Лекция 2. Проблемы построения ИС

Самой первой проблемой является проблема проектирования. Нельзя начинать техническую разработку, не имея тщательно проработанного проекта. Если начинать с решения наиболее очевидных задач, не обращая внимания на потенциально существующие, то такая система будет непрерывно находиться в стадии разработки и переделки.

Первой стадией проектирования должен быть анализ требований корпорации. Для этого на основе экспертных запросов необходимо выявить все актуальные и потенциальные потребности корпорации, которые должны удовлетворяться проектируемой информационной системой, понять какие потоки данных существуют внутри корпорации, оценить объемы информации, которые должны поддерживаться и обрабатываться информационной системой. Эта стадия, как правило, носит неформальный характер, хотя, конечно, очень важно сохранить полученную информацию, поскольку она должна входить в документацию системы. Существуют CASE-средства верхнего уровня, которые помещают полученные данные в общий репозиторий проекта и позволяют использовать их на следующих стадиях проектирования.

Следующая стадия проектирования - выработка концептуальной схемы базы данных, которая будет лежать в основе информационной системы. Сначала придется выбрать систему *нотаций*, в которой будет представляться концептуальная схема (правильнее было бы сказать - выбрать концептуальную модель). Таких нотаций существует великое множество, и хотя практически все они диаграммные (в духе ER-модели), они отличаются одна от другой. Заметим, что даже в случае использования некоторых CASE-средств вам все равно предлагается на выбор несколько нотаций. Заметим, что хотя, на взгляд автора, выбор нотации - дело вкуса (по возможностям они почти эквивалентны), это ответственный выбор. Концептуальное представление базы данных должно сохраняться как часть документации информационной системы на все время ее существования и будет использоваться при ее сопровождении и развитии.

Далее, с большой вероятностью в основе информационной системы будет лежать реляционная база данных. Несмотря на очевидную привлекательность объектно-ориентированных (ObjectStore, Objectivity, O2, Jasmin и т.д.) и объектно-реляционных (Illustra, UniSQL) СУБД, в ближайшие годы придется работать с хорошо отлаженными, развитыми, сопровождаемыми системами, поддерживающими стандарт SQL-92 (например, Oracle, Informix, CA-OpenIngres, Sybase, DB2). Просто потому, что должно пройти время, чтобы эти системы устоялись, обрели необходимую надежность, стали бы опираться на какие-либо стандарты и т.д.

Поэтому, с большой вероятностью, на следующей стадии проектирования понадобится на основе имеющейся концептуальной схемы произвести набор определений схемы реляционной базы данных в терминах языка SQL. К сожалению, несмотря на наличие стандарта языка, на этой стадии иногда невозможно не учитывать специфику сервера баз данных, который будет использоваться. Вы спросите, почему "к сожалению"? Да потому, что на самом деле мы еще не дошли до той стадии, когда конкретные особенности сервера действительно необходимо учитывать. В принципе, на данной стадии мы все еще находимся на уровне абстрактной реляционной модели. Но все дело в том, что когда производители серверов баз данных провозглашают соответствие своих серверных продуктов стандарту языка SQL-92, то в основном они понимают соответствие так называемому "ядру" стандарта. К сожалению, ядро стандарта не включает средств определения схемы базы данных. Поэтому диалекты SQL, реализуемые разными

производителями, различаются в деталях соответствующих языковых средств. По этой причине необходимо внимательно изучить "целевой" диалект SQL, если трансляция концептуальной схемы в реляционную производится вручную (например, на основе методологии, предлагаемой компанией Oracle), или указать название используемого серверного продукта, если используется продукто-независимое CASE-средство (например, Silverrun).

На этой же стадии необходимо решить, какие таблицы будут реально хранимыми, а какие - представляемыми (view).

После того, как выработана общая реляционная схема базы данных, необходимо определиться с архитектурой системы. В частности, очень важно решить, какой будет база данных - централизованной или распределенной (другими словами будет ли использоваться только один сервер баз данных или их будет несколько). Если принимается решение о распределенном характере базы данных, то необходимо произвести соответствующую декомпозицию набора определений схемы базы данных. (Заметим, что, вообще говоря, принятие решения об архитектуре системы возможно и до выработки общей реляционной схемы базы данных. Тогда декомпозиция производится на уровне концептуальной схемы, а затем для каждой отдельной части концептуальной схемы создается реляционная схема в терминах языка SQL.)

