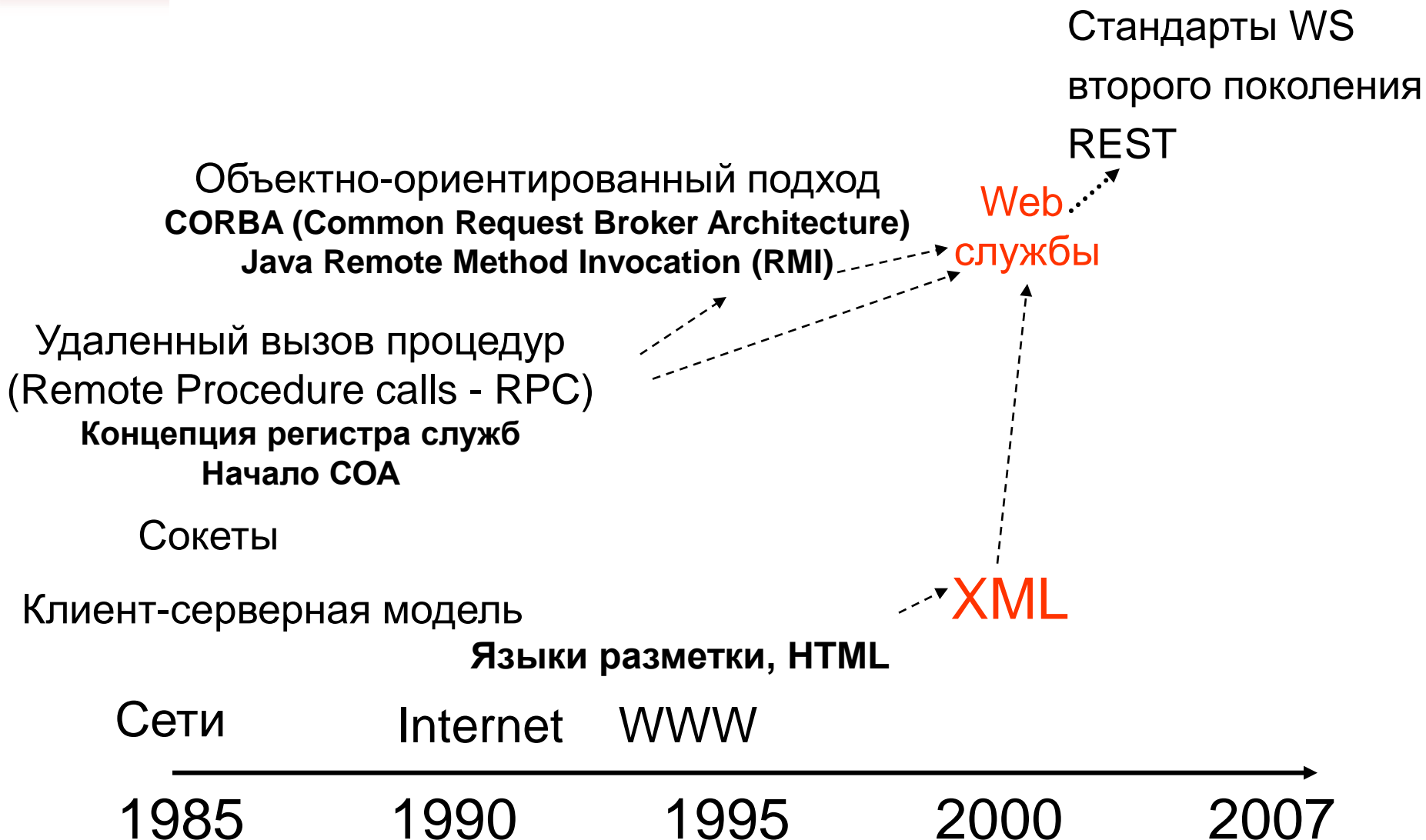


РАСПРЕДЕЛЕННЫЕ ОБЪЕКТНЫЕ ТЕХНОЛОГИИ

Сервис-ориентированная архитектура

КРАТКАЯ ИСТОРИЯ РВС

РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ



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

