Евгений Шуремов

Методологические подходы и средства поддержки процессов разработки программного обеспечения организационно-экономических систем

Коротко о главном

Евгений Шуремов

Методологические подходы и средства поддержки процессов разработки программного обеспечения организационно-экономических систем. Коротко о главном

«Издательские решения»

Шуремов Е. Л.

Методологические подходы и средства поддержки процессов разработки программного обеспечения организационно-экономических систем. Коротко о главном / Е. Л. Шуремов — «Издательские решения»,

ISBN 978-5-44-836096-1

Книга содержит краткое изложение основной проблематики, связанной с процессами промышленной разработки программного обеспечения информационных систем организационно-экономического управления. Рассматриваются основные существующие методологические подходы и методики организации разработки, а также наиболее популярные поддерживающие их инструментальные средства.Изложение ориентировано на минимально подготовленных читателей, желающих получить общее представление по данной теме.

ISBN 978-5-44-836096-1

Содержание

Методологические подходы и средства поддержки процессов разработки программного обеспечения организационно-экономических систем Коротко о главном Евгений Шуремов

ISBN 978-5-4483-6096-1
Создано в интеллектуальной издательской системе Ridero

Введение

Данная публикация содержит краткое изложение основной проблематики, связанной с процессами промышленной разработки программного обеспечения информационных систем организационно-экономического управления. Рассматриваются основные существующие методологические подходы и методики организации разработки, а также наиболее популярные поддерживающие их инструментальные средства.

Изложение ориентировано на минимально подготовленных читателей, желающих получить общее представление по данной теме. В основу книги положены тезисы лекций, на протяжении нескольких лет читавшихся автором для магистрантов Финансового университета при Правительстве РФ.

Книга имеет Интернет-поддержку на сайте http://shurem.ru. На страницах поддержки будут публиковаться уточнения и дополнительные материалы к тексту книги. Кроме того, там будут размещаться средства автоматизированного самотестирования на знание основных положений представленного в книге материала, а также инструкции по его проведению. Однако страницы Интернет-поддержки доступны только авторизованным пользователям сайта http://shurem.ru. «Случайным» посетителям сайта эти страницы недоступны. То есть необходимо зарегистрироваться, авторизоваться на сайте и далее пройти по пути **Публикации => Поддержка книг => Методологические подходы и средства поддержки процессов разработки программного обеспечения организационно-экономических систем**.

С пожеланиями и предложениями можно обратиться к автору по адресу shurem@mail.ru

Роль программного обеспечения в развитии организационно-экономических систем

В организационно-экономических информационных системах (ИС) процедуры информационного процесса интегрированы с функциями управления. Наряду со своими основными функциями, их непосредственно выполняет управленческий персонал. Поэтому в современных условиях ИС следует рассматривать как неотъемлемую часть инфраструктуры организационно-экономической системы, инструмент решения всего комплекса задач управления организацией.

Развитие любой организации так или иначе требует развития ее управленческой инфраструктуры. Отсюда следует, что ИС постоянно нуждаются в развитии. В условиях информационной экономики это направление развития считается приоритетным по отношению к другим направлениям.

Далее (если отдельно не оговорено противное) рассматриваются ИС организационно-экономического управления.

Бурное развитие информационных технологий постоянно расширяет вариативность выбора стратегий развития ИС.

50—70 гг. ИС (АСУ) на базе мэйнфреймов. Создавались в крупных организациях для решения отдельных задач управления, требующих массового ввода и оперативной обработки больших объемов информации. Полностью централизованная обработка данных в вычислительных центрах. Выбор вариантов развития ИС ограничен возможностями приобретения новых вычислительных мощностей на мэйнфреймах и разработкой для них программ для автоматизации решения новых задач.

70—80 гг. ИС на базе мини-ЭВМ. Создавались на средних и даже относительно небольших предприятиях для решения широкого круга задач. Выбор шире: можно создать ИС на основе мэйнфреймов или на базе мини-ЭВМ. При этом выбор технической платформы предопределяет возможность использования программного обеспечения (либо для одной, либо для другой аппаратной платформы).

80 гг. ИС с распределенной обработкой данных на персональных компьютерах. Массовое распространение офисных программ. Начало формирования электронного документооборота. Выбор еще шире: можно по-старинке создать ИС на базе мэнфреймов или мини-ЭВМ; полностью распределенную систему обработки данных на ПК; часть задач решать на мини-ЭВМ, а часть на ПК; для ПК можно выбрать программы разных поставщиков и т. д.

80—90 гг. Создание и массовое распространение ERP-систем. Массовый переход к клиент-серверным технологиям. Выбор еще шире: наряду с возможностями предыдущих этапов, можно выбрать разные варианты организации локальной сети, включая разделение доступа к периферийным устройствам; разные технологии функционирования прикладных программ в сети; разное обеспечивающее и прикладное ПО и т. д.

Середина 90 – конец нулевых гг. Широкое распространение электронного документооборота. Развитие электронной коммерции и коммуникаций с сетями внешних партнёров и клиентов через Интернет (B2B, B2C, SCM, CSRP и др.). Дополнительно к предыдущему: разные способы взаимодействия локальных сетей разных организаций через Интернет; выбор стандартов и форматов обмена данными; дальнейшее расширение возможностей прикладных программ; выбор способов взаимодействия сотрудников через глобальные сети и т. д.

