Архитектурные стили

Юрий Литвинов
yurii.litvinov@gmail.com

16.11.2017г

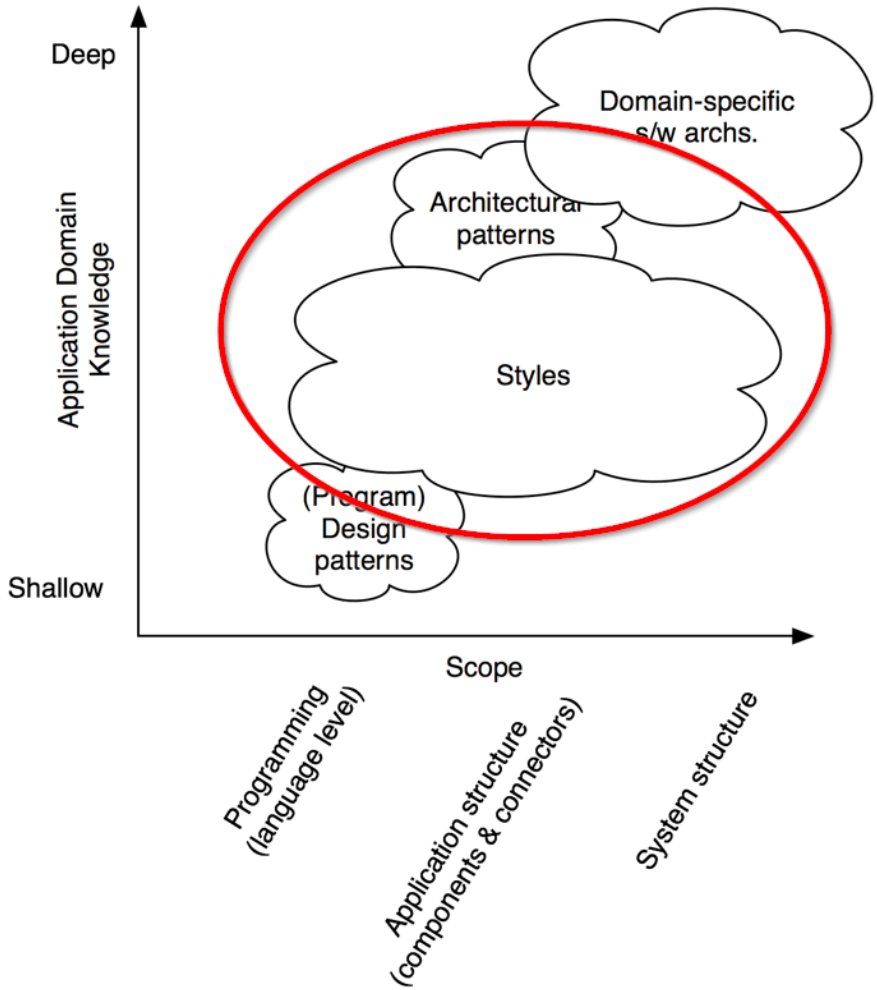
Архитектурные шаблоны и стили

Архитектурный стиль — набор решений, которые

1. применимы в выбранном контексте разработки,
2. задают ограничения на принимаемые архитектурные решения, специфичные для определённых систем в этом контексте,
3. приводят к желаемым положительным качествам получаемой системы.

Архитектурный шаблон — именованный набор ключевых проектных решений по эффективной организации подсистем, применимых для повторяемых технических задач проектирования в различных контекстах и предметных областях