5

- ⊙ 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

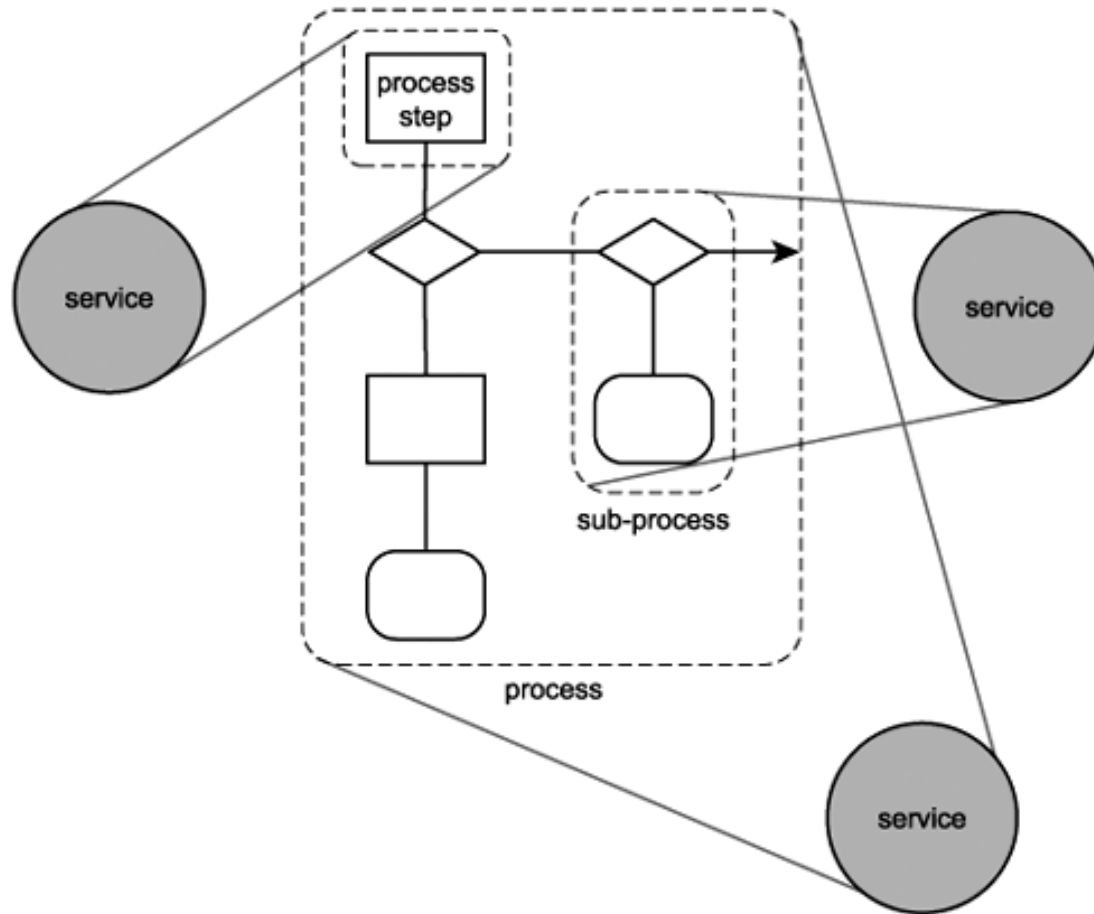
СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА

- ◎ ***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)

