Распределенные вычислительные системы

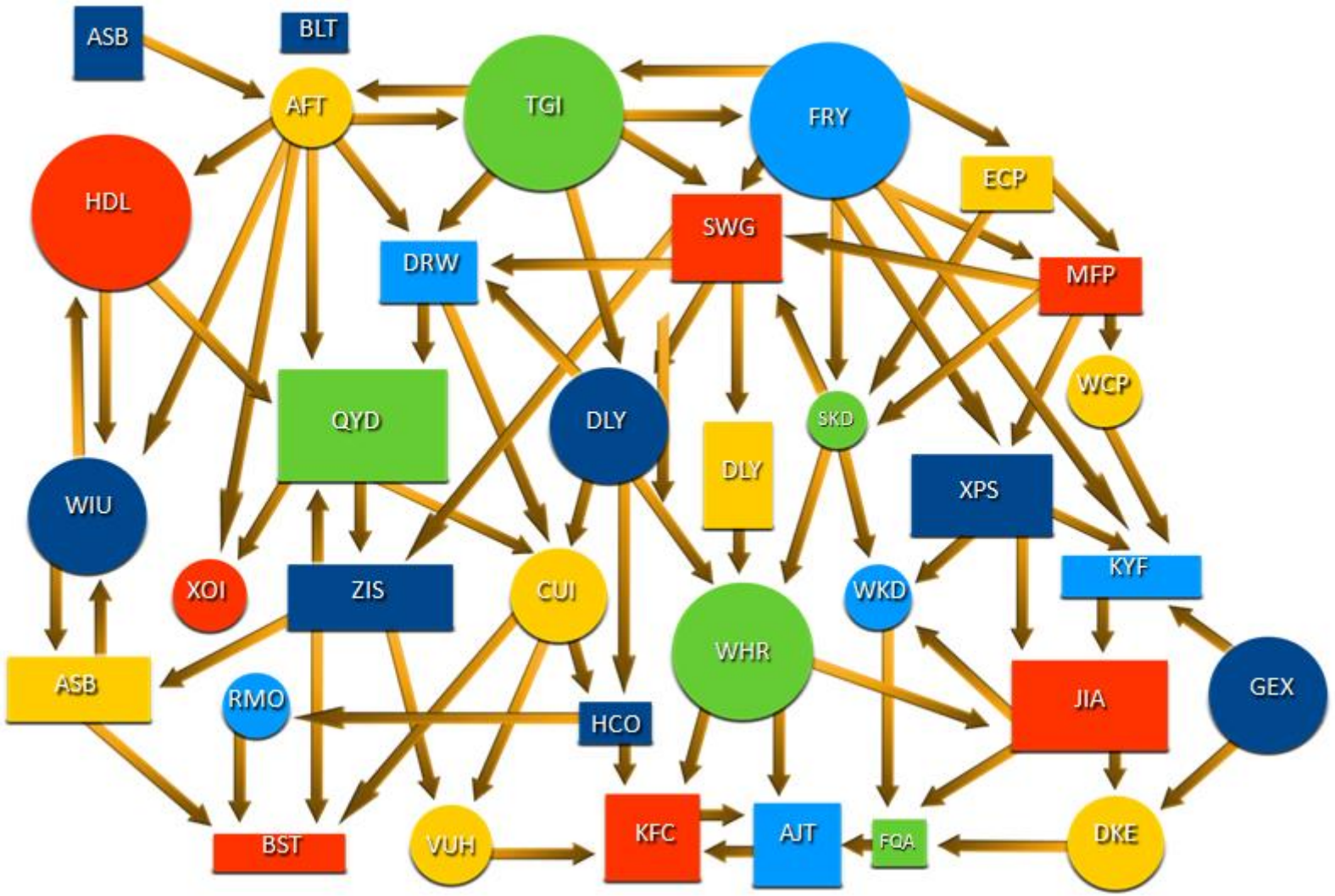
Сервис-ориентированная архитектура

Подход SOA

Принципы SOA

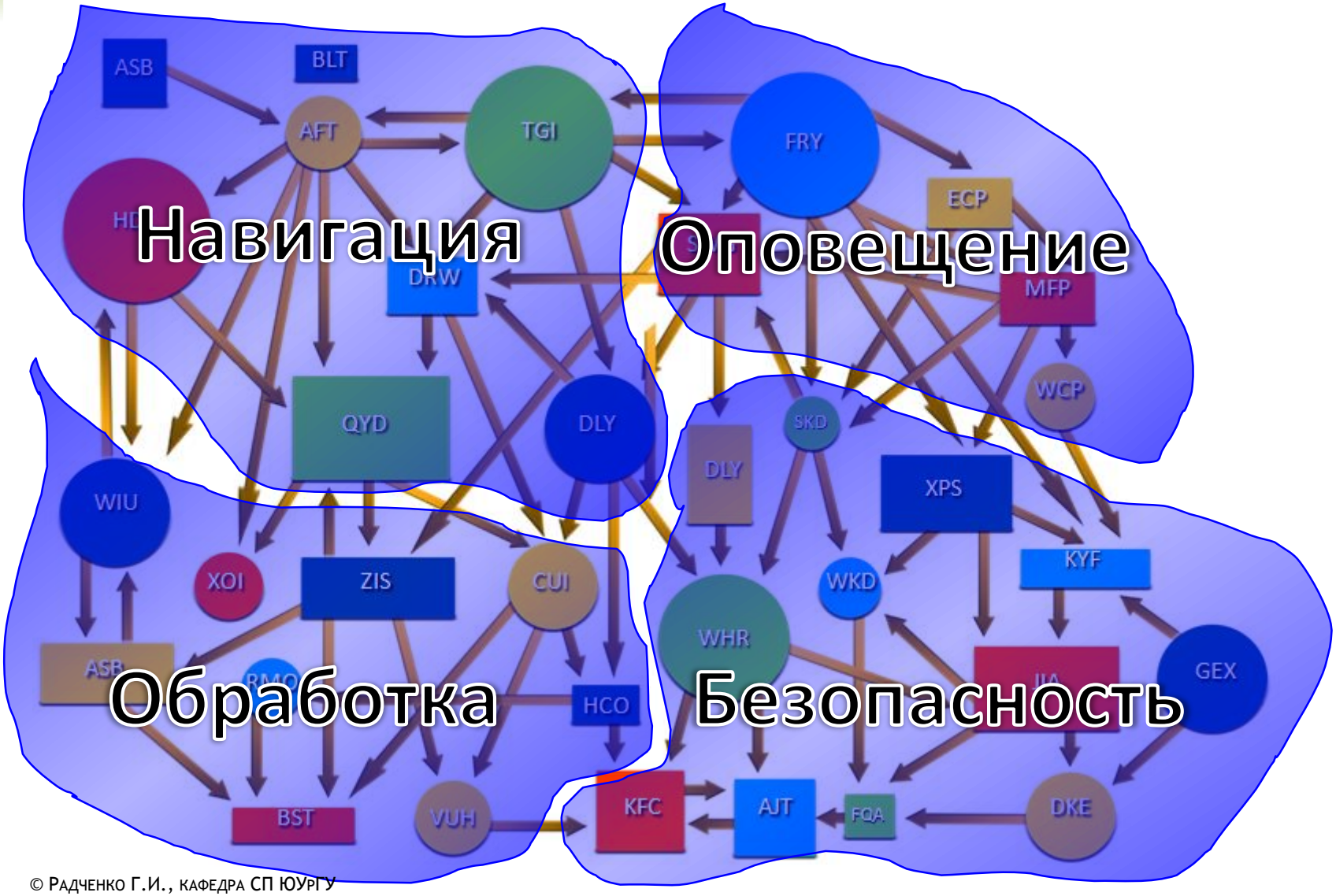
- Фундаментальных, жестко закрепленных и прописанных в стандарте «принципов SOA» не существует
- Существует ряд практик, которых стоит придерживаться при разработке SOA-систем