Архитектурные шаблоны и стили, классификация

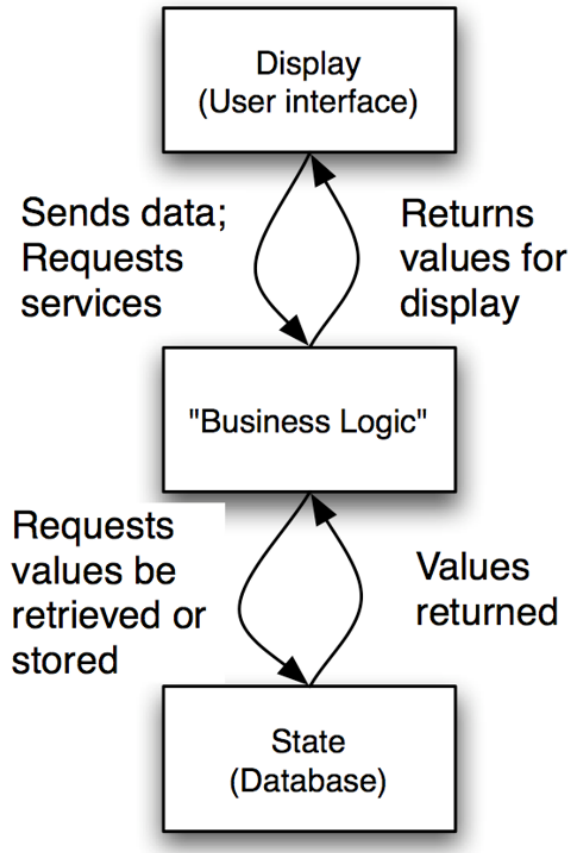


Пример: трёхзвенная архитектура

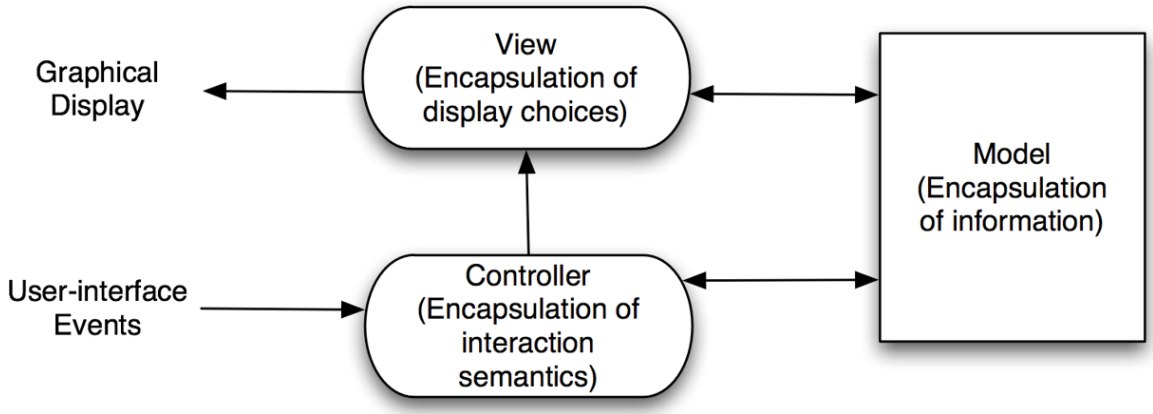
State-Logic-Display

Примеры применения

- Бинес-приложения
- Многопользовательские игры
- Веб-приложения



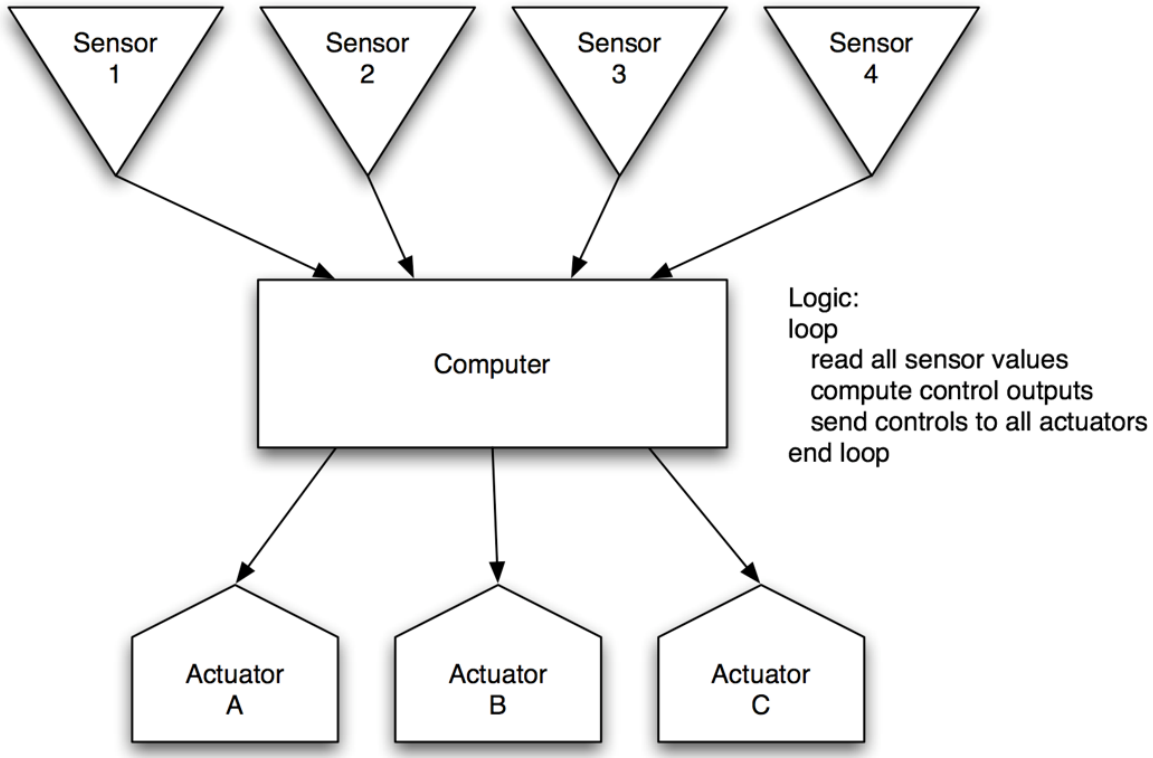
Пример: Model-View-Controller



© N. Medvidovic

- Разделяет данные, представление и взаимодействие с пользователем
- Если в модели что-то меняется, она оповещает представление (представления)
- Через контроллер проходит всё взаимодействие с пользователем
  - Естественное место для паттерна “Команда” и Undo/Redo

Пример: Sense-Compute-Control



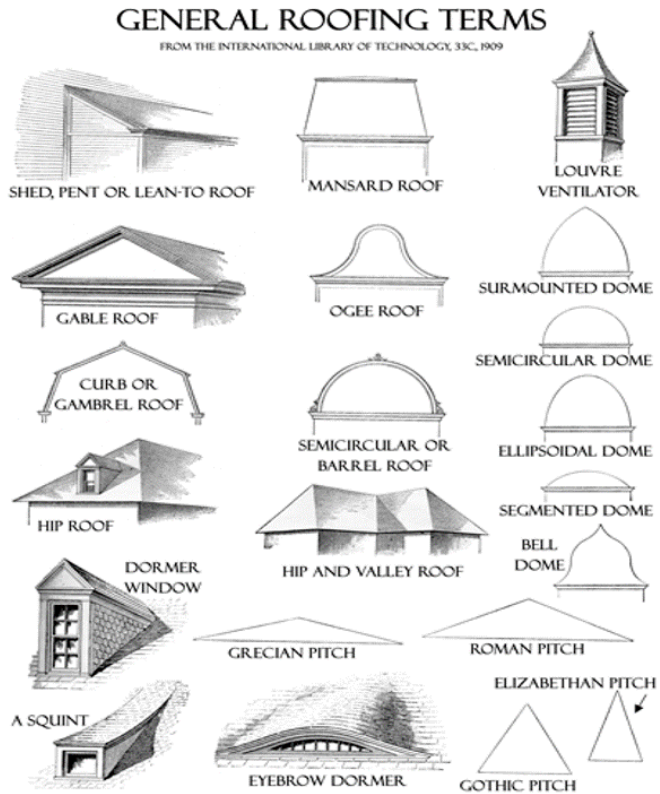
- Применяется во встроенных системах и робототехнике

Архитектурные стили

- Именованная коллекция архитектурных решений
- Менее узкоспециализированные, чем архитектурные паттерны

Архитектурные стили

- Одна система может включать в себя несколько архитектурных стилей
- Понятие стиля применимо и к подсистемам



© N. Medvidovic

Преимущества использования стилей

- Переиспользование архитектуры
  - Для новых задач можно применять хорошо известные и изученные решения
- Переиспользование кода
  - Часто у стилей бывают неизменяемые части, которые можно один раз реализовать
- Упрощение общения и понимания системы
- Упрощение интеграции приложений
- Специфичные для стиля методы анализа
  - Возможны благодаря ограничениям на структуру системы
- Специфичные для стиля методы визуализации

Основные характеристики стилей

- Набор используемых элементов архитектуры
  - Типы компонентов и соединителей, элементы данных
    - Например, объекты, фильтры, сервера и т.д.
- Набор правил конфигурирования
  - “Топологические” ограничения на соединение элементов
    - Например, компонент может быть соединён с максимум двумя компонентами
- Семантика, стоящая за элементами