Настоящее время. Широкое распространение мобильных устройств. Рост популярности «облачных» вычислений. Дополнительно к предыдущему: выбор вариантов взаимо-

действия с мобильными пользователями (сотрудниками, партнерами, клиентами); иметь собственные вычислительные мощности или арендовать на стороне; использовать «монолитную» систему обработки данных или набор web-сервисов и т. д.

Различные стратегии развития ИС могут требовать разных затрат, иметь следствием разную ***совокупную стоимость*** владения системой, обеспечивать разный уровень отдачи на инвестиции. Однако какой бы вариант развития ИС ни был выбран, ключевым элементом развития является увеличение функциональности ИС, возможность автоматизации решения более широкого круга задач, чем раньше. А поскольку функциональность ИС обеспечивается, прежде всего, ее программным обеспечением, то при любых условиях развитие программного обеспечения является важнейшим элементом развития любой ИС.

Программное обеспечение как объект развития

Программное обеспечение (ПО) – все или часть программ, процедур, правил и соответствующей документации системы обработки информации (ISO/IEC 2382—1: 1993. Information technology – Vocabulary – Part 1: Fundamental terms).

Другие определения из международных и отечественных стандартов:

Компьютерные программы, процедуры и, возможно, соответствующая документация и данные, относящиеся к функционированию компьютерной системы (FCD ISO/IEC 24765. Systems and Software Engineering Vocabulary).

Совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ (ГОСТ 19781—90).

Программное обеспечение является объектом изучения информатики, программирования и программной инженерии.

Термин **software** впервые применил математик из Принстонского университета Джон Тьюки (англ. John W. Tukey) в статье в American Mathematical Monthly в 1958 году.

Первая теория, касающаяся программного обеспечения, была предложена английским математиком Аланом Тьюрингом в 1935 году в эссе «Computable numbers with an application to the Entscheidungsproblem (Decision problem)». Он создал так называемую машину Тьюринга, математическую модель абстрактной машины, способной выполнять последовательности рудиментарных операций, которые переводят машину из одного фиксированного состояния в другое. Главная идея заключалась в математическом доказательстве факта, что любое наперёд заданное состояние системы может быть всегда достигнуто последовательным выполнением конечного набора элементарных команд (программы) из фиксированного набора команд.

Архитектура программного обеспечения – структура программы или вычислительной системы, которая включает программные компоненты, видимые снаружи свойства этих компонентов, а также отношения между ними.

Качество программного обеспечения – весь объем признаков и характеристик программ, который относится к их способности удовлетворять установленным или предполагаемым потребностям.

Тестирование программного обеспечения – процесс исследования программного обеспечения (ПО) с целью получения информации о качестве продукта.

С точки зрения **ISO 9126**, качество программного обеспечения можно определить как совокупную характеристику исследуемого ПО с учётом следующих составляющих:

– Надёжность;
– Сопровождаемость;
– Практичность;
– Эффективность;
– Мобильность;
– Функциональность.

Лицензия на программное обеспечение – правовой инструмент, определяющий использование и распространение программного обеспечения, защищённого авторским правом.

Обычно лицензия на программное обеспечение разрешает получателю использовать одну или несколько копий программы, причём без лицензии такое использование рассматривалось бы в рамках закона как нарушение авторских прав издателя. По сути, лицензия выступает гарантией того, что издатель ПО, которому принадлежат исключительные права на программу, не подаст в суд на того, кто ею пользуется.

Математические и алгоритмические методы, содержащиеся в программном обеспечении, могут быть запатентованы.

По определению, предложенному Фондом за свободную информационную инфраструктуру, программный патент – это «патент на что-либо, выполняемое компьютером посредством программного обеспечения».

Защитники программных патентов считают, что они позволяют:

– защитить сложное ПО от подражателей, которым не нужно тратить время и деньги на проектные работы;

– защитить изобретателей-одиночек от крупных компаний;

– труднодоступность запатентованных технологий стимулирует создание более совершенных технологий.

Документация – печатные руководства пользователя, диалоговая (оперативная) документация и справочный текст, описывающие как пользоваться программой.

Документация состоит из отдельных документов.

Документ как элемент документации – это целевая информация, предназначенная для конкретной аудитории, размещённая на конкретном носителе в заданном формате.

Программный документ – документ, содержащий в зависимости от назначения данные, необходимые для разработки, производства, эксплуатации и сопровождения программы.

Разработка программного обеспечения (software development) – это процесс, направленный на создание и поддержание работоспособности, качества и надежности программного обеспечения.

Как и традиционные инженерные дисциплины, разработка программного обеспечения имеет дело с проблемами качества, стоимости и надёжности. Некоторые программы содержат миллионы строк исходного кода, которые должны правильно исполняться в изменяющихся условиях. Сложность ПО сравнима со сложностью наиболее совершенных из современных машин, таких как самолёты.

