Архитектура программных систем

Алексей Островский

Физико-технический учебно-научный центр НАН Украины

7 ноября 2014 г.

Архитектура ПО

Определение

Архитектура программного проекта — высокоуровневое представление структуры системы и спецификация ее компонентов и логики их взаимодействия.

Преимущества использования архитектуры ПО:

- основа для анализа системы на ранних этапах ее разработки;
- основа для повторного использования компонентов и решений;
- упрощение принятия решений касательно разработки, развертывания и поддержки ПО;
- упрощение диалога с заказчиком;
- уменьшение рисков и снижение затрат на производство ПО.

Виды архитектуры ПО

- Архитектура отдельных программ.

  Цель: определить разбиение программы на составляющие.

  Составляющие = функциональные требования

  Архитектура = нефункциональные требования

- Архитектура сложных систем.

  Цель: определить структуру взаимодействия системы с другими системами, программами и компонентами; определить местоположение системы в распределенной среде.

  (будет рассмотрена в весеннем семестре)

Архитектурные решения

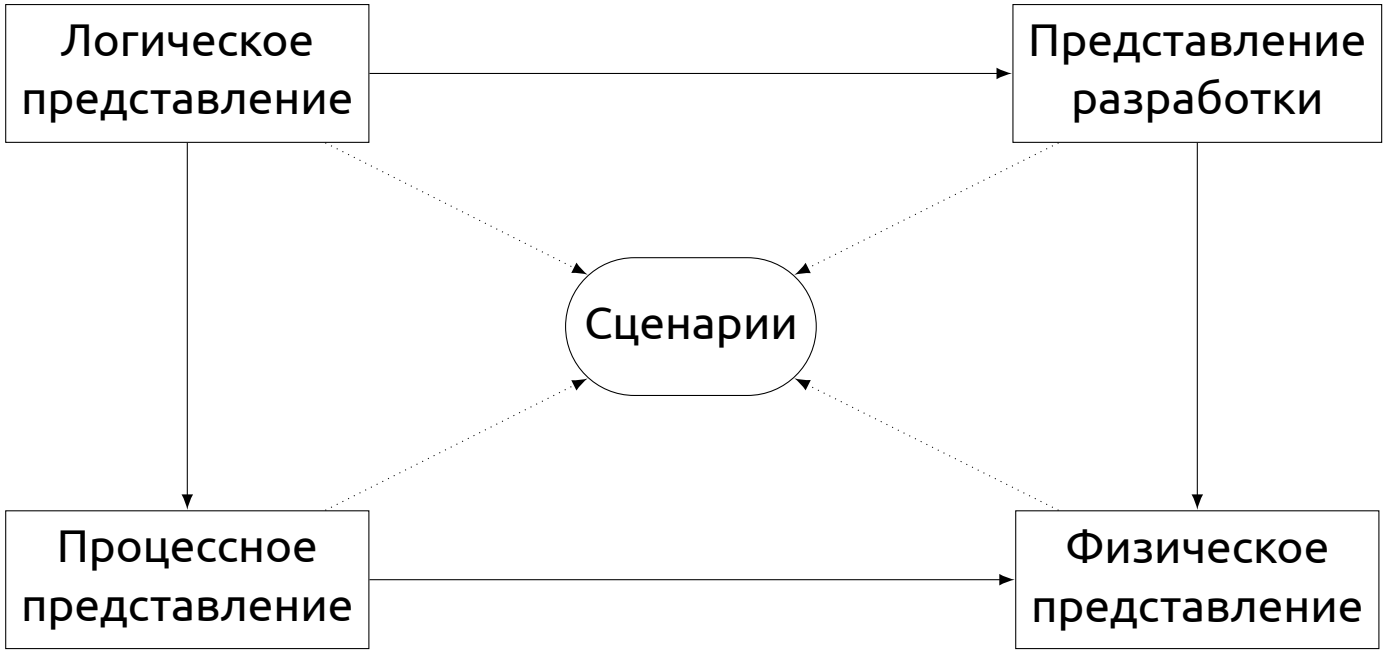
Вопросы, касающиеся архитектуры ПО:

- распределение программы по нескольким потокам выполнения;
- определение структуры системы (выделение компонентов и субкомпонентов);
- контроль над выполнением компонентов;
- выполнение нефункциональных требований к программе;
- использование готовых шаблонов архитектуры и проектирования;
- документирование архитектуры.