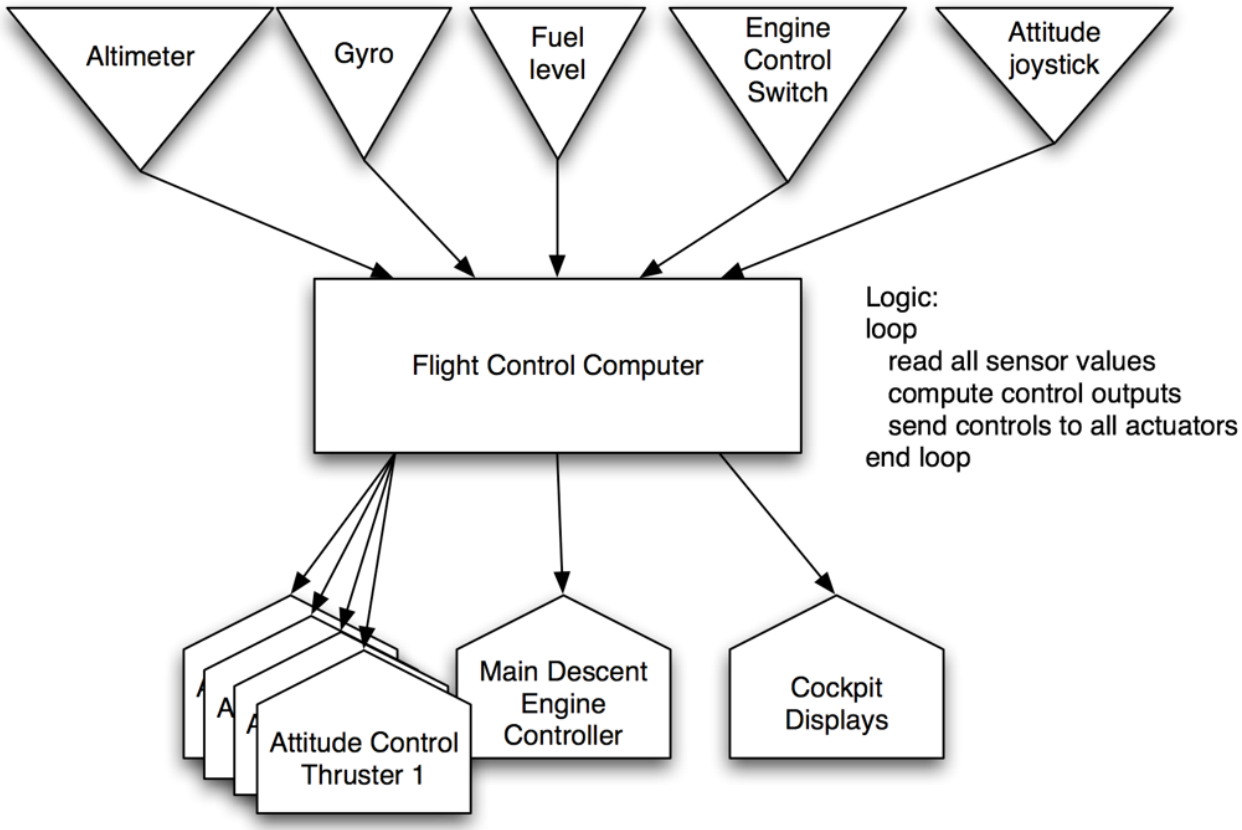
Игра “Посадка на луну”

Lunar Lander

- Игрок управляет двигателем спускаемого аппарата
- Топливо ограничено
- Заданы начальная высота и скорость
- Победа засчитывается, если скорость при касании грунта меньше заданной
- Продвинутая версия позволяет управлять горизонтальным движением