Накопленный к настоящему времени опыт создания программных систем (ПС) показывает, что это сложная и трудоемкая работа, требующая высокой квалификации участвующих в ней специалистов. Однако до настоящего времени создание таких систем нередко выполняется на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ПО. По данным Института программной инженерии (Software Engineering Institute, SEI) в последние годы до 80% всего эксплуатируемого ПО разрабатывалось вообще без использования какой-либо дисциплины проектирования, методом «code and fix» (кодирования и исправления ошибок).

Проблемы создания ПО следуют из его свойств. Еще в 1975 г. Фредерик Брукс, проанализировав свой уникальный по тем временам опыт руководства крупнейшим проектом разработки операционной системы OS/360, определил перечень неотъемлемых свойств ПО: сложность, согласованность, изменяемость и незримость.

Современные крупномасштабные проекты создания и развития ПС характеризуются следующими особенностями.

Характеристики объекта внедрения:

– структурная сложность (многоуровневая иерархическая структура организации) и территориальная распределенность;

– функциональная сложность (многоуровневая иерархия и большое количество функций, выполняемых организацией; сложные взаимосвязи между ними);

– информационная сложность (большое количество источников и потребителей информации (министерства и ведомства, местные органы власти, организации-партнеры),

разнообразные формы и форматы представления информации, сложная информационная модель объекта – большое количество информационных сущностей и сложные взаимосвязи между ними), сложная технология прохождения документов;

– сложная динамика поведения, обусловленная высокой изменчивостью внешней среды (изменения в законодательных и нормативных актах, нестабильность экономики и политики) и внутренней среды (структурные реорганизации, текучесть кадров).

Технические характеристики проектов создания ПО:

– различная степень унифицированности проектных решений в рамках одного проекта;

– высокая техническая сложность, определяемая наличием совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (транзакционных приложений, предъявляющих повышенные требования к надежности, безопасности и производительности, и приложений аналитической обработки (систем поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);

– отсутствие полных аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем, высокая доля вновь разрабатываемого ПО;

– большое количество и высокая стоимость унаследованных приложений (существующего прикладного ПО), функционирующих в различной среде (персональные компьютеры, миникомпьютеры, мэйнфреймы), необходимость интеграции унаследованных и вновь разрабатываемых приложений;

– большое количество локальных объектов внедрения, территориально распределенная и неоднородная среда функционирования (СУБД, операционные системы, аппаратные платформы);

– большое количество внешних взаимодействующих систем – различных организаций с различными форматами обмена информацией (налоговая служба, налоговая полиция, Госстандарт, Госкомстат, Министерство финансов, МВД, местная администрация).

Организационные характеристики проектов создания ПО:

– различные формы организации и управления проектом: централизованно управляемая разработка тиражируемого ПО, экспериментальные пилотные проекты, инициативные разработки, проекты с участием как собственных разработчиков, так и сторонних компаний на контрактной основе;

– большое количество участников проекта как со стороны заказчиков (с разнородными требованиями), так и со стороны разработчиков (более 100 человек), разобщенность и разнородность отдельных групп разработчиков по уровню квалификации, сложившимся традициям и опыту использования тех или иных инструментальных средств;

– значительная длительность жизненного цикла системы, в том числе значительная временная протяженность проекта, обусловленная масштабами организации-заказчика, различной степенью готовности отдельных ее подразделений к внедрению ПО и нестабильностью финансирования проекта;

– высокие требования со стороны заказчика к уровню технологической зрелости организаций-разработчиков (наличие сертификации в соответствии с международными и отечественными стандартами).

Уже в конце 60-х годов прошлого века в США было отмечено явление под названием «**software crisis**» (кризис ПО). Это выражалось в том, что большие проекты стали выполняться с отставанием от графика или с превышением сметы расходов, разработанный продукт не обладал требуемыми функциональными возможностями, производительность его была низка, качество получаемого программного обеспечения не устраивало потребителей.

Аналитические исследования и обзоры, выполняемые в течение ряда последних лет ведущими зарубежными аналитиками, показывали не слишком обнадеживающие результаты. Так, например, результаты исследований, выполненных в 1995 году компанией Standish Group, которая проанализировала работу 364 американских корпораций и итоги выполнения более 23 тысяч проектов, связанных с разработкой ПО, выглядели следующим образом:

– только 16,2% завершились в срок, не превысили запланированный бюджет и реализовали все требуемые функции и возможности;

– 52,7% проектов завершились с опозданием, расходы превысили запланированный бюджет, требуемые функции не были реализованы в полном объеме;

– 31,1% проектов были аннулированы до завершения;

– для двух последних категорий проектов бюджет среднего проекта оказался превышенным на 89%, а срок выполнения – на 122%.

В соответствии с исследованиями 1998 года процентное соотношение трех перечисленных категорий проектов лишь немного изменилось в лучшую сторону (26%, 46% и 28% соответственно).

В последние годы процентное соотношение трех перечисленных категорий проектов также незначительно изменяется в лучшую сторону, однако, по оценкам ведущих аналитиков, это происходит в основном за счет снижения масштаба выполняемых проектов, а не за счет повышения управляемости и качества проектирования.