Требования и архитектура

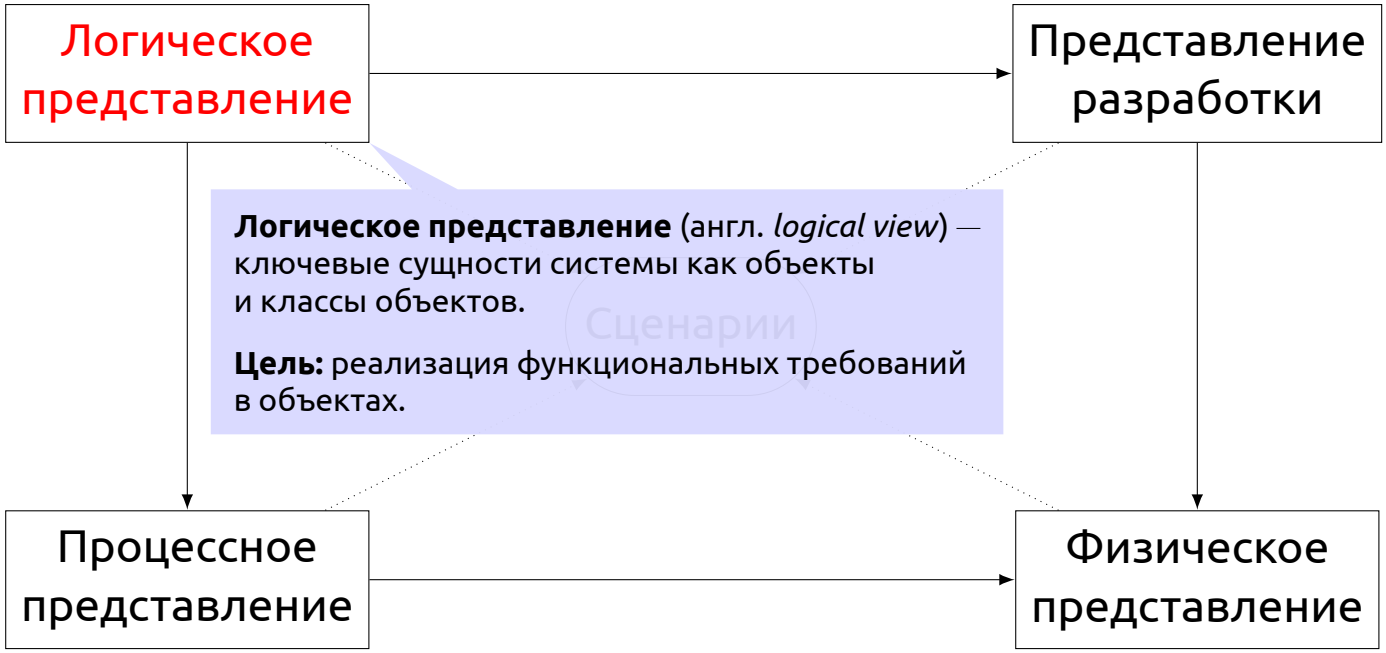
Требование	Влияние на архитектуру
Производительность	преимущественно большие компоненты, отказ от распределенности, параллельное исполнение кода.
Безопасность	использование многослойной архитектуры; локализация механизмов авторизации в небольшом числе компонентов.
Доступность (*availability*)	наличие избыточных компонентов; локализация сбоев.
Удобство сопровождения (*maintainability*)	использование малых компонентов; минимизация зависимостей; отсутствие совместного использования данных.

Представления архитектуры



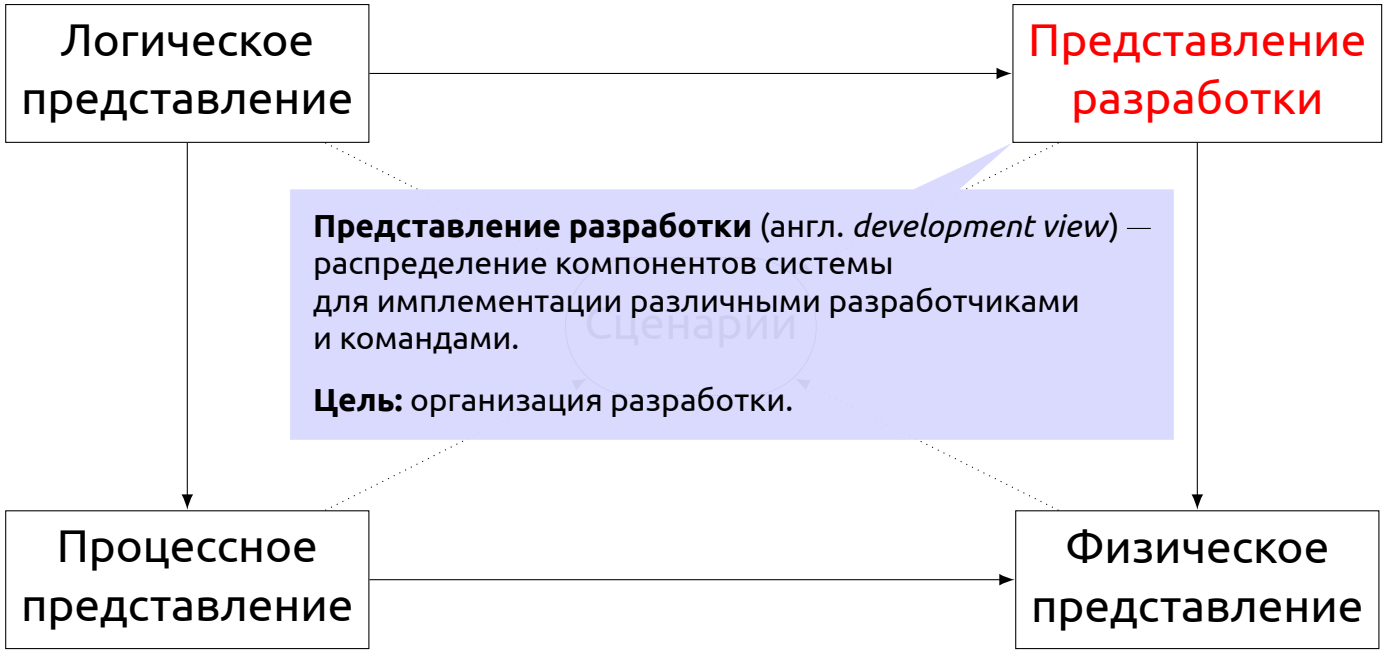
Модель представлений 4+1 [Kruchten, 1995]

