Распределенные объектные технологии

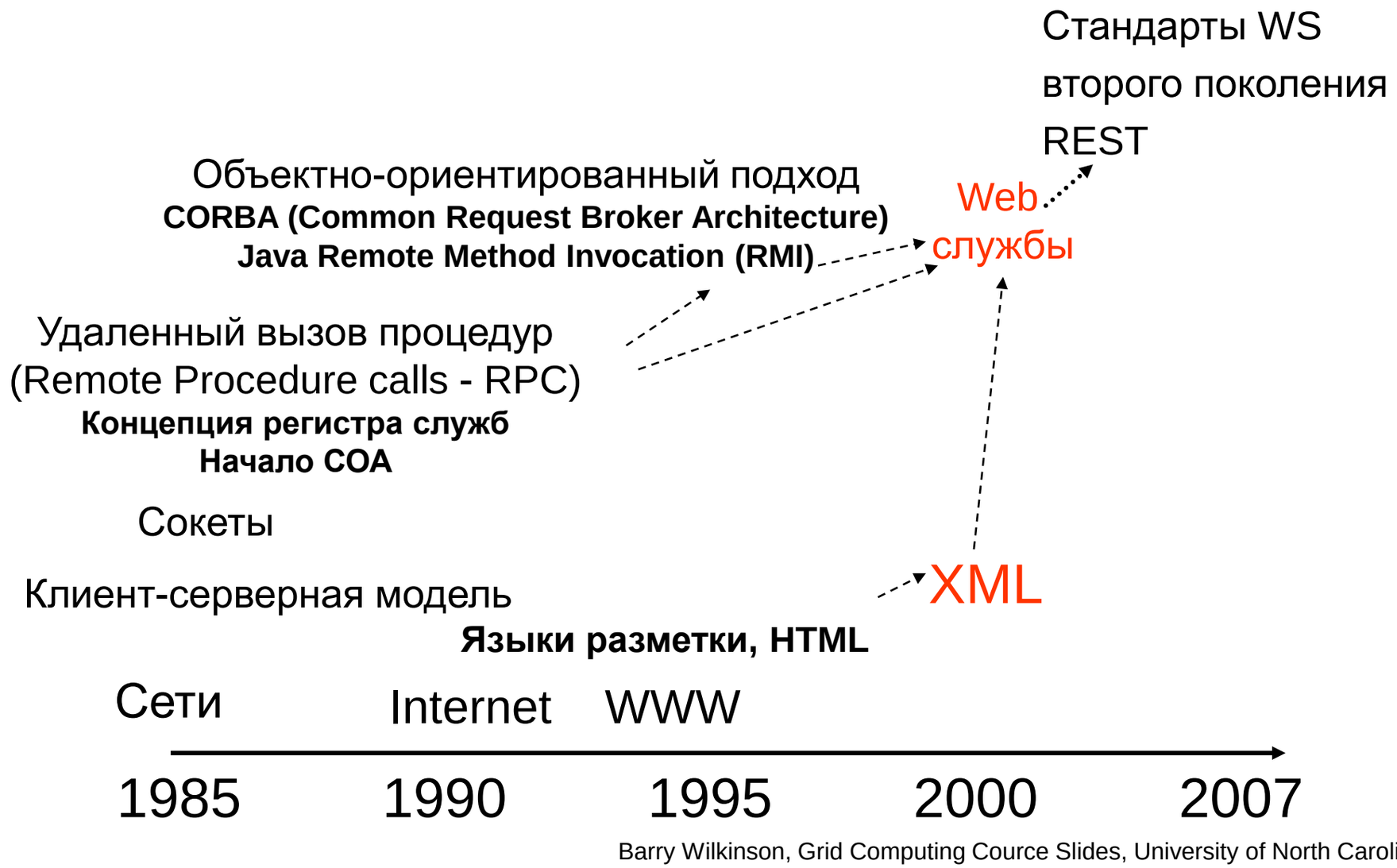
Лекция 2
История РВС; SOA

Радченко Глеб Игоревич, каф. СП, ЮУрГУ

http://dom.susu.ru

История и виды распределенных вычислений

Распределенные вычисления



Barry Wilkinson, Grid Computing Cource Slides, University of North Carolina

Развитие распределенных вычислений и грид

- Первый этап.
  - Середина 1990-х – становление и развитие систем распределенных вычислений.
- Второй этап.
  - 1998 г. – определение термина Grid (грид) в рамках книги «Грид. Новая инфраструктура вычислений» . Развитие распределенных систем, ориентированных на массивные объемы передачи информации и вычислительные затраты.
- Третий этап.
  - 2001 г. – уклон в сторону “Виртуальных организаций”. Развитие сервисно-оринентированных подходов (SOA), автоматизация методов управления ресурсами.

Первый этап – становление распределенных вычислений (1980 – 1998)

- Основные технологии – сокеты, RMI, CORBA

- Первые проекты по распределенным вычислениям (начало 1990-х) основывались на объединении вычислительных ресурсов суперкомпьютеров.

- Проект FAFNER: Factoring via Network-Enabled Recursion (Сетевое разложение на множители посредством рекурсии).

- Проект I-WAY: Information Wide Area Year (Год Информации Глобальных Сетей) – экспериментальная высокопроизводительная сеть, которая объединяла множество высокопроизводительных компьютеров и передовые средства визуализации. Прообраз Globus.

Второй этап – развитие, появление Грид и P2P (1999 – 2004)

- Начало 2000-х приносит бурное развитие Грид-систем, начинаются разработки средств создания и управления Грид-сетями и вычислительными ресурсами.

- В ходе исследований систем распределенных вычислений, Ян Фостер вывел 3 основных требования, которым они должны удовлетворять.
  - Гетерогенность.
  - Масштабируемость.
  - Адаптируемость.

- Появляются первые проекты P2P систем (Napster)

- Основные проекты: Globus, SETI@home, Napster (P2P)...

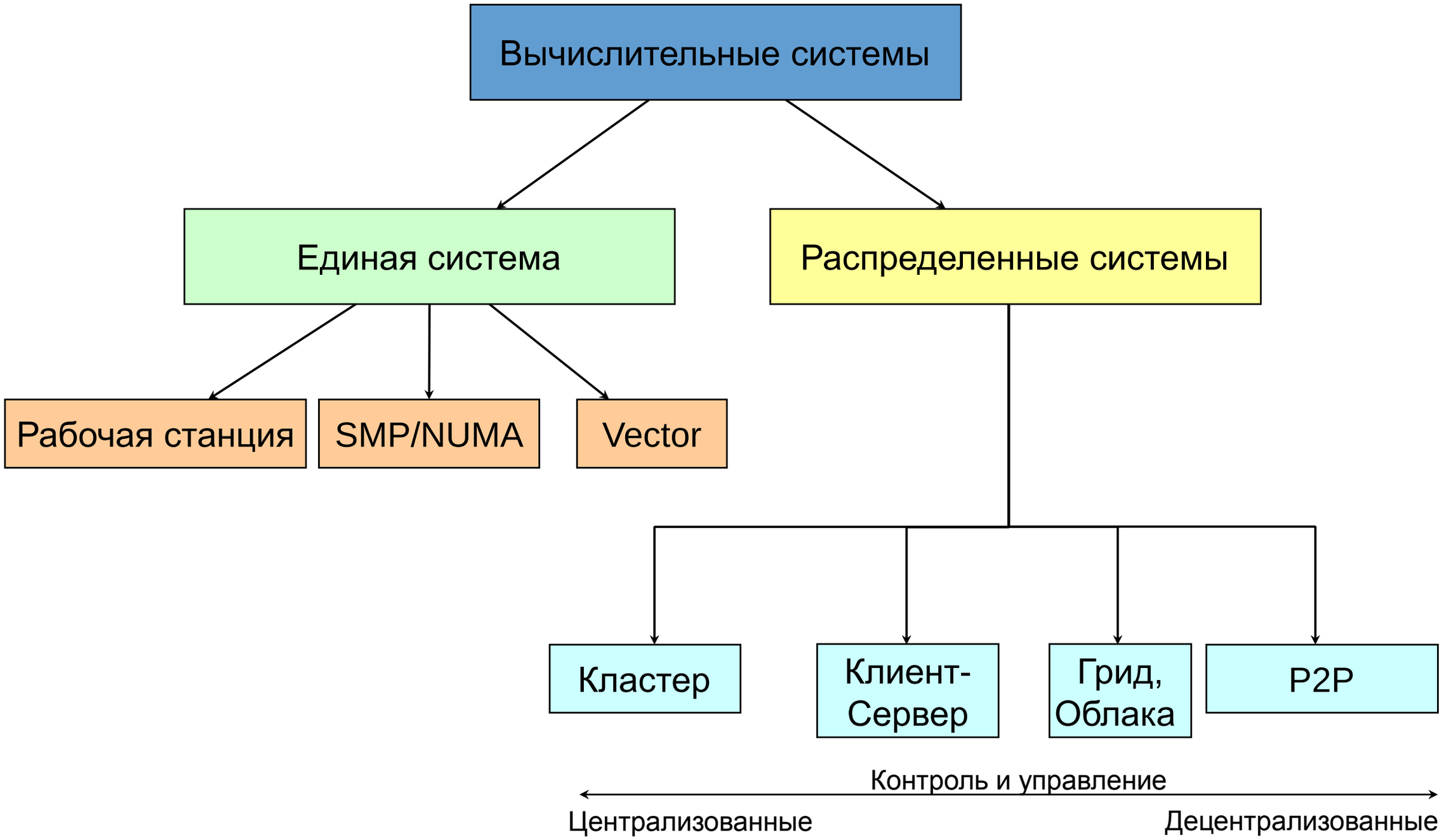
Третий этап – SOA (2005 – на сегодняшний день)