© N. Medvidovic

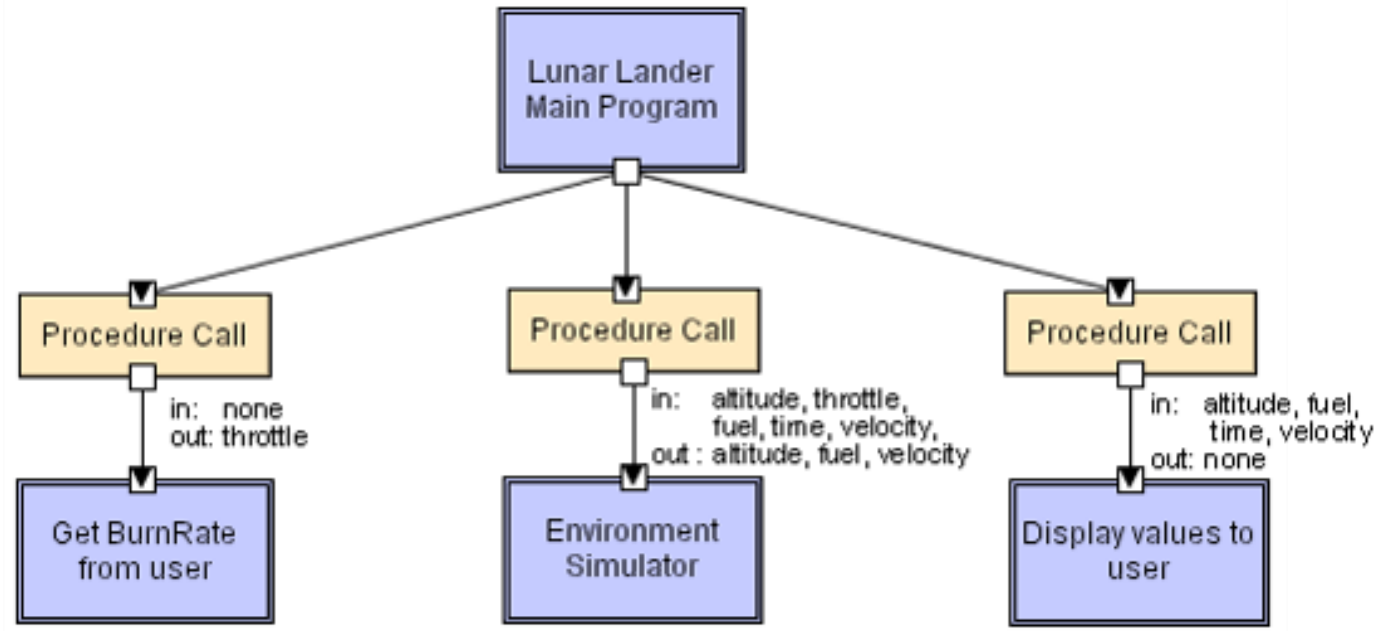
Sense-Compute-Control-реализация



Некоторые известные стили

- “Традиционные”, связанные с языком
  - Главная программа/подпрограммы
  - Объектно-ориентированный
- Уровневый стиль
  - Виртуальные машины
  - Клиент-сервер
- Стили, ориентированные на поток данных
  - Пакетное исполнение
  - Каналы и фильтры
- Peer-to-peer
- Общая память
  - Blackboard
  - Ориентированные на правила
- Интерпретаторы
  - Интерпретатор
  - Мобильный код
- Неявный вызов
  - Событийно-ориентированный
  - Издатель-подписчик
- “Производные” стили
  - Распределённые объекты
  - REST
  - C2

