

Сложность программных систем

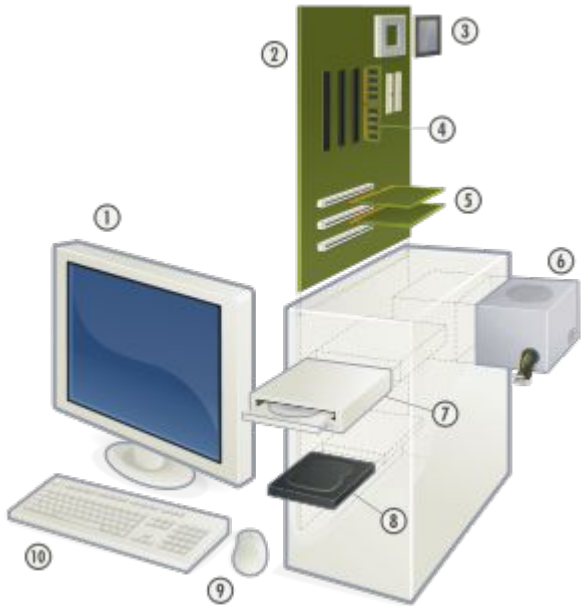
1. Признаки сложных систем
2. Причины сложности разработки программных систем
3. Способы борьбы со сложностью: абстракция, декомпозиция, иерархия

Преподаватель:

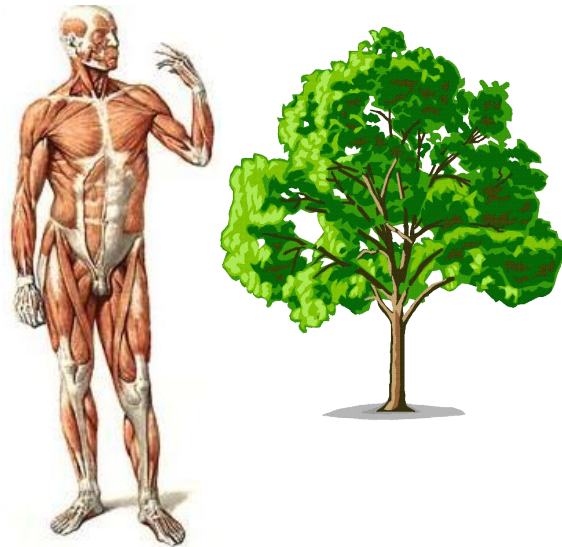
Ботов Дмитрий Сергеевич

Примеры сложных систем

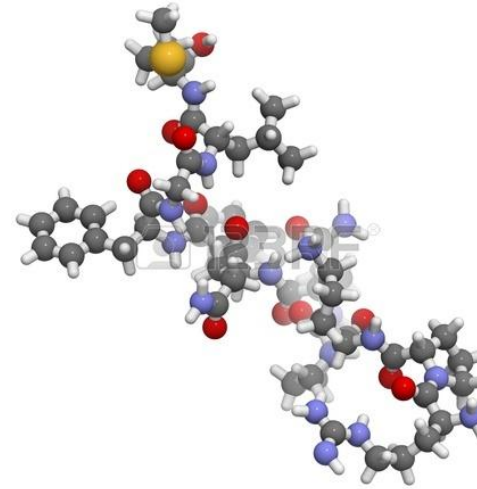
Структура
персонального
компьютера



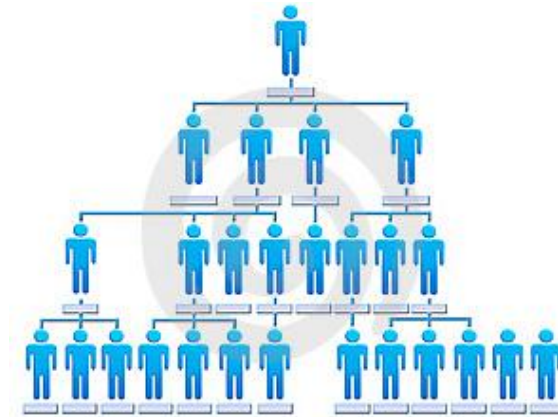
Структура живых
организмов



Структура
вещества



Структура
социальных
институтов

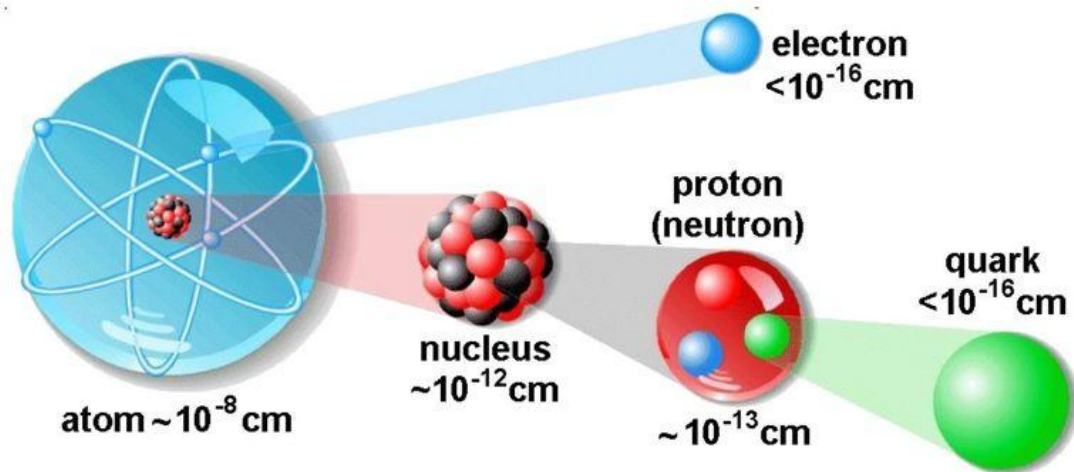


Признаки сложной системы (по Саймону)

1. Сложность часто представляется в виде **иерархии**
2. Выбор, какие компоненты в данной системе считаются **элементарными**, относительно произволен (на усмотрение наблюдателя)
3. Внутрикомпонентная связь обычно сильнее, чем связь между компонентами
4. Иерархические системы обычно состоят из **немногих типов подсистем**, по-разному скомбинированных и организованных
5. Работающая сложная система является результатом развития работавшей **простой системы**