- Развитие сервисно-ориентированных подходов (как противоположность удаленным объектам) позволяет гибко использовать одни и те же вычислительные ресурсы многими пользователями.

- Стандарты и концепции сервис-ориентированного мира (WSDL, SOAP, REST) позволяют осуществить взаимодействие внутрикорпоративных и меж-корпоративных программных систем, независимо от базовой платформы

- Основные проекты и технологии: XML Веб-сервисы, платформа Globus, UNICORE, стандарты OGSA, WSRF…

Виды вычислительных систем

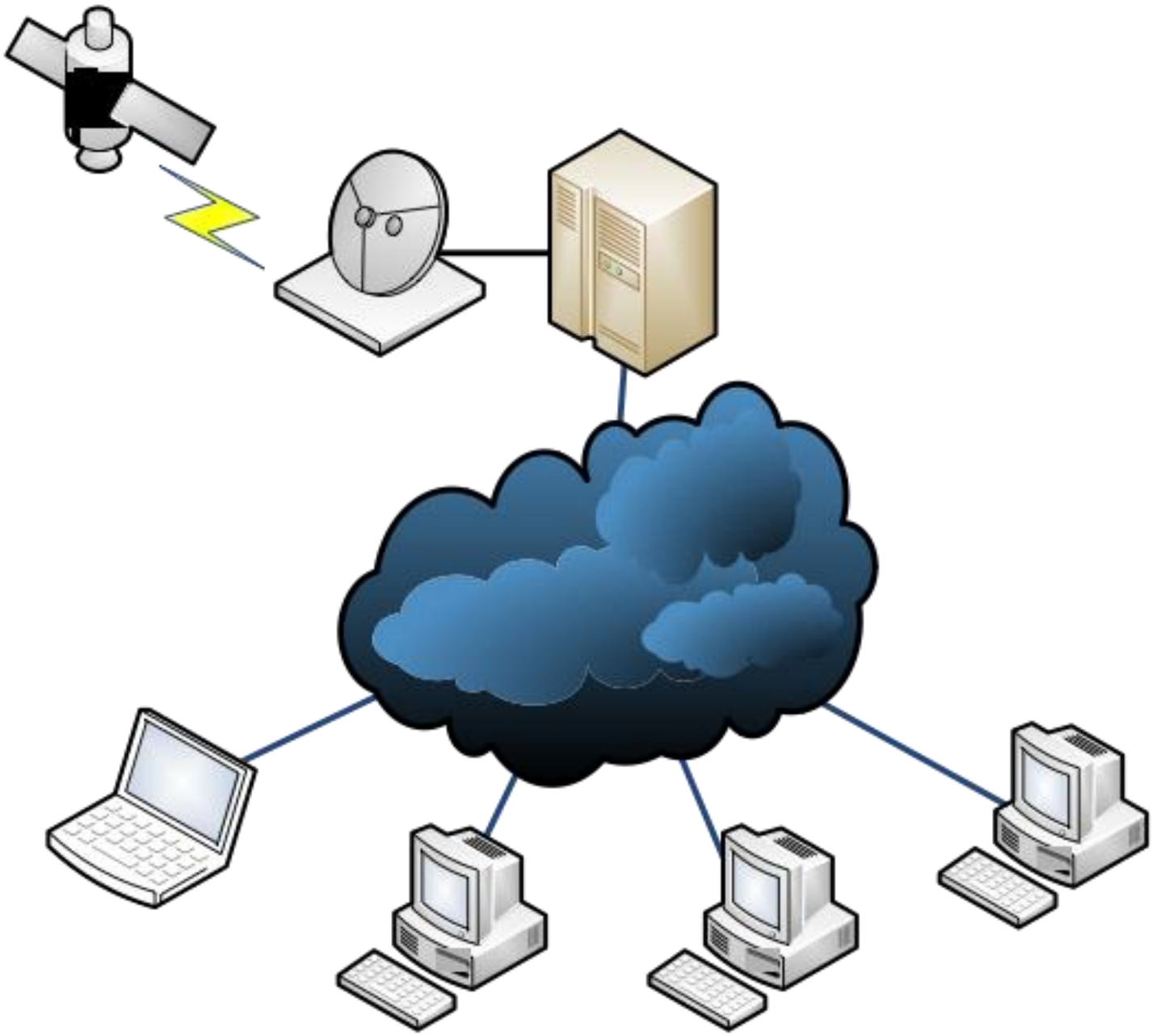


Кластер

Интернет вычисления

Примеры проектов:

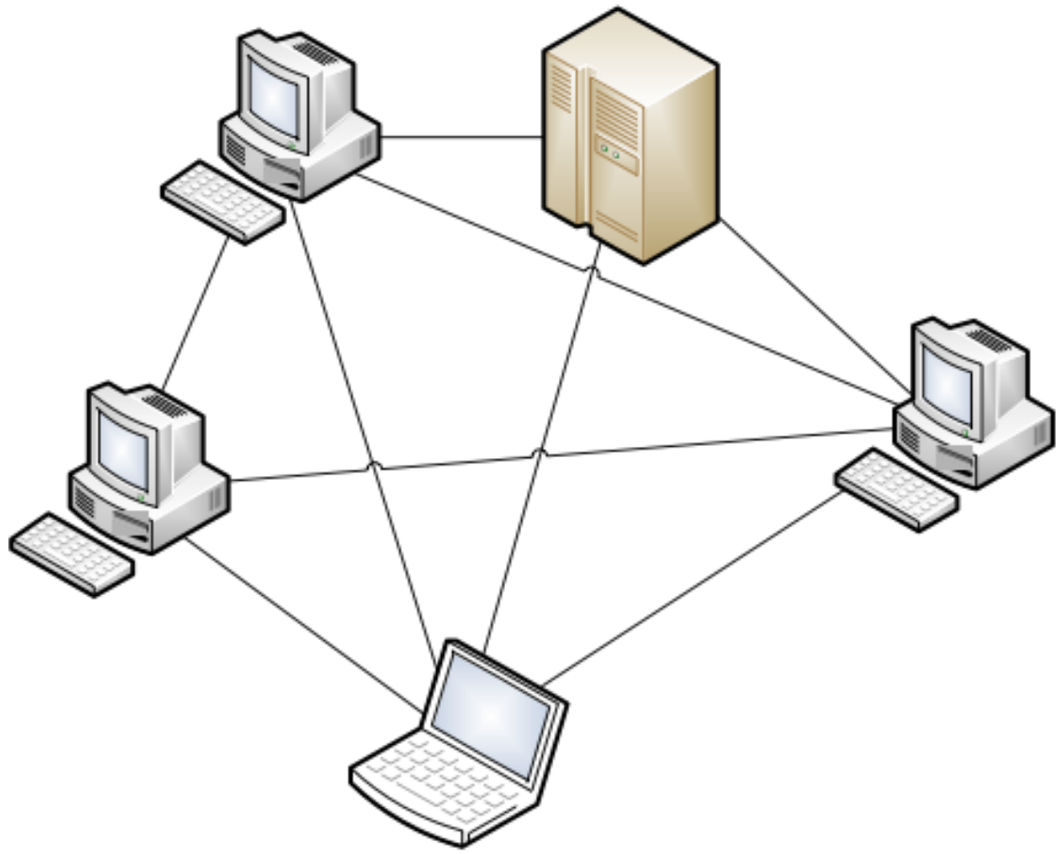
- Платформа BOINC (Berkeley Open Infrastructure for Network Computing):
  - SETI@home
  - Genome@home
  - Folding@home