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

СЕРВИСЫ ИНКАПСУЛИРУЮТ ДЕЙСТВИЕ

11



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

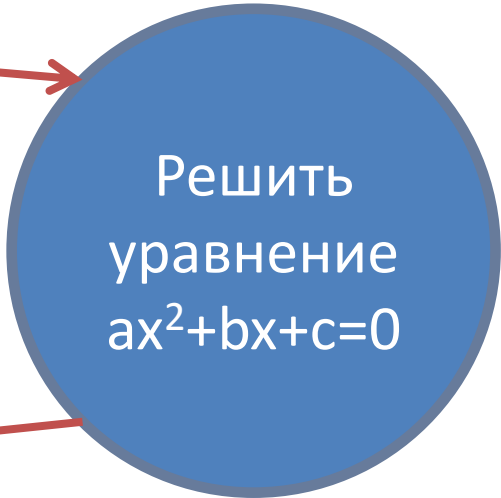
- ◎ ***Мелкомодульный сервис***
предоставляет элементарный объем функциональной нагрузки и обеспечивает высокую степень повторного использования.
- ◎ Необходимо координировать работу нескольких сервисов для получения результата

Крупномодульный сервис

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

Крупномодульный сервис РЕШЕНИЯ КВАДРАТНЫХ УРАВНЕНИЙ

$$a=5; b=10; c=3$$

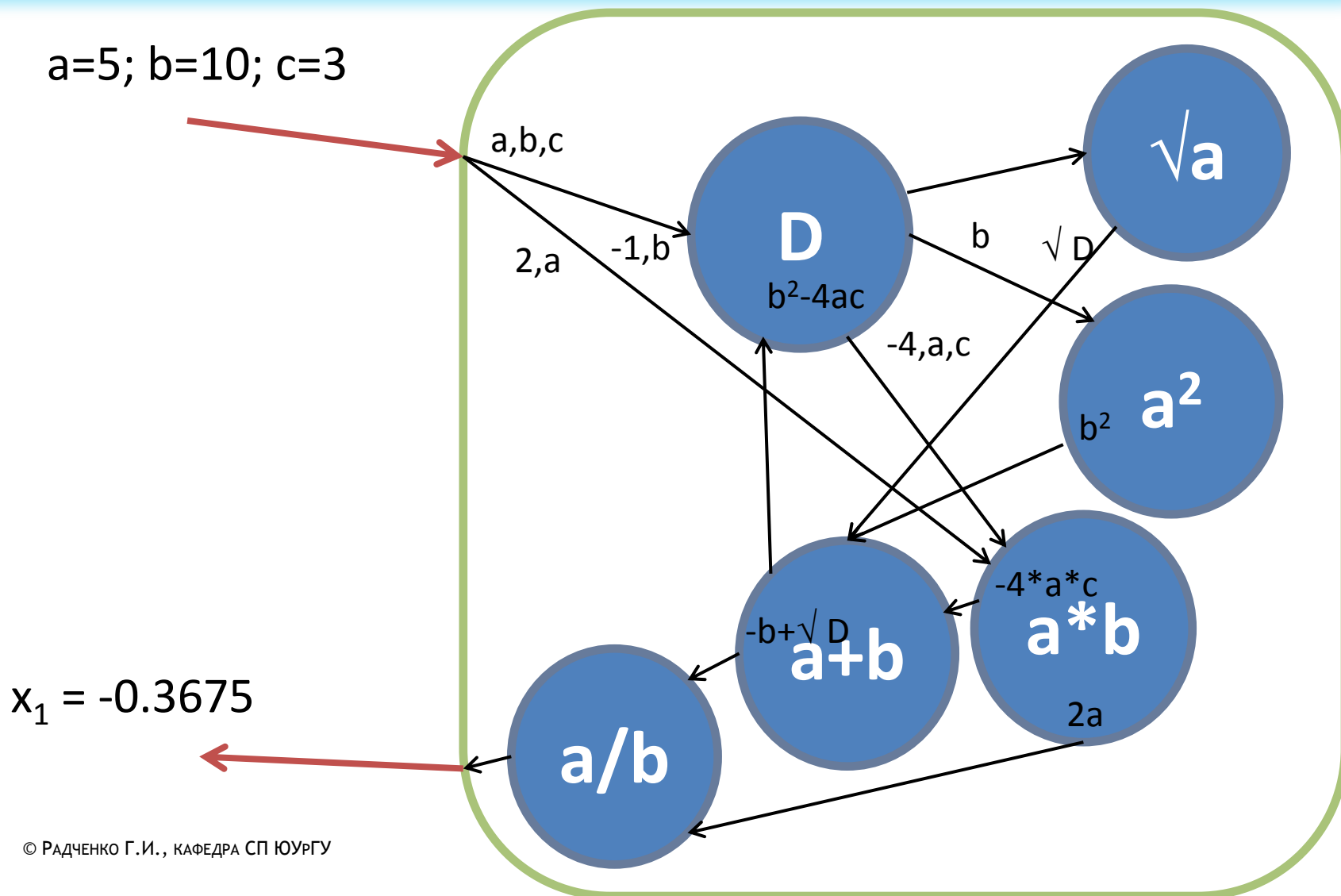


Решить
уравнение
 $ax^2+bx+c=0$

$$x_1 = -0.3675$$
$$x_2 = -1.6325$$

ОРКЕСТРАЦИЯ МЕЛКОМОДУЛЬНЫХ СЕРВИСОВ ДЛЯ РЕШЕНИЯ КВАДРАТНЫХ УРАВНЕНИЙ

$a=5; b=10; c=3$



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

22

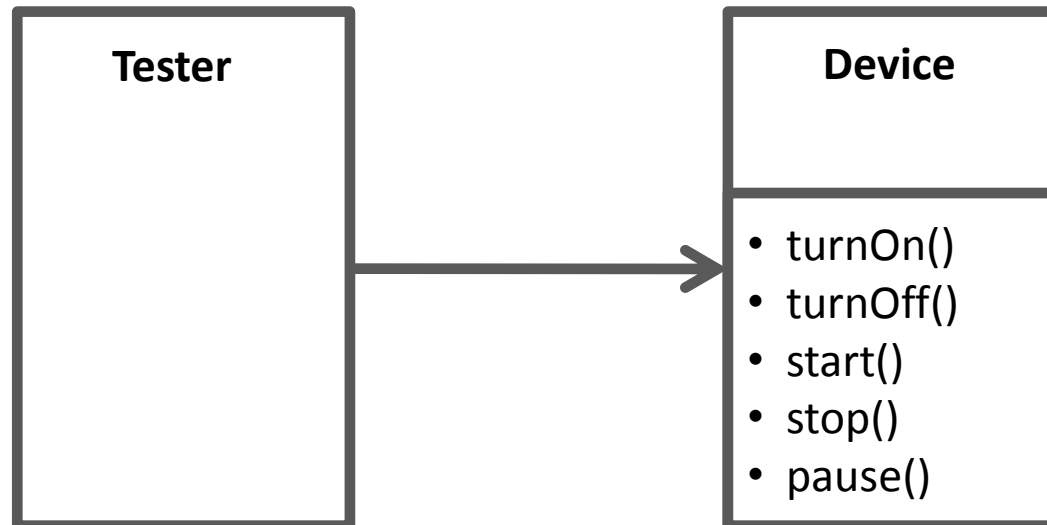
Связанность – это степень знания и зависимости одного объекта от внутреннего содержания другого.