Представления архитектуры



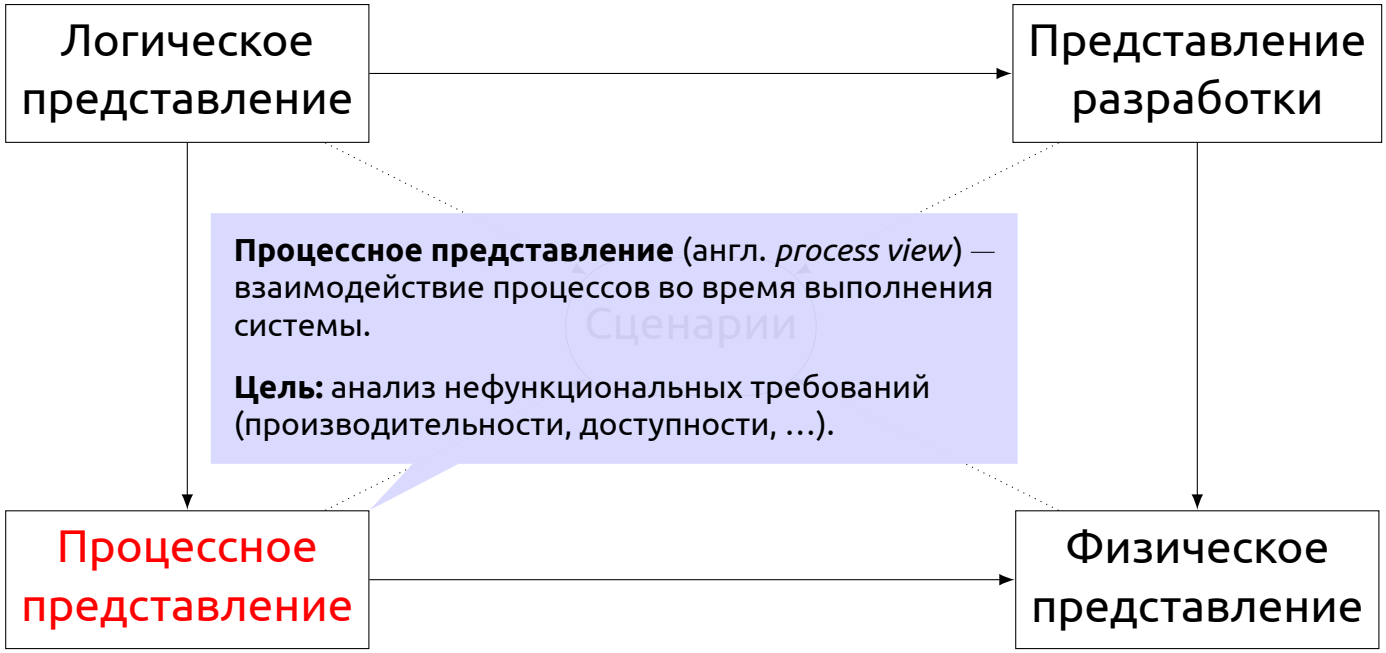
Модель представлений 4+1 [Kruchten, 1995]

Представления архитектуры



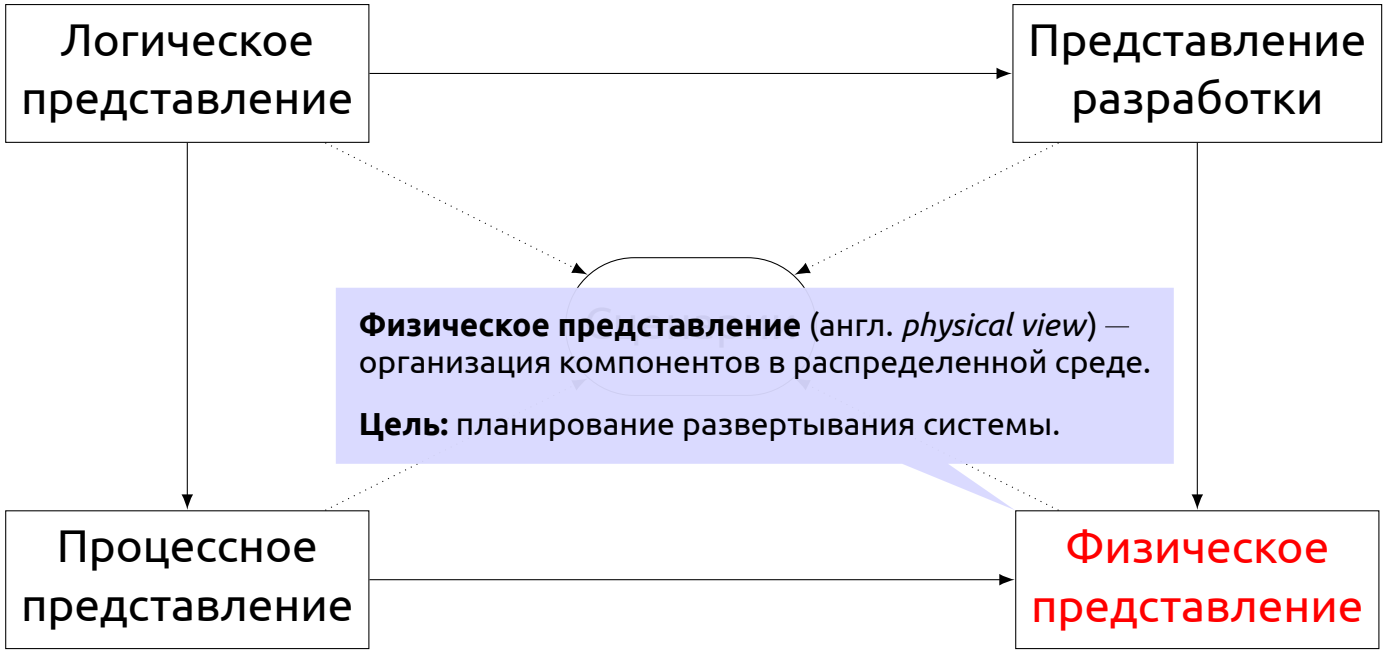
Модель представлений 4+1 [Kruchten, 1995]

