

Распределенные вычислительные системы

Лекция №2: Эволюция распределенных технологий, метамодель

Алексей В. Бурдаков, к.т.н.
burdakov@usa.net

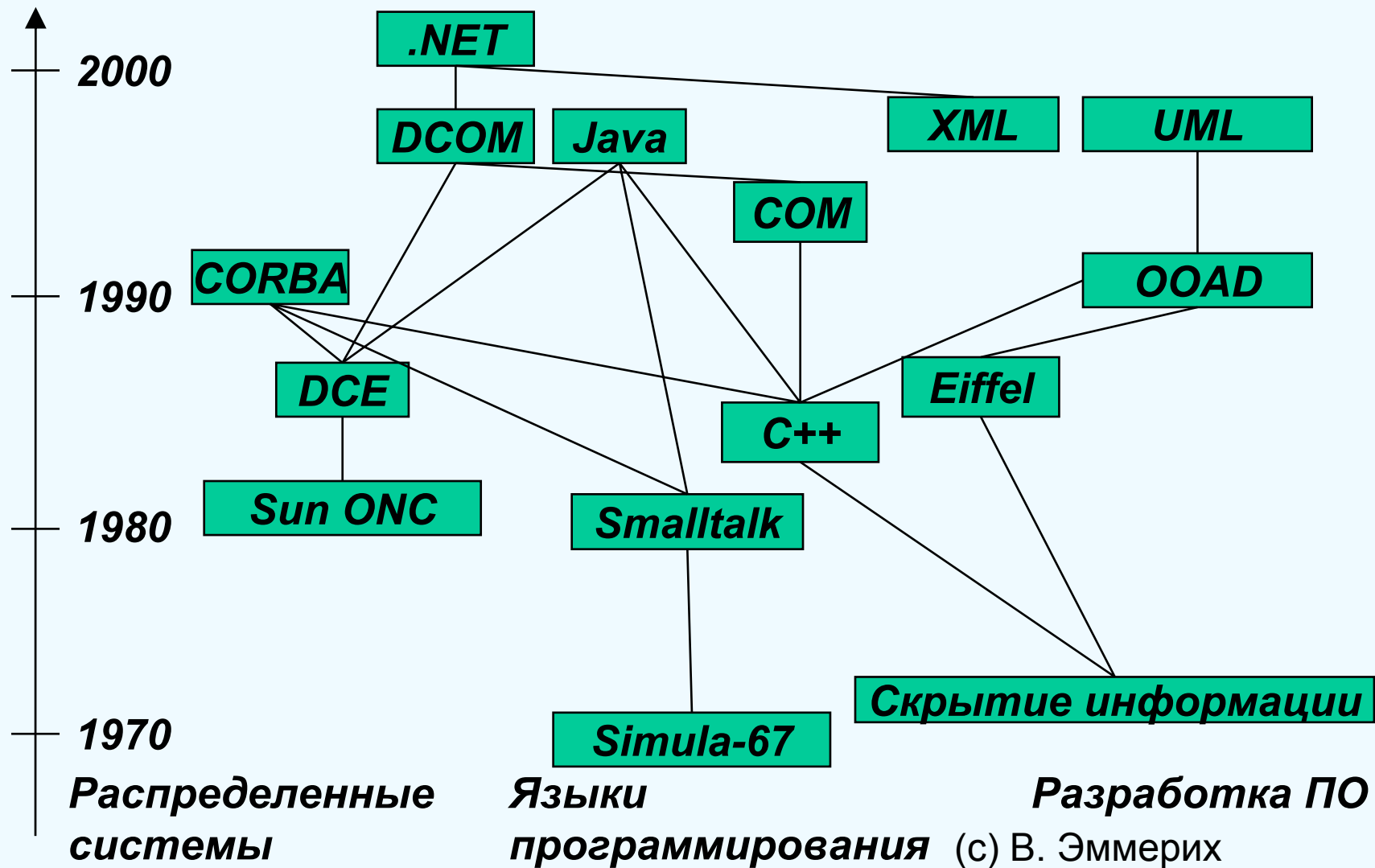
План лекций

№	Дата	Тема
1	04.09	Вводная лекция
2	11.09	Эволюция распределенных технологий, метамодель

План лекции

- Эволюция распределенных технологий
- Мета модель распределенной системы
- Особенности разработки распределенных систем