Подход SOA:
1. Анализ бизнес-процессов



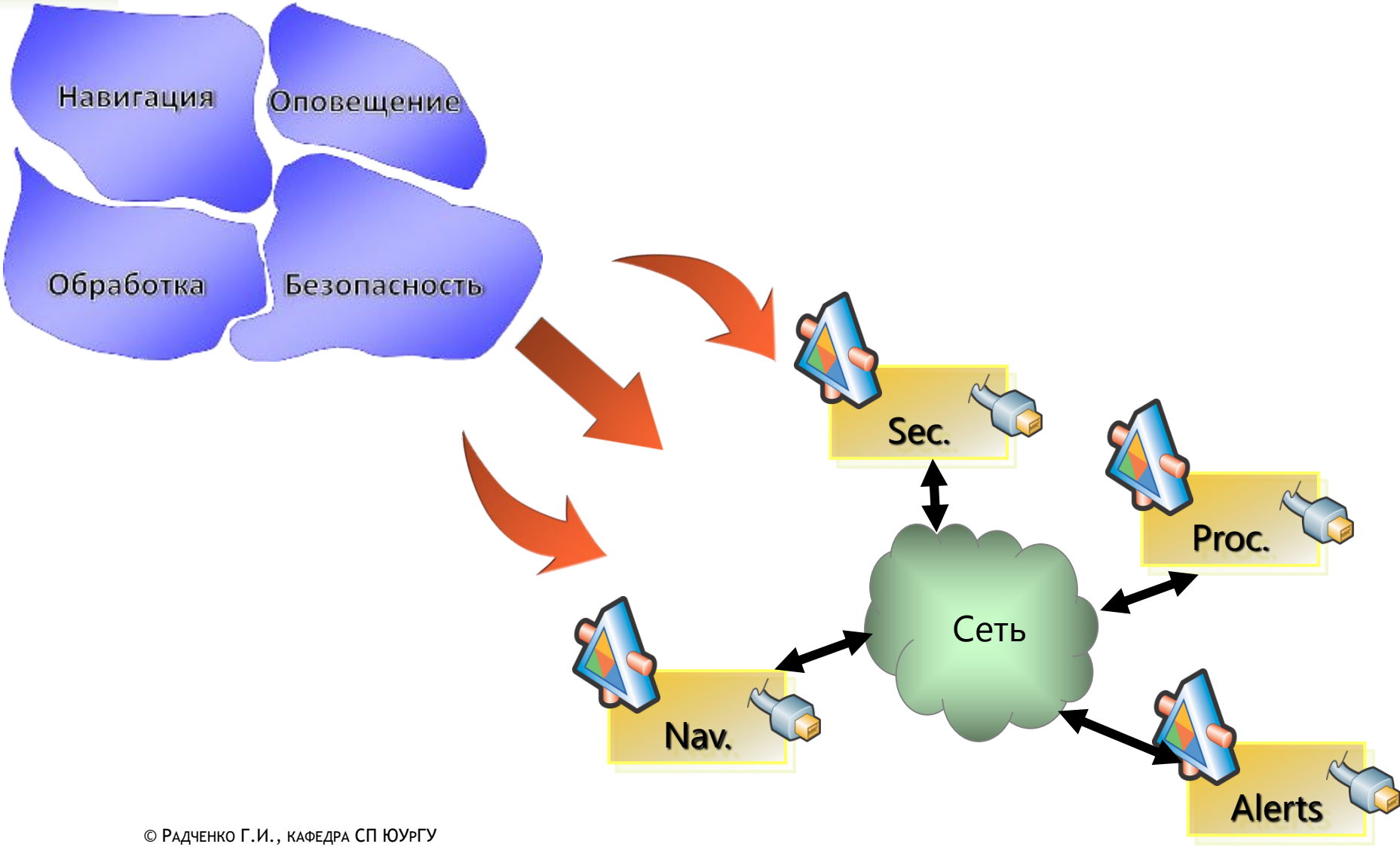
© Радченко Г.И., кафедра СП ЮУрГУ

Подход SOA:
2. Выделение областей



© Радченко Г.И., кафедра СП ЮУрГУ

Подход SOA: 3. Проекция на сервисы



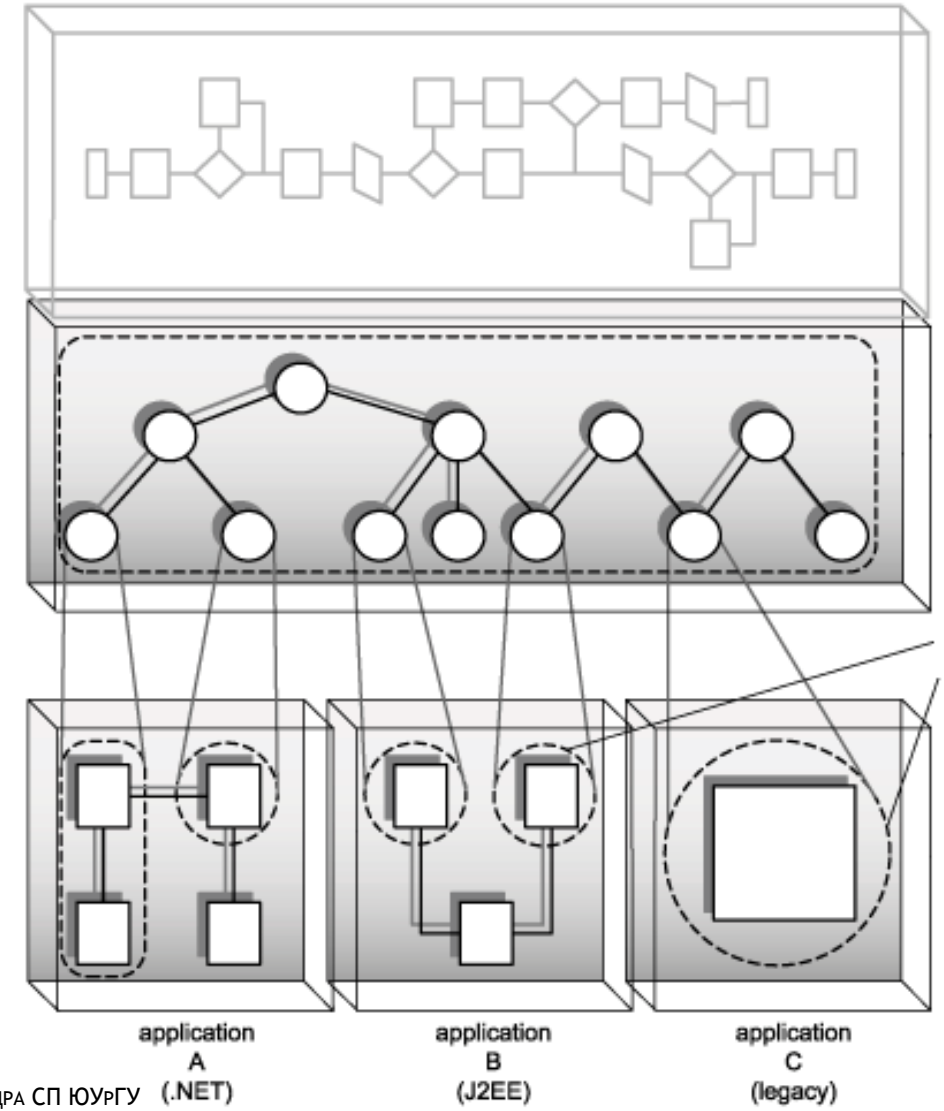
© Радченко Г.И., кафедра СП ЮУрГУ

Подход SOA: Реализация сервисов

Уровень бизнес-логики

Уровень интерфейса сервисов

Уровень приложений



Физическая инкапсуляция сервисов

Базовые принципы SOA

- Распределенное проектирование
- Постоянство изменений
- Последовательное совершенствование
- Рекурсивность