Наиболее простым случаем декомпозиции является тот, когда образующиеся разделы базы данных логически автономны. В терминах концептуальной схемы это означает, что между разделенными сущностями отсутствуют прямые или транзитивные (через другие сущности) связи. В терминах реляционной схемы: ни в одном разделе не присутствует таблица, ссылающаяся на таблицу, которая располагается в другом разделе (рисунки 1 и 2). Если требование логической автономности компонентов распределенной базы данных выполнено, то дальнейшее проектирование можно производить для каждого компонента независимо.

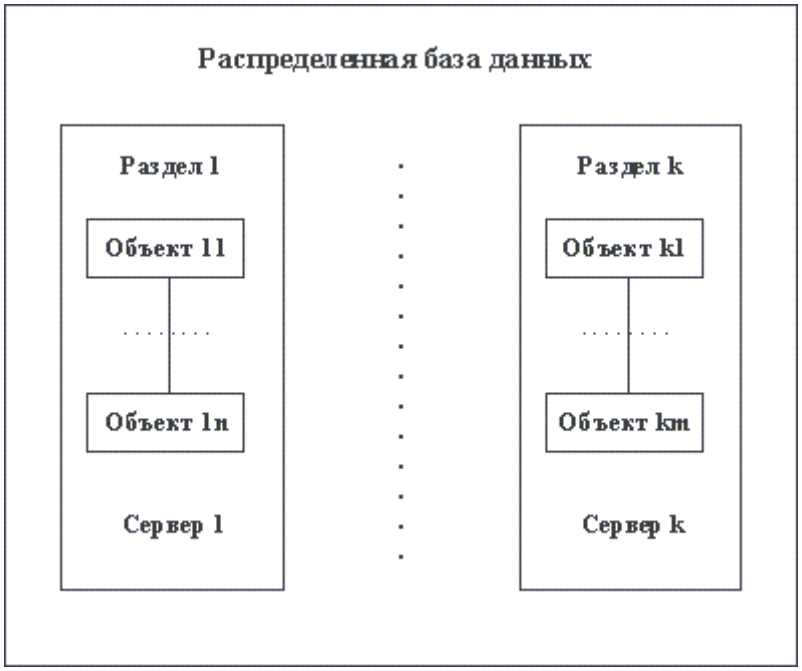

Рис. 1. Распределенная база данных с логически автономными разделами

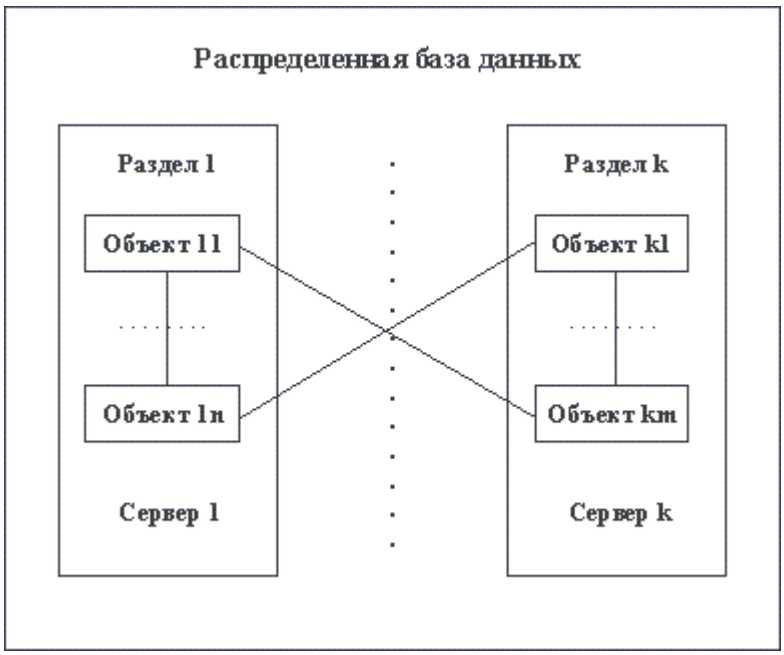

Рис. 2. Распределенная база данных с логически неавтономными разделами

В чем, собственно, состоит проблема распределенных баз данных с логически неавтономными разделами? Все дело в том, что любая связь, отраженная в схеме базы данных (между сущностями в концептуальной модели или между таблицами в реляционной модели), соответствует ограничению целостности, которое впоследствии должно сохраняться в базе данных и поддерживаться СУБД. В настоящее время общая технология организации распределенных баз данных (даже однородных, основанных на SQL) отсутствует. Некоторые производители решают эту задачу путем расширения функций сервера, и тогда, вообще говоря, в базе данных могут присутствовать ограничения целостности, содержащие ссылки на таблицы, которые располагаются в других разделах. В других системах задача управления распределенной базой данных решается на стороне клиента. В этом случае сервер, управляющий разделом распределенной базы данных ничего не знает о существовании других разделов и не может поддерживать ограничения целостности, включающие ссылки на объекты "чужих" разделов. Тогда целостность распределенной базы данных приходится поддерживать за счет написания явного кода в приложении. Другими словами, если невозможно произвести декомпозицию схемы общей базы данных в набор схем логически независимых разделов, то снова необходимо учитывать возможности используемого серверного продукта и двигаться дальше соответствующим образом.

Следующая стадия проектирования состоит в дополнении реляционных схем разделов распределенной базы данных определениями общих ограничений целостности, триггеров и хранимых процедур. На каждом из этих компонентов схемы следует остановиться отдельно. Начнем с общих ограничений целостности.

Требования к техническим средствам, поддерживающим ИС

Естественно, что требования к техническим средствам определяются требованиями к информационной системе в целом. Какие бы информационные возможности не требовались сотрудникам корпорации, окончательное решение всегда принимается ее руководством, которое корректирует требования к информационной системе и формирует окончательное представление об аппаратной среде. На наш взгляд, имеются четыре возможных позиции руководства по поводу места информационной системы в

корпорации: пессимистическая, пессимистически-оптимистическая, оптимистически-пессимистическая и оптимистическая.