В числе причин возможных неудач, по мнению разработчиков, фигурируют:

– нечеткая и неполная формулировка требований к ПО;

– недостаточное вовлечение пользователей в работу над проектом;

– отсутствие необходимых ресурсов;

– неудовлетворительное управление проектом;

– частое изменение требований и спецификаций;

– новизна и несовершенство используемой технологии;

– недостаточная поддержка со стороны высшего руководства;

– недостаточно высокая квалификация разработчиков, отсутствие необходимого опыта.

Объективная потребность контролировать процесс разработки сложных систем ПО, прогнозировать и гарантировать стоимость разработки, сроки и качество результатов привела в конце 60-х годов прошлого века к необходимости перехода от кустарных к индустриальным способам создания ПО и появлению совокупности инженерных методов и средств создания ПО, объединенных общим названием «программная инженерия» (**software engineering**). В основе программной инженерии лежит следующая фундаментальная идея: проектирование ПО является формальным процессом, который можно изучать и совершенствовать. Освоение и правильное применение методов и средств создания ПО позволяет повысить его качество, обеспечить управляемость процесса проектирования ПО и увеличить срок его жизни.

В то же время, попытки чрезмерной формализации процесса, а также прямого заимствования идей и методов из других областей инженерной деятельности (строительства, производства) привели к ряду серьезных проблем. После двух десятилетий напрасных ожиданий повышения продуктивности процессов создания ПО, возлагаемых на новые методы и технологии, специалисты в индустрии ПО пришли к пониманию, что фундаментальная проблема в этой области – **неспособность эффективного управления проектами создания ПО**.

Выяснилось, что невозможно достичь удовлетворительных результатов от применения даже самых совершенных технологий и инструментальных средств, если они применяются

бессистемно. Часто разработчики не обладают необходимой квалификацией для работы с ними, а сам проект выполняется и управляется хаотически, в режиме «тушения пожара». Бессистемное применение технологий создания ПО (ТС ПО), в свою очередь, порождает разочарование в используемых методах и средствах. Анализ мнений разработчиков показывает, что среди факторов, влияющих на эффективность создания ПО, используемым методам и средствам придается гораздо меньшее значение, чем квалификации и опыту разработчиков. Если в таких условиях отдельные проекты завершаются успешно, то этот успех достигается за счет героических усилий фанатично настроенного коллектива разработчиков. Постоянное повышение качества создаваемого ПО и снижение его стоимости может быть обеспечено только при условии достижения организацией необходимой технологической зрелости, создании эффективной инфраструктуры как в сфере разработки ПО, так и в управлении проектами. В соответствии с моделью SEI CMM (Capability Maturity Model), в хорошо подготовленной (зрелой) организации персонал обладает технологией и инструментарием оценки качества процессов создания ПО на протяжении всего жизненного цикла ПО и на уровне всей организации.

Одна из причин распространенности «хаотического» процесса создания ПО – стремление сэкономить на стадии разработки, не затрачивая времени и средств на обучение разработчиков и внедрение технологического процесса создания ПО. Эти затраты до недавнего времени были довольно значительными и составляли, по различным оценкам (в частности, Gartner Group), более $100 тыс. и около трех лет на внедрение развитой ТС ПО, охватывающей большинство процессов жизненного цикла ПО, в многочисленной команде разработчиков (до 100 чел.). Причина – в сложности «тяжелых» технологических процессов, имеющих следующие особенности:

– необходимость документировать каждое действие разработчиков;

– множество рабочих продуктов (в первую очередь – документов), создаваемых в бюрократической атмосфере;

– отсутствие гибкости;

– детерминированность (долгосрочное детальное планирование и предсказуемость всех видов деятельности, а также распределение человеческих ресурсов на длительный срок, охватывающий большую часть проекта.

Альтернативой «тяжелому» процессу является адаптивный (гибкий) процесс, основанный на принципах «быстрой разработки ПО.

Современные тенденции в программной инженерии

В начале 2001 года века ряд ведущих специалистов в области программной инженерии (Алистер Коберн, Мартин Фаулер, Джим Хайсмит, Кент Бек и другие) сформировали группу под названием Agile Alliance. Слово agile (быстрый, ловкий, стремительный) отражало в целом их подход к разработке ПО, основанный на богатом опыте участия в разнообразных проектах в течение многих лет. Этот подход под названием «Быстрая разработка ПО» (Agile software development) базируется на четырех идеях, сформулированных ими в документе «Манифест быстрой разработки ПО» (Agile Alliance's Manifesto) и заключающихся в следующем:

– индивидуумы и взаимодействия между ними ценятся выше процессов и инструментов;

– работающее ПО ценится выше всеобъемлющей документации;

– сотрудничество с заказчиками ценится выше формальных договоров;

– реагирование на изменения ценится выше строгого следования плану.

При таком подходе технология занимает в процессе создания ПО вполне определенное место и повышает эффективность деятельности разработчиков при наличии следующих условий:

– технология позволяет людям легче выразить свои мысли;

– технология выполняет задачи, невыполнимые вручную;

– технология автоматизирует утомительные и подверженные ошибкам действия;

– технология облегчает общение между людьми.

Технология не должна действовать против характера культурных ценностей и познавательной способности человека.