Эволюция объектной технологии



Эволюция объектной технологии

- **Simula** (1966, Скандинавия) - начало объектной ориентации
 - разработка программ имитационного моделирования,
 - понятие класса как типа,
 - экземпляры могут реагировать на сообщения
- **Скрытие информации** (1972, Д. Парнас)
 - сокрытие структур данных наиболее часто изменяемых при неизменном интерфейсе
 - идея модуля (module)

Эволюция объектной технологии

- **Smalltalk** (конец 70-х, начало 80-х) - для создания нового пользовательского интерфейса в центре Xerox
 - концепция - «все есть объект»,
 - наследование,
 - Полиморфизм
- **C++** (середина 80-х) – Бьярн Строструп вместе с группой исследователей из AT&T Bell Labs
 - строгое супермножество C - приемственность
 - статическое определение типов означающее контроль на стадии компиляции
 - более успешен нежели Smalltalk

Эволюция объектной технологии

- **RPC** разработанная Sun Microsystems как часть архитектуры **ONC** (Open Network Communications - Открытое сетевое взаимодействие):
 - удаленный вызов процедур,
 - описание интерфейсов с помощью специализированного языка,
 - использование клиентских и серверных стабов (stub),
 - передача параметров по сетевым протоколам
- **DCE** (Distributed Computing Environment - Среда распределенных вычислений) - стандарт OSF (Open Software Foundation);
 - стандарт RPC был включен в DCE
 - среди прочего определяет внешнее представление данных (External Data Representation, XDR),
 - служба высокого уровня: 1) имен и 2) безопасности

Эволюция объектной технологии

- **CORBA** – Common Request Broker Architecture – Общая архитектура брокера объектных запросов
 - версия 1.0 принята в 1991 г.
 - удаленный вызов и объектно-ориентированная парадигма
 - брокер объектных запросов – средний слой, обеспечивающий удаленные вызовы, независимость от расположения объектов, аппаратных и программных средств, языков прогр.
- **Java/RMI** (Remote Method Invocation - Удаленный вызов методов)
 - удаленный вызов методов интегрированы в Java
 - компоненты Java могут мигрировать и выполняться на различных платформах
 - нет проблем с неоднородностью, так как «Весь мир - это Java»

Эволюция объектной технологии

- **COM** (Component Object Model - Компонентная модель объектов)
- модель взаимодействия приложений, написанных на разных языках
 - несколько интерфейсов (множественное наследование)
 - Только локальное взаимодействие
- **DCOM** (Distributed Component Object Model - Объектная модель распределенных компонентов)
 - представлена в Windows NT 4.0
 - доступ к удаленным объектам по OSF/RPC
- **.NET** – набор технологий и архитектурных моделей, предложенный Microsoft
 - XML – eXtensible Markup Language
 - SOAP – Simple Object Access Protocol
 - Web-сервисы

Эволюция объектной технологии: сравнение

Программный аспект

- DCE не является объектно-ориентированным, независимым от средств разработки, не поддерживает состояние объектов;
- DCOM зависим от платформы, имеет слабую независимость от языков и не поддерживает состояние объектов.

Коммуникационный аспект

- DCE, DCOM и .NET не поддерживают методы вызовов: асинхронный, отложенный синхронный, передача сообщений;
- DCE кроме того не поддерживает динамических вызовов, а DCOM – свободу перемещения объектов в вычислительной сети.

Эволюция объектной технологии

Инфраструктура:

- DCE определяет намного более слабую инфраструктуру, DCOM не определяет стандарта на инфраструктуру.
- .NET определяют только сервис именования UDDI

