Глава 10

Проектирование программных систем

Существует два вида проектирования программных систем:

1. Архитектурное – проектирование определяется как создание общей структуры системы
2. Детализированное – проектирование определяется как конкретизация и реализация модулей, операций, отношений. На этой же стадии дорабатывается структура системы.

10.1. Цели проектирования программных систем

В контексте инженерного программирования цель проектирования можно определить как создание программной системы, которая удовлетворяет:

- Данным функциональным требованиям в их формальной или неформальной форме;
- Явным и неявным требованиям по эксплуатационным качествам и ресурсопотреблению;
- Явным и неявным требованиям дизайна;
- Требованиям к самому процессу разработки, таким, например, как его стоимость и продолжительность.

- *Программное обеспечение* — это совокупность программ, процедур работы и соответствующей документации для информационной системы.
- *Проектирование программных систем* — это такое применение естественных и математических наук, в результате которого потенциальные возможности технических средств реализуются на пользу человеку с помощью машинных программ, организационных процедур и соответствующей документации.

Современный уровень программного обеспечения информационных систем предъявляет существенные требования к проектированию:

- Повысить производительность труда при разработке программного обеспечения;
- повысить эффективность сопровождения программного обеспечения, так как оно составляет около половины стоимости.

Также требования к проектированию состоят в необходимости разработки и сопровождения такого программного обеспечения, которое гарантирует следующие свойства информационных систем:

- исключительную надежность;
- удобство в использовании;
- естественность;
- проверяемость:
- труднодоступностъ для злоупотреблений:
- ставку на человека, а не на машину.

Эффективность проектирования основывается на осуществлении двух основных подцелей:

- получения качественного программного изделия:
- реализации эффективного процесса разработки и сопровождения программного обеспечения.

Каждая из этих подцелей состоит из следующих трех

- Учета человеческих факторов
- управления ресурсами;
- программотехники (технология программирования).

Эффективность проектирования программных систем представляется возможным оценить, исходя из следующих групп факторов:

- Качество программного изделия
- Эффективность процесса разработки программной системы

Три квалификационные категории пользователей:

- К первому уровню относятся разработчики программного обеспечения — специалисты в области применения аппаратно-программных комплексов, способные разрабатывать базовые методы, средства и оснащение программных систем
- Ко второму уровню — те, кто хорошо знает тонкости построения системы и может ее модифицировать, то есть прикладные программисты
- К третьему относятся пользователи, работающие в системе с помощью ориентированного на них языка взаимодействия

10.2. Принципы разработки программных систем

- *Частотный принцип.* Основан на выделении в алгоритмах и в обрабатываемых структурах действий и данных по частоте использования. Для действий, которые часто встречаются при работе программной системы, обеспечиваются условия их быстрого выполнения.
- *Принцип модульности.* Под модулем в общем случае понимают функциональный элемент рассматриваемой системы, имеющий оформление, законченное и выполненное в пределах требований системы, и средства сопряжения с подобными элементами или элементами более высокого уровня данной или другой системы.

- *Принцип функциональной избирательности.* Этот принцип является логическим продолжением частотного и модульного принципов и используется при проектировании программных систем, объем которых существенно превосходит имеющийся объем функционирования. В программной системе выделяется некоторая часть важных модулей, которые постоянно должны быть в состоянии готовности для эффективной организации вычислительного процесса. Этот принцип чаще всего реализуется путем разбиения программной системы на исполняемые модули, между которыми осуществляется обмен информацией или организация динамически загружаемых модулей

- *Принцип генерируемости.* Основное положение этого принципа определяет такой способ исходного представления программной системы, который бы позволял осуществлять настройку на конкретную конфигурацию технических средств, круг решаемых проблем, условия работы пользователя.
- *Принцип функциональной избыточности.* Этот принцип учитывает возможность проведения одной и той же работы различными средствами. Особенно важен учет этого принципа при разработке пользовательского интерфейса для выдачи данных

- *Принцип «по умолчанию».* Применяется для облегчения организации связей с системой как на стадии генерации, так и при работе с уже готовым программным обеспечением. Принцип основан на хранении в системе некоторых базовых описаний структур, модулей, конфигураций и данных, определяющих условия работы с программой.

