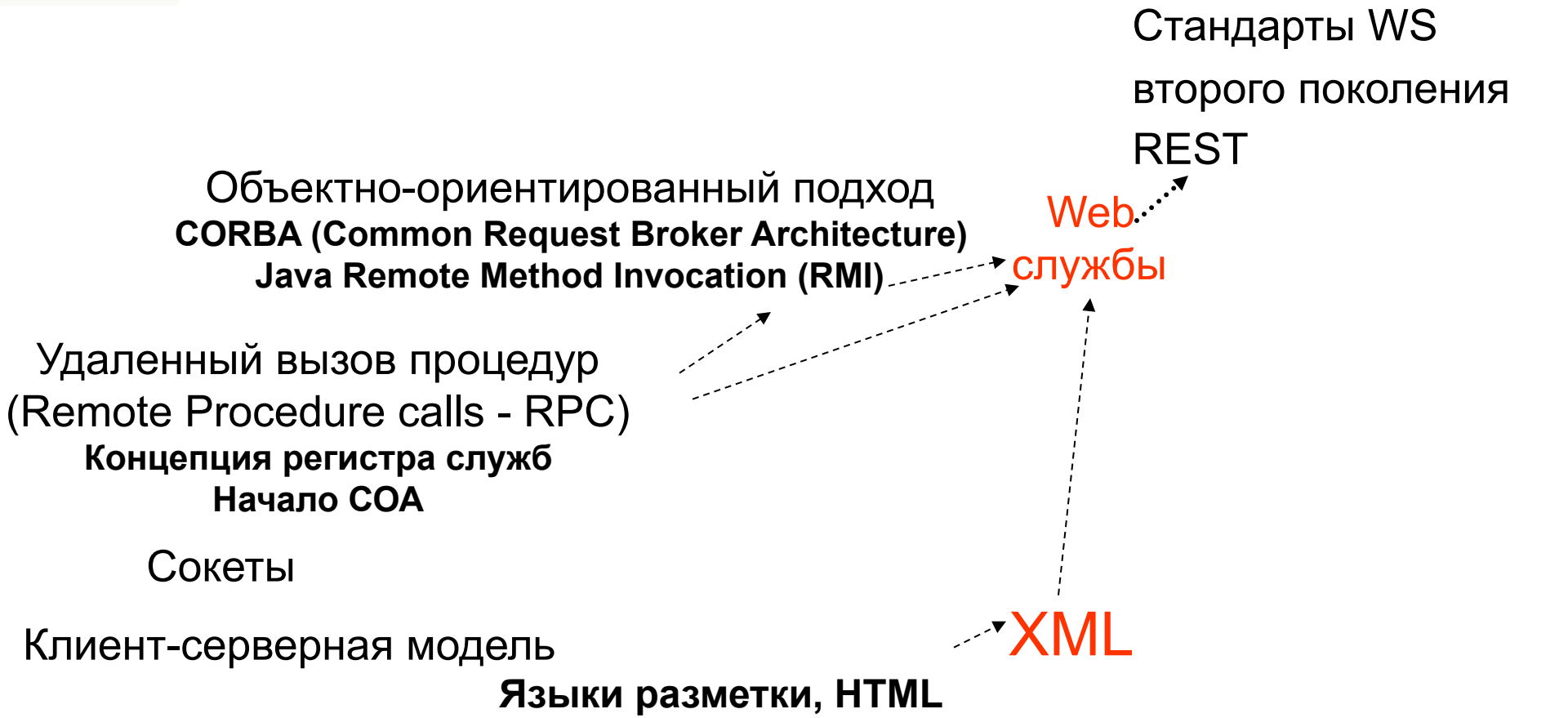
Распределенные вычислительные системы

Сервис-ориентированная архитектура

Краткая история PBC

Распределенные вычисления



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

Технология RPC (удаленный вызов процедур)

- Реализация поддержки клиент-серверной архитектуры
- Обеспечивает возможность локального приложения вызвать процедуру на удаленном компьютере и получить результаты работы процедуры
- Примеры реализации:
  - Sun RPC; .Net Remouting; Java RMI; CORBA

Недостатки RPC

- Отсутствие универсальных, стандартизованных интерфейсов
- Двоичный формат обмена информацией
- Отсутствие функциональной совместимости различных стандартов
- Сложности с организацией взаимодействия посредством Internet

Появление Web служб (Web Services)

- Представлены в 2000 г.
- Это основанная на XML платформенно-независимая технология поддержки распределенных вычислений.
- Позволяет взаимодействовать клиентам и серверам независимо от платформы реализации и языка программирования.

