Основные понятия технологии проектирования информационных систем.

Основные понятия в последние годы не претерпели сильных изменений, формулировки стали более точными и лаконичными, исключающими неоднозначность понятий.

Наиболее полные определения представлены в Федеральных законах Российской Федерации и стандартах.

Информация – это сведения (сообщения, данные) независимо от формы их представления». Информационные технологии – «процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов.

Информационная система – это совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств. Проектирование информационных систем – это упорядоченная совокупность методологий и средств создания или модернизации информационных систем.

Управление информационными системами – это применение методов управления процессами планирования, анализа, дизайна, создания, внедрения и эксплуатации информационной системы организации для достижения ее целей.

Жизненный цикл информационной системы – это развитие рассматриваемой системы во времени, начиная от замысла и кончая списанием.

Модель жизненного цикла – структурная основа процессов и действий, относящиеся к жизненному циклу, которая служит в качестве общей ссылки для установления связей и взаимопонимания сторон.

Архитектура информационных систем – это концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы.

Бизнес-процесс – это цепочка взаимосвязанных действий, направленных на создание товарной продукции или услуги.

Регламент бизнес-процесса – это четко определенный порядок выполнения бизнес-процесса, определяющий состав и действия участников.

Модель данных – это система организации данных и управления ими.

Методология проектирования информационных систем – это совокупность принципов проектирования (моделирования), выраженная в определенной концепции.

Средства моделирования – это программы описания и моделирования систем.

Типовое проектное решение (ТПР) – это многократно используемое проектное решение.

Нотации – это определенные способы представления элементов информационной системы. Реинжиниринг бизнес-процессов – это фундаментальная реорганизация бизнес-процессов с целью повышения их эффективности.

Системный подход – процесс рассмотрения любой системы в качестве совокупности взаимосвязанных элементов.

Процессный подход – представление любой системы в качестве совокупности процессов. Функциональный подход – предусматривает четкое закрепление за каждой структурной единицей набора функций.

Техническое задание – документ, используемый заказчиком в качестве средства для описания и определения задач, выполняемых при реализации договора

Начало зарождения индустрии ПО - середина 50-х годов XIX в. В 1970 г годовой оборот всех фирм-разработчиков ПО в США составлял около 3,7% всего оборота компьютерного бизнеса. Серьезный рост начался в 70-х годах XX в., начиная с принятого фирмой IBM в 1969 г. решения о развязывании цен (раздельном назначении цен на аппаратуру, ПО и услуги), и продолжился до конца декады и появления персонального компьютера. К 1979 г. годовой объем продаж фирм-разработчиков ПО в США составлял около $2 млрд. В 80-х годах рост составлял 20% в год и более.

Сегодня общий объем продаж ПО превышает $100 млрд. Производство ПО сегодня — крупнейшая отрасль мировой экономики, в которой занято около трех миллионов специалистов, называющих себя программистами, разработчиками ПО и т.п. Еще несколько миллионов человек занимают рабочие места, напрямую зависящие от благополучия корпоративных информационных подразделений либо от производителей ПО, таких, как корпорации Microsoft или IBM.

Накопленный к настоящему времени опыт создания систем ПО показывает, что это логически сложная, трудоемкая и длительная 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. Кроме того, в процессе создания и функционирования ПО потребности пользователей постоянно изменяются или уточняются, что еще более усложняет разработку и сопровождение таких систем.

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 последние годы ведущими зарубежными аналитиками, показывали не слишком обнадеживающие результаты.

Например, результаты исследований, выполненных в 1995 г компанией Standish Group, которая проанализировала работу 364 американских корпораций и итоги выполнения более 23 тыс. 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 1998 г процентное соотношение трех перечисленных категорий проектов лишь немного изменилось в лучшую сторону (26%, 46% и 28% соответственно).

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 планирование и отсутствие грамотного управления проектом;

• частое изменение требований и спецификаций;

• новизна и несовершенство используемой технологии;

• недостаточная поддержка со стороны высшего руководства;

• недостаточно высокая квалификация разработчиков, отсутствие необходимого опыта.

В последнее время ведущие зарубежные аналитики отмечают как одну из причин многих неудач тот факт, что множество проектов выполняется в экстремальных условиях. В англоязычной литературе с легкой руки Эдварда Йордона одного из ведущих мировых специалистов в области ПО, утвердилось название «death march», буквально -- «смертельный марш». Под ним понимается такой проект, параметры которого отклоняются от нормальных значений, по крайней мере, на 50%. По отношению к проектам создания ПО это означает наличие одного или более из следующих ограничений:

• план проекта сжат более чем наполовину по сравнению с нормальным расчетным планом; таким образом, проект, требующий в нормальных условиях 12 календарных месяцев, приходится выполнять за 6 или менее месяцев. Жесткая конкуренция на мировом рынке делает такую ситуацию наиболее распространенной;

• количество разработчиков уменьшено более чем наполовину по сравнению с действительно необходимым для проекта данного размера и масштаба. На сегодняшний день наиболее обшей причиной уменьшения количества разработчиков является сокращение штатов компании в результате кризиса, реорганизации и т.д.;

• бюджет и связанные с ним ресурсы урезаны наполовину. Зачастую это результат сокращения компании и других противозатратных мер, хотя это может быть и результатом конкурентной борьбы за выгодный контракт. Такое ограничение часто непосредственно влияет на количество нанимаемых разработчиков, однако последствия могут быть и менее явными - например, может быть принято решение привлечь относительно недорогих и неопытных молодых разработчиков вместо опытных дорогостоящих специалистов;

• требования к функциям, возможностям, производительности и другим техническим характеристикам вдвое превышают значения, которые они могли бы иметь в нормальных условиях.