Программные системы можно разделить на 2 типа:

- 1. Сильносвязанные системы (*Strong coupling*)**
 - ⊙ Java RMI, .NET Remoting и т.п.
- 2. Слабосвязанные системы (*Loose coupling*)**
 - ⊙ SOA

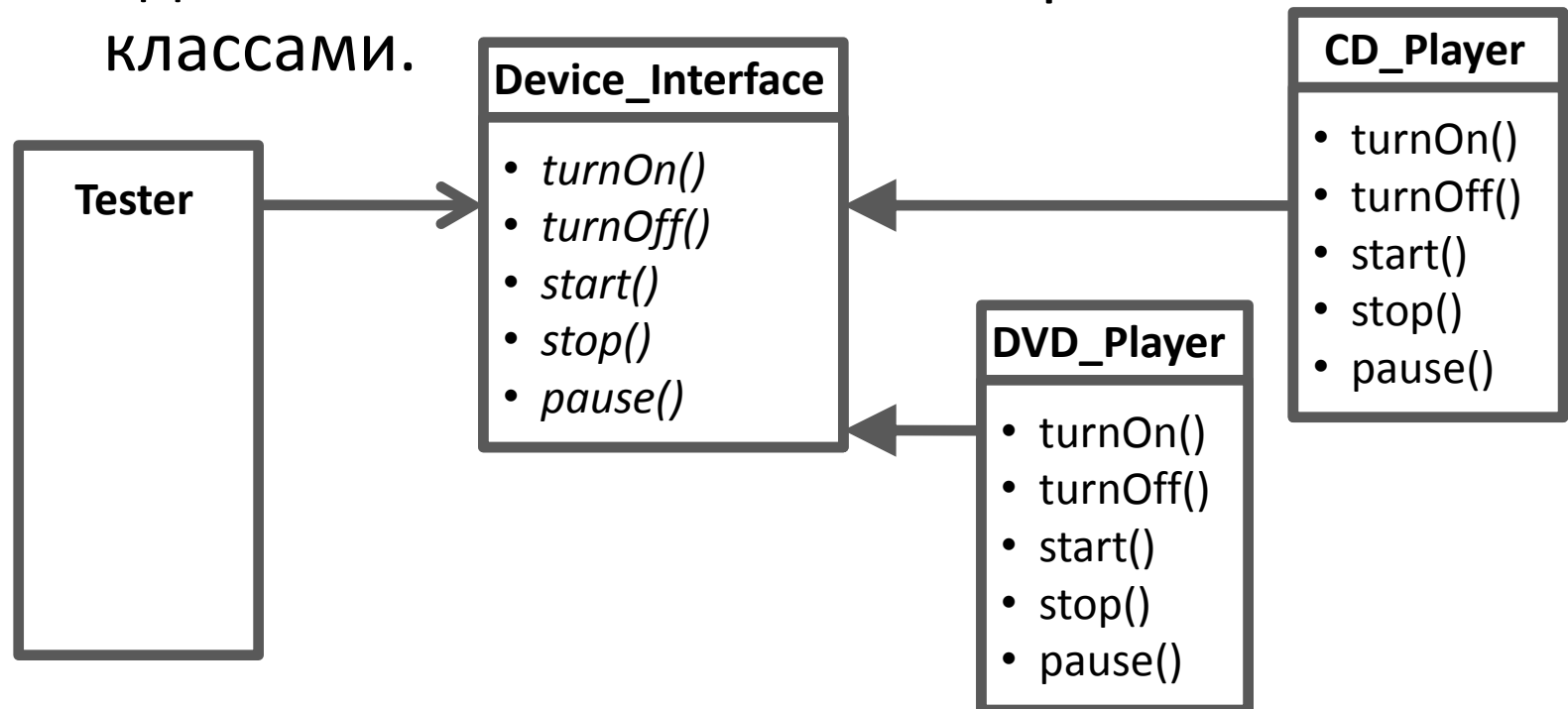
СИЛЬНОСВЯЗАННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

- ◎ *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

- ① Уменьшается количество связей между компонентами системы, уменьшая объем возможных последствий в связи со сбоями
- ① Обеспечивается возможность расширения рабочей системы, путем создания новых классов, обладающих единым интерфейсом

СРАВНЕНИЕ СЛАБО- И СИЛЬНОСВЯЗАННЫХ СИСТЕМ

	Сильно связанные 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