Общесистемные принципы

- *Принцип включения* предусматривает ситуацию, когда требования к созданию, функционированию и развитию программного обеспечения определяются со стороны более сложной, включающей его в себя системы.
- *Принцип системного единства* состоит в том, что на всех стадиях создания, функционирования и развития программного обеспечения его целостность будет обеспечиваться связями между подсистемами, а также функционированием подсистемы управления

- *Принцип развития* предусматривает в программной системе возможность ее наращивания и совершенствования компонентов и связей между ними. Данный принцип предусматривает также реализацию свойств адаптивности к изменениям как внутренним, так и внешним.
- *Принцип комплексности* заключается в том, что программное обеспечение осуществляет связность обработки информации как для отдельных элементов, так и для всего объема данных в целом на всех стадиях обработки.
- *Принцип информационного единства* состоит в том, что во всех подсистемах, средствах обеспечения и компонентах программного обеспечения используются единые термины, *символы*, условные обозначения и способы представления.

- *Принцип совместимости* состоит в том, что язык, символы, коды и средства обеспечения программных подсистем информационной системы согласованы, обеспечивают совместное функционирование всех его подсистем и сохраняют открытой структуру системы в целом.
- *Принцип инвариантности* предопределяет, что подсистемы и компоненты программного обеспечения инвариантны к обрабатываемой информации, то есть являются универсальными или типовыми.

10.3.1. Общие требования к методологии и технологии

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), используемых для описания проектируемой системы.

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

- технология должна поддерживать полный жизненный цикл разработки;
- 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
- технология должна обеспечивать возможность ведения работ по проектированию отдельных подсистем небольшими группами (3-7 человек).

- технология должна обеспечивать минимальное время получения работоспособного программного обеспечения информационной системы.
- 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 программного обеспечения

Виды стандартов:

- Стандарт проектирования

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

  - правила фиксации проектных решений на диаграммах, в том числе правила наименования объектов

  -требования к конфигурации рабочих мест разработчиков

  -механизм обеспечения совместной работы над проектом, в том числе правила интеграции подсистем проекта, правила поддержания проекта в одинаковом для всех разработчиков состоянии

- Стандарт оформления проектной документации:

  - комплектность, состав и структуру документации на каждой стадии проектирования;

  - требования к се 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-срсдств 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.

10.3.2. Методология RAD

Методология быстрой разработки приложений RAD (Rapid Application Development) - процесс разработки программного обеспечения, содержащий 3 элемента:

- небольшую команду программистов (от 2 до 10 человек);
- короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
- повторяющийся цикл, при котором разработчики, по мере того как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.

Жизненный цикл программного обеспечения по методологии RAD состоит из четырех фаз:

- Анализа и планирования требований:

на фазе анализа и планирования требований пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь

- Проектирования:

на фазе проектирования часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков.

- Построения:

на фазе построения выполняется непосредственно сама быстрая разработка приложения

- Внедрения:

на фазе внедрения производится обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой

Методология RAD неприменима для построении сложных расчетных программ, операционных систем, т.е. программ, требующих написания большого объема (сотни тысяч строк) уникального кода.

Основные принципы методологии RAD:

- разработка приложений итерациями;
- необязательность полного завершения работ на каждом из этапов жизненного цикла;
- обязательное вовлечение пользователей в процесс разработки программного обеспечения;
- необходимое применение CASE-средств, обеспечивающих целостность проекта;
- применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
- необходимое использование генераторов кода;
- использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
- тестирование и развитие проекта, осуществляемое одновременно с разработкой;
- ведение разработки немногочисленной хорошо управляемой командой профессионалов;
- грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

10.3.3. Моделирование программных систем

При моделировании необходимым условием успешного результата является соблюдение принципов декомпозиции, абстракции, иерархии .

- *Абстрагирование.* Абстрагирование является одним из главных способов, используемым для решения сложных задач.

Абстракция — это такие существенные характеристики некоторого объекта, которые отличают его от всех других видов объектов и, таким образом, четко определяют особенности данного объекта с точки зрения дальнейшего рассмотрения и анализа.

Существует определенный спектр абстракций по принципу выделения:

- абстракция сущности объекта — объект представляет собой модель существенных сторон предметной области;
- абстракция поведения — объект состоит из обобщенного множества операций, каждая из которых выполняет определенную функцию;
- абстрагирование в виде виртуальной машины — объект объединяет группы операций, которые либо используются для управления объектом, либо соответствуют функциям нижнего уровня;
- произвольная абстракция — объект включает в себя набор независимых по отношению друг к другу операций. Выбор достаточного множества абстракций (не больше и не меньше целесообразного количества) является основной проблемой проектирования.

- Декомпозиция.