Все знание о способах разработки ПО основано исключительно на пробах и ошибках. Конечно, со времени зарождения программирования сообщество программистов добилось некоторых успехов. Признанные корифеи программирования изобрели в помощь разработчикам ПО успешные методы и правила. Но, подобно средневековым архитекторам, разработчики ПО изучили и испытали эти методы путем проб и ошибок. Современным разработчикам не хватает системы основных принципов, на основе которых можно было бы строить свои правила и методы. Замечено, что ключевым фактором успеха проекта является хорошая архитектура. Именно неспособность регулярно создавать хорошую архитектуру не дает права разработке ПО называться сложившейся инженерной дисциплиной. Для доказательства адекватности проекта до завершения любых строительных работ инженер, в отличие от программиста, использует систему основных принципов. Разработчик ПО, с другой стороны, при оценке качества архитектуры должен полагаться на тестирование. Он должен искать хорошую архитектуру путем проб и ошибок. Это объясняет, почему двумя наиболее явными проблемами неудачных программных проектов являются переделка программ и обнаружение негодности проекта на его поздних стадиях.

Разработчик проектирует архитектуру на ранних стадиях разработки ПО, но не имеет возможности сразу же оценить ее качество. У него отсутствуют под рукой основные принципы для доказательства адекватности проекта. Тестирование программного обеспечения постепенно выявляет все дефекты архитектуры, но только на поздних стадиях разработки, когда исправление

Почему же идеология проб и ошибок так глубоко проникла в разработку ПО? Для ответа на этот вопрос необходимо понять, что разработка ПО изначально является проектированием и не имеет признаков строительства WRVL производства. Это утверждение трудно принять, но оно может быть легко обосновано. Все хорошо понимают, что такое проектирование, где оно заканчивается и где начинается строительство или производство. Рассмотрим два следующих аргумента:

1. Граница между проектированием и строительством всегда четко обозначена чертежом. Проектирование включает в себя все операции, необходимые для создания чертежа, а строительство охватывает все операции, необходимые для создания продуктов по этому чертежу. В идеальном случае чертеж должен определять создаваемый продукт во всех подробностях, что, конечно же, бывает очень редко. Тем не менее, целью чертежа является настолько подробное описание конструируемого продукта, насколько это возможно. Описывает ли проект архитектуры программной системы создаваемый продукт «во всех подробностях»? — Нет. Проект архитектуры предназначен для описания существенных, но, безусловно, не всех подробностей программной системы. Поэтому очевидно, что проект архитектуры не является чертежом. Все подробности программной системы описываются только кодом на языке высокого уровня, который, таким образом, является чертежом программы. А поскольку все операции, ведущие к созданию чертежа, являются проектированием, то и вся разработка ПО должна считаться проектированием.

2. Объем работ (время, деньги, ресурсы), необходимый для создания продукта, всегда может быть разделен на проектировочную и производственную составляющие. В чем разница? — Объем работ проектирования является общим для всех копий продукта и должен быть затрачен только один раз. Объем работ для производства должен затрачиваться при создании каждой копии продукта. Программный продукт обычно представляет собой двоичный исполняемый файл программы, поставляемой на компакт-диске. Ясно, что усилия по созданию исходного кода программы, включая проект архитектуры, подробный проект и код на языке высокого уровня, должны быть затрачены лишь однажды, независимо от количества выпущенных копий программного обеспечения. Следовательно, усилия по созданию исходного кода программы являются целиком проектировочными, а вся разработка ПО является проектированием.

Разработчики не строят ПО - они его проектируют. Конечный результат проектирования — код на языке высокого уровня — является чертежом ПО. Компилятор и компоновщик механически строят программный продукт — двоичный исполняемый файл - по этому спроектированному коду. Проект архитектуры программной системы наиболее близко соответствует картонным моделям или эскизам проекта, используемым в некоторых инженерных дисциплинах

Чтобы понять, почему разработчики ПО до сих пор не увидели и не нашли основных принципов, представим себе мир, в котором создание небоскреба не требует ничего, кроме подробного чертежа. Имея чертеж, архитектор мог бы одним нажатием кнопки построить небоскреб — мгновенно и практически без затрат. Затем архитектор мог бы проверить небоскреб и сравнить его с техническими требованиями. Если бы он разрушился или не смог пройти проверку, архитектор мог бы его снести и убрать обломки - мгновенно и опять же без затрат. Стал бы этот архитектор тратить много времени на формальную проверку согласованности проекта с физическими законами? Или хотя бы пытаться исследовать и понять эти законы? — Вряд ли. Он, вероятно, смог бы получить результаты быстрее путем многократного строительства, проверки и сноса небоскреба, каждый раз внося исправления в чертеж. В мире, где строительство и разрушение бесплатны, выбирается метод проб и ошибок, а фундаментальные исследования остаются на будущее. ПО разрабатывается именно в таком мире. Программист создает чертеж в виде программы на языке высокого уровня. Затем он позволяет компилятору и компоновщику в мгновение ока и почти без затрат построить программный продукт. Создание чертежа требует значительных усилий, но строительство с помощью компилятора и компоновщика практически бесплатно. Программисту вообще не надо беспокоиться о сносе и уборке обломков, по крайней мере до тех пор, пока ему достаточно дискового пространства. Неудивительно, что идеология проб и ошибок так глубоко укоренилась в процессе разработки ПО, а сообщество программистов не удосужилось исследовать основные принципы разработки ПО. Метод проб и ошибок завел достаточно далеко. Но рост сложности современных программных систем подводит нас к жесткому пределу. За пределами определенного уровня сложности создание качественных архитектур методом проб и ошибок становится невозможным.