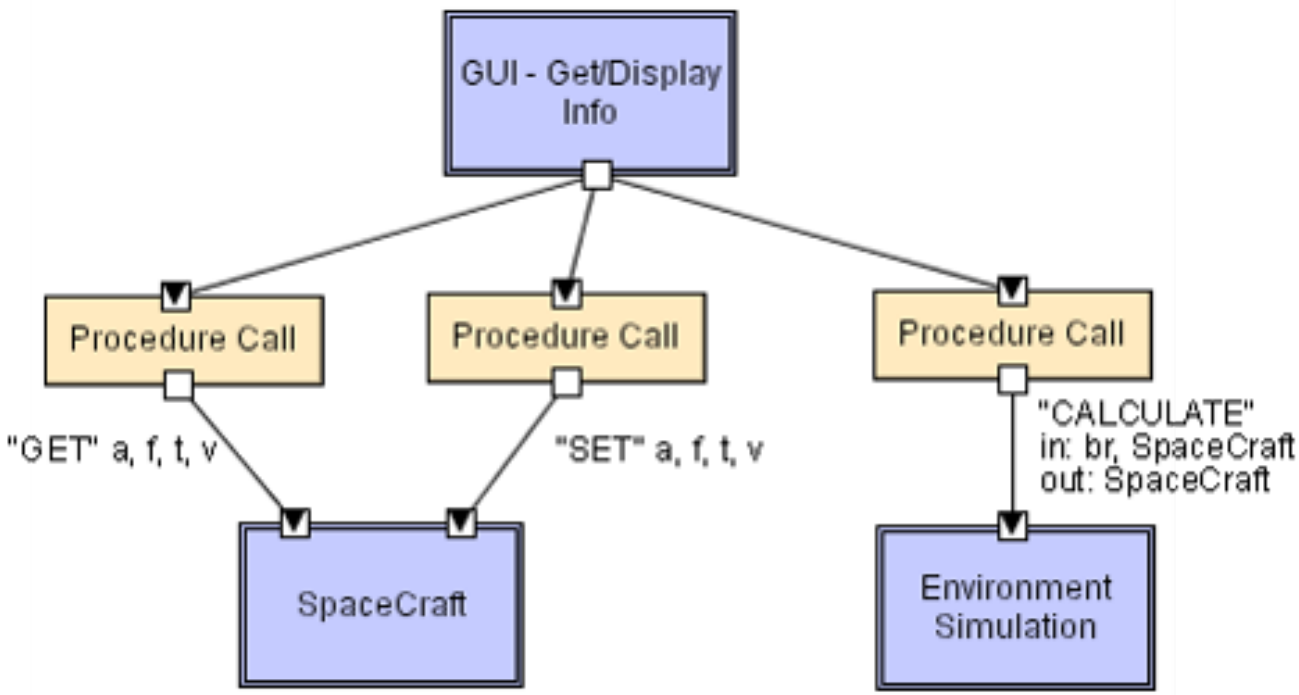
Главная программа/подпрограммы



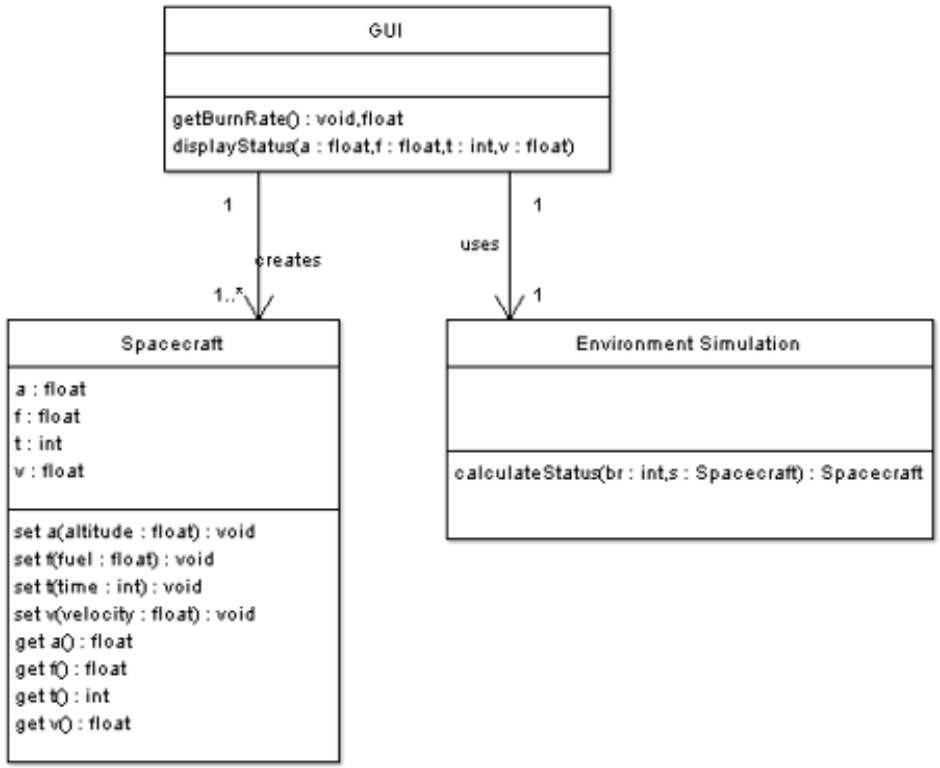
Объектно-ориентированный стиль

- Компоненты — объекты
- Соединители — сообщения и вызовы методов
- Инварианты:
  - Объекты отвечают за своё внутреннее состояние
  - Реализация скрыта от других объектов
- Преимущества:
  - Декомпозиция системы в набор взаимодействующих агентов
  - Внутреннее представление объектов можно менять независимо
  - Близко к предметной области
- Недостатки:
  - Побочные эффекты при вызове методов
  - Объекты вынуждены знать обо всех, от кого зависят

