Лекция 12-14

Темы:

12 Основы проектирования ПС.

13 Роли и участники проекта.

14 Ключевые вопросы проектирования .Параллелизм. Асинхронные агенты.

Проектирование программной системы. Этот процесс можно определить как процесс создания *проекта* программной системы (ПС) — набора схем, диаграмм, технических заданий и другой документации, содержащих описание разрабатываемого ПП в объеме, достаточном для его конструирования. Проект необходим для того, чтобы все его участники понимали цель разработки, какой продукт и с какими характеристиками будет создан в результате их деятельности.

Целью проектирования является определение внутренних свойств системы и детализации ее внешних (видимых) свойств на основе выданных заказчиком требований к ПС (исходных условий задачи).

Процесс проектирования — инженерная деятельность в рамках ЖЦ ПП, в рамках которой анализируются требования и создается описание внутренней структуры ПС, которое является основой для его дальнейшего конструирования.

В зависимости от сложности создаваемого ПП процесс проектирования может обеспечиваться как «ручным» проектированием, так и различными средствами его автоматизации.

В процессе проектирования ПС также могут использоваться различные графические средства: блок-схемы, ER-диаграммы, UML-диаграммы, а также *макеты*.

Блок-схемы — распространенный тип схем (графических моделей), описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности их выполнения.

ER-диаграммы используются при высокоуровневом проектировании БД.

UML-диаграммы (UML, Unified Modeling Language — унифицированный язык моделирования) — графическое описание процесса моделирования в области разработки ПО.

Проектированию обычно подлежат архитектура ПО, устройство компонентов ПО, пользовательские интерфейсы.

Первоначально программа рассматривается как черный ящик. Ход процесса проектирования и его результаты зависят не только от состава требований, но и от выбранной модели процесса, опыта проектировщика.

Проектирование является сложным и, как правило, трудоемким процессом, требующим большого опыта и знаний, которые позволяют находить компромиссные решения при разработке требуемого программного обеспечения.

Роли участников процесса проектирования. При проектировании должны быть задействованы определенные люди, которые примут участие в анализе и разработке программного проекта.

Участникам проекта принято назначать роли — функции, которые они будут выполнять в проекте. В зависимости от квалификации и величины проектной команды один человек может совмещать различные роли (например, если программу пишет один разработчик, то он может совместить их все). Для осуществления процесса проектирования выделяют следующие основные роли:

1. заказчик;
2. руководитель проекта;
3. системный администратор;
4. администратор базы данных;
5. системный архитектор;
6. архитектор базы данных;
7. бизнес-аналитик;
8. аналитик (системный аналитик);
9. тестировщик.

Заказчиком является лицо, которому нужно разрабатываемое ПО. В качестве заказчика может выступать один человек, группа людей либо какое-либо предприятие, юридическое лицо. Заказчик определяет требования к разрабатываемому программному продукту, которые влияют на то, каким будет конечный результат разработки.

Руководитель проекта управляет процессом разработки ПП. Он отвечает за выполнение всех задач в срок перед заказчиком, обеспечивает взаимодействие всех участников программного проекта, контролирует процесс выполнения.

Системный администратор создает условия для непрерывной работы над разрабатываемым программным продуктом, следит за работоспособностью серверов, выявляет и устраняет неполадки. При проектировании он выступает в качестве

консультанта по требованиям к аппаратному обеспечению (серверам, рабочим компьютерам, мощности процессоров, необходимых для работы, объему оперативной памяти и т.д.), необходимому для нормального функционирования программного продукта.

Администратор базы данных обеспечивает непрерывность работы сервера базы данных, контролирует его работу. При проектировании выступает в качестве консультанта по требованию к серверу базы данных, а также к системе управления базой данных, необходимой для программного продукта.

Системный архитектор проектирует архитектуру всего программного продукта в целом и более детально производит проектирование самого приложения, не вдаваясь в подробности проектирования базы данных.

Архитектор базы данных проектирует укрупненную структуру базы данных. Более детальная проработка архитектуры базы данных производится на этапе конструирования ПО.

Бизнес-аналитик описывает требуемое поведение системы с точки зрения конечных пользователей. Например, с точки зрения пользователя, покупка выбранного товара будет заключаться в щелчке левой кнопкой мыши по кнопке «Купить» и заполнении данных банковской карты. С точки зрения разработчика, данное действие будет заключаться в получении события щелчка левой кнопкой мыши по кнопке «ЫпВиу», вызове формы «бгтВиу» и проверке заполненных полей.

Аналитик (системный аналитик) пишет технические задания для разработчиков. В зависимости от проекта технические задания могут быть различной степени детализации — от подробных, по которым разработчикам остается только закодировать написанное ТЗ на выбранном языке программирования, до общих, которые требуют дополнительного проектирования (проработки структуры таблиц, классов, методов и пр.).