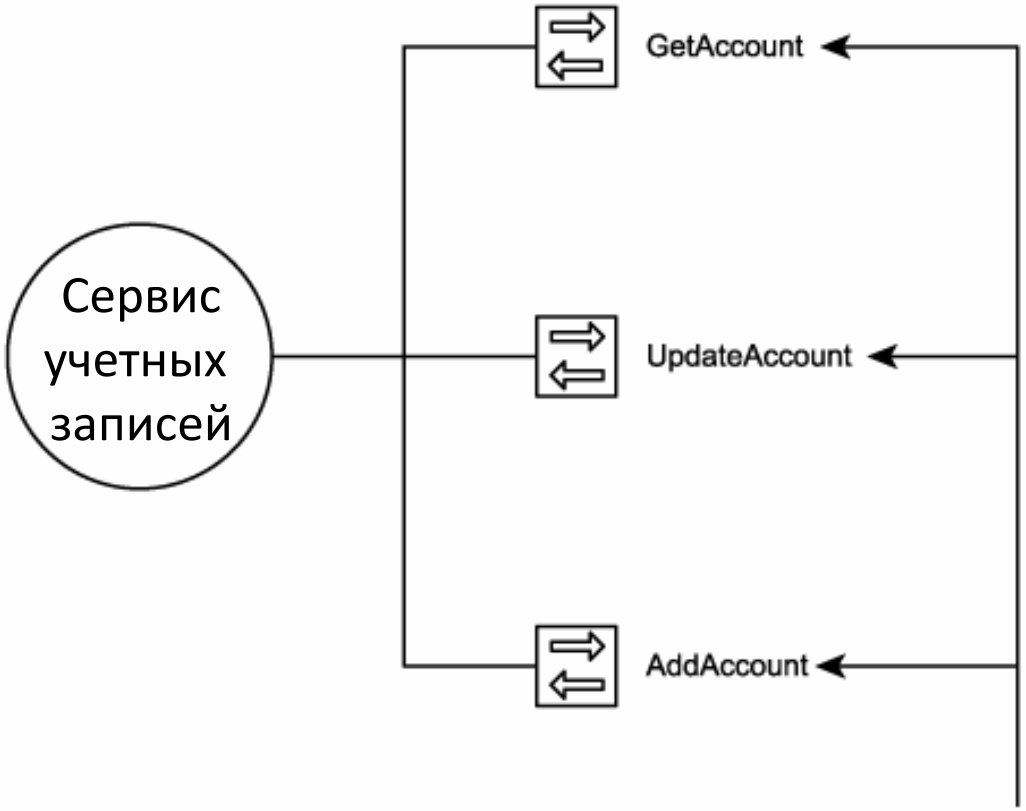
Основные принципы SOA

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

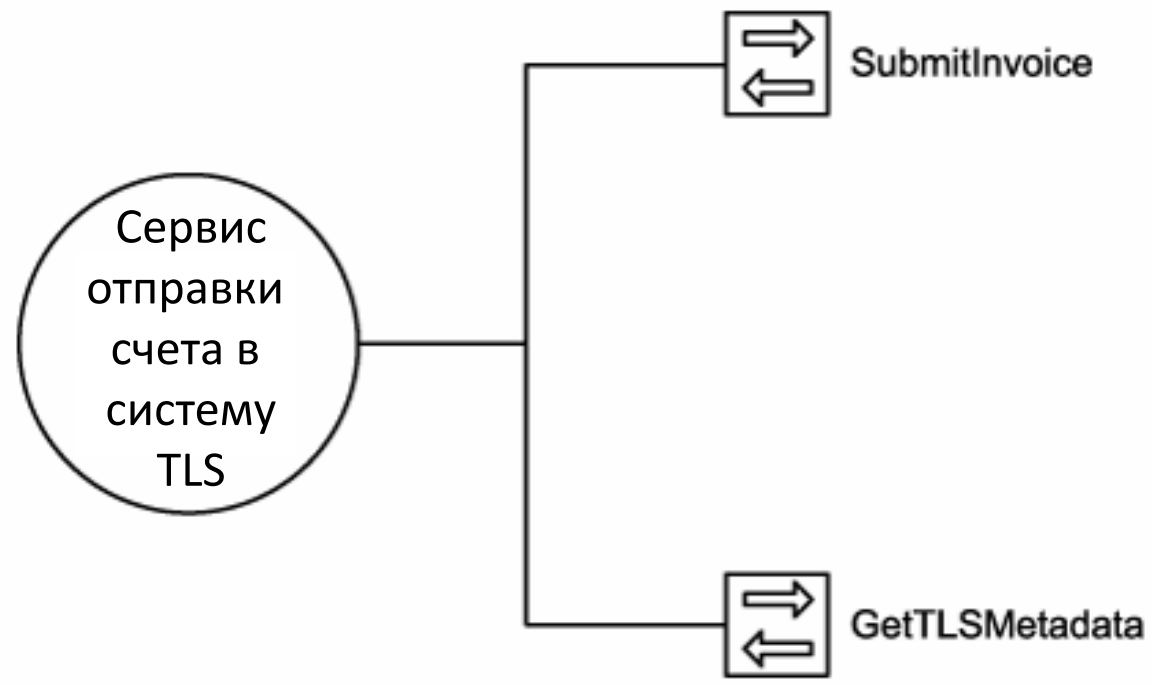
© Радченко Г.И., кафедра СП ЮУрГУ

1. Сервис должен допускать повторное использование (2)