1. Сложность часто представляется в виде иерархии

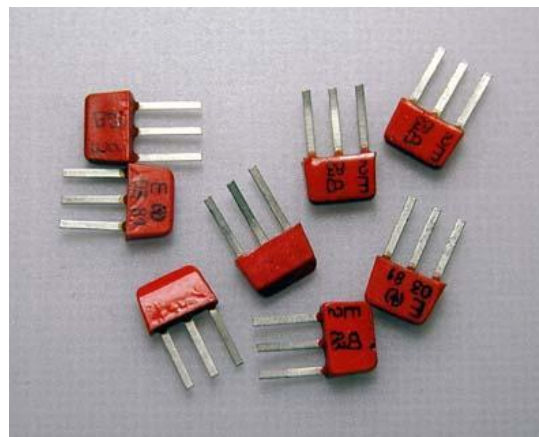
Структура вещества



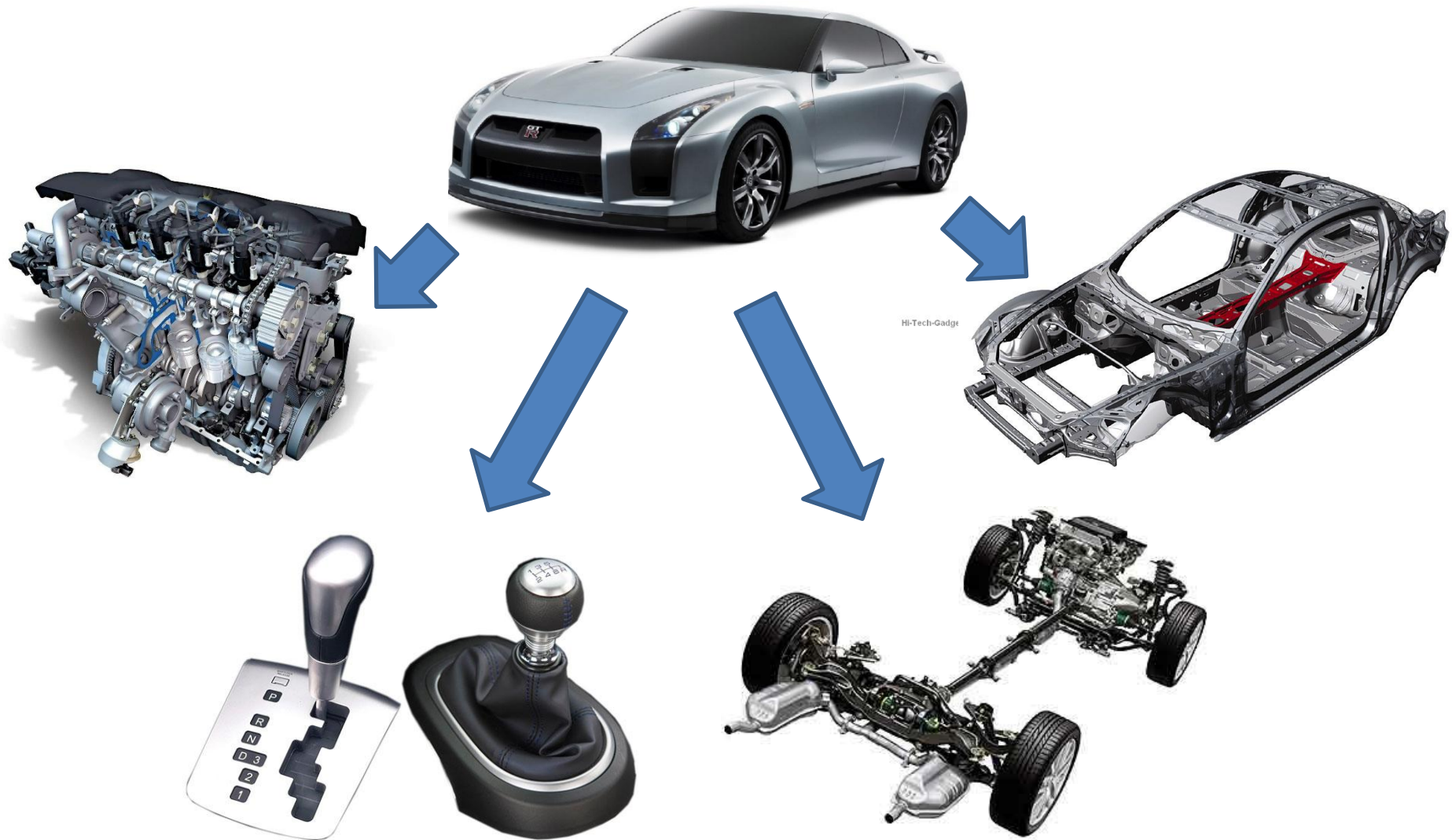
Структура социальных институтов



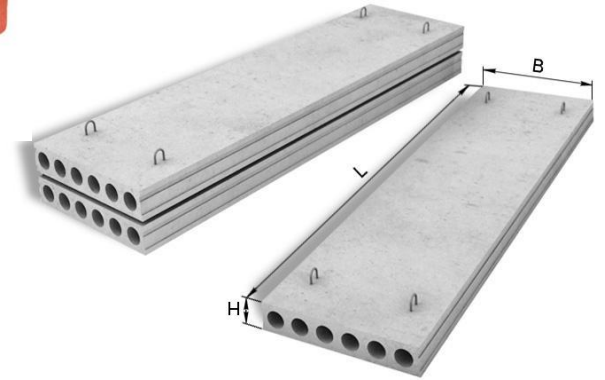
2. Выбор, какие компоненты в данной системе считаются **элементарными**, относительно произволен (на усмотрение наблюдателя)