Области применения Web-служб

- Осуществление совместного использования информации путем объединения внутренних ресурсов отдельных компаний (B2B)
- Готовые модули для разработчиков
- Обеспечение дополнительных возможностей клиентских приложений
- Инструменты для обеспечения взаимодействия программ в рамках одной компании

Сервис-ориентированная архитектура распределенных систем

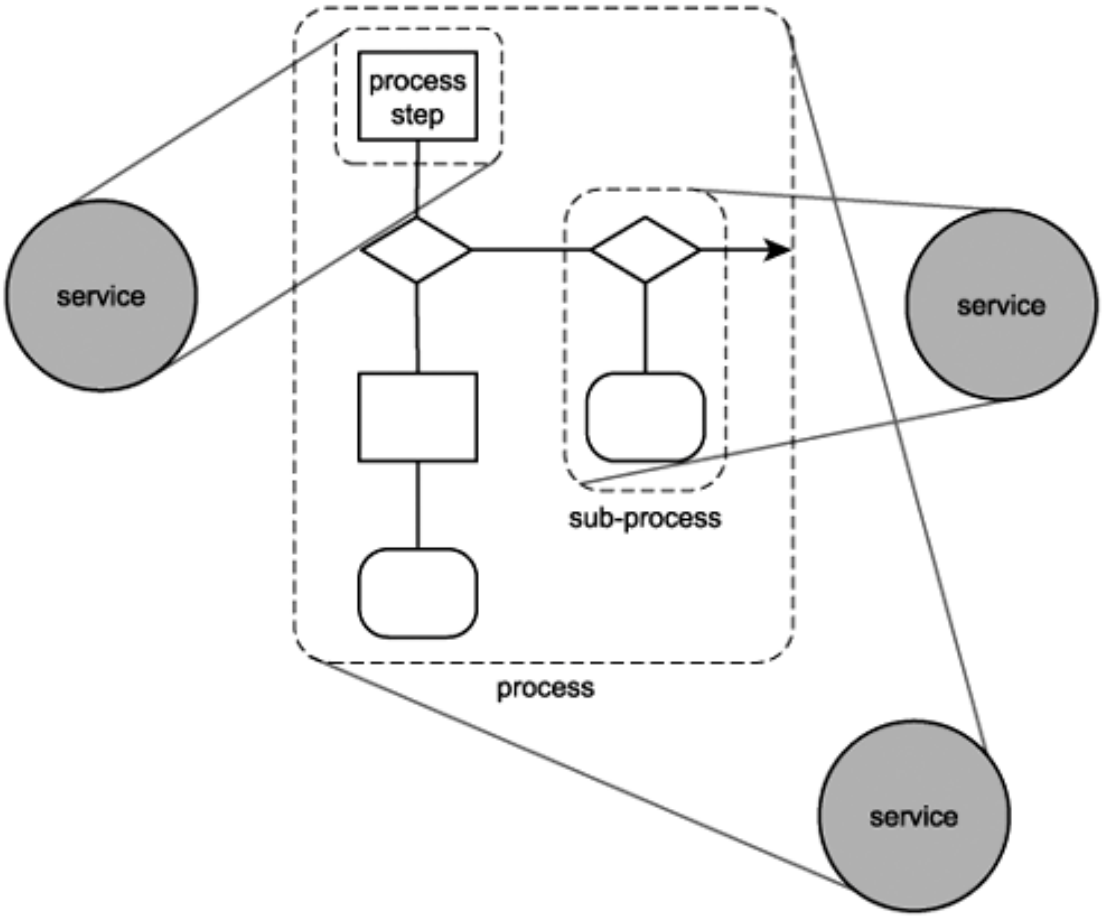
Сервис-ориентированная архитектура

- ***Сервис ориентированная архитектура*** – это парадигма организации и использования распределенных функциональных возможностей, которые могут предоставляться различными владельцами
  - Определение организации OASIS (Organization for the Advancement of Structured Information Standards)

SOA VS ООП

- В основе объектно-ориентированной архитектуры лежит сущность «объекта»
- В основе SOA лежит «действие»

Сервисы инкапсулируют действие



Составляющие сервис-ориентированной архитектуры

1. Сервисные компоненты (сервисы)
2. Контракты сервисов (интерфейсы)
3. Соединители сервисов (транспорт)
4. Механизмы обнаружения сервисов (регистры)

Сервисные компоненты

- Это ***программные компоненты***, обеспечивающие прозрачную сетевую адресацию.
- ***Сервисы*** – это открытые, самоопределяющиеся компоненты, поддерживающие быстрое построение распределенных приложений.
- Сервисы могут быть *мелкомодульными* и *крупномодульными*.