Может использоваться несколькими абонентами

1. Сервис должен допускать повторное использование (3)

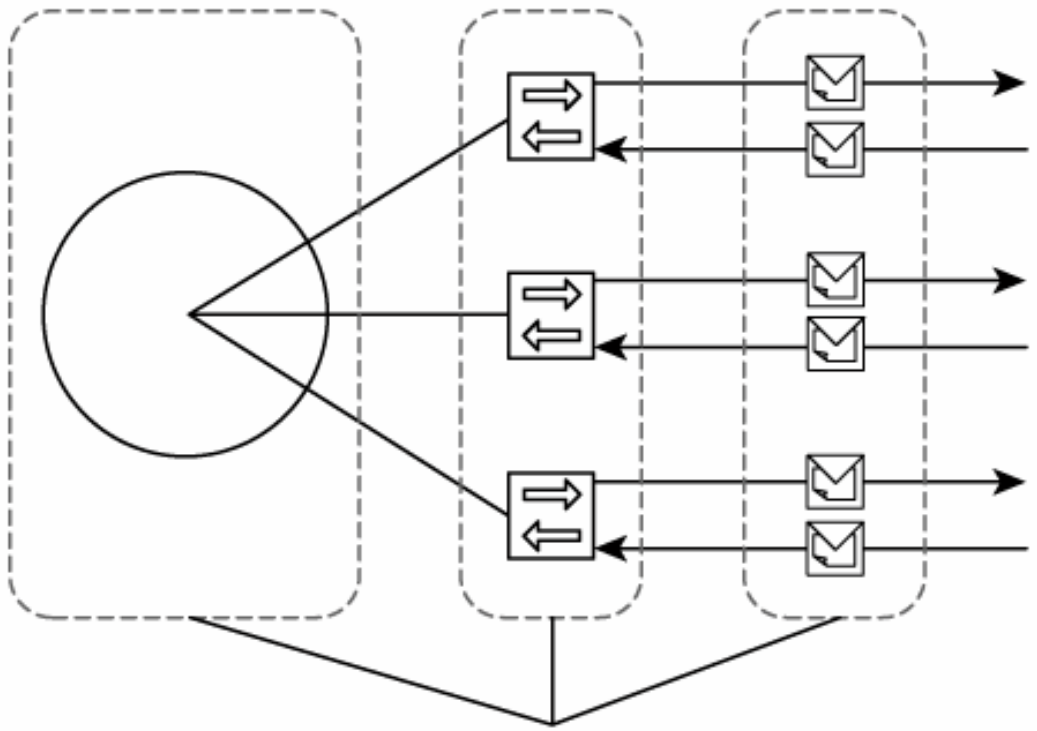


2. Сервисы должны обеспечивать формальный контракт использования (1)

- 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.

© Радченко Г.И., кафедра СП ЮУрГУ

2. Сервисы должны обеспечивать формальный контракт использования (2)



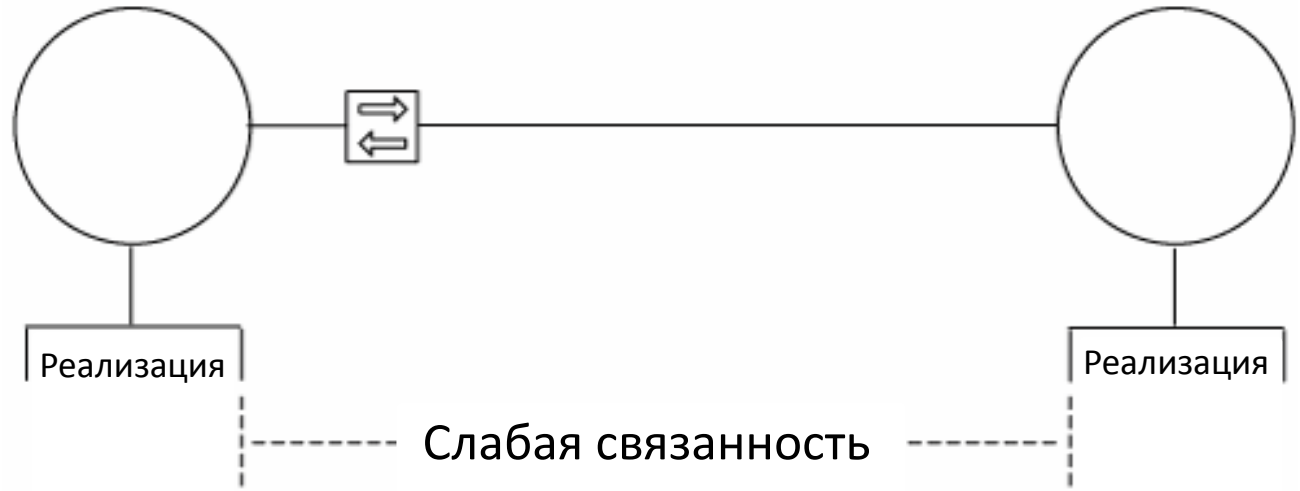
Определяется контрактом сервиса

© Радченко Г.И., кафедра СП ЮУрГУ

3. Сервисы должны быть слабосвязанны(1)