Руководитетель-пессимист рассуждает следующим образом. Корпорации нужно продержаться хотя бы какое-то время. Без информационной системы это невозможно. Нужно выбрать самое дешевое решение, которое может быть реализовано максимально быстро. Руководитель не думает, что будет с корпорацией через два года (вернее, поскольку он пессимист, то думает, что, скорее всего, через два года корпорация просто не будет существовать или у нее сменится руководитель). При такой позиции наиболее подходящим является некоторое закрытое и законченное техническое решение. Например, это может быть полностью сбалансированная локальная сеть Novell с выделенным файл-сервером и фиксированным числом рабочих станций. Жесткость решения затем закрепляется соответствующим программным обеспечением информационной системы. Возможности расширения системы отсутствуют, реинжиниринг требует практически полной переделки системы.

Пессимистически-оптимистический руководитель не ожидает краха корпорации или своего собственного увольнения. Но он не надеется на изменение статуса компании, например, на появление зарубежных филиалов. Возможно, корпорация будет несколько развиваться, возможно, появятся новые виды бизнеса, возможно, увеличится число сотрудников. Но поскольку руководитель все-таки более пессимист, чем оптимист, то он не очень высоко оценивает шансы на развитие (дай-то Бог, чтобы за два года мы выросли на 20%). Такой позиции руководства больше всего подходит закрытое решение, обладающее ограниченными возможностями расширения. Например, это может быть локальная сеть Novell, в которой пропускная способность превосходит потребности имеющихся рабочих станций, а файл-сервер может быть оснащен дополнительными магнитными дисками. Если пессимизм руководителя окажется неоправданным, то корпорация встретится с потребностью сложного реинжиниринга.

Руководитель с оптимистически-пессимистической позицией ставит своей целью развитие корпорации. Он учитывает, что при развитии корпорации потребуется соответствующее развитие информационной системы, для чего, вообще говоря, может понадобиться сменить сервер баз данных. Он учитывает, что при развитии корпорации могут образоваться территориально разнесенные офисы, в результате чего, возможно, потребуется перейти к использованию распределенной базы данных. Он учитывает, в конце концов, что корпорации могут потребоваться развитые средства телекоммуникации с удаленными филиалами и/или партнерами. Пессимизм руководителя состоит только в том, что он заранее делает ставку на одного производителя (эти-то продукты я знаю). Например, может быть сделана установка на использование только Intel-платформ в среде Microsoft или только Alpha-платформ с VAX/VMS. Вообще говоря, это здоровый пессимизм, поскольку однородность аппаратно-программной среды существенно облегчает ее администрирование. Но как обидно будет этому руководителю, если ему предложат дешево купить прекрасный аппаратный продукт другого производителя. Придется либо отказаться, либо снова производить изнурительный реинжиниринг.