Однако при всех достоинствах быстрой разработки ПО этот подход не является универсальным и применим только в проектах определенного класса. Для характеристики таких проектов Алистер Коберн ввел два параметра – критичность и масштаб. Критичность определяется последствиями, вызываемыми дефектами в ПО. Ее уровень может иметь одно из четырех значений:

– C – дефекты вызывают потерю удобства;

– D – дефекты вызывают потерю возместимых средств (материальных или финансовых);

– E – дефекты вызывают потерю невозместимых средств;

– L – дефекты создают угрозу человеческой жизни.

Масштаб определяется количеством разработчиков, участвующих в проекте:

– от 1 до 6 человек – малый масштаб;

– от 6 до 20 человек – средний масштаб;

– свыше 20 человек – большой масштаб.

По оценке Коберна, быстрая разработка ПО применима только в проектах малого и среднего масштаба с низкой критичностью (C или D). Общие принципы оценки технологий в таких проектах заключаются в следующем:

– интерактивное общение лицом к лицу – это самый дешевый и быстрый способ обмена информацией;

– избыточная «тяжесть» технологии стоит дорого;

– более многочисленные команды требуют более «тяжелых» и формальных технологий;

– большая формальность подходит для проектов с большей критичностью;

– возрастание обратной связи и коммуникации сокращает потребность в промежуточных и конечных продуктах;

– дисциплина, умение и понимание противостоят процессу, формальности и документированию;

– потеря эффективности в некритических видах деятельности вполне допустима.

Одним из наиболее известных примеров практической реализации подхода быстрой разработки ПО является «Экстремальное программирование» (**Extreme Programming** – XP). Этот метод предназначен для небольших компактных команд, нацеленных на получение как можно более высокого качества и продуктивности, что достигается посредством насыщенной, неформальной коммуникации, придания на персональном уровне особого значения умению, навыкам, дисциплине, пониманию.

Однако задача поиска повторяемого, предсказуемого процесса или методологии, которые бы улучшили продуктивность, качество и надёжность разработки до сих пор не решена. Одни подходы делают акцент на систематизации и формализации процессов. Другие ставят во главу угла методы управления проектами и методы программной инженерии. Третьи исходят из необходимости постоянного контроля со стороны заказчика.

При выборе методологии разработки программного обеспечения следует руководствоваться тем, что сложность методологии сравнима со сложностью структуры программного продукта, и неоправданная для продукта данной сложности сложность методологии только неоправданно увеличит стоимость разработки.

Инструментальные средства бизнес-моделирования

Моделирование бизнес-процессов является важной составной частью проектов по созданию крупномасштабных систем ПО организационно-экономических систем. Отсутствие таких моделей является одной из главных причин неудач многих проектов.

Назначением будущего ПО организационно-экономических систем является, в первую очередь, решение проблем бизнеса. Требования к ПО формируются на основе бизнес-модели, а критерии проектирования систем прежде всего основываются на наиболее полном их удовлетворении.

Модели бизнес-процессов являются не просто промежуточным результатом, используемым консультантом для выработки каких-либо рекомендаций и заключений. Они представляют собой самостоятельный результат, имеющий большое практическое значение.

Бизнес-моделирование (деловое моделирование) – деятельность по формированию моделей организаций, включающая описание деловых объектов (подразделений, должностей, ресурсов, ролей, процессов, операций, информационных систем, носителей информации и т. д.) и указание связей между ними. Требования к формируемым моделям и их соответствующее содержание определяются целями моделирования.

Бизнес-моделированием также называют дисциплину и отдельный подпроцесс в процессе разработки программного обеспечения, в котором описывается деятельность компании и определяются требования к системе. То есть те подпроцессы и операции, которые подлежат автоматизации в разрабатываемой информационной системе.

Нередко бизнес-моделирование сочетается с **управленческим консалтингом**, призванным выработать рекомендации по совершенствованию системы управления. В этом случае производится:

– анализ опыта других организаций (близких по профилю, отрасли, рынку, методам ведения бизнеса и т.д.), связанного с внедрением информационных технологий;

– определение целей проекта в контексте повышения эффективности решения существующих управленческих задач и возможности внедрения принципиально новых управленческих технологий;

– определение укрупненных показателей эффективности бизнес-процессов, подлежащих автоматизации (целевых бизнес-процессов), и формирование первоначальных критериев успешности проекта реорганизации управления на основе внедрения новых программных средств;

– определение приемлемого объема финансирования проекта.

При проведении управленческого консалтинга, направленного на выработку рекомендаций по совершенствованию системы управления предприятием проводятся следующие работы.

– Диагностика текущего состояния и тенденций развития предприятия.

– Выявление ключевых внутренних и внешних проблем.

– Анализ баланса сил, интересов и целей, распределения полномочий и ответственности среди участников (учредителей) и руководства предприятия и разработка рекомендаций по их корректировке.

– Подготовка рекомендаций по целевому планированию.

– Разработка предложений по корректировке стратегии развития организации.

– Анализ функционирования основных подсистем управления организацией и подготовка предложений по их совершенствованию.

– Анализ соответствия организационно-функциональной структуры организации стратегии ее развития и подготовка рекомендаций по совершенствованию оргструктуры.