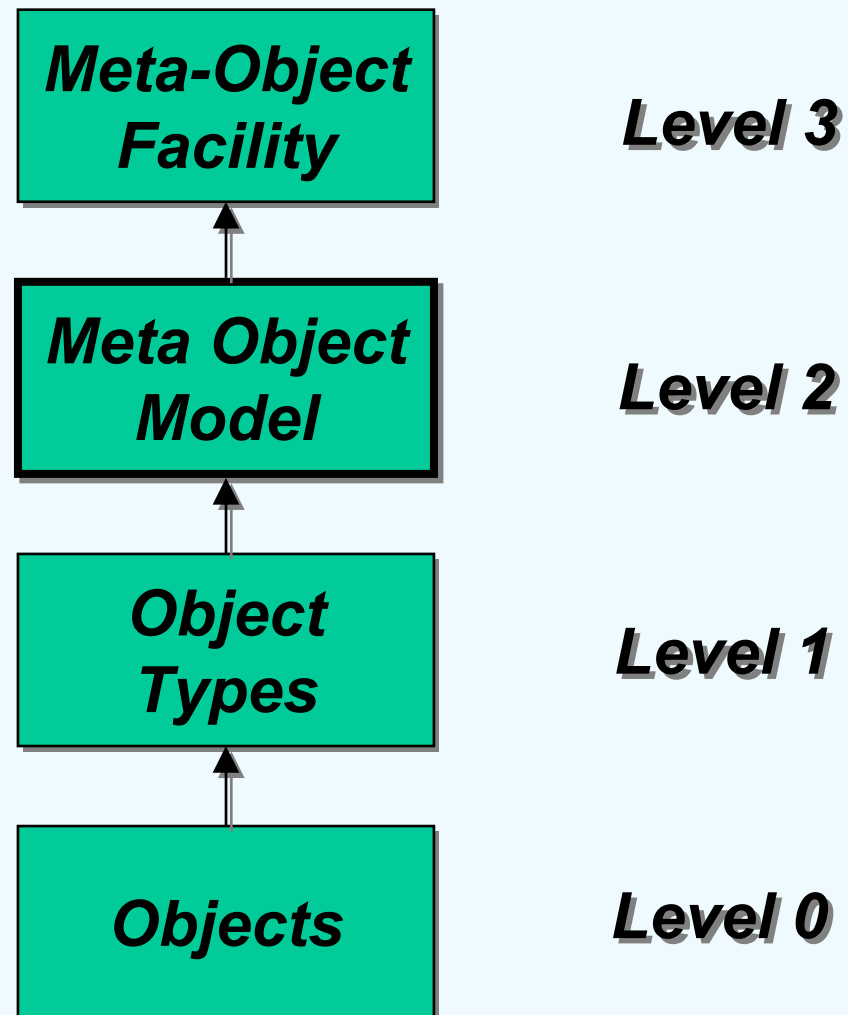
Признание и поддержка индустрией:

- CORBA – наибольшая поддержка.

План лекции

- Эволюция распределенных технологий
- Метамодель распределенной системы
- Особенности разработки распределенных систем

Различные уровни абстракции систем



Метамодель распределенной системы

- Распределенные системы состоят из множества компонентов
- Компоненты являются гетерогенными
- Компоненты взаимодействуют между собой
- Должна быть общая модель компонентов, отражающая:
 - Состояние компонентов
 - Сервисы компонентов
 - Взаимодействие между компонентами

Объект (компонент)

- Реализует функциональность с помощью операций
- Обладает набором атрибутов, которые в совокупности определяют состояние объекта
 - Частные (private) – доступны внутри
 - Общие (public) – доступны из других компонентов
 - Только для чтения – доступны из других компонентов для чтения
- Имеет уникальный объектный идентификатор
 - Присваивается при создании
 - Сохраняется на протяжении всей жизни
- Объектная ссылка – дескриптор объекта
 - Используются для доступа к объектам
 - Может существовать несколько ссылок на данный объект

Объект (компонент)

- Равенство и идентичность:
 - Равные объекты (в соответствии с критерием равенства)
 - Идентичные объекты (имеющие одинаковые идентификаторы)
 - Идентичные объекты равны, но не наоборот
- Может вызывать исключительные события

Типы

- Типы объектов определяют те характеристики, которые являются общими для похожих объектов
 - Задается с помощью интерфейса
 - Определяют набор операций и атрибутов
- Объекты являются экземплярами типов
 - Экземпляр одного или
 - Нескольких типов
 - Связь с типом позволяет получить информацию об атрибутах и операциях

Необъектные типы

- Высокие издержки на передачу объектов (ссылок на объекты)
 - Информация о местоположении
 - Информация о безопасности
- Поддержка простых типов данных (логические значения, символы и числа)
 - Являются значениями
 - Не имеют идентификаторов
 - А) Атомарные
 - Б) Сложные (сконструированные)
 - Записи
 - Массивы
 - Последовательности
 - ...