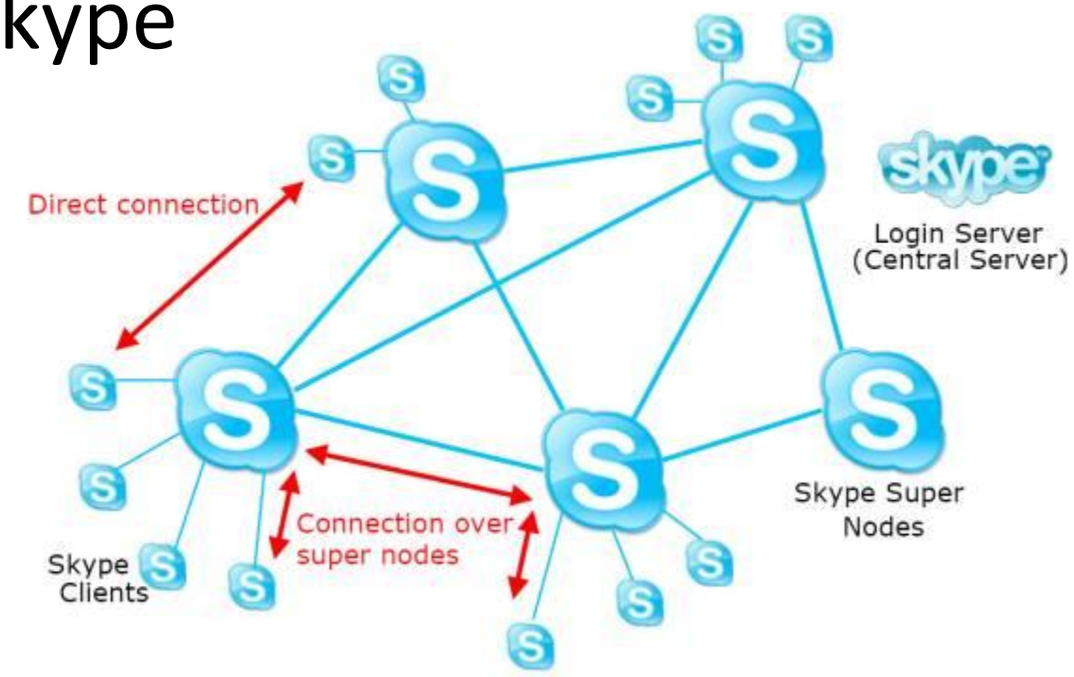


P2P

Примеры проектов:

- eDonkey
- Kazaa, Napster (RIP)
- BitTorrent
- Jabber
- Skype





Сервис-ориентированная архитектура

Классификация систем по связности

Связанность – это степень знания и зависимости одного объекта от внутреннего содержания другого.

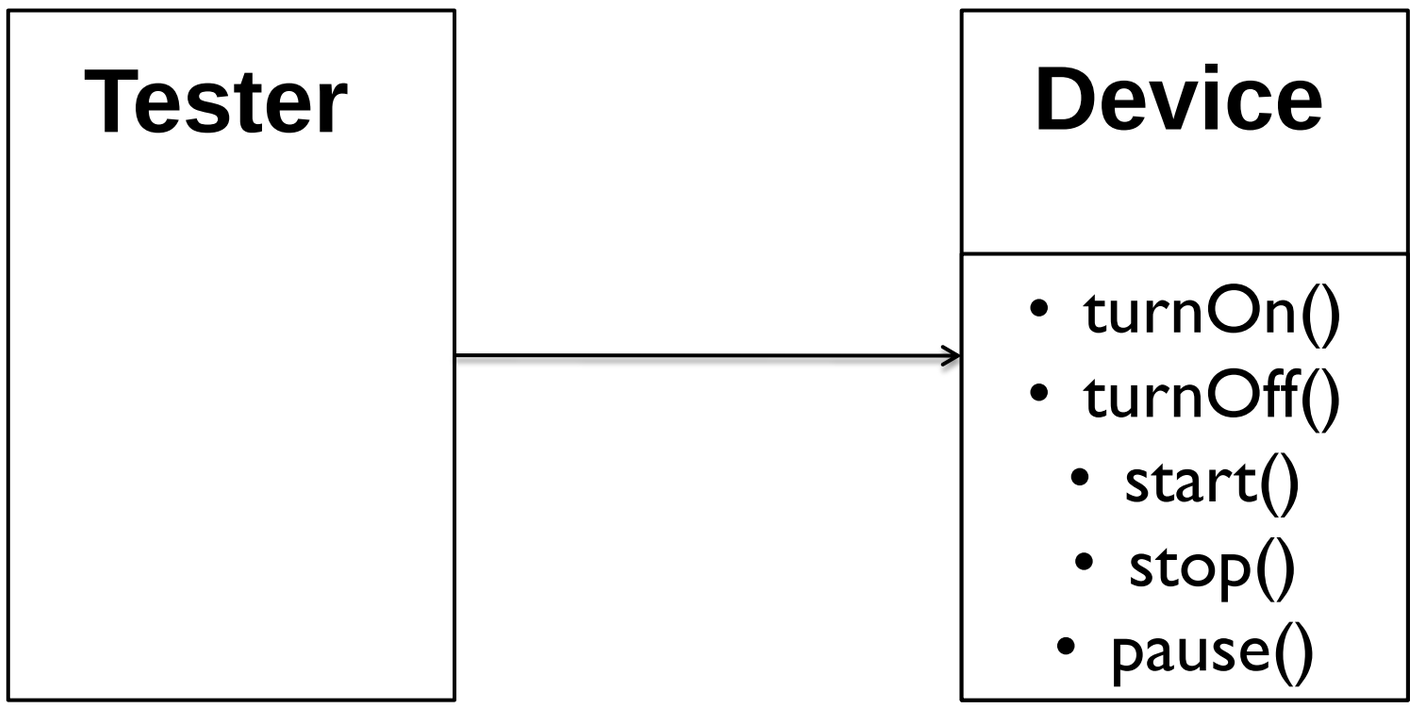
Программные системы можно разделить на 2 типа:

1. **Сильносвязанные системы** *(Strong coupling)*

2. **Слабосвязанные системы** *(Loose coupling)*

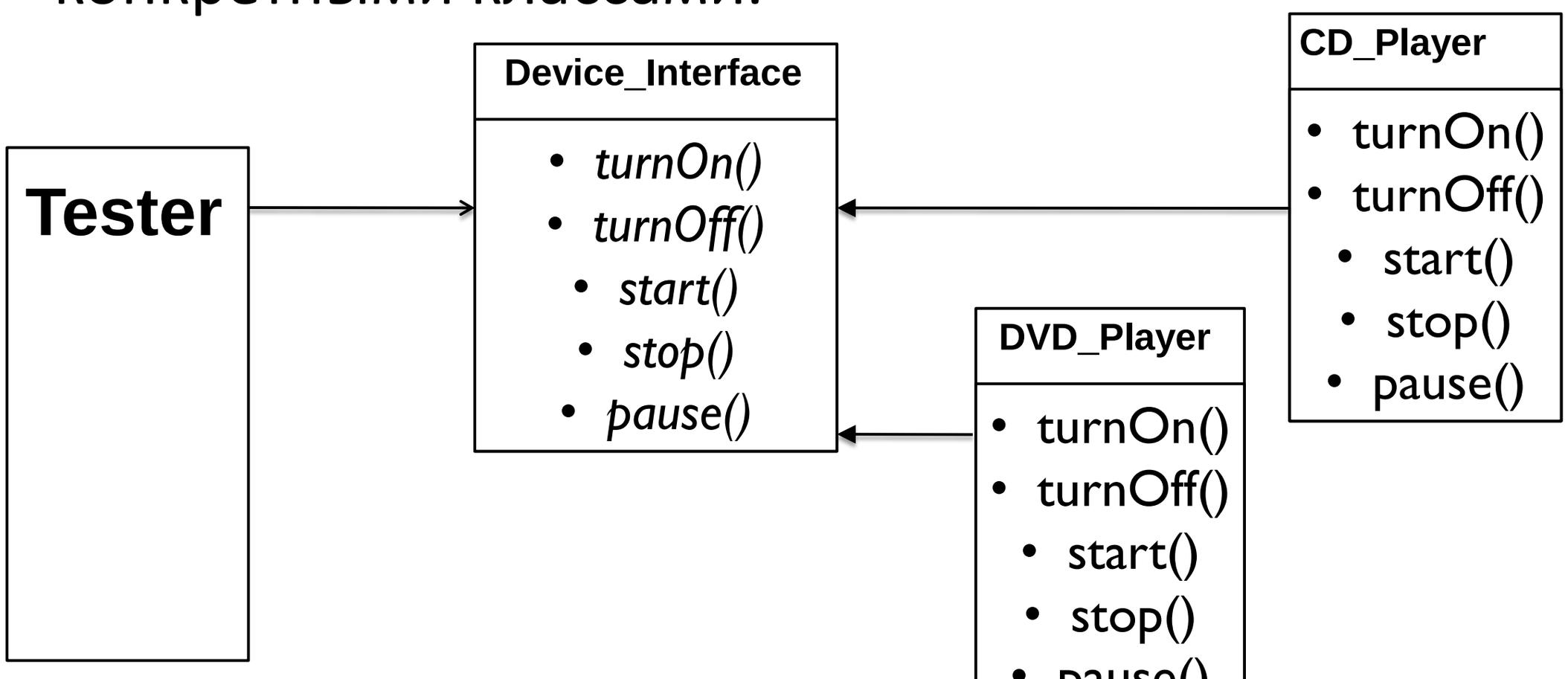
Сильносвязанные вычислительные системы

- *Сильная связанность* возникает, когда зависимый класс содержит ссылку непосредственно на ***определенный класс***, предоставляющий некоторые возможности.



Слабосвязанные вычислительные системы

- *Слабая связанность* возникает, когда зависимый класс содержит ссылку на ***интерфейс*** который может быть реализован одним или несколькими конкретными классами.



Преимущества слабосвязанных систем

- Уменьшается количество связей между компонентами системы, уменьшая объем возможных последствий в связи со сбоями
- Обеспечивается возможность расширения рабочей системы, путем создания новых классов, обладающих единым интерфейсом

Сервис-ориентированная архитектура

- СОА означает, что компоненты приложения являются слабосвязанными и реализуются в виде совместимых сервисов (служб), которые могут быть использованы независимо друг от друга и объединены с другими приложениями и другими разработчиками.

- Антоним – монолитный программный продукт, который не предоставляет открытых и описанных API ко своим компонентам

Patterson, David; Fox, Armando (2012-01-12). Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing, Alpha Edition (Kindle Locations 557-561). Strawberry Canyon LLC. Kindle Edition.

Amazon:
От монолитных систем – к SOA

- В 1995 году Amazon стартовал как неунифицированная, закрытая система онлайн-продаж.

- В 2002 году CEO Amazon перевел компанию на русло сервис-ориентированной системы посредством простого письма, разосланного всем разработчикам.

Patterson, David; Fox, Armando (2012-01-12). Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing, Alpha Edition (Kindle Locations 557-561). Strawberry Canyon LLC. Kindle Edition.

1. С настоящего момента все меж-процессные взаимодейсвтия должны производиться исключительно на основе открытых интерфейсов служб.
2. Команды должны взаимодействовать друг с другом на основе таких интерфейсов.
3. Запрещено: прямое чтение или модификация данных из хранилищ других команд; использование общей памяти; «черных ходов» и тому подобное
4. Все интерфейсы должны быть расширяемы и разработаны таким образом, чтобы быть готовыми к использованию разработчиками из внешнего мира.
5. Все кто не выполняет данные требования будет уволен.

Patterson, David; Fox, Armando (2012-01-12). Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing, Alpha Edition (Kindle Locations 557-561). Strawberry Canyon LLC. Kindle Edition.

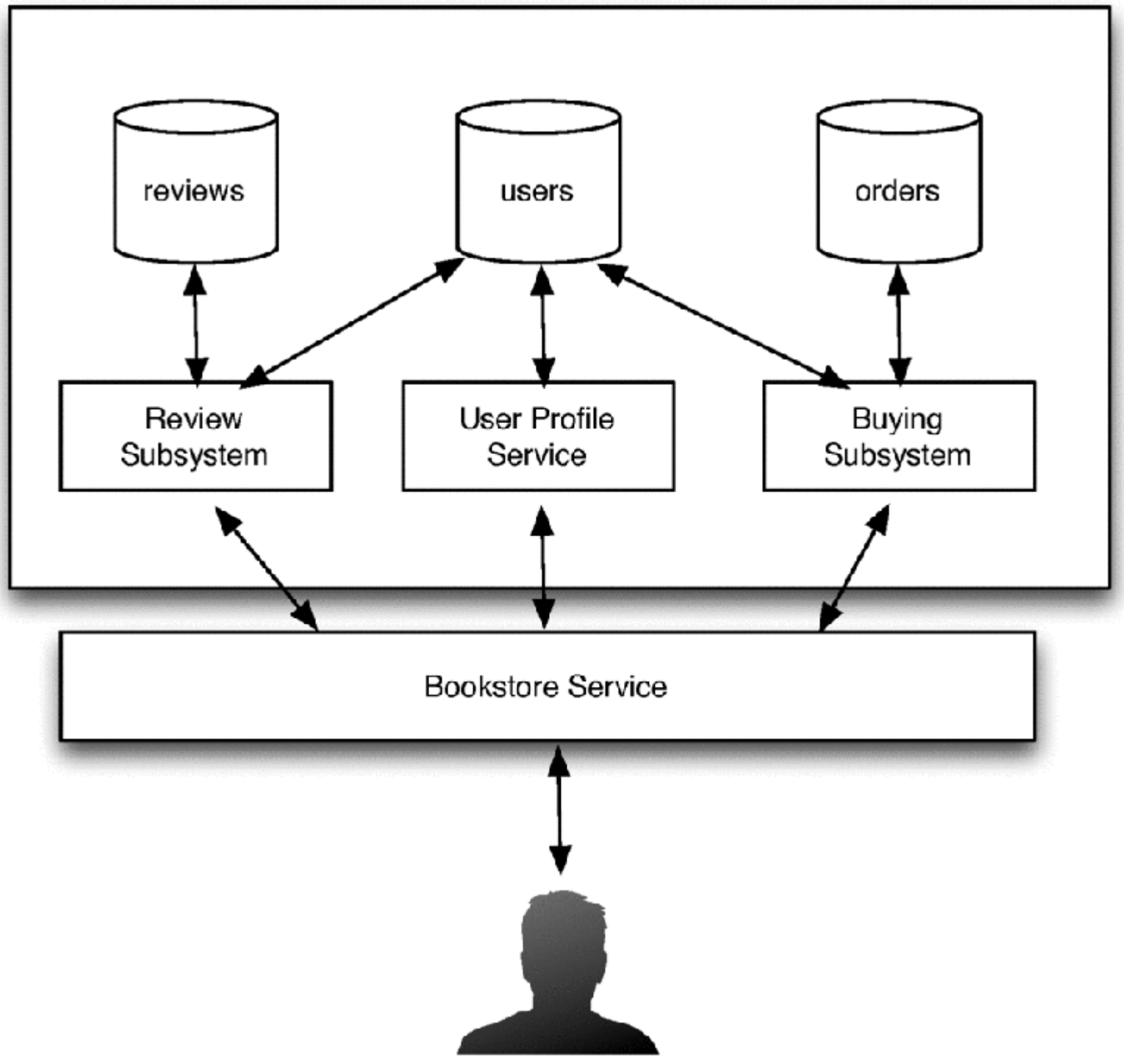
Facebook (2007) и Google+

- Аналогичный переход сделал Facebook в 2007 году
- Была запущена Facebook Platform, позволив сторонним программистам разрабатывать приложения, которые могли получать доступ к социальному графу и другим данным пользователей в Facebook.

- Google+ очень долго ругали за отсутствие API, но когда он был выпущен в сентябре 2011 года он оказался «монолитным» и абсолютно не «SOA-like», обеспечивая лишь простейший доступ к профилю пользователя и всему, что он видит.