Мелкомодульный сервис

- ***Мелкомодульный сервис*** предоставляет элементарный объем функциональной нагрузки и обеспечивает высокую степень повторного использования.

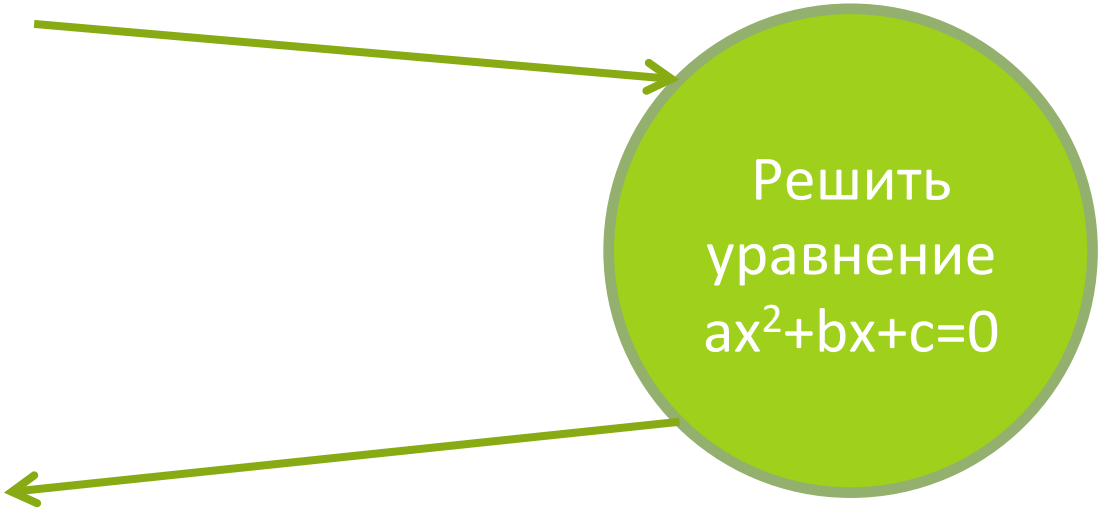
- Необходимо координировать работу нескольких сервисов для получения результата

Крупномодульный сервис

- ***Крупномодульный сервис*** обеспечивает высокую степень *инкапсуляции* функциональности.
- Но затрудняет повторное использование, в связи с узкой специализацией