Объектно-ориентированный стиль, Lunar Lander



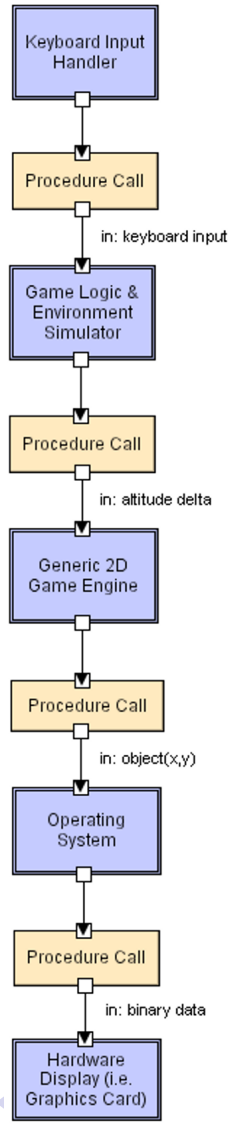
Или то же на UML



Слоистый стиль

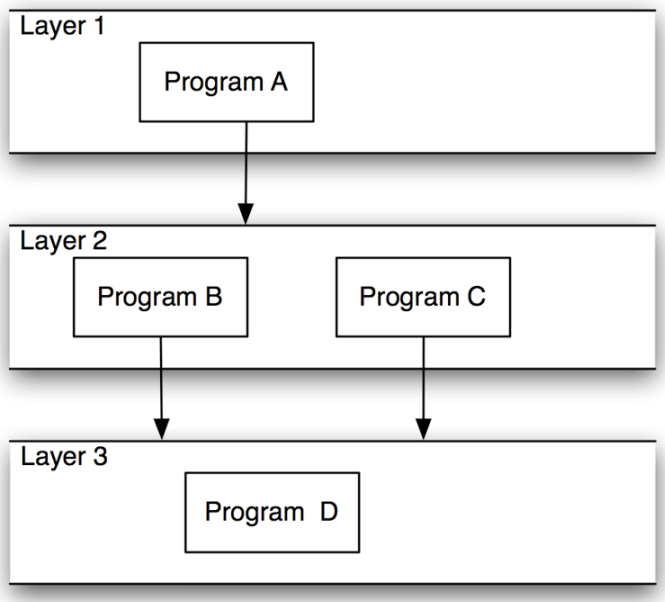
Layered style

- Иерархическая организация системы
  - “Многоуровневый клиент-сервер”
  - Каждый слой предоставляет интерфейс для использования слоями выше
- Каждый слой работает как:
  - Сервер — предоставляет функциональность слоям выше
  - Клиент — использует функциональность слоёв ниже
- Соединители — протоколы взаимодействия слоёв
- Пример — операционные системы, сетевые стеки протоколов



Слоистый стиль, подробности

- Преимущества:
  - Повышение уровня абстракции
  - Лёгкость в расширении
  - Изменения в каждом уровне затрагивают максимум два соседних
  - Возможны разные реализации уровня, если они удовлетворяют интерфейсу
- Недостатки:
  - Не всегда применим
  - Проблемы с производительностью

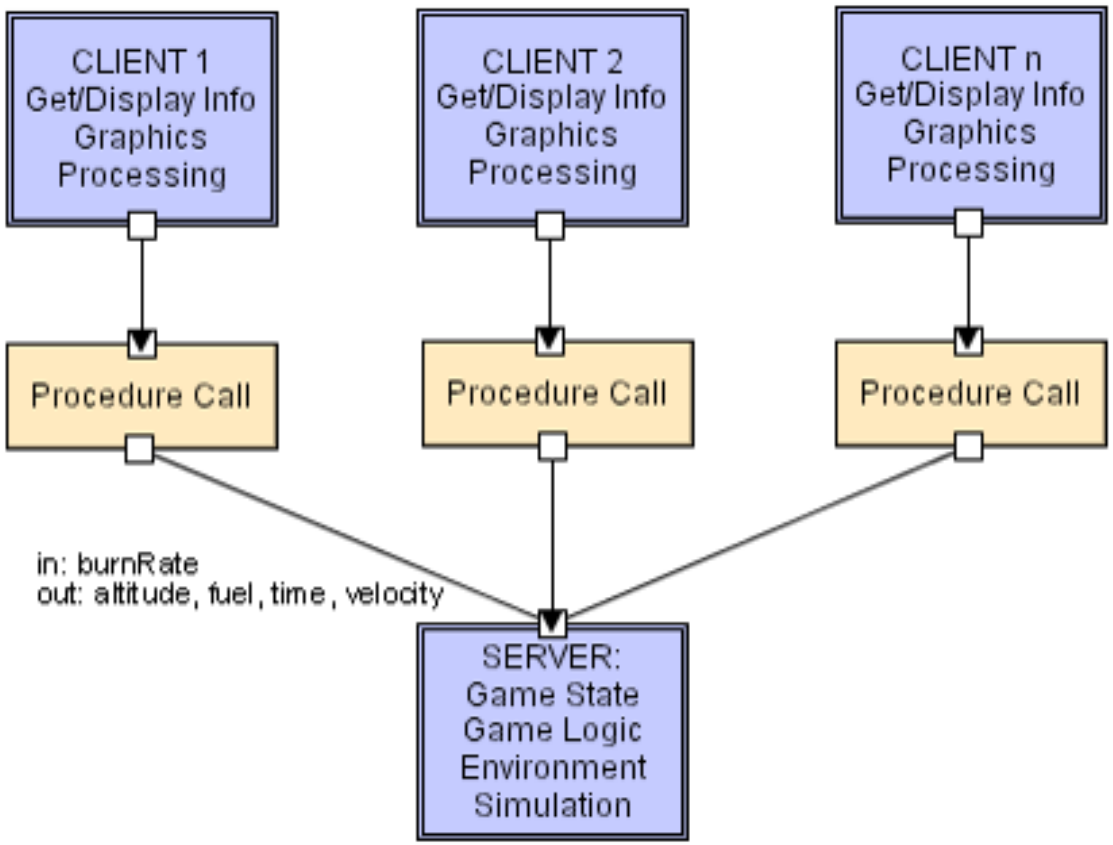


© N. Medvidovic

“Клиент-сервер”