Заявки

- Объектная заявка – триада из объектной ссылки, имени операции и набора фактических параметров
- Значительно большие издержки по сравнению с обычным вызовом
 - Устранение неоднородности данных
 - Синхронизирование клиента и сервера
 - Установление связи по сети

Исключения

- Отказы происходят
- Отказы должны скрываться промежуточным слоем
- Отказы, которые не могут быть скрыты передаются в прикладные программы в виде исключений (exceptions)
- Исключение – структура данных, предназначенная для хранения информации об отказах
- Исключение генерируется объектом-сервером и передается объекту-клиенту

Исключения

- Исключения
 - 1) Системные (генерируются промежуточным слоем)
 - 2) Типо-зависимые (генерируются прикладным компонентом)

Операции

- Операция определяется
 - Именем
 - Списком входных и выходных параметров
 - Типом возвращаемого значения
 - Списком исключений

Подтипы и множественное наследование

- Подтип содержит (говорят – «наследует») все операции, атрибуты и исключения своего «супертипа»
- Подтип содержит дополнительные элементы, т.е. расширяет определение супертипа
- Объектный тип может наследоваться от более чем одного супертипа образуя множественное наследование
- Преимущества
 - Повторное использование типов
 - Облегчение изменений в иерархической структуре
 - Облегчение восприятия проекта
 - Полиморфизм (вместо типа использование подтипа)

План лекции

- Эволюция распределенных технологий
- Мета модель распределенной системы
- Особенности разработки распределенных систем

Особенности распределенных систем с точки зрения проектировщика

- Ссылки
- Задержки заявки
- Активация/деактивация
- Миграция
- Постоянное хранение
- Одновременный доступ
- Связь
- Безопасность

Ссылки

- Ссылки на объекты в программных модулях на ОО языках программирования (например, С++) являются указателями в памяти
- Ссылки на объекты в распределенных системах в противоположность являются более комплексными:
 - Содержат информацию о размещении
 - Информацию о безопасности
 - Ссылки на объектные типы
- Ссылки на распределенные объекты значительно больше (40 байт для Orbix)

Задержки выполнения запросов

- Локальные вызовы требуют порядка пары сотен наносекунд
- Запрос к объекту требует от 0.1 до 10 миллисекунд
- Интерфейсы в распределенной системе должны быть спроектированы так чтобы снизить время выполнения запросов
 - Снизить частоту обращения
 - Укрупнить выполняемые функции

Активация/деактивация

- Объекты в ОО языках находятся в виртуальной памяти от создания до уничтожения
- В распределенных системах
 - Больше объектов
 - Объекты могут не использоваться на протяжении долгого времени
- Реализации распределенных объектов
 - Переносятся в память при активации
 - Удаляются из памяти при деактивации