– Выявление проблем информационного обеспечения системы управления организацией и оценка затрат на их решение.

На сегодняшний день в моделировании бизнес-процессов преобладает процессный подход. Его основной принцип заключается в структурировании деятельности организации в соответствии с ее бизнес-процессами, а не организационно-штатной структурой. При этом, модель, основанная на бизнес-процессах, содержит в себе и организационно-штатную структуру предприятия.

Наиболее популярными специализированными инструментами описания бизнес-процессов являются:

– Business studio (см. Приложение 1);

– ARIS (см. Приложение 2);

– Casewise Corporate Modeler Suite (см. Приложение 3);

– AllFusion Process Modeler (см. Приложение 4);

– PayDox (см. Приложение 5).

Наряду с ними применяются инструменты для описания, имитации и анимации бизнес-процессов:

– AnyLogic (см. Приложение 6);

– GPSS;

– IBM WebSphere Business Modeler.

Жизненный цикл программной системы

Жизненный цикл программного обеспечения (ПО) – это период времени с момента принятия решения о необходимости создания программной системы до момента ее полного изъятия из эксплуатации.

Основным международным стандартом определения жизненного цикла ПО является ISO/IEC 12207:2008 «System and software engineering – Software life cycle processes» (российский аналог – ГОСТ Р ИСО/МЭК 12207—2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств). Наряду с ним в России применяется ГОСТ 34.601—90.

ГОСТ 34.601—90

Стандарт ГОСТ 34.601—90 был создан еще в СССР и предусматривает следующие стадии и этапы создания автоматизированной системы (АС).

1. Формирование требований.

– Обследование объекта и обоснование необходимости создания АС.

– Формирование требований пользователя к АС.

– Оформление отчета о выполнении работ и заявки на разработку АС.

– Разработка концепции АС.

– Изучение объекта.

– Проведение необходимых научно-исследовательских работ.

– Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователей.

– Оформление отчета о проделанной работе.

2. Техническое задание – разработка и утверждение технического задания на создание АС.

3. Эскизный проект.

– Разработка предварительных проектных решений по системе и ее частям.

– Разработка документации на АС и ее части.

4. Технический проект.

– Разработка проектных решений по системе и ее частям.

– Разработка документации на АС и ее части.

– Разработка и оформление документации на поставку комплектующих изделий.

– Разработка заданий на проектирование в смежных частях проекта.

– Рабочая документация.

– Разработка рабочей документации на АС и ее части.

– Разработка и адаптация программ.

5. Ввод в действие.

– Подготовка объекта автоматизации.

– Подготовка персонала.

– Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями).

– Строительно-монтажные работы.

– Пусконаладочные работы.

– Проведение предварительных испытаний.

– Проведение опытной эксплуатации.

– Проведение приемочных испытаний.

6. Сопровождение АС.

– Выполнение работ в соответствии с гарантийными обязательствами.

– Послегарантийное обслуживание.

Эскизный, технический проекты и рабочая документация – это последовательное построение все более точных проектных решений. Допускается исключать стадию «Эскизный проект» и отдельные этапы работ на всех стадиях, объединять стадии «Технический проект» и «Рабочая документация» в «Технорабочий проект», параллельно выполнять различные этапы и работы, включать дополнительные.

Данный стандарт не вполне подходит для проведения разработок в настоящее время: многие процессы отражены недостаточно, а некоторые положения устарели.

ГОСТ Р ИСО/МЭК 12207—2010

Федеральным агентством по техническому регулированию и метрологии РФ 01.03.2012 г. принят стандарт ГОСТ Р ИСО/МЭК 12207—2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств», идентичный международному стандарту ISO/IEC 12207:2008 «System and software engineering – Software life cycle processes».

Данный стандарт, используя устоявшуюся терминологию, устанавливает общую структуру процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии. Стандарт определяет процессы, виды деятельности и задачи, которые используются при приобретении программного продукта или услуги, а также при поставке, разработке, применении по назначению, сопровождении и прекращении применения программных продуктов.

Стандарт ГОСТ Р ИСО/МЭК 12207—2010 не предлагает конкретную модель жизненного цикла. Его положения являются общими для любых моделей жизненного цикла, методов и технологий создания ИС. Он описывает структуру процессов жизненного цикла, не конкретизируя, как реализовать или выполнить действия и задачи, включенные в эти процессы.

Стандарт группирует различные виды деятельности, которые могут выполняться в течение жизненного цикла программных систем, в семь групп процессов. Каждый из процессов жизненного цикла в пределах этих групп описывается в терминах цели и желаемых выходов, списков действий и задач, которые необходимо выполнять для достижения этих результатов.

Группы процессов:

процессы соглашения – два процесса;

процессы организационного обеспечения проекта – пять процессов;

процессы проекта – семь процессов;

технические процессы – одиннадцать процессов;

процессы реализации программных средств – семь процессов;

процессы поддержки программных средств – восемь процессов;

процессы повторного применения программных средств – три процесса.

Каждый процесс включает ряд действий. Например, процесс приобретения охватывает следующие действия.

Инициирование приобретения.

Подготовка заявочных предложений.

Подготовка и корректировка договора.

Надзор за деятельностью поставщика.