Тестировщик производит тестирование ПП. При проектировании его необходимо ознакомить с подготовленной документацией по программному продукту для подготовки к процессу тестирования. В настоящее время существуют определенные технологии разработки программных проектов, которые основываются на тестах. При проектировании пишутся тесты для разрабатываемого функционала, а при разработке систему конструируют в соответствии с написанными тестами.

Для каждого программного продукта набор ролей будет различным. Например, при разработке интернет-магазина могут потребоваться все перечисленные роли, а при

разработке мобильного приложения может быть достаточно заказчика, руководителя проекта, системного архитектора и системного аналитика.

Ключевые вопросы проектирования

На этапе проектирования программного продукта необходимо определиться с ключевыми вопросами, которые оказывают существенное влияние на архитектуру программного продукта.

Параллелизм. *Параллельные вычисления* — способ организации компьютерных вычислений, при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно (воспринимаемых пользователем как одновременные).

Существуют различные способы реализации параллельных вычислений.

Например, каждый вычислительный процесс может быть реализован в виде процесса операционной системы (ОС) либо вычислительные процессы могут представлять собой набор потоков выполнения внутри одного процесса ОС.

Параллельные программы могут физически исполняться либо последовательно на единственном процессоре, осуществляя по очереди шаги выполнения каждого вычислительного процесса, либо параллельно, выделяя каждому вычислительному процессу один или несколько процессоров (находящихся рядом или объединенных в компьютерную сеть).

Основной сложностью при проектировании параллельных программ является

1. обеспечение правильной последовательности взаимодействий между различными вычислительными процессами, а также координация ресурсов, разделяемых между процессами.

Например, для интернет-магазина по продаже сотовых телефонов можно выделить следующие параллельные процессы:

1) формирование и выдача страниц по запросам пользователей (обеспечивается на уровне архитектуры Web-сервера);
2) выдача запросов базой данных (обеспечивается на уровне архитектуры базы данных).

2. ряд действий нельзя выполнять параллельно.

Например, при регистрации, когда резервируется уникальный код доступа к учетной записи, нельзя одновременно давать регистрироваться двум пользователям, так как это может вызвать конфликт кодов. Функции обработки заказов также нельзя делать параллельно, так как товар на складе может закончиться при обработке одного из заказов.

Асинхронные агенты. «Тяжелые» (продолжительные) задачи необходимо выделять в отдельные потоки, что позволяет разгрузить потоки, обрабатывающие сообщения от графического интерфейса. Например, это позволяет отделить прорисовку графической информации от вычислений, занимающих большой объем времени. К категории задач, которые необходимо выделять на обработку в отдельные потоки, можно отнести следующие: печать, лингвистические проверки, КОМПИЛЯЦИЮ и т.д.

Основным принципом для взаимодействия данных процессов является асинхронный обмен сообщениями. В случае с графическим интерфейсом данный подход является наиболее предпочтительным, так как работа графического интерфейса основана на сообщениях.

Для WEB-приложений выделить асинхронные события несколько трудней, так как обработка графической информации производится в браузерах клиентов.

Но для них также возможно создание асинхронных событий на основе использования специальных возможностей встроенного языка разметки.

Например, для интернет-магазина сотовых телефонов в качестве асинхронного события можно выделить автоматическое обновление статуса заказов на страницах без их перерисовки.

Контроль и обработка событий. *Событийно-ориентированная архитектура* позволяет реализовать механизм работы приложения не на основе прямого вызова методов классов, а на основе реагирования на те или иные события. Таким образом можно уменьшить связность в системе.

Изначально событийность была присуща графическому интерфейсу, где вся работа основывалась на передаче сообщений. Событийность возможна как асинхронное решение для обмена информацией в многослойных приложениях (когда архитектура делится на несколько слоев), особенно это касается приложений, слои которых разделены по разным серверам, так как в этом случае прямой вызов методов невозможен.

Обеспечение отказоустойчивости. Одной из задач проектирования является обеспечение корректной обработки ошибок. В частности, при проектировании необходимо описать, как обеспечить корректную работу системы, даже если произошел сбой.

Существует два наиболее распространенных метода контроля ошибок.

1. *Исключения* — метод, позволяющий разработчикам получить наиболее подробную информацию по ошибкам и достаточно легко их локализовать,

классифицировать и обрабатывать централизованно. Недостатком этого способа является невысокая скорость об-

работки ошибок, поскольку, чтобы собрать необходимые для исключения данные, необходимо проделать значительный объем вычислений. Поэтому данный метод не рекомендуется использовать при обработке больших потоков информации, когда выдвигаются требования к скорости.

2. *Коды ошибок* — метод, широко использующийся при обработке большого потока данных, когда приоритетом ставится скорость обработки информации. При появлении ошибки функции возвращают лишь ее код, «надеясь» на то, что в вызывающих процедурах есть логика по их обработке. Это менее надежный, но более производительный метод.

Не все языки программирования поддерживают обе возможности работы с ошибками. Например, часть языков баз данных имеют возможность работы только с кодами ошибок.