Наконец, руководитель-оптимист разделяет позицию оптимистически-пессимистического руководителя по поводу перспектив развития корпорации, но при этом желает сохранить возможность использования различных аппаратных платформ. Он стимулирует построение открытой корпоративной информационной системы, которая может неограниченно наращиваться за счет подключения новых сегментов сети, включения новых серверов и рабочих станций. Оптимистический подход, естественно, требует применения международных стандартов, что облегчает комплексирование аппаратного

комплекса и обеспечивает реальное масштабирование информационной системы. Если начать построение корпоративной системы с сети Ethernet с использованием стека протоколов TCP/IP, то это начало оптимистического решения. В этом случае проблемы реинжиниринга практически отсутствуют (пока не сменятся стандарты).

Конечно, приведенная классификация является несколько утрированной. В жизни все гораздо сложнее. В частности, нельзя не учитывать влияние на руководителя технических специалистов. Чем грамотнее составляется обоснование на приобретение технических средств, тем обоснованнее оптимизм или пессимизм руководителя. Конечно, многое определяется возможными денежными затратами. Но стоит заметить, что оптимистический подход (по сути дела, это подход открытых систем) требует минимальных затрат на начальном этапе становления корпоративной системы (если, конечно, не учитывать необходимость приобретения хорошего программного сервера баз данных).

Понятия метода и технологии проектирования ПО

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем (ИС), создаваемых в различных областях экономики. Современные крупные проекты ИС характеризуются, как правило, следующими особенностями:

- сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;
- наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);
- отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;
- необходимость интеграции существующих и вновь разрабатываемых приложений;
- функционирование в неоднородной среде на нескольких аппаратных платформах;
- разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;
- существенная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков, и, с другой стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИС.

Для успешной реализации проекта объект проектирования (ИС) должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких

систем. В 70-х и 80-х годах при разработке ИС достаточно широко применялась структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания ИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей ИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако, широкое применение этой методологии и следование ее рекомендациям при разработке конкретных ИС встречалось достаточно редко, поскольку при неавтоматизированной (ручной) разработке это практически невозможно. Действительно, вручную очень трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость, и тем более изменить. Если все же удается создать строгую систему проектных документов, то ее переработка при появлении серьезных изменений практически неосуществима. Ручная разработка обычно порождала следующие проблемы:

- неадекватная спецификация требований;
- неспособность обнаруживать ошибки в проектных решениях;
- низкое качество документации, снижающее эксплуатационные качества;
- затяжной цикл и неудовлетворительные результаты тестирования.

С другой стороны, разработчики ИС исторически всегда стояли последними в ряду тех, кто использовал компьютерные технологии для повышения качества, надежности и производительности в своей собственной работе (феномен "сапожника без сапог").

Общая характеристика и классификация Case-средств. Технология внедрения Case-средств. Характеристики Case-средств

Перечисленные факторы способствовали появлению программно-технологических средств специального класса - CASE-средств, реализующих CASE-технологию создания и сопровождения ИС. Термин CASE (Computer Aided Software Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС. Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т.д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

- подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;
- широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;

- внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Согласно обзору передовых технологий (Survey of Advanced Technology), составленному фирмой Systems Development Inc. в 1996 г. по результатам анкетирования более 1000 американских фирм, CASE-технология в настоящее время попала в разряд наиболее стабильных информационных технологий (ее использовала половина всех опрошенных пользователей более чем в трети своих проектов, из них 85% завершились успешно). Однако, несмотря на все потенциальные возможности CASE-средств, существует множество примеров их неудачного внедрения, в результате которых CASE-средства становятся "полочным" ПО (shelfware). В связи с этим необходимо отметить следующее:

- CASE-средства не обязательно дают немедленный эффект; он может быть получен только спустя какое-то время;
- реальные затраты на внедрение CASE-средств обычно намного превышают затраты на их приобретение;
- CASE-средства обеспечивают возможности для получения существенной выгоды только после успешного завершения процесса их внедрения.

Ввиду разнообразной природы CASE-средств было бы ошибочно делать какие-либо безоговорочные утверждения относительно реального удовлетворения тех или иных ожиданий от их внедрения. Можно перечислить следующие факторы, усложняющие определение возможного эффекта от использования CASE-средств:

- широкое разнообразие качества и возможностей CASE-средств;
- относительно небольшое время использования CASE-средств в различных организациях и недостаток опыта их применения;
- широкое разнообразие в практике внедрения различных организаций;
  отсутствие детальных метрик и данных для уже выполненных и текущих проектов;
- широкий диапазон предметных областей проектов;
- различная степень интеграции CASE-средств в различных проектах.