Активация/деактивация

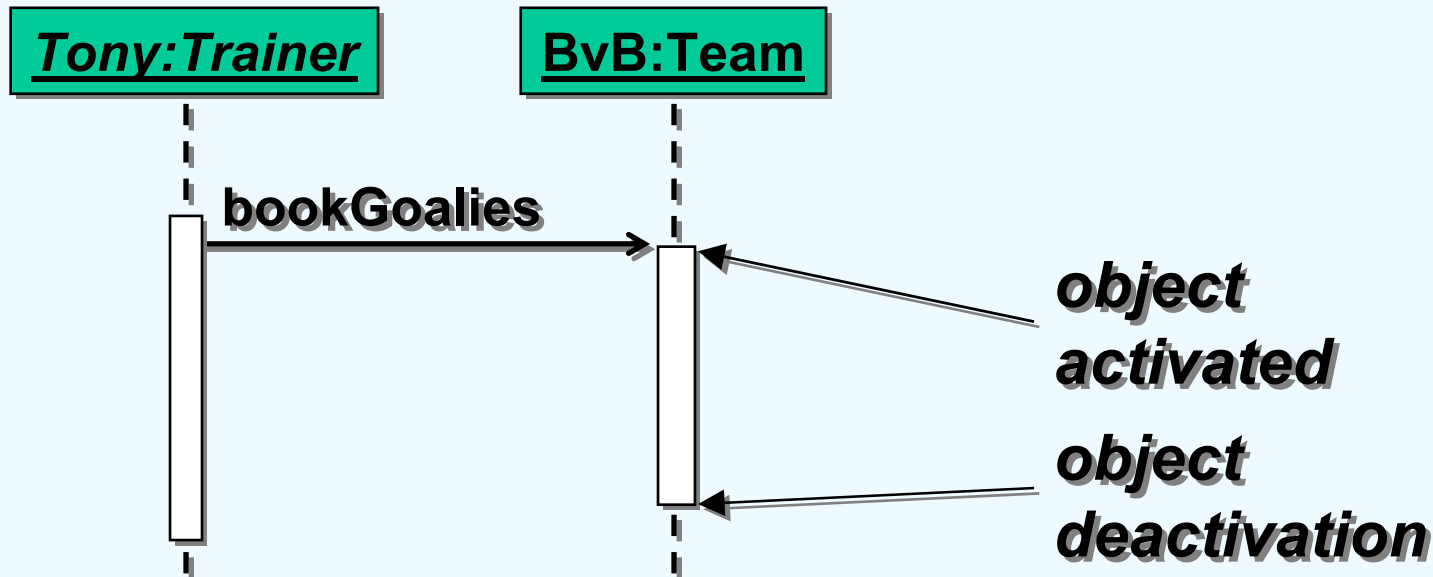


Диаграмма последовательностей
(sequence diagram) UML

Активация/деактивация

- Вопросы, которые должны быть решены:
 - Репозиторий реализаций
 - Связывание объектов и процессов
 - Явная и неявная активация
 - Когда можно деактивировать объекты
 - Как обрабатывать одновременные запросы
- Кто должен решать эти вопросы?
 - Проектировщик
 - Разработчик
 - Администратор
- Как документировать решения?

Постоянное хранение

- Объекты имеющие и не имеющие состояния
- Объекты имеющие состояние должны сохранять его на постоянный носитель между:
 - Деактивацией объекта
 - Активацией объекта
- Может быть достигнуто
 - Экстернализацией на файловую систему
 - Отражением на реляционные БД
 - С помощью объектных БД
- Должно приниматься во внимание в процессе проектирования

Параллельное исполнение

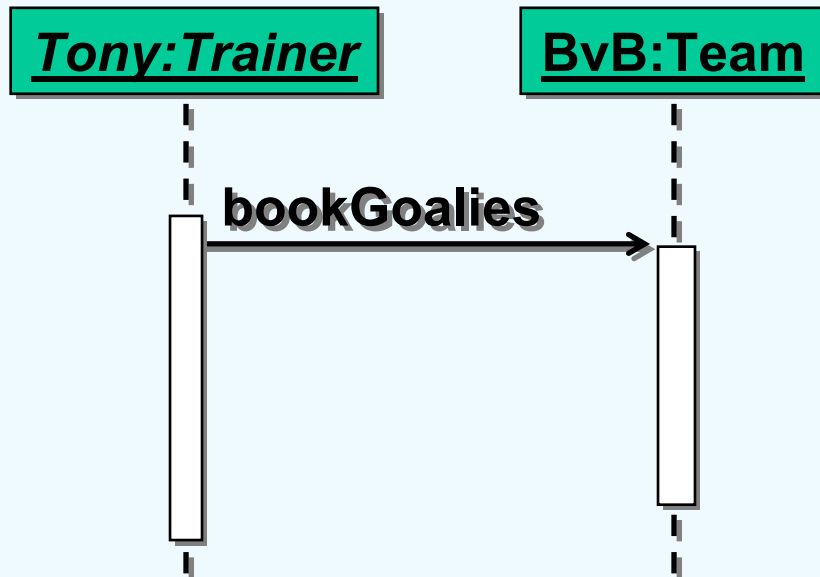
- В нераспределенных системах исполнение в основном последовательное, иногда конкурентное в разных нитях процессов
- Распределенные компоненты выполняются параллельно

Взаимодействие

- В нераспределенных системах вызовы являются синхронными
- Альтернативы для распределенных объектов:
 - Синхронные
 - Односторонние
 - Отложенные синхронные
 - Асинхронные запросы
 - Множественные заявки

Взаимодействие

- Односторонние



- Асинхронные
- Отложенные
- синхронные
- Синхронные

Отказы

- Запросы в распределенных системах имеют большую вероятность отказов
- Клиенты обязаны проверять факт выполнения сервером запросов

Безопасность

- Безопасность в ОО приложениях может выполняться на основе контроля сеансов
- Распределенные объекты:
 - Кто запрашивает выполнение операции?
 - Как мы можем удостовериться, что субъект является именно тем за кого он себя выдает?
 - Как мы примем решение предоставлять или нет субъекту право на выполнение сервиса?
 - Как мы можем неопровержимо доказать, что сервис был предоставлен?

Литература / Internet ИСТОЧНИКИ

- В. Эммерих ***Конструирование распределенных объектов.*** - М.:Мир. - 2002.
- М.Р. Когаловский ***Энциклопедия технологий баз данных.*** - М.: ФС. - 2002.
- Ю.А. Григорьев, А.Д. Плутенко. - ***Жизненный цикл проектирования распределенных баз данных.*** - Благовещенск. - 1999.
- www.omg.org