Представления архитектуры



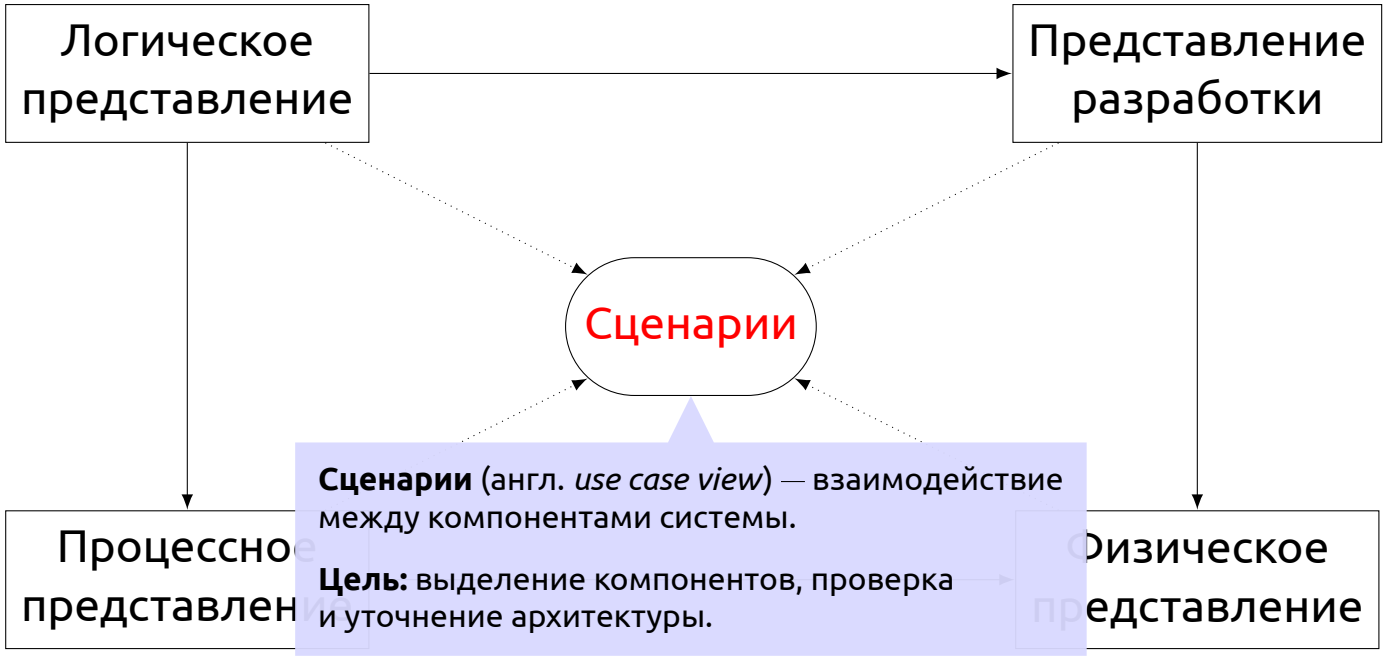
Модель представлений 4+1 [Kruchten, 1995]

Представления архитектуры



Модель представлений 4+1 [Kruchten, 1995]

Представления архитектуры



Модель представлений 4+1 [Kruchten, 1995]

Создание представлений

Место представлений в проектировании ПО:

- Упрощение обсуждения решений, касающихся проектирования системы.

  Потребители: заинтересованные стороны (напр., заказчики), менеджеры.

  Средства: неформальные представления (напр., блочные диаграммы).

- Документирование *уже разработанной* архитектуры для улучшения ее понимания и развития приложения.

  Потребители: разработчики, отдел сопровождения.

  Средства: формальные представления (UML, специализированные языки).

Agile development: детализованные представления *не нужны*.

Инструменты создания представлений

UML:

Представление	**Диаграмма UML**
логическое	классов, последовательности, коммуникации
процессное	деятельности
разработка	компонентов, пакетов
физическое	развертывания
сценарии	вариантов использования

Другие средства: языки описания архитектуры (англ. *architecture description language, ADL*) — AADL, C2, Darwin, Wright.

Архитектурные шаблоны

Определение

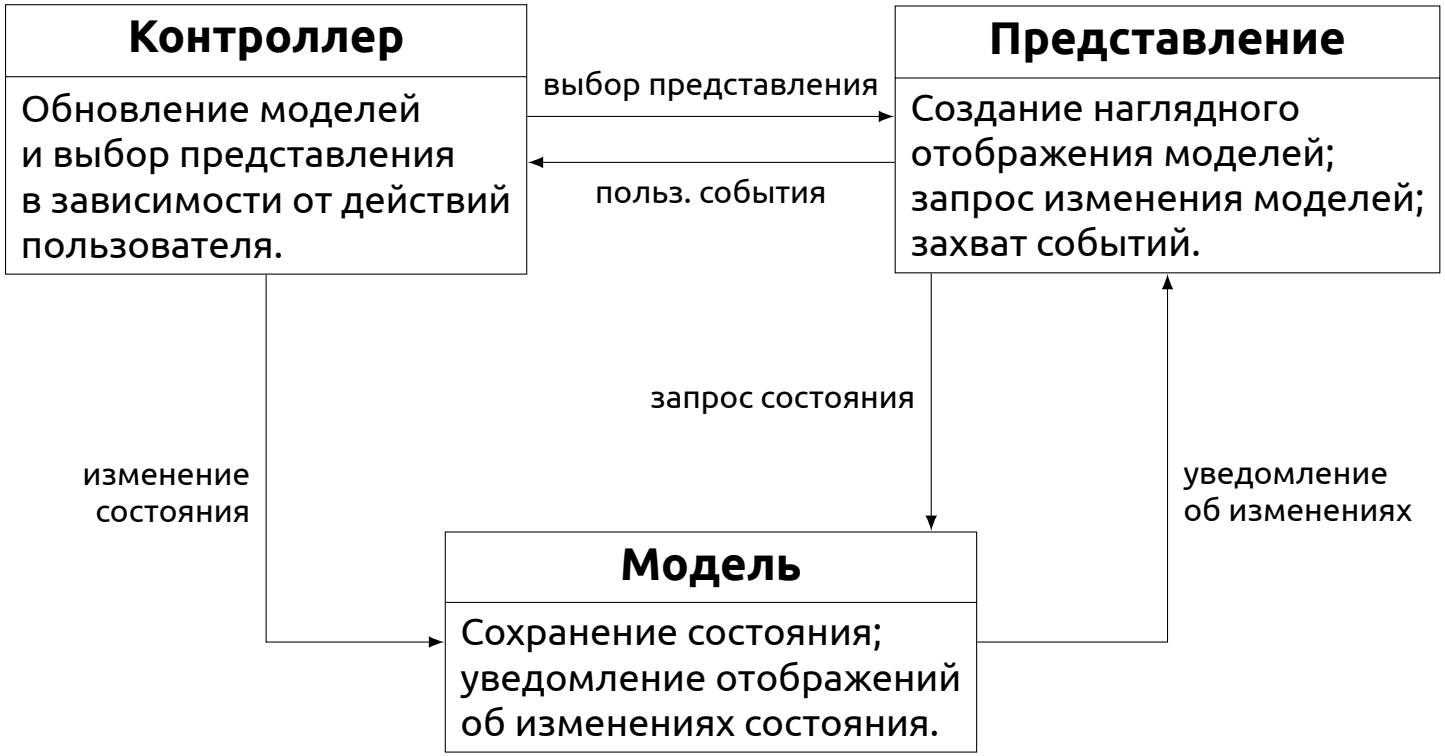
Архитектурный шаблон (англ. *architecture pattern*) — абстрактное описание применяющегося на практике подхода к организации программной системы.