Вследствие этих сложностей доступная информация о реальных внедрениях крайне ограничена и противоречива. Она зависит от типа средств, характеристик проектов, уровня сопровождения и опыта пользователей. Некоторые аналитики полагают, что реальная выгода от использования некоторых типов CASE-средств может быть получена только после одно- или двухлетнего опыта. Другие полагают, что воздействие может реально проявиться в фазе эксплуатации жизненного цикла ИС, когда технологические улучшения могут привести к снижению эксплуатационных затрат.

Для успешного внедрения CASE-средств организация должна обладать следующими качествами:

Технология. Понимание ограниченности существующих возможностей и способность принять новую технологию;

Культура. Готовность к внедрению новых процессов и взаимоотношений между разработчиками и пользователями;

Управление. Четкое руководство и организованность по отношению к наиболее важным этапам и процессам внедрения.

Если организация не обладает хотя бы одним из перечисленных качеств, то внедрение CASE-средств может закончиться неудачей независимо от степени тщательности следования различным рекомендациям по внедрению. Для того, чтобы принять взвешенное решение относительно инвестиций в CASE-технологию, пользователи вынуждены производить оценку отдельных CASE-средств, опираясь на неполные и противоречивые данные. Эта проблема зачастую усугубляется недостаточным знанием всех возможных "подводных камней" использования CASE-средств. Среди наиболее важных проблем выделяются следующие:

- достоверная оценка отдачи от инвестиций в CASE-средства затруднительна ввиду отсутствия приемлемых метрик и данных по проектам и процессам разработки ПО;
- внедрение CASE-средств может представлять собой достаточно длительный процесс и может не принести немедленной отдачи. Возможно даже краткосрочное снижение продуктивности в результате усилий, затрачиваемых на внедрение. Вследствие этого руководство организации-пользователя может утратить интерес к CASE-средствам и прекратить поддержку их внедрения;
- отсутствие полного соответствия между теми процессами и методами, которые поддерживаются CASE-средствами, и теми, которые используются в данной организации, может привести к дополнительным трудностям;
- CASE-средства зачастую трудно использовать в комплексе с другими подобными средствами. Это объясняется как различными парадигмами, поддерживаемыми различными средствами, так и проблемами передачи данных и управления от одного средства к другому;
- некоторые CASE-средства требуют слишком много усилий для того, чтобы оправдать их использование в небольшом проекте, при этом, тем не менее, можно извлечь выгоду из той дисциплины, к которой обязывает их применение;
- негативное отношение персонала к внедрению новой CASE-технологии может быть главной причиной провала проекта.

Пользователи CASE-средств должны быть готовы к необходимости долгосрочных затрат на эксплуатацию, частому появлению новых версий и возможному быстрому моральному старению средств, а также постоянным затратам на обучение и повышение квалификации персонала.

Несмотря на все высказанные предостережения и некоторый пессимизм, грамотный и разумный подход к использованию CASE-средств может преодолеть все перечисленные трудности. Успешное внедрение CASE-средств должно обеспечить такие выгоды как:

- высокий уровень технологической поддержки процессов разработки и сопровождения ПО;
- положительное воздействие на некоторые или все из перечисленных факторов:
  производительность, качество продукции, соблюдение
  стандартов, документирование;
  приемлемый уровень отдачи от инвестиций в CASE-средства.

Методологии, технологии и инструментальные средства проектирования (CASE-средства) составляют основу проекта любой ИС. Методология реализуется через конкретные технологии и поддерживающие их стандарты, методики и инструментальные средства, которые обеспечивают выполнение процессов жизненного цикла.

Технология проектирования определяется как совокупность трех составляющих:

- пошаговой процедуры, определяющей последовательность технологических операций проектирования;
- критериев и правил, используемых для оценки результатов выполнения технологических операций;

- нотаций (графических и текстовых средств), используемых для описания проектируемой системы.

Технологические инструкции, составляющие основное содержание технологии, должны состоять из описания последовательности технологических операций, условий, в зависимости от которых выполняется та или иная операция, и описаний самих операций.