При проектировании программной системы необходимо составлять ее из небольших подсистем, каждую из которых представляется возможным отладить независимо от других.

При проектировании программной системы как вида обеспечения информационных систем возможна реализация двух подходов к декомпозиции.

Первый подход можно определить как декомпозицию программной системы в соответствии со структурой информационной системы.

Вторым подходом к декомпозиции программной системы является декомпозиция на основании выделения тесно связанных и логически единых функций информационной системы, подлежащих реализации в программной системе

- Иерархия.

Значительное упрощение понимания сложных задач в процессе проектирования достигается за счет образования иерархической структуры из абстракций и(или) модулей.

Иерархия – ранжированная или упорядоченная система абстракций.

Формирование иерархической структуры необходимо для понимания взаимосвязей и подчиненности объектов и модулей программной системы.

Этот процесс неразрывно связан с процессами абстрагирования и декомпозиции

10.3.4 Использование формальных спецификаций

Одним из методов выполнения этапов анализа и проек тирования программного обеспечения является использование формальных спецификаций, описывающих абстракции процедуры, данные и итераторы в соответствии с приведенными ниже шаблонами:

Процедурная абстракция.

Это отображение набора значений входных аргументов в выходной набор результатов с возможной модификацией входных значений.

Спецификация процедуры состоит из заголовка и описания функции, выполняемой процедурой.

- *Абстракции данных.*

Абстракции данных — наиболее важный метод в проектировании программ. Выбор правильных структур данных играет решающую роль для создания эффективной программы.

Они позволяют отложить окончательный выбор структур данных до момента, когда эти структуры станут вполне ясны.

- *Абстракции итерации.*

Для работы с набором данных требуется некоторый общий метод итерации, который удобен и эффективен и который сохраняет абстракцию через спецификацию. Итератор обеспечивает эти требования. Он вызывается как процедура, но вместо окончания с выдачей всех результатов имеет много результатов, которые каждый раз выдает по одному.

Системный подход при проектировании представляет собой комплексное, взаимосвязанное, пропорциональное рассмотрение всех факторов, путей и методов решения сложной многофакторной и многовариантной задачи проектирования интерфейса взаимодействия. В отличие от классического инженерно-технического проектирования, при использовании системного подхода учитываются все факторы проектируемой системы — функциональные, психологические, социальные и даже эстетические.

Автоматизация неизбежно влечет за собой осуществление системного подхода.

10.4. Архитектура ИС

Целью пункта является формализация представления архитектуры проектируемой информационной системы. Рассматривается схема Захмана, предназначенная для формирования взгляда на архитектуру информационной системы с точки зрения участников ее разработки. Предлагается аналогичная схема, в которой информационная система рассматривается в терминах различных *подходов* к моделированию бизнеса. Схема позволяет осознать место каждой из таких подходов в создании информационной системы, а также определить границы деятельности каждого из ее создателей.

10.4.1. Моделирование бизнеса и архитектура ИС

Рассмотрим таблицу, иллюстрирующую основные методы технологии моделирования на разных стадиях развития системы:

Таблица использования методов моделирования

Методы	Уровень бизнеса	Систем-ный уровень	Програм-мно-про-цедурный уровень
Функциональная иерархия	о	о	о
Анализ состояний	н	о	н
Диаграммы потоков данных	н	о	н
Событийное моделирование	н	о	о
Функциональная логика	о	о	н

Примечание:

о — обязательное использование;

н — необязательное, но возможное использование.

Схема Захмана. В 1987 г. Джон Захман опубликовал полезную схему развития архитектуры информационной системы. Схема создает контекст для описания различных представлений архитектуры разрабатываемой системы. Эти представления соответствуют тому, как видят систему ее заказчик, проектировщик и разработчик, причем, в разрезе трех выбранных аспектов. Эти три аспекта — данные, функции и сетевая структура. В схеме Захмана строке соответствует точка зрения какого-либо участника проекта по созданию системы. Аспекты представлены в схеме колонками.