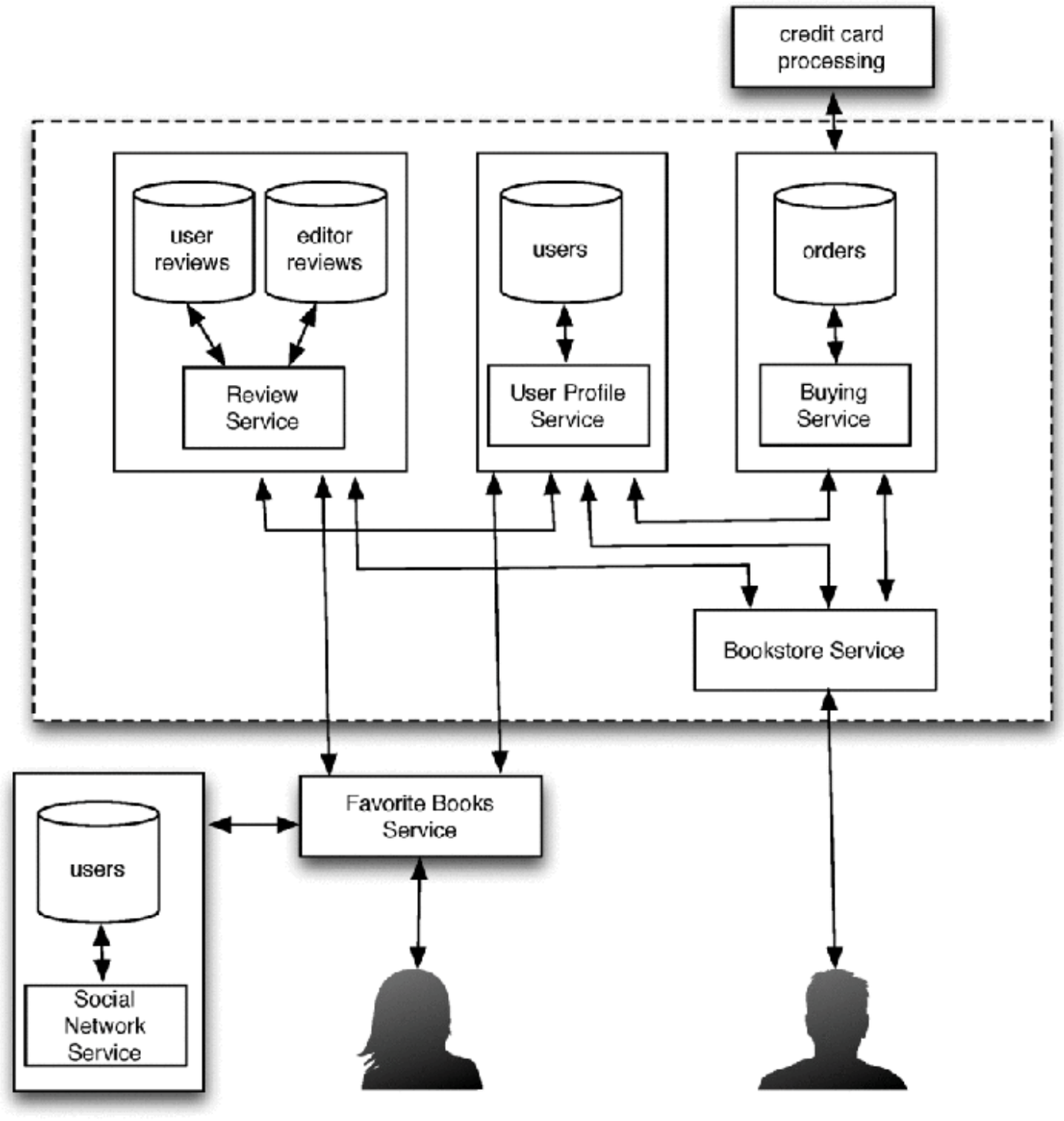
Patterson, David; Fox, Armando (2012-01-12). Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing, Alpha Edition (Kindle Locations 557-561). Strawberry Canyon LLC. Kindle Edition.

Пример архитектуры: магазин книг (1)



Patterson, David; Fox, Armando (2012-01-12). Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing, Alpha Edition (Kindle Locations 557-561). Strawberry Canyon LLC. Kindle Edition.

Пример архитектуры: магазин книг (2)



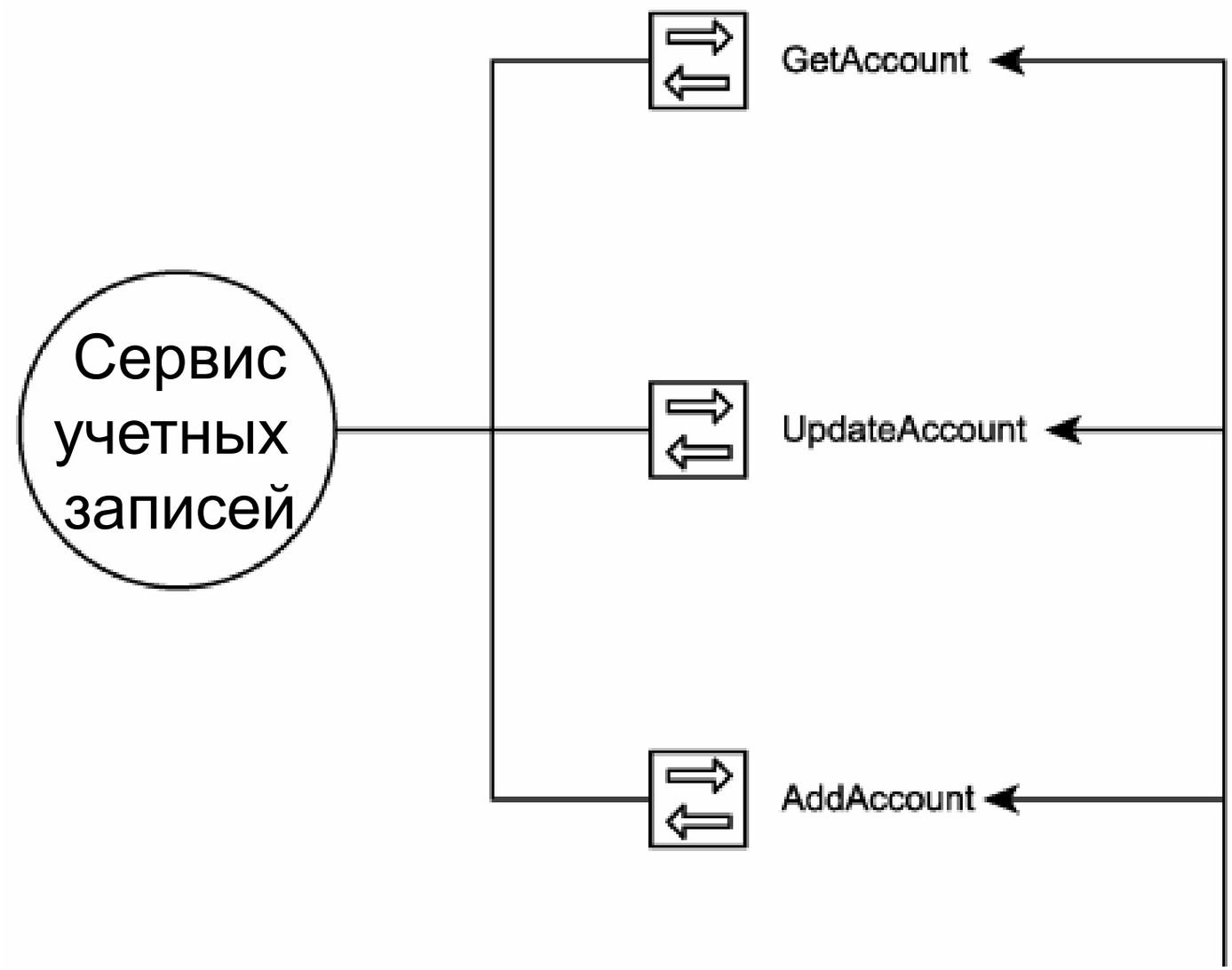
Patterson, David; Fox, Armando (2012-01-12). Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing, Alpha Edition (Kindle Locations 557-561). Strawberry Canyon LLC. Kindle Edition.