3. Внутриконтонентная связь обычно сильнее, чем связь между компонентами

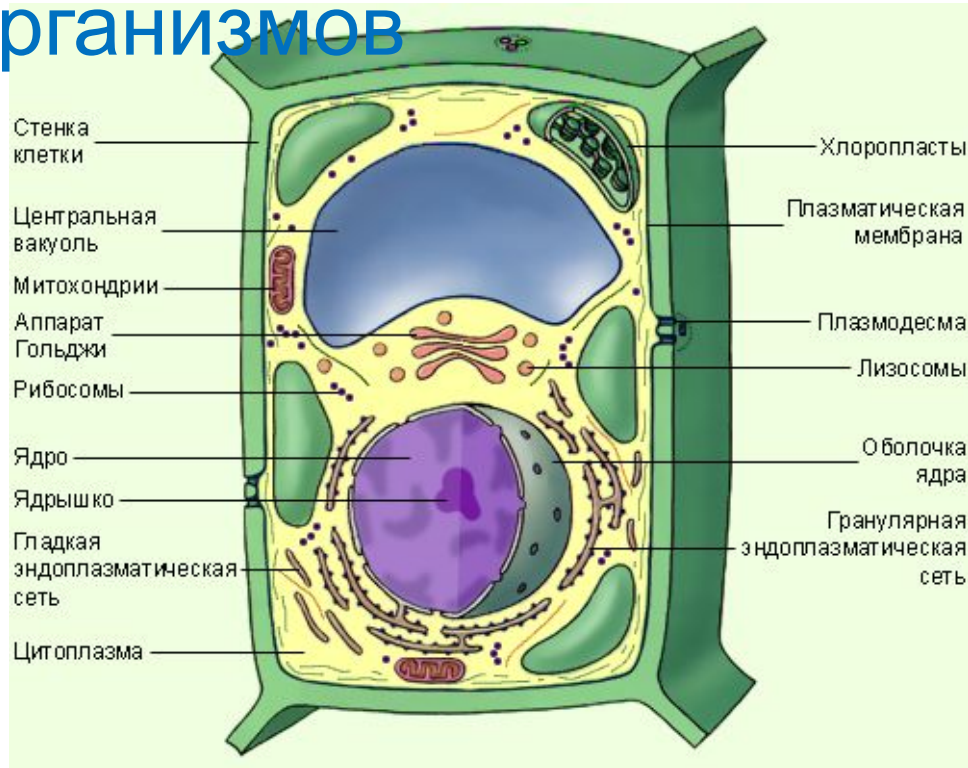


4. Иерархические системы обычно состоят из **немногих типов подсистем**, по-разному скомбинированных и организованных



4. Иерархические системы обычно состоят из **немногих типов подсистем**, по-разному скомбинированных и организованных

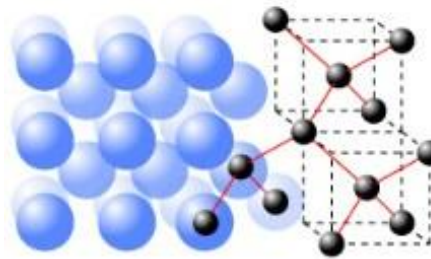
Структура живых организмов



Структура вещества

КРИСТАЛЛЫ

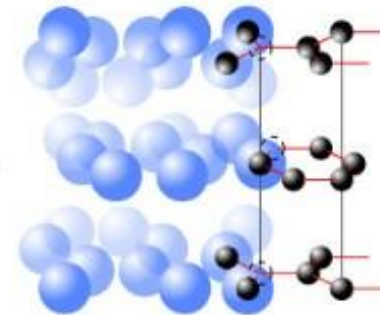
УПАКОВКА АТОМОВ И ПРОСТРАНСТВЕННАЯ РЕШЕТКА АЛМАЗА



АЛМАЗ



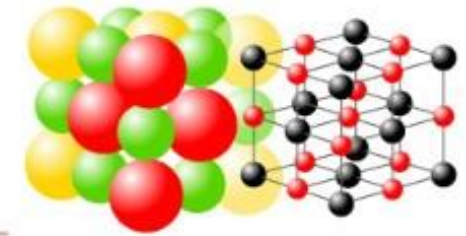
УПАКОВКА АТОМОВ И ПРОСТРАНСТВЕННАЯ РЕШЕТКА ГРАФИТА



ГРАФИТ