Архитектурное представление — это ячейка таблицы, соответствующая пересечению выбранного столбца и выбранной строки. Например, с точки зрения разработчика

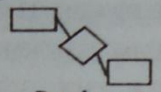
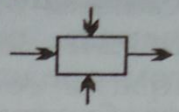
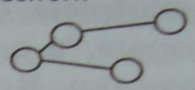
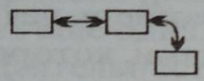
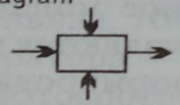
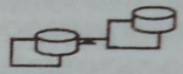
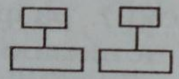
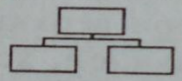
	DATA	FUNCTION	NETWORK
OBJECTIVES/ SCOPE	List of Things important to the Business  Entity = Class of Business Things	List of Processes the Business Performs  Processes = Class of Business Process	List of Location in Which the Business Operates
MODEL OF THE BUSINESS	e.g., Entity Relationship Diagram  Ent. = Business Entity Rel. = Data Rel.	e.g., Data Flow Diagram  Proc. = Bus. Process I/O = Bus. Resource (including info.)	e.g., Logistics Network  Node = Business Unit Link = Bus. Relationship (Org. Product, info.)
MODEL OF THE INFORMATION SYSTEM	e.g., Data Model  Ent. = Data Entity Rel. = Data Rel.	e.g., Function Diagram  Proc. = Appl. Functions I/O = User Views (set of Data Elements)	e.g., distributed Sys. Arch  Node = I/S Function (Process, Storage, etc.) Link = Line Char.
TECHNOLOGY MODEL	e.g., Data Design  Ent. = Segment/Row Rel. = Pointer/Key	e.g., Structure Chart  Proc. = Computer Function I/O = Screen/Devise Formats	e.g., System Arch  Node = Hardware/Sys. Software Link = Line Specs.
DETAILED REPRESENTATION	e.g., Data Design Description  Ent. = Fields Rel. = Addresses	e.g., Program Description  Proc. = Language Stmts. I/O = Control Blocks	e.g., Network Arcitecture  Node = Addresses Link = Protocols
FUNCTIONING SYSTEM	e.g., DATA	e.g., FUNCTION	e.g., COMMUNICA-TIONS

Рис. 10.2. *Схема Захмана*

10.4.2 Конфигурация и архитектура ИС

Архитектура информационной системы — концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы.

При декомпозиции системы на функциональные фрагменты можно выделить следующие компоненты:

- презентационная логика (Presentation Layer — PL);
- бизнес-логика (Business Layer — BL);
- логика доступа к ресурсам (Access Layer — AL).

- ***Клиент-серверная технология*** — это стиль работы приложений, где клиентский процесс запрашивает обслуживание у процесса сервера. Проще говоря, сервер — это программа, предоставляющая доступ к каким-либо услугам, например, к электронной почте, файлам, Web или данным (в качестве сервера баз данных). Клиент — это приложение, которое соединяется с сервером, чтобы воспользоваться предоставляемыми им услугами.

Модели клиент-серверного взаимодействия:

- ***«Толстый» клиент***. Наиболее часто встречающийся вариант реализации архитектуры клиент-сервер в уже внедренных и активно используемых системах. Такая модель подразумевает объединение в клиентском приложении как PL, так и BL. Серверная часть, при описанном подходе, представляет собой сервер баз данных, реализующий AL.
- ***«Тонкий» клиент.*** Модель, начинающая активно использоваться в корпоративной среде в связи с распространением Internet-технологий и, в первую очередь, Web-браузеров. В этом случае клиентское приложение обеспечивает реализацию PL, а сервер объединяет BL и AL.
- ***Сервер бизнес-логики***. Модель с физически выделенным в отдельное приложение блоком BL.

Первоначально системы базировались на классической двухуровневой клиент-серверной архитектуре (Two-tier architecture). Минусов у данной архитектуры было огромное множество. Решение данной проблемы не заставило себя ждать и впоследствии разработчиками была создана трехуровневая клиент-серверная архитектура(Three-tier architecture).

Несомненными достоинствами стали: возможность подключения различных баз данных, снизились требования к клиентским машинам. Однако повышенные требования к пропускной возможности сети никуда не исчезли, накладывая больше ограничения на использования таких систем в сетях с неустойчивой связью и пропускной способностью.

Благодаря работе программистов стала применяться распределенная архитектура, которая позволяет решать задачи, которые формулируются недостатками многоуровневой системы. При таком способе построения архитектуры ИС более 95% данных могут быть размещены на оном персональном компьютере. Это позволяет не только снизить нагрузку на сеть но и в связи с уменьшением трафика снизить расходы на эксплуатацию. Каждый АРМ(автоматизированное рабочее место) содержит только ту информацию с которой должен работать, а актуальность поддерживается благодаря непрерывному обмену сообщениями с другими АРМами.