Составляющие шаблона: описание, границы применения, сильные и слабые стороны, примеры.

Примеры шаблонов:

- Model — View — Controller (MVC);
- многослойная архитектура;
- клиент-серверная архитектура;
- конвейерная архитектура.

MVC



Общая модель архитектуры MVC

MVC — описание

Описание: Отделяет представление данных и взаимодействие с пользователем от хранимых данных.

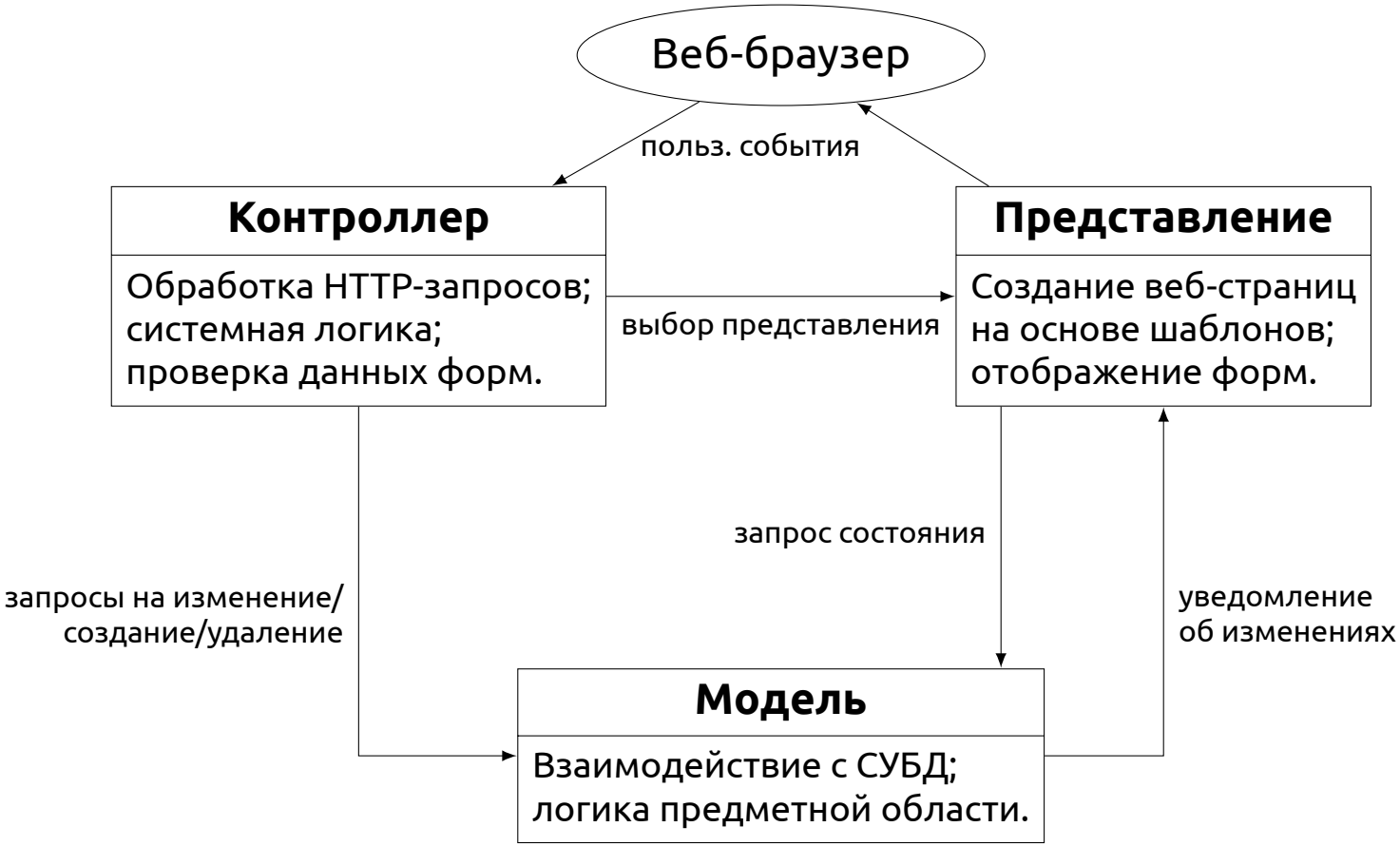
Применение: если необходимо несколько способов взаимодействия и/или отображения данных.

Примеры: веб-приложения; графические приложения (напр., библиотеки Qt или Android Development Kit).

Преимущества: независимость данных от их представления; изменение данных в одном представлении обновляет другие представления.

Недостатки: избыточное усложнение системы, если модель данных и взаимодействия ограничены.

MVC — пример



Архитектура MVC для веб-приложений

Многослойная архитектура

Общая модель четырехслойной архитектуры ПО

Многослойная архитектура — описание

Описание: Выделяет в системе несколько слоев, представляющих различные уровни детализации. Компоненты из каждого слоя используются в следующем.