Основные принципы СОА

1. Сервис должен допускать повторное использование (1)

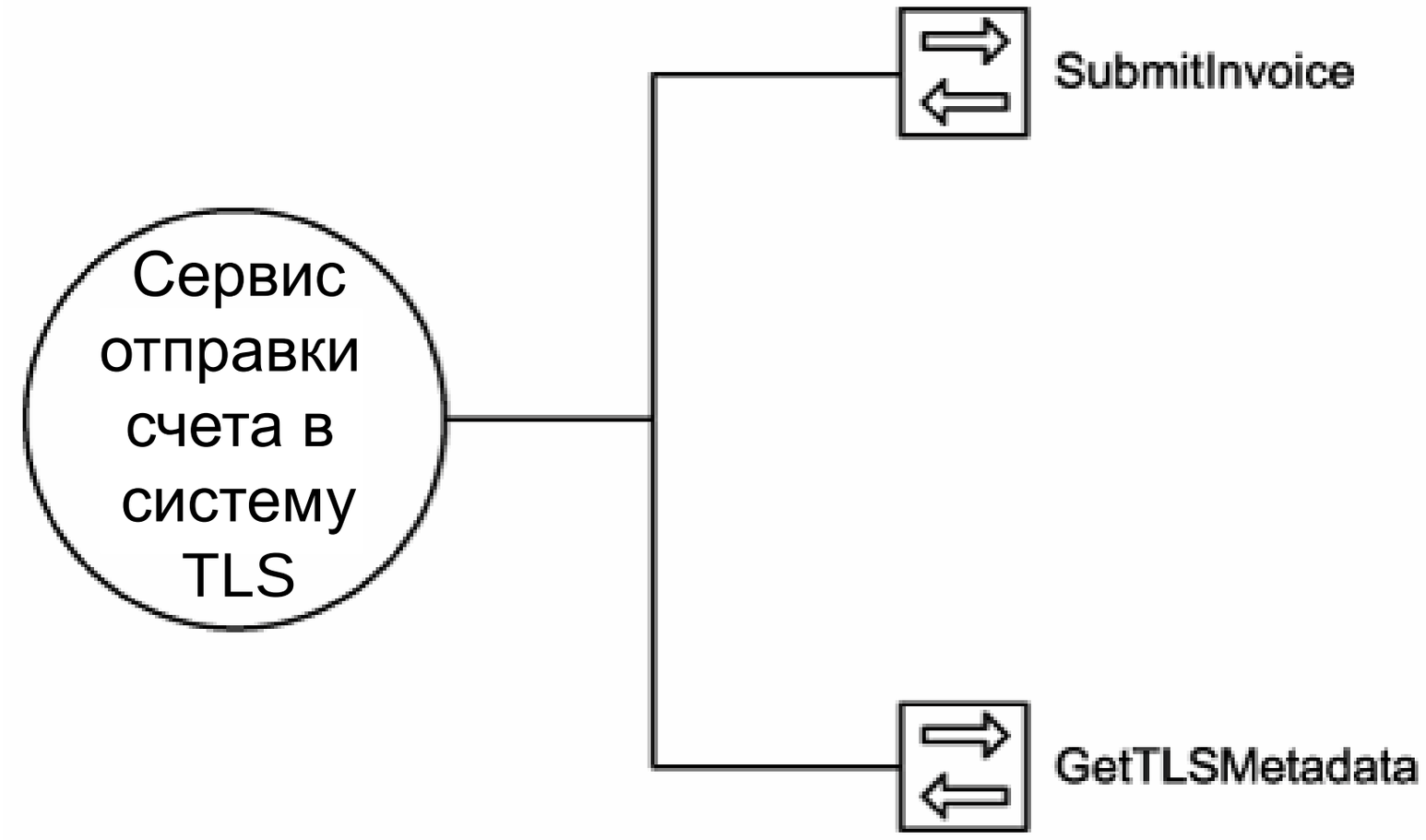
- SOA-системы должны поддерживать повторное использование всех сервисов, независимо от сиюминутных требований к их функциональным особенностям.
  - Это позволит упростить расширение и развитие системы, отказаться от «оберток» над сервисами для их подстройки на решение новых задач.
- Таким образом, каждая операция сервиса должна поддерживать повторное использование

1. Сервис должен допускать повторное использование (2)



Может использоваться несколькими абонентами

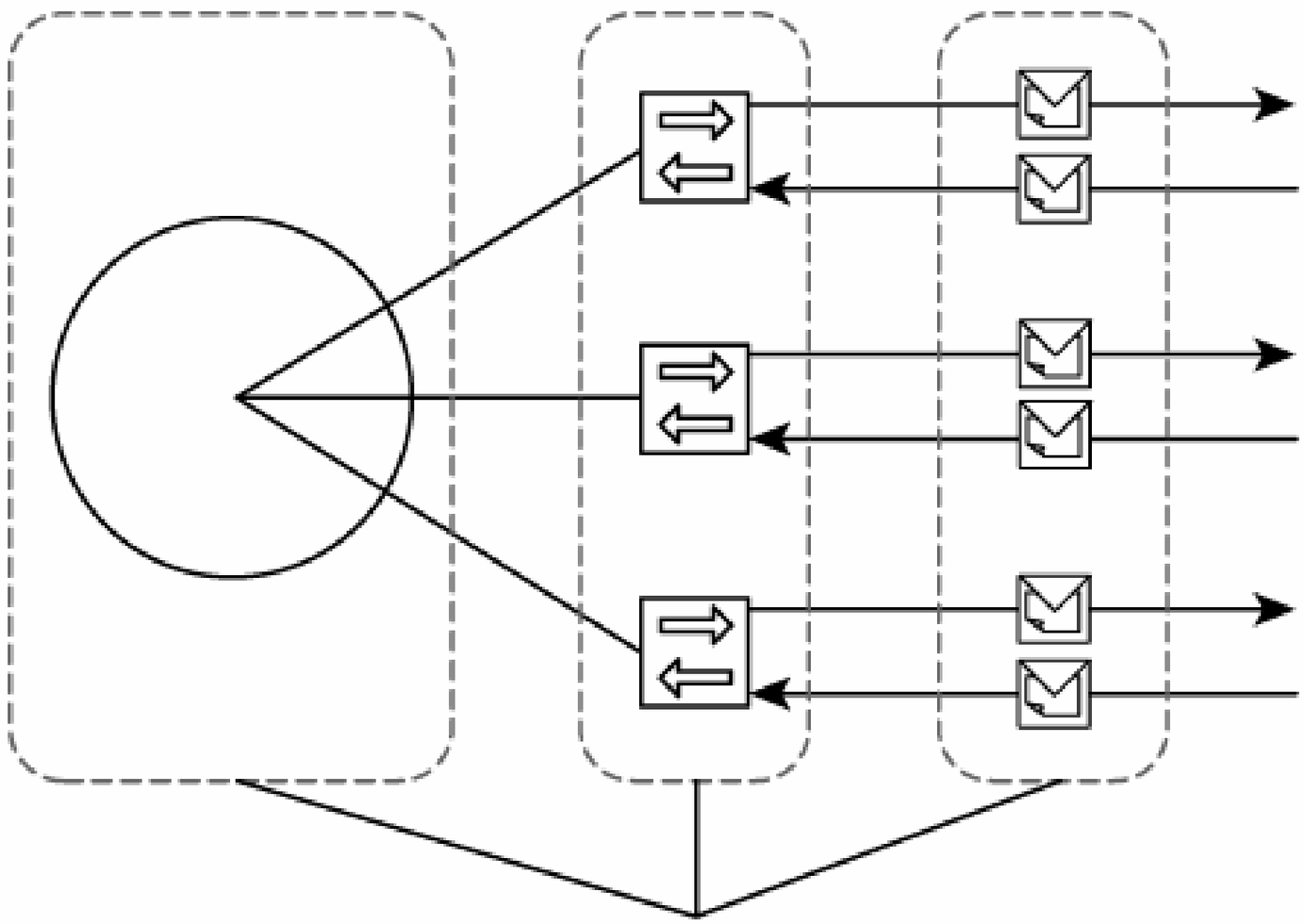
1. Сервис должен допускать повторное использование (3)



2. Сервисы должны обеспечивать формальный контракт использования (1)

- Контракт сервиса предоставляет следующую информацию:
  - Адрес конечной точки (service endpoint)
  - Все операции, предоставляемые сервисом
  - Все сообщения, поддерживаемые каждой операцией
  - Правила и характеристики сервиса и его операций.