Крупномодульный сервис решения квадратных уравнений

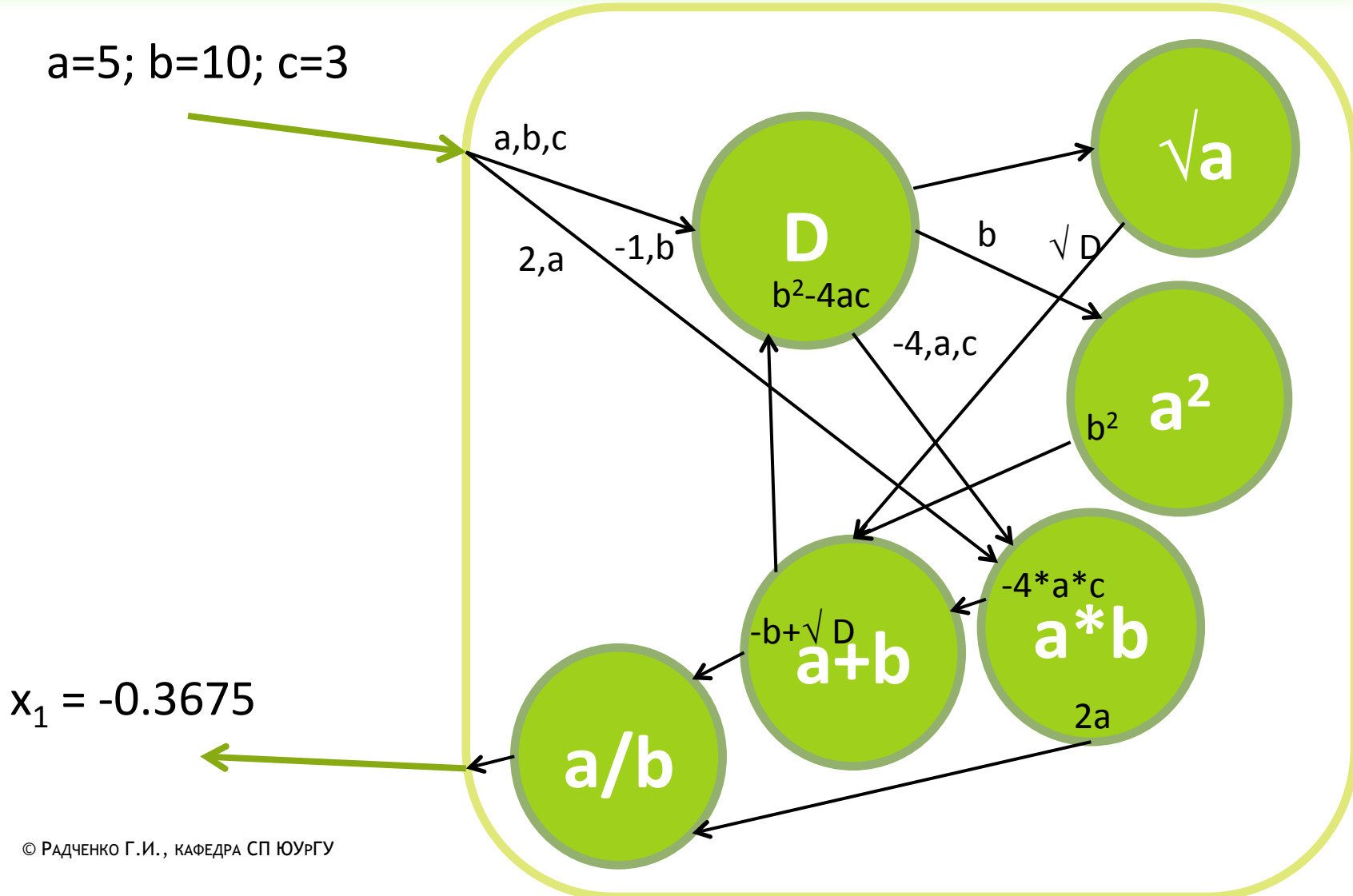
a=5; b=10; c=3



$x_1 = -0.3675$

$x_2 = -1.6325$

Оркестрация мелкомодульных сервисов для решения квадратных уравнений



Контракты сервиса

- ***Интерфейс*** – это описание возможностей, предоставляемых конкретным сервисом.

- В интерфейсе определяется:
  - Формат сообщений
  - Входные и выходные параметры методов

Соединитель сервисов

- ***Транспорт*** обеспечивает обмен информацией между компонентами.
- Использование гибких транспортных протоколов для обмена информацией между сервисными компонентами позволяет повысить программную совместимость сервис ориентированной системы

Механизмы обнаружения (регистры сервисов)

- Используются для поиска сервисов с требуемой функциональностью.
- *Статическое* обнаружение: ориентированы на хранение информации о редко изменяющихся системах
  - телефонная АТС, UDDI
- *Динамическое* обнаружение: системы в которых наблюдается частое появление и исчезновение сервисных компонентов:
  - P2P, мобильные агенты.

Слабо- и сильносвязанные программные системы

Классификация систем по связности

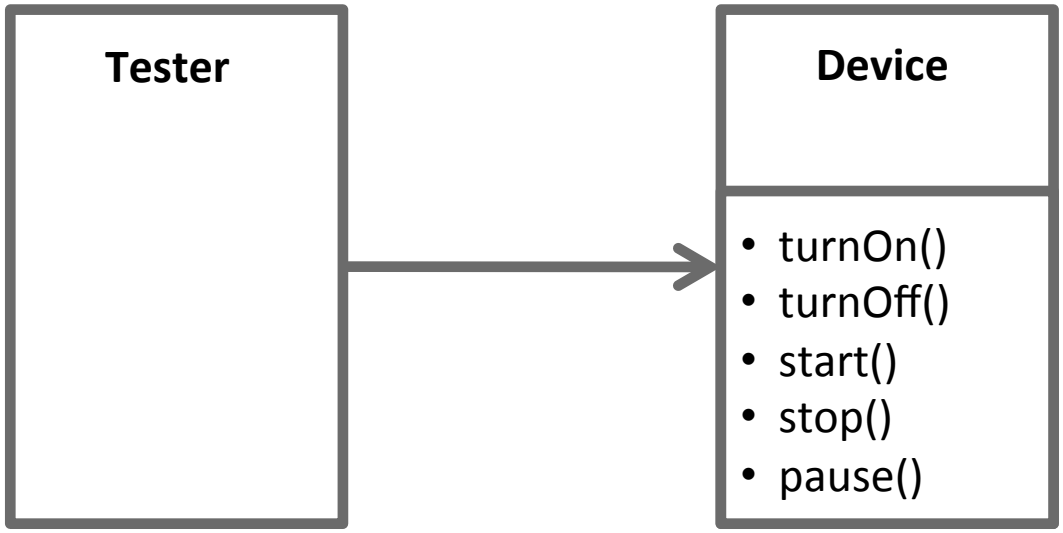
Связанность – это степень знания и зависимости одного объекта от внутреннего содержания другого.