- Компоненты — клиенты и серверы
- Серверы не знают ничего о клиентах, даже их количество
- Клиенты знают только про сервера и не могут общаться друг с другом
- Соединители — сетевые протоколы

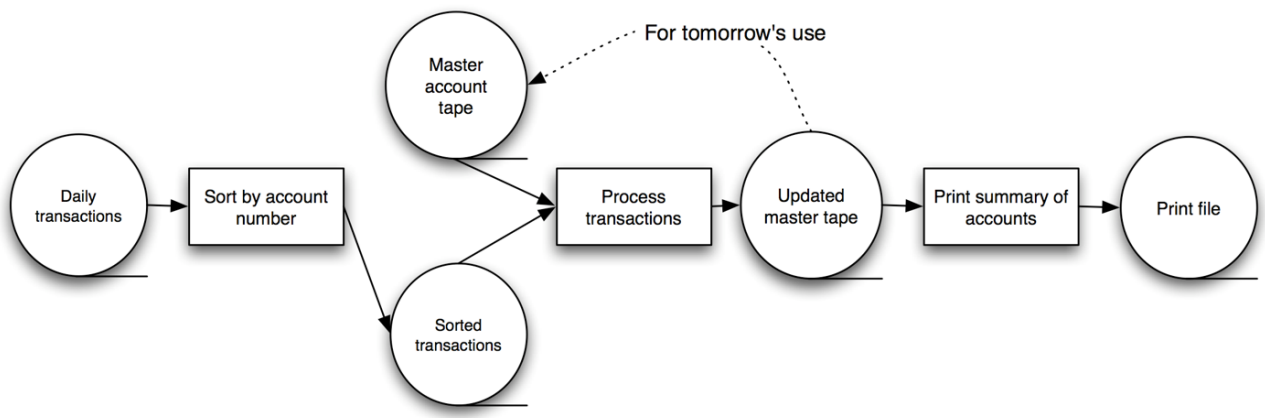
“Клиент-сервер”, Lunar Lander



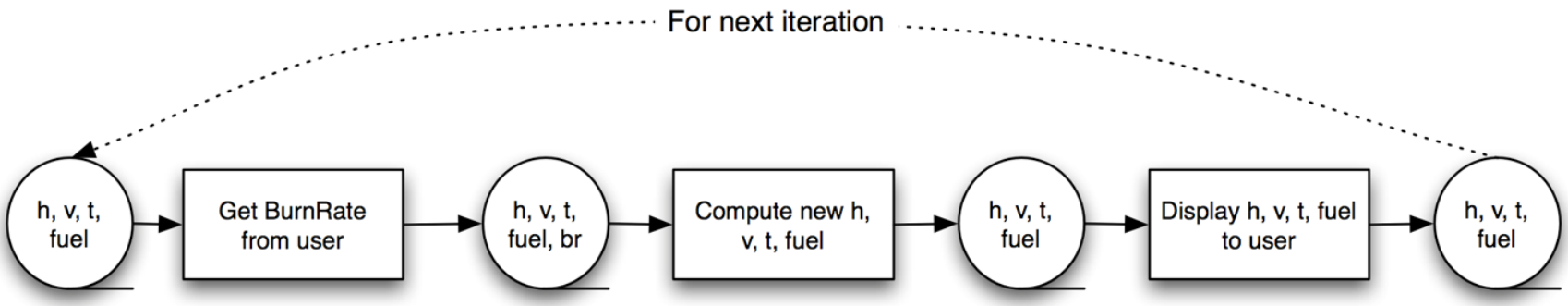
Пакетная обработка

- Система строится как набор отдельных программ, выполняющихся последовательно
- Данные стандартным для ОС способом передаются от программы к программе
  - Pipes, named pipes, файлы
- Данные — в явном виде всё, необходимое для работы

Типичен для финансовых систем глубокой древности (“Прадедушка стилей”)



Пакетная обработка, Lunar Lander



Play-by-email?