2. Сервисы должны обеспечивать формальный контракт использования (2)

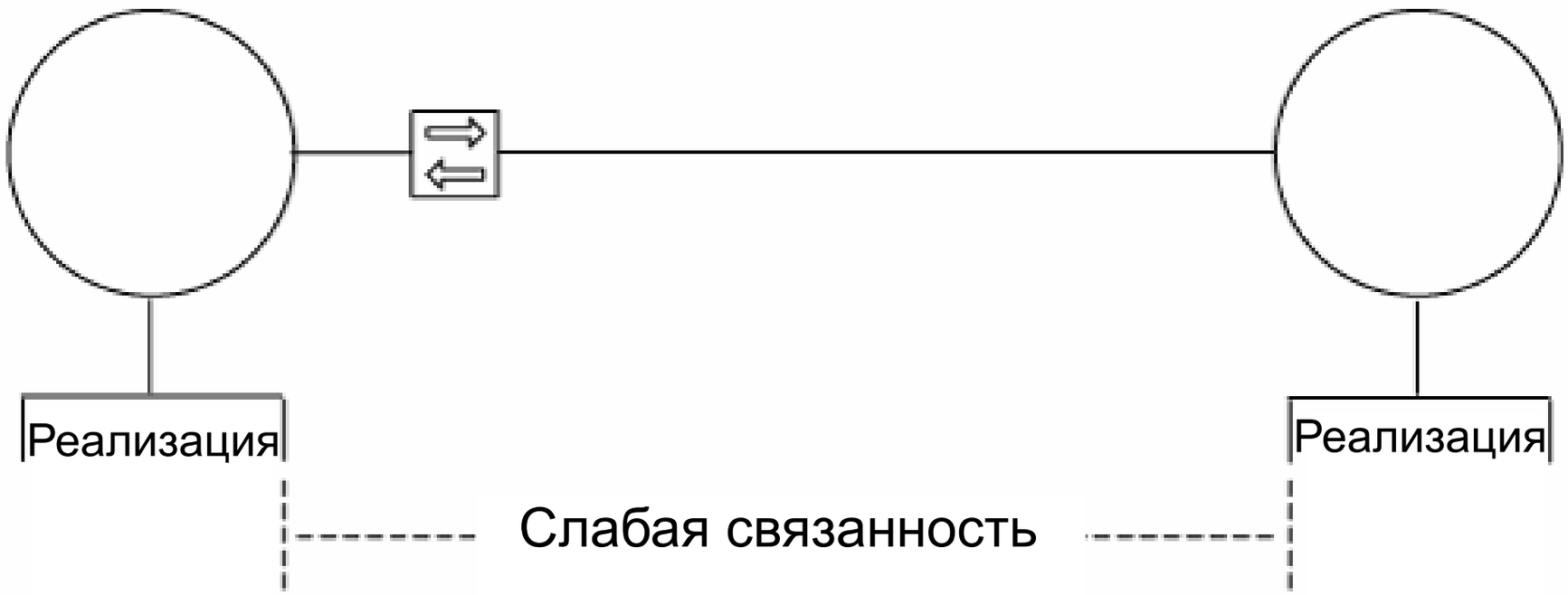


Определяется контрактом сервиса

3. Сервисы должны быть слабосвязанны(1)

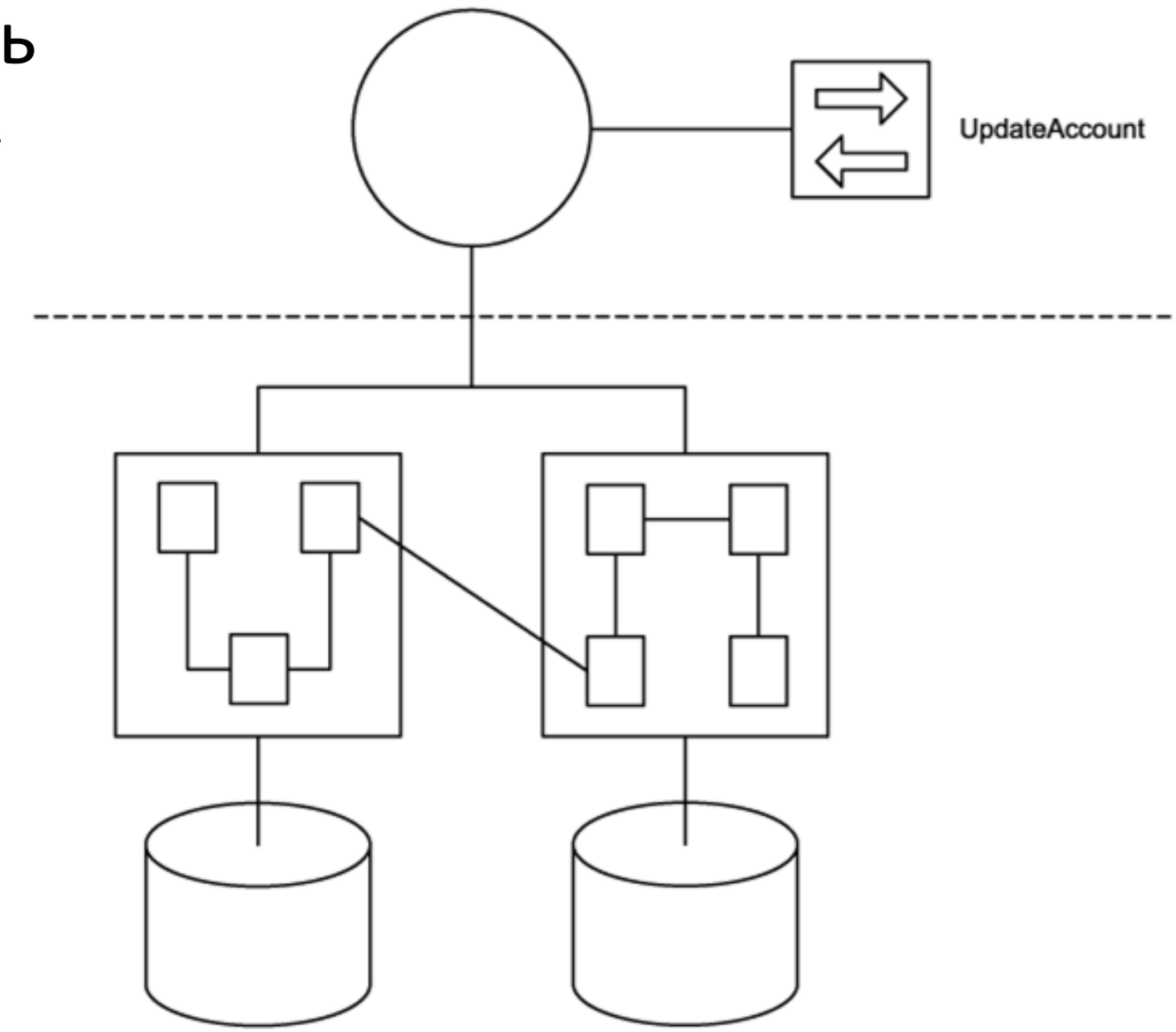
- Необходимо обеспечить целостность системы в рамках развития системы, независимо от пути развития
- Система сервисов является слабо связанной если сервис может приобретать знания о другом сервисе, оставаясь независимым от внутренней реализации логики данного сервиса.