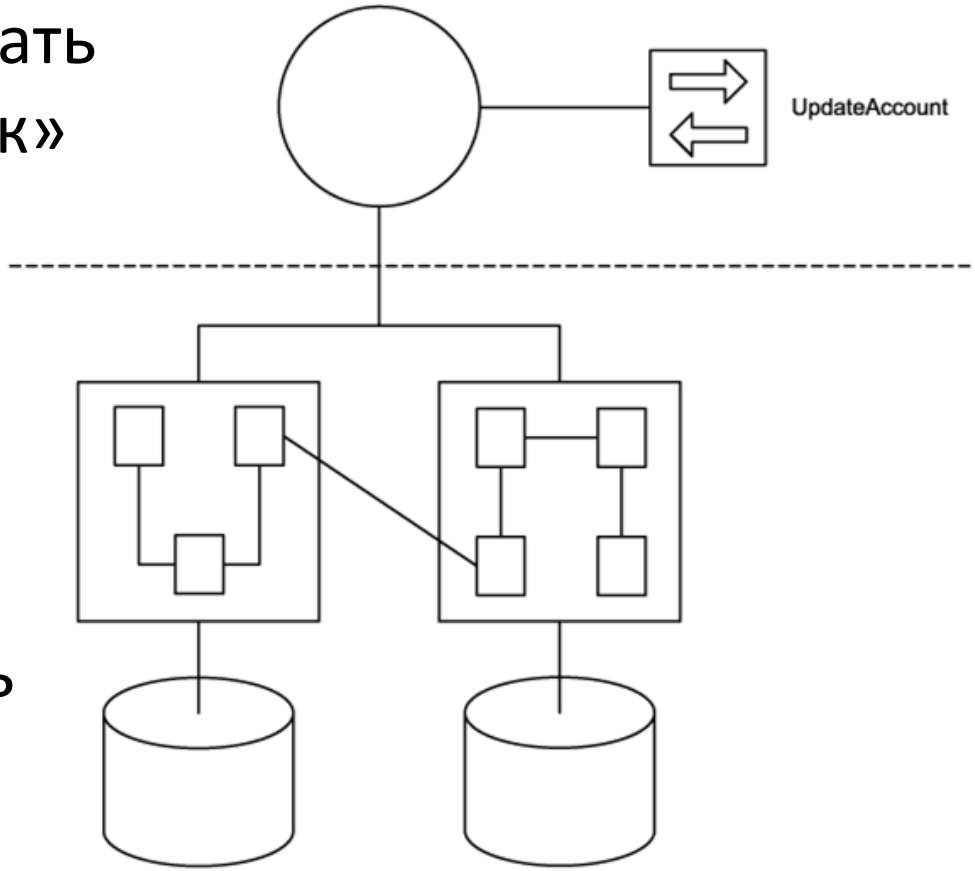
- Необходимо обеспечить целостность системы в рамках развития системы, независимо от пути развития
- Система сервисов является слабо связанной если сервис может приобретать знания о другом сервисе, оставаясь независимым от внутренней реализации логики данного сервиса.

3. Сервисы должны быть слабосвязанны(2)



4. Сервисы должны абстрагировать внутреннюю логику

- Каждый сервис должен действовать как «черный ящик»