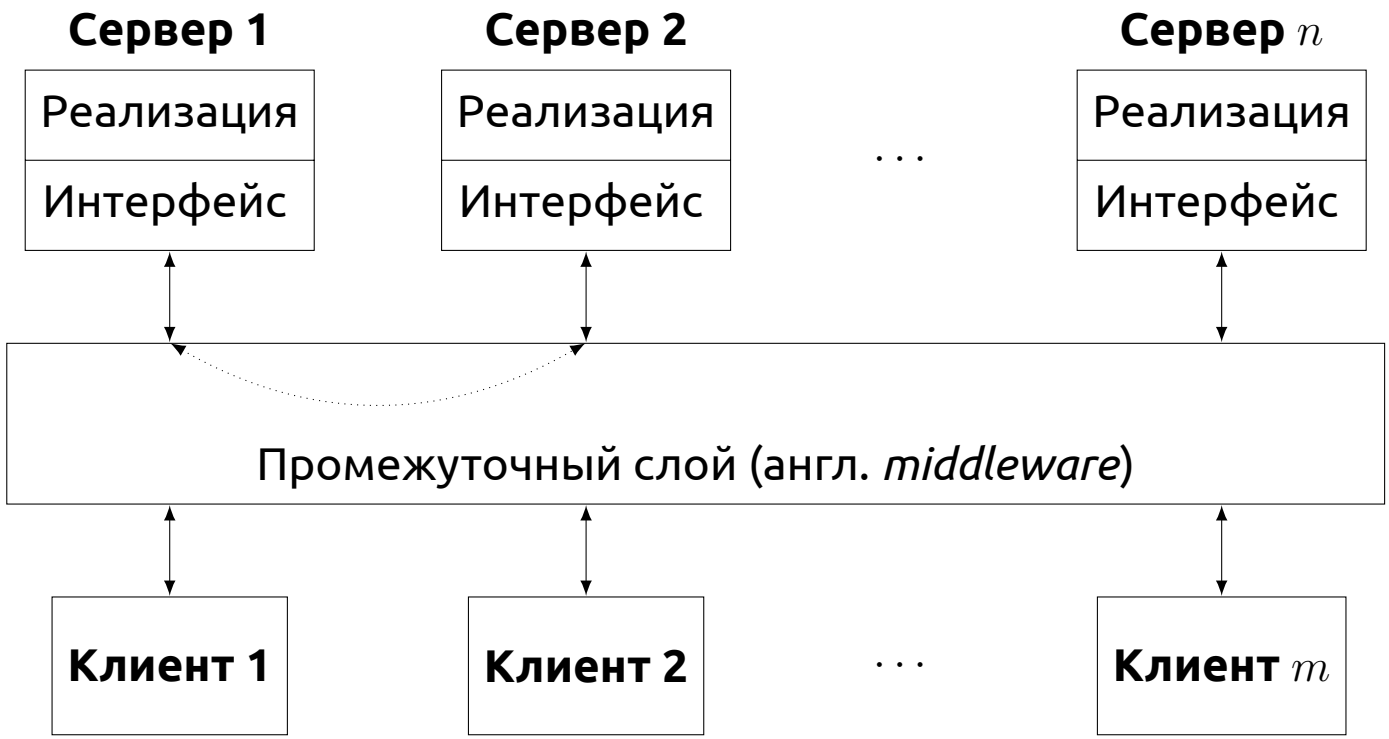
Применение: разработка на основе существующей системы; разработка несколькими командами; высокие требования к защищенности.

Примеры: операционные системы.

Преимущества: модульность системы (возможность замены отдельных слоев); дублирование функциональности на разных уровнях для повышения отказоустойчивости.

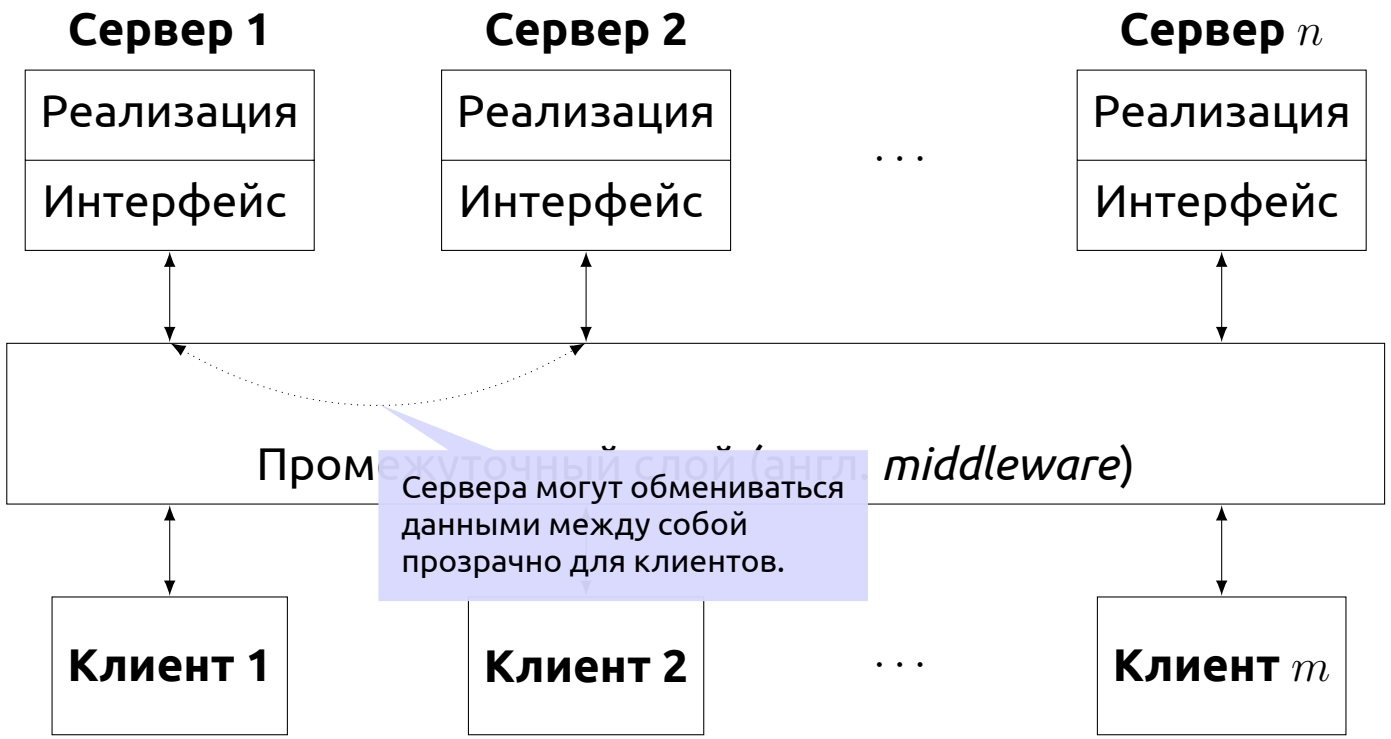
Недостатки: сложность размежевания уровней; снижение производительности.