3. Сервисы должны быть слабосвязанны(2)



4. Сервисы должны абстрагировать внутреннюю логику

- Каждый сервис должен действовать как «черный ящик»

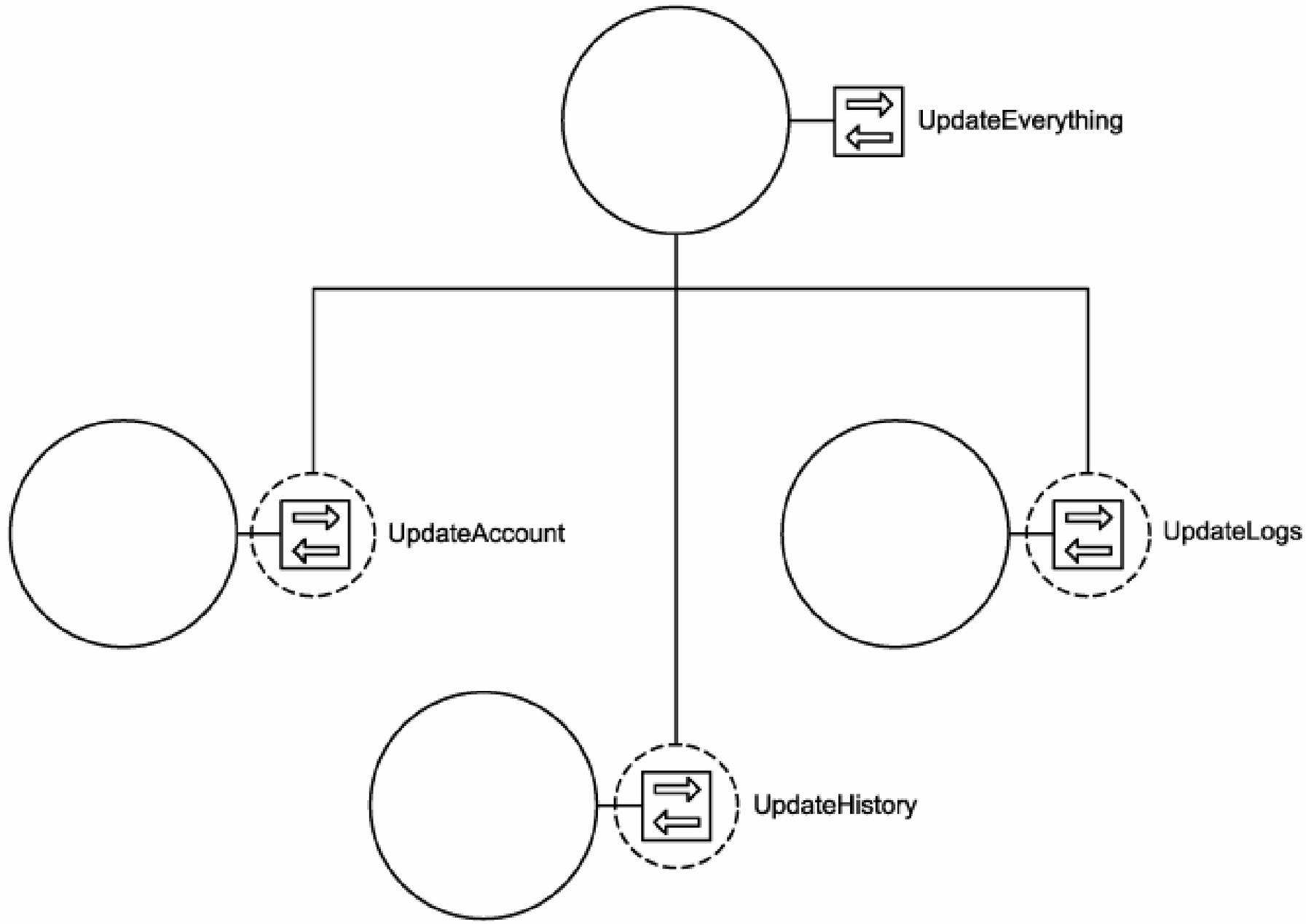


- Это одно из требований, обеспечивающих слабосвязанность сервисов.

5. Сервисы должны быть совместимы (1)

- Сервис может как самостоятельно реализовывать логику, так и применять другие сервисы для ее реализации
- Сервисы должны быть спроектированы таким образом, чтобы поддерживать возможность их использования в качестве элементов другого сервиса
- Такой процесс называется «Оркестрацией сервисов» (Service orchestration).

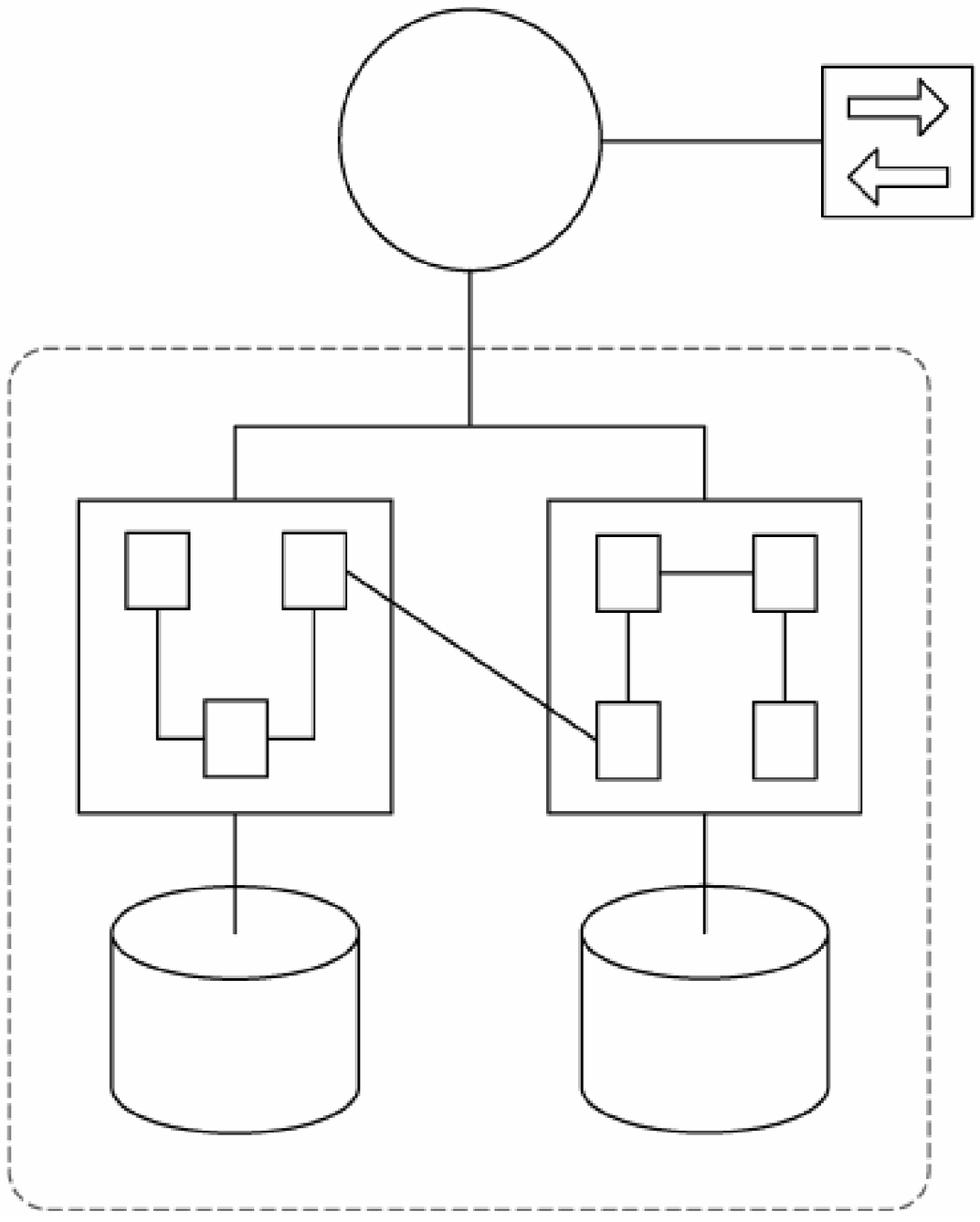
5. Сервисы должны быть совместимы (2)



6. Сервисы должны быть автономны (1)

- Область бизнес-логики и ресурсов, используемых сервисом должны быть ограничены явными пределами
- Вопрос автономности – наиболее важный аргумент при распределении бизнес-логики на отдельные сервисы
- Выделяют 2 типа автономности:
  - *Автономность на уровне сервиса*: границы ответственности сервисов отделены, но они могут использовать общие ресурсы
  - *Чистая автономность:* бизнес-логика и ресурсы находятся под полным контролем сервиса

6. Сервисы должны быть автономны (2)



7. Сервисы не должны использовать информацию о состоянии (1)

- «Чистые» сервисы должны значительно ограничивать объем и время хранения информации (в идеале – только на время вычислений)
- Не зависимость от состояния (Statelessness) позволяет повысить возможности масштабируемости и повторного использования сервисов
- Но это очень жесткое ограничение, которое не всегда удается удовлетворить.

8. Сервисы должны поддерживать обнаружение

- Обнаружение сервисов позволяет избежать случайного создания избыточного сервиса, обеспечивающего избыточную логику
- Все методы сервиса должны содержать метаданные, описывающие их возможности в системе поиска
- Каждый сервис должен предоставлять как можно больше информации о своих возможностях