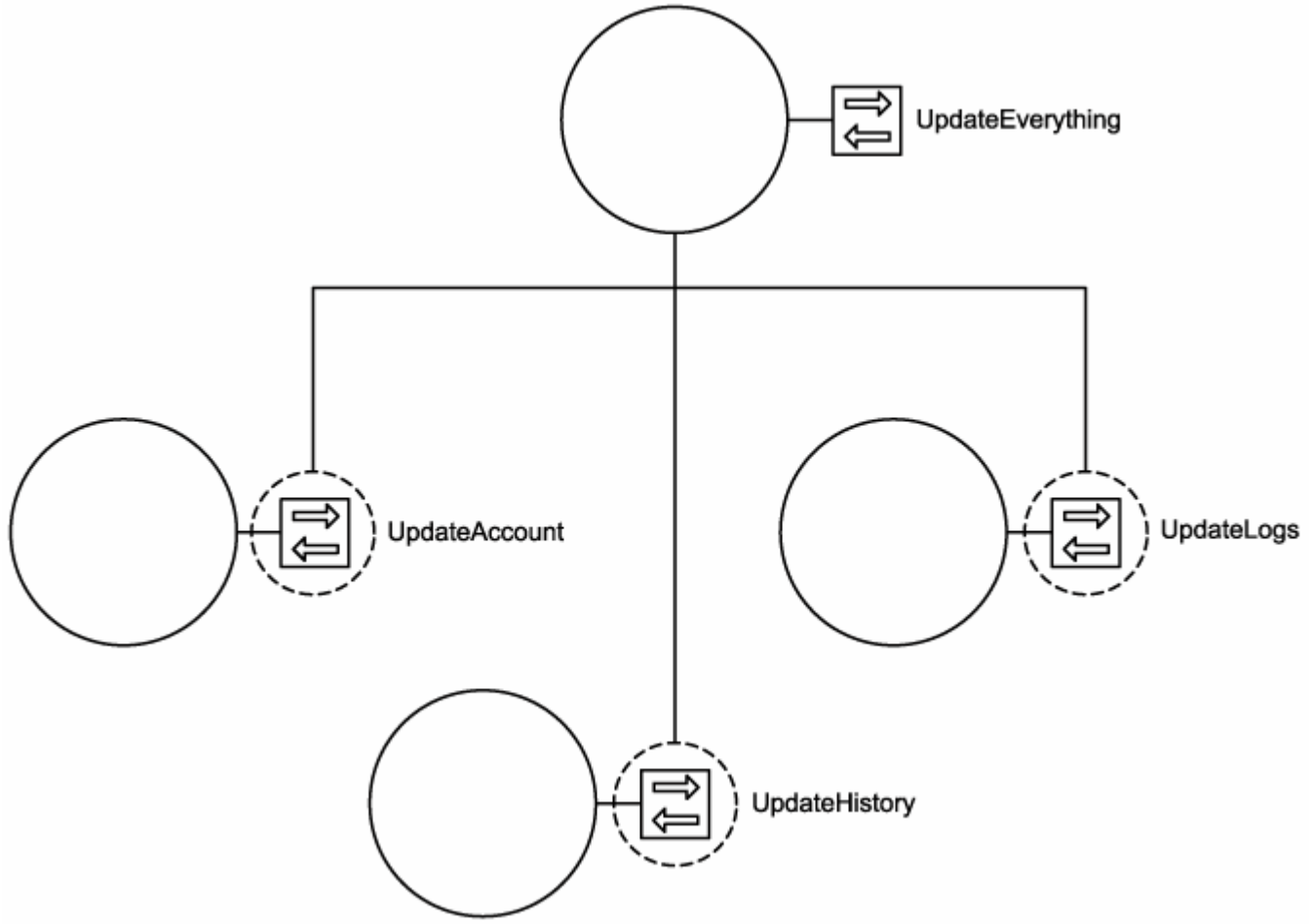


- Это одно из требований, обеспечивающих слабосвязанность сервисов.

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).

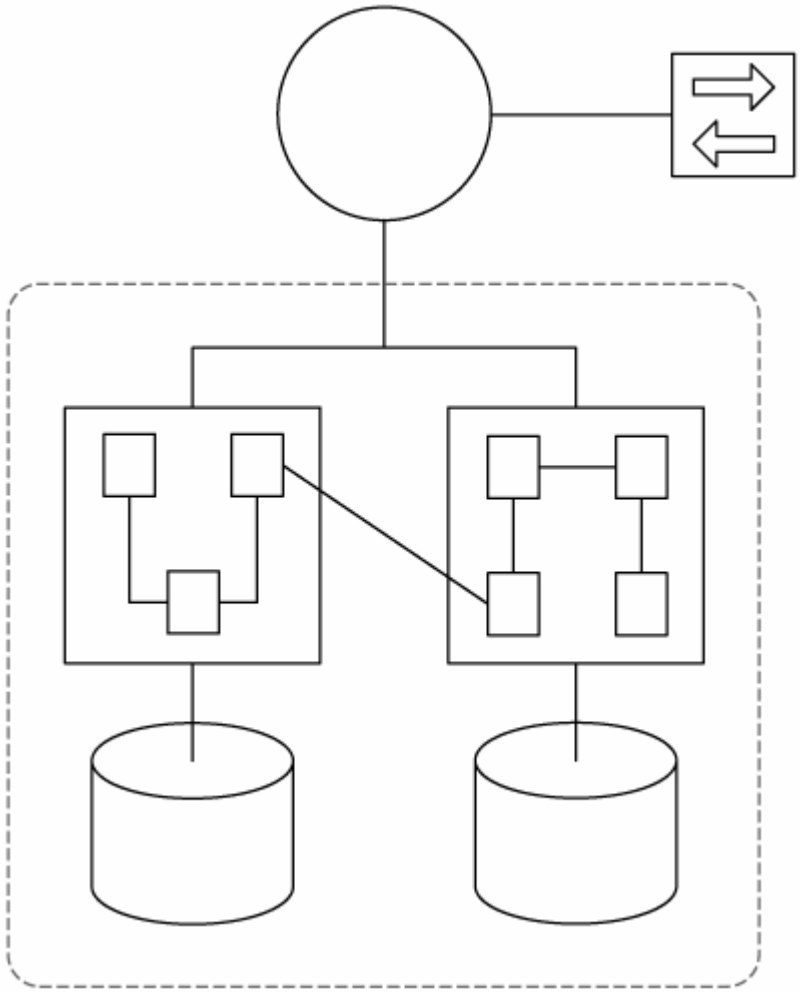
5. Сервисы должны быть совместимы (2)



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)