Технология проектирования, разработки и сопровождения ИС должна удовлетворять следующим общим требованиям:

- технология должна поддерживать полный жизненный цикл программного обеспечения;
- технология должна обеспечивать гарантированное достижение целей разработки ИС с заданным качеством и в установленное время;
- технология должна обеспечивать возможность выполнения крупных проектов в виде подсистем (т.е. возможность декомпозиции проекта на составные части, разрабатываемые группами исполнителей ограниченной численности с последующей интеграцией составных частей). Опыт разработки крупных ИС показывает, что для повышения эффективности работ необходимо разбить проект на отдельные слабо связанные по данным и функциям подсистемы. Реализация подсистем должна выполняться отдельными группами специалистов. При этом необходимо обеспечить координацию ведения общего проекта и исключить дублирование результатов работ каждой проектной группы, которое может возникнуть в силу наличия общих данных и функций;
- технология должна обеспечивать возможность ведения работ по проектированию отдельных подсистем небольшими группами (3-7 человек). Это обусловлено принципами управляемости коллектива и повышения производительности за счет минимизации числа внешних связей;
- технология должна обеспечивать минимальное время получения работоспособной ИС. Речь идет не о сроках готовности всей ИС, а о сроках реализации отдельных подсистем. Реализация ИС в целом в короткие сроки может потребовать привлечения большого числа разработчиков, при этом эффект может оказаться ниже, чем при реализации в более короткие сроки отдельных подсистем меньшим числом разработчиков. Практика показывает, что даже при наличии полностью завершенного проекта, внедрение идет последовательно по отдельным подсистемам;
- технология должна предусматривать возможность управления конфигурацией проекта, ведения версий проекта и его составляющих, возможность автоматического выпуска проектной документации и синхронизацию ее версий с версиями проекта;
- технология должна обеспечивать независимость выполняемых проектных решений от средств реализации ИС (систем управления базами данных (СУБД), операционных систем, языков и систем программирования);
- технология должна быть поддержана комплексом согласованных CASE-средств, обеспечивающих автоматизацию процессов, выполняемых на всех стадиях жизненного цикла.

Реальное применение любой технологии проектирования, разработки и сопровождения ИС в конкретной организации и конкретном проекте невозможно без выработки ряда стандартов (правил, соглашений), которые должны соблюдаться всеми участниками проекта. К таким стандартам относятся следующие:

- стандарт проектирования;
- стандарт оформления проектной документации;
- стандарт пользовательского интерфейса.

Стандарт проектирования должен устанавливать:

- набор необходимых моделей (диаграмм) на каждой стадии проектирования и степень их детализации;

- правила фиксации проектных решений на диаграммах, в том числе: правила именования объектов (включая соглашения по терминологии), набор атрибутов для всех объектов и правила их заполнения на каждой стадии, правила оформления диаграмм, включая требования к форме и размерам объектов, и т. д.;
- требования к конфигурации рабочих мест разработчиков, включая настройки операционной системы, настройки CASE-средств, общие настройки проекта и т. д.;
- механизм обеспечения совместной работы над проектом, в том числе: правила интеграции подсистем проекта, правила поддержания проекта в одинаковом для всех разработчиков состоянии (регламент обмена проектной информацией, механизм фиксации общих объектов и т.д.), правила проверки проектных решений на непротиворечивость и т. д.

Стандарт оформления проектной документации должен устанавливать:

- комплектность, состав и структуру документации на каждой стадии проектирования;
- требования к ее оформлению (включая требования к содержанию разделов, подразделов, пунктов, таблиц и т.д.), правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков для каждой стадии;
- требования к настройке издательской системы, используемой в качестве встроенного средства подготовки документации;
- требования к настройке CASE-средств для обеспечения подготовки документации в соответствии с установленными требованиями.

Стандарт интерфейса пользователя должен устанавливать:

- правила оформления экранов (шрифты и цветовая палитра), состав и расположение окон и элементов управления;
- правила использования клавиатуры и мыши;
- правила оформления текстов помощи;
- перечень стандартных сообщений;

правила обработки реакции пользователя.

Вопросы для самопроверки.

1. Каковы особенности современных информационных систем?
2. Какие проблемы возникают при ручной разработке проектов?
3. Какие факторы способствовали появлению CASE-технологий?
4. Что необходимо для внедрения CASE-средств?
5. Какие проблемы возникают при внедрении CASE-средств?
6. Что устанавливает стандарт проектирования?