Клиент-серверная архитектура



Общая модель клиент-серверной архитектуры

Клиент-серверная архитектура



Общая модель клиент-серверной архитектуры

Клиент-серверная архитектура — описание

Описание: Функциональность системы организована в виде сервисов, соответствующих различным серверам.

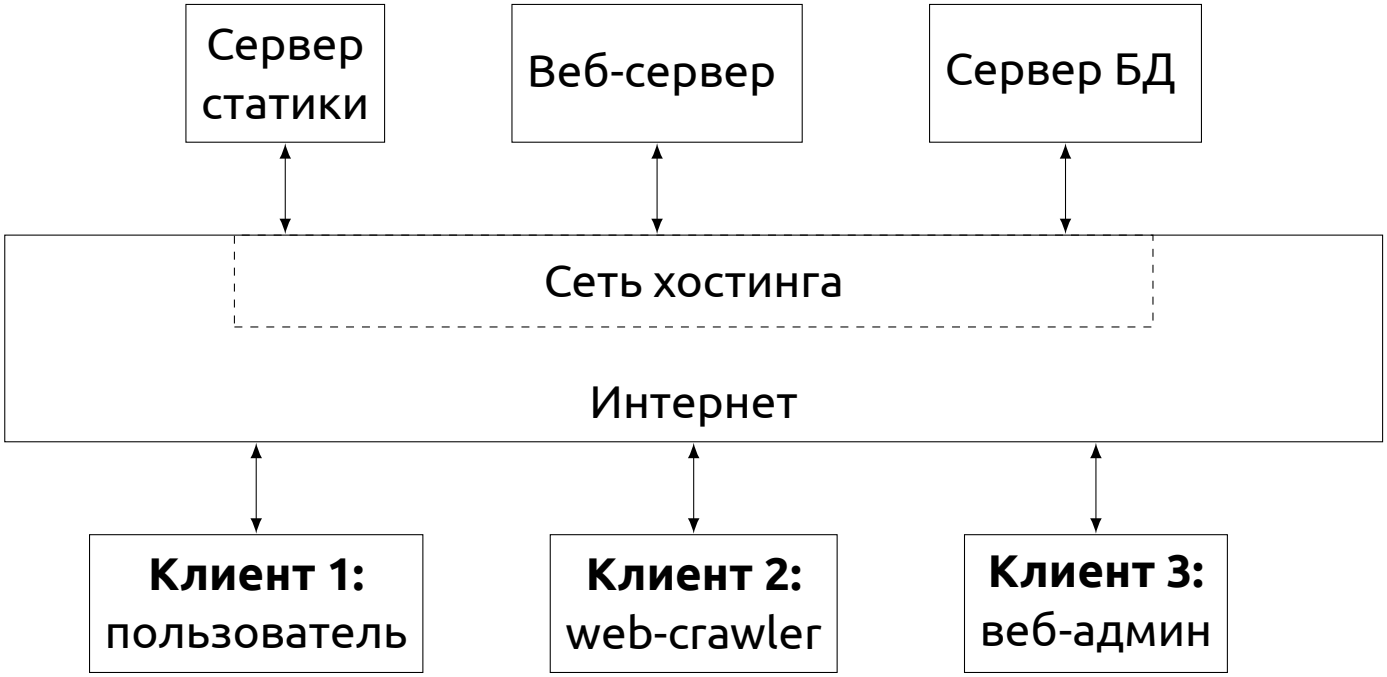
Применение: большой объем и распределенность данных; необходимость удаленного доступа к данным; балансирование нагрузки.

Примеры: веб-приложения.

Преимущества: высокая доступность; снижение требований к клиентам.

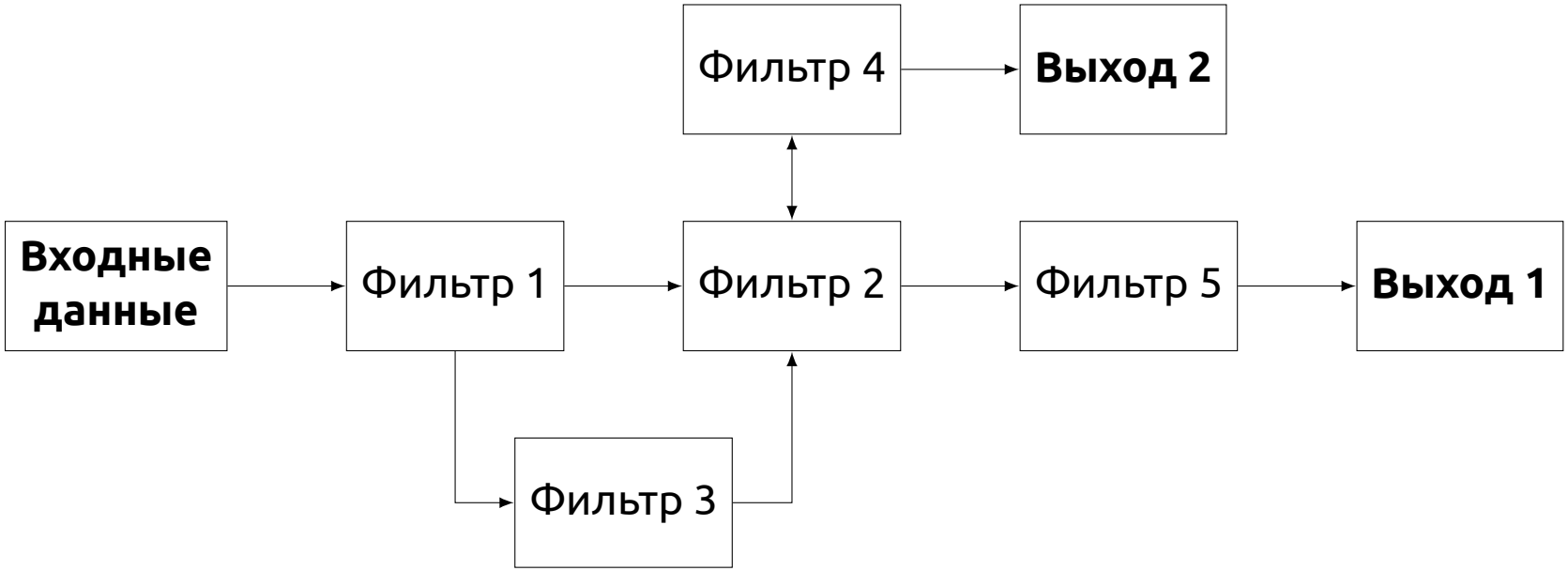
Недостатки: уязвимость к DoS-атакам; потенциальное снижение производительности; сложность управления.