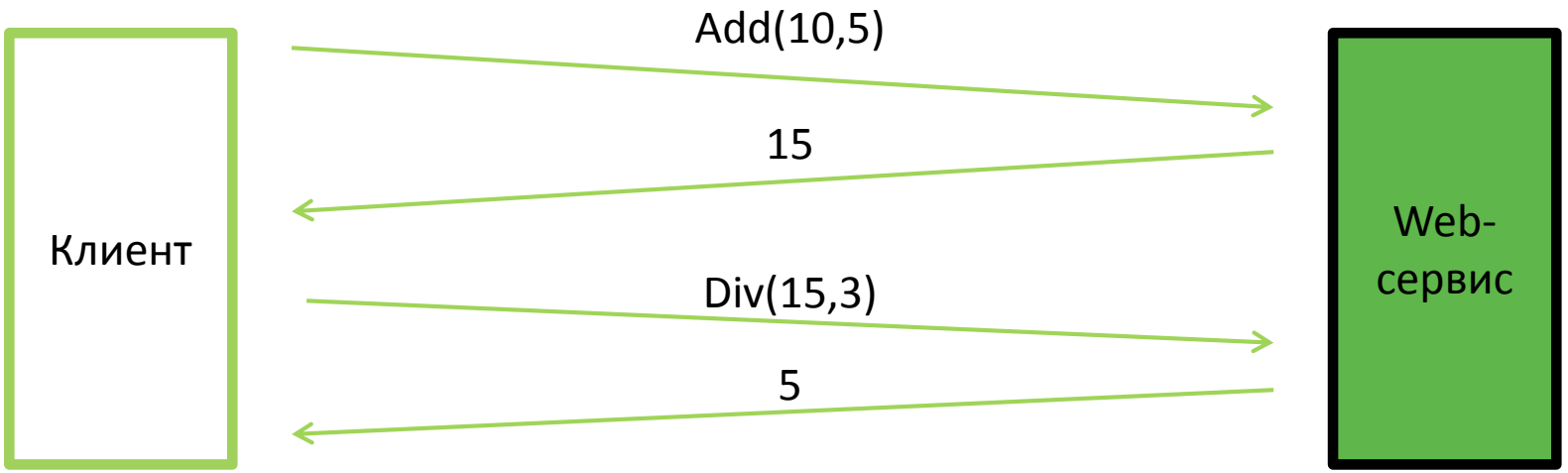


7. Сервисы не должны использовать информацию о состоянии (1)

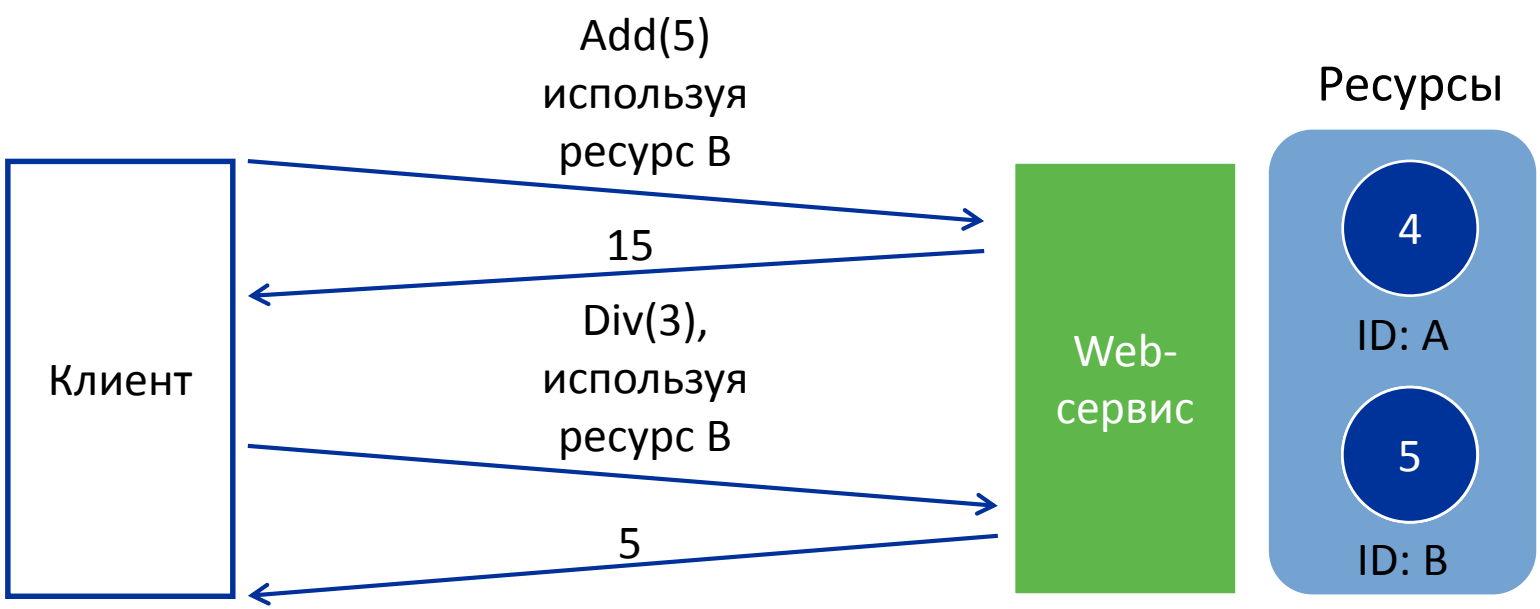
- «Чистые» сервисы должны значительно ограничивать объем и время хранения информации (в идеале – только на время вычислений)
- Не зависимость от состояния (Statelessness) позволяет повысить возможности масштабируемости и повторного использования сервисов
- Но это очень жесткое ограничение, которое не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

© Радченко Г.И., кафедра СП ЮУрГУ

7. Сервисы не должны использовать информацию о состоянии (2)



7. Сервисы не должны использовать информацию о состоянии (3)



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

© Радченко Г.И., кафедра СП ЮУрГУ