Программные системы можно разделить на 2 типа:

1. **Сильносвязанные системы** *(Strong coupling)*
2. **Слабосвязанные системы** *(Loose coupling)*

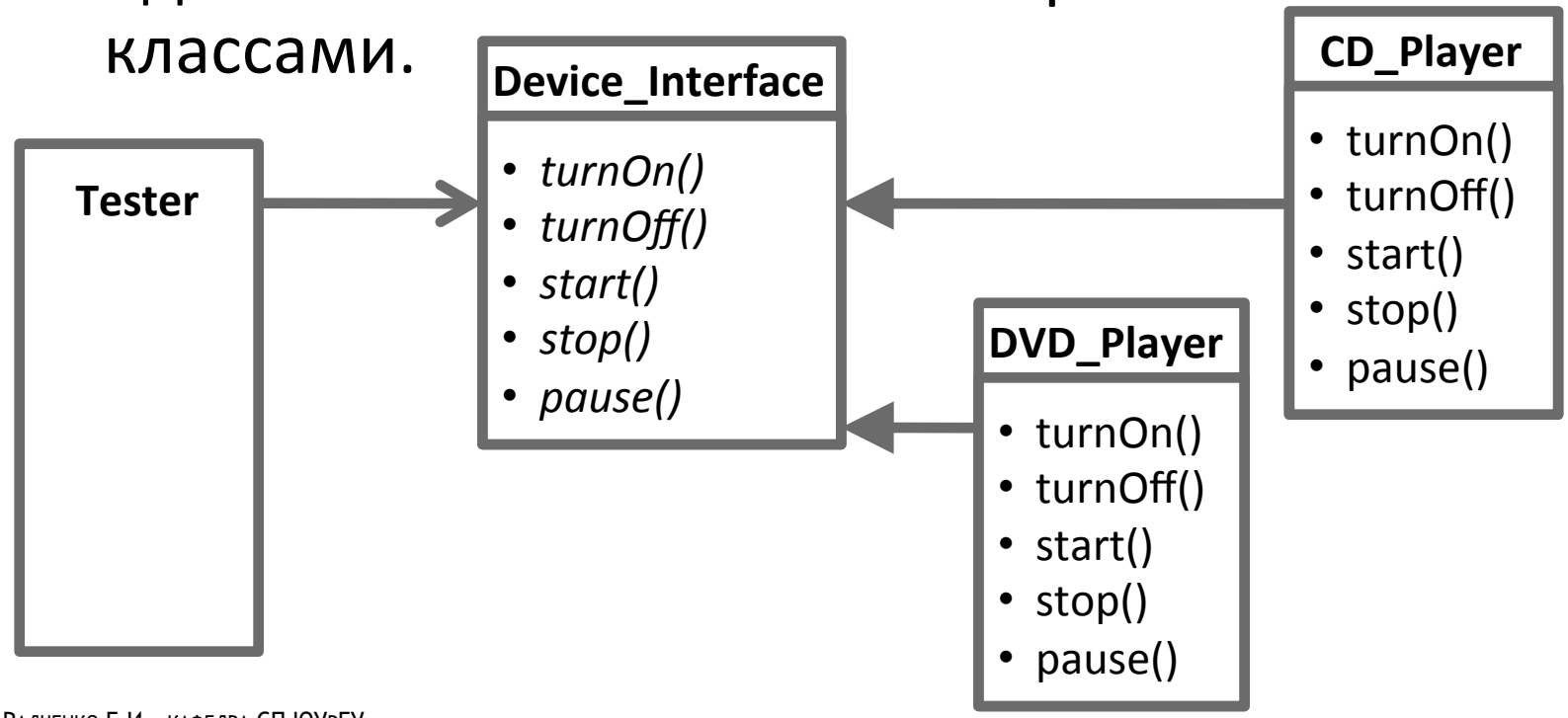
Сильносвязанные вычислительные системы

- *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

Сравнение слабо- и сильносвязанных систем

	Сильносвязанные системы	Слабосвязанные системы
Физические соединения	Точка-точка	Через посредника
Стиль взаимодействий	Синхронные	Асинхронные
Модель данных	Общие сложные типы	Простые типы
Связывание	Статическое	Динамическое
Платформа	Сильная зависимость от базовой платформы	Независимость от платформы
Развертывание	Одновременное	Постепенное