Клиент-серверная архитектура — пример



Клиент-серверная архитектура для веб-сервера

Конвейерная архитектура



Абстрактная модель конвейерной архитектуры (англ. *pipe and filter architecture*)

Конвейерная архитектура — описание

Описание: Организация обработки данных в виде конвейера, в котором каждый компонент (фильтр) выполняет операции одного типа.

Применение: приложения, ориентированные на данные (обработка транзакций, интеллектуальный анализ); большое количество однотипных данных.

Примеры: pipe в *NIX; обработка звука и видео; системы Map/Reduce.

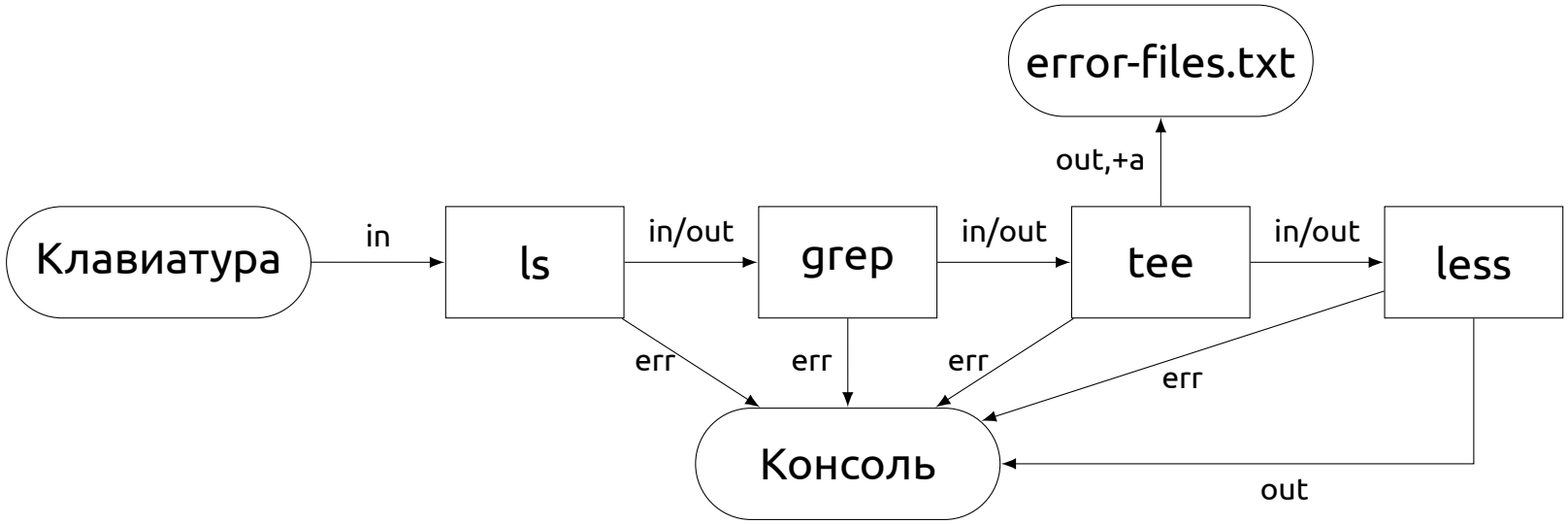
Преимущества: хорошая масштабируемость; простота и понятность; возможности повторного использования компонентов.

Недостатки: необходимость преобразования входных/выходных данных к стандартному виду.

Конвейерная архитектура — пример

Команды: `% ls -l | grep error | tee -a error-files.txt | less`

Эффект: Выводит информацию о файлах, имя которых содержит *error* с разбиением текста на страницы, и добавляет эту информацию в конец файла *error-files.txt*.



Пример конвейера для цепочки команд Linux с указанием направления стандартных потоков

Выводы

1. Архитектура программной системы определяет ее организацию и разбиение на компоненты. На архитектуру влияют нефункциональные требования к системе (напр., производительность и надежность).
2. Помимо составляющих системы, архитектура может определять распределение работ между разработчиками, взаимодействие компонентов и их организацию в распределенной системе. Для представления архитектуры могут использоваться формальные (UML, ADL) и неформальные (диаграммы) методы.
3. Опыт разработки архитектуры различных систем собран в архитектурные шаблоны. Часто используемые шаблоны: MVC, многослойная архитектура, клиент-сервер и конвейер.

Материалы

- Sommerville, Ian

  Software Engineering.

  Pearson, 2011. — 790 p.

- Лавріщева К. М.

  Програмна інженерія (підручник).

  К., 2008. — 319 с.

Спасибо за внимание!