Приемка и завершение работ.

Каждое действие включает ряд задач. Например, подготовка заявочных предложений должна предусматривать:

Формирование требований к системе.

Формирование списка программных продуктов.

Установление условий и соглашений.

Описание технических ограничений (среда функционирования системы).

Модель жизненного цикла ПО – структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Она зависит от специфики, масштаба и сложности проекта и особенностей условий, в которых система создается и функционирует.

Модель ЖЦ ПО включает несколько стадий.

Стадия – часть процесса создания ПО, ограниченная определенными временными рамками и заканчивающаяся выпуском конкретного продукта (моделей, программных компонентов, документации), определяемого заданными для данной стадии требованиями.

На каждой стадии могут выполняться несколько процессов, определенных в стандарте ГОСТ Р ИСО/МЭК 12207—2010, и наоборот, один и тот же процесс может выполняться на различных стадиях. Соотношение между процессами и стадиями также определяется используемой моделью жизненного цикла ПО.

Модели жизненного цикла ПО

Выделяют следующие возможные модели ЖЦ ПО:

– Водопадная

– Итерационная

– Спиральная

Водопадная (каскадная, последовательная) модель

Водопадная модель жизненного цикла (англ. waterfall model) была предложена в 1970 г. Уинстоном Ройсом. Она предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе. Требования, определенные на стадии формирования требований, строго документируются в виде технического задания и фиксируются на все время разработки проекта. Каждая стадия завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Этапы проекта в соответствии с каскадной моделью:

Формирование требований;

Проектирование;

Реализация;

Тестирование;

Внедрение;

Эксплуатация и сопровождение.

Преимущества каскадной модели:

– полная и согласованная документация на каждом этапе;

– легко определить сроки и затраты на проект.

В водопадной модели переход от одной фазы проекта к другой предполагает полную корректность результата (выхода) предыдущей фазы. Однако неточность какого-либо требования или некорректная его интерпретация в результате приводит к тому, что приходится возвращаться к ранней фазе проекта. В результате требуемая переработка приводит не только к нарушению графика, но и к существенному росту затрат.

По мнению современных специалистов, основное заблуждение авторов водопадной модели состоит в предположениях, что проект проходит через весь процесс один раз, спроектированная архитектура хороша и проста в использовании, проект осуществления разумен, а ошибки в реализации легко устраняются по мере тестирования. Эта модель исходит из того, что все ошибки будут сосредоточены в реализации, а потому их устранение происходит равномерно во время тестирования компонентов и системы. Таким образом, водопадная модель для крупных проектов мало реалистична и может быть эффективно использована только для создания небольших систем.

Итерационная модель

Альтернативой последовательной модели является так называемая модель итеративной и инкрементальной разработки (англ. iterative and incremental development, IID), получившей также название эволюционной модели.

Модель IID предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «мини-проект», включая все процессы разработки в применении к созданию меньших фрагментов функциональности, по сравнению с проектом в целом. Цель каждой итерации – получение работающей версии программной системы, включающей функциональность, определённую интегрированным содержанием всех предыдущих и текущей итерации. Результат финальной итерации содержит всю требуемую функциональность продукта. Таким образом, с завершением каждой итерации продукт получает приращение – инкремент – к его возможностям, которые, следовательно, развиваются эволюционно.

По выражению Т. Гилба, «эволюция – прием, предназначенный для создания видимости стабильности. Шансы успешного создания сложной системы будут максимальными, если она реализуется в серии небольших шагов и если каждый шаг заключает в себе четко определённый успех, а также возможность „отката“ к предыдущему успешному этапу в случае неудачи. Перед тем, как пустить в дело все ресурсы, предназначенные для создания системы, разработчик имеет возможность получать из реального мира сигналы обратной связи и исправлять возможные ошибки в проекте».

Недостатки подхода IID:

– отсутствие целостного понимания возможностей и ограничений проекта на ранних итерациях;

– необходимость переделывать часть сделанной ранее работы;

– снижение добросовестности специалистов при выполнении работ (всё равно всё можно будет переделать и улучшить позже).

Различные варианты итерационного подхода реализованы в большинстве современных методологий разработки (RUP, MSF, XP).

Спиральная модель

Спиральная модель (англ. spiral model) была разработана в середине 1980-х годов Барри Боэмом. Она основана на классическом цикле Деминга PDCA (plan-do-check-act). При использовании этой модели ПО создается в несколько итераций (витков спирали) методом прототипирования.

Каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации.

На каждой итерации оцениваются:

– риск превышения сроков и стоимости проекта;

– необходимость выполнения ещё одной итерации;

– степень полноты и точности понимания требований к системе;

– целесообразность прекращения проекта.

Спиральная модель является усовершенствованным вариантом эволюционной модели (модели IID). Ее отличительной особенностью является специальное внимание, уделяемое рискам, влияющим на организацию разработки. Наиболее распространёнными являются следующие группы рисков.

– Дефицит специалистов.

– Нереалистичные сроки и бюджет.

– Реализация несоответствующей функциональности.

– Разработка неправильного пользовательского интерфейса.

– Перфекционизм, ненужная оптимизация и оттачивание деталей.

– Непрекращающийся поток изменений.

– Нехватка информации о внешних компонентах, определяющих окружение системы.

– Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.

– Недостаточная производительность получаемой системы.

– Разрыв в квалификации специалистов разных областей.

Для преодоления рисков при реализации проектов предлагается использовать так называемые контрольные точки, характеризующие разные стадии готовности проекта.

– Concept of Operations (COO) – концепция (использования) системы;

– Life Cycle Objectives (LCO) – цели и содержание жизненного цикла;

– Life Cycle Architecture (LCA) – архитектура жизненного цикла; здесь же возможно говорить о готовности концептуальной архитектуры целевой программной системы;

– Initial Operational Capability (IOC) – первая версия создаваемого продукта, пригодная для опытной эксплуатации;

– Final Operational Capability (FOC) – готовый продукт, развернутый (установленный и настроенный) для реальной эксплуатации.

Формирование требований и проектирование программной системы

Требования к программному обеспечению – совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации.

Требования могут выражаться в виде текстовых утверждений и графических моделей.

В классическом техническом подходе совокупность требований используется на стадии проектирования ПО. Требования также используются в процессе проверки ПО, так как тесты основываются на определённых требованиях.

Этапу разработки требований часто предшествует технико-экономическое обоснование, или концептуальная фаза анализа проекта. Для ЭИС это, как правило, бизнес-моделирование.

Стадии фазы разработки требований:

– выявление;

– оценка целостности и реализуемости;

– документирование;

– согласование.

Разработка требований к ПО – процесс выявления, формулирования, анализа, документирования и верификации требований, подлежащих выполнению в программном обеспечении (ПО). В его ходе системный аналитик формирует реестр требований, который оформляется в виде документа, а также может быть занесен в автоматизированную систему управления требованиями.

Уровни требований к ПО.

– **Бизнес-требования** – определяют назначение ПО, описываются в документе о видении (vision) и границах проекта (scope).

– **Пользовательские требования** – определяют набор пользовательских задач, которые должна решать программная система, а также способы (сценарии) их решения.

– **Функциональные требования** – характеризуют предполагаемое поведение системы в виде набора определенных действий, которые описываются в системной спецификации (англ. system requirement specification, SRS).

– **Нефункциональные требования** – набор условий, которым должна соответствовать программная система.

К **нефункциональным** требованиям относятся:

– Ограничения на программные интерфейсы, в том числе к внешним системам.

– Требования к атрибутам качества.

– Требования к применяемому оборудованию и ПО.

– Требования к документированию.

– Требования к дизайну.

– Требования к безопасности и надёжности.

– Требования к показателям назначения (производительность, устойчивость к сбоям и т.п.).

– Требования к эксплуатации и персоналу.

– Прочие требования и ограничения (мобильность, автономность и т.п.).

Источники требований:

– Федеральное и муниципальное отраслевое законодательство (конституция, законы, распоряжения)

– Нормативное обеспечение организации (регламенты, положения, уставы, приказы)

– Текущая организация деятельности объекта автоматизации
– Модели деятельности (диаграммы бизнес-процессов)
– Представления и ожидания потребителей и пользователей системы
– Журналы использования существующих программно-аппаратных систем
– Конкурирующие программные продукты

Свойства требований

Характеристика	Объяснение
Единичность	Требование упоминается единожды.
Завершённость	Требование полностью определено в одном месте.
Последовательность	Требование не противоречит другим требованиям.
Атомарность	Требование не может быть разбито на ряд более детальных требований без потери завершённости.
Отслеживаемость	Требование документировано.
Актуальность	Требование не стало устаревшим с течением времени.
Выполнимость	Требование может быть реализовано в пределах проекта.
Недвусмысленность	Требование кратко определено без обращения к техническому жаргону, акронимам и другим скрытым формулировкам. Оно выражает объективные факты, а не субъективные мнения. Возможна одна и только одна интерпретация. Определение не содержит нечётких фраз. Использование отрицательных утверждений и составных утверждений запрещено.
Обязательность	Требование представляет характеристику, отсутствие которой приведёт к неполноценности решения.
Проверяемость	Реализованность требования может быть определена посредством осмотра, демонстрации, теста или анализа

Методы выявления требований

– Интервью, опросы, анкетирование.
– Мозговой штурм, семинар.
– Наблюдение за производственной деятельностью.
– Анализ нормативной документации.
– Анализ моделей деятельности.
– Анализ конкурентных продуктов.
– Анализ статистики использования предыдущих версий системы.

Все требования должны поддаваться проверке. Наиболее распространенная методика проверки – тесты. Если проверка тестами невозможна, тогда должен использоваться другой метод проверки (анализ, демонстрация, обзор дизайна).

Нефункциональные требования, неподдающиеся проверке на программном уровне, должны быть отдельно документированы. Во многих случаях они могут быть преобразованы из требования к продукту в требования к процессу. Например, нефункциональное требование, чтобы ПО не содержало «потайных ходов», может быть удовлетворено заменой на требование использовать парное программирование. Сложные требования безопасности

авиационного программного обеспечения могут быть удовлетворены следованием процессу разработки DO-178B.

Конец ознакомительного фрагмента.