. Но этот путь страдает как минимум тем недостатком, что разные переводы и адаптации стандартов, сделанные разными разработчиками и заказчиками, будут отличаться массой деталей. Эти отличия неизбежно касаются не только наименований, но и их содержательных определений, вводимых и используемых в стандартах. Таким образом, на этом пути неизбежно постоянное возникновение путаницы, а это прямо противоположно целям не только стандартов, но и любых грамотных методических документов.

В настоящее время во ВНИИ стандартов подготовлены предложения по совершенствованию и развитию комплекса стандартов по документированию ПС.

Список источников

1. Буч Градди Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд. / Буч Градди, Максимчук Роберт А., Энгл Майкл У., Янг Бобби Дж., Коналлен Джим, Хьюстон Келли А.: Пер с англ. — М.: ООО «И.Д. Вильямс», 2010. — 720 с.
2. Грекул В. И. Проектирование информационных систем / В. И. Грекул, Н. Г. Денищенко, Н. Л. Коровкина. – Интернет-университет информационных технологий – ИНТУИТ.ру, 2008. – 300 с.
3. Вендров А. М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2006. – 544 с.
4. Липаев В. В., Филинов Е. Н. Мобильность программ и данных в открытых информационных системах. М., РФФИ, 1997
5. Лясин Д.Н. Объектно-ориентированный анализ и программирование [Электронный ресурс] Сборник "Учебные пособия". Выпуск 1. [Электронный ресурс] / Д.Н. Лясин, О.Ф. Абрамова. - Волгоград: ВолгГТУ, 2014. - номер гос. регистрации 0321400870- 1 электрон. опт. диск (CD-ROM).
6. Методологии разработки программных продуктов [Электронный ресурс], https://studme.org/263333/informatika/ metodologii_ razrabotki_ programmnyh_produktov
7. Г.С. Иванова, Т.Н. Ничушкина Проектирование программного обеспечения Учебное пособие// Москва, 2002 – 74 стр.
8. А. Кайгородова, Модели и методологии разработки ПО продуктов [Электронный ресурс], https://geekbrains.ru/posts/methodologies

9. Ральф Джонсон, Джон Влиссидес, Ричард Хелм, Эрих Гамма, Приемы объектно-ориентированного проектирования. Паттерны проектирования// Библиотека программиста (Питер), 2015 - 368 стр.

10. Ив Пинье Александр Остервальдер Построение бизнес-моделей. Настольная книга стратега и новатора// Серия «Сколково»// 2013 – 330 стр.

11. Стандарты, регламентирующие жизненный цикл информационных систем [Электронный ресурс], https://studopedia.ru/3_20558_standarti-reglamentiruyushchie-zhiznenniy-tsikl-informatsionnih-sistem.html

12. А. М. Гудов, С.Ю. Завозкин, С. Н. Трофимов Технология разработки программного обеспечения// Учебное пособие// Кемерово, 2009 – 138 стр.

13. А. В. Рудаков Технология разработки программных продуктов. Учебник// Academia, 2013 – 208 стр.

14. Аллен Э. Типичные ошибки проектирования / Э.Аллен: Пер. с англ. – СПб.: Питер, 2003. – 224 стр.

15. Брукс, Ф. Мифический человеко-месяц или как создаются программные системы / Ф. Брукс. — М.: СПб: Символ-Плюс, 2016. — 304 с.

Электронное учебное издание

Оксана Федоровна **Абрамова**

Индустриальная разработка программных продуктов Часть 1

Учебное пособие

Электронное издание сетевого распространения

Редактор Матвеева Н.И.

Темплан 2020 г. Поз. № 4.
Подписано к использованию 28.04.2020. Формат 60x84 1/16.
Гарнитура Times. Усл. печ. л. 5,19.

Волгоградский государственный технический университет.
400005, г. Волгоград, пр. Ленина, 28, корп. 1.

ВПИ (филиал) ВолгГТУ.
404121, г. Волжский, ул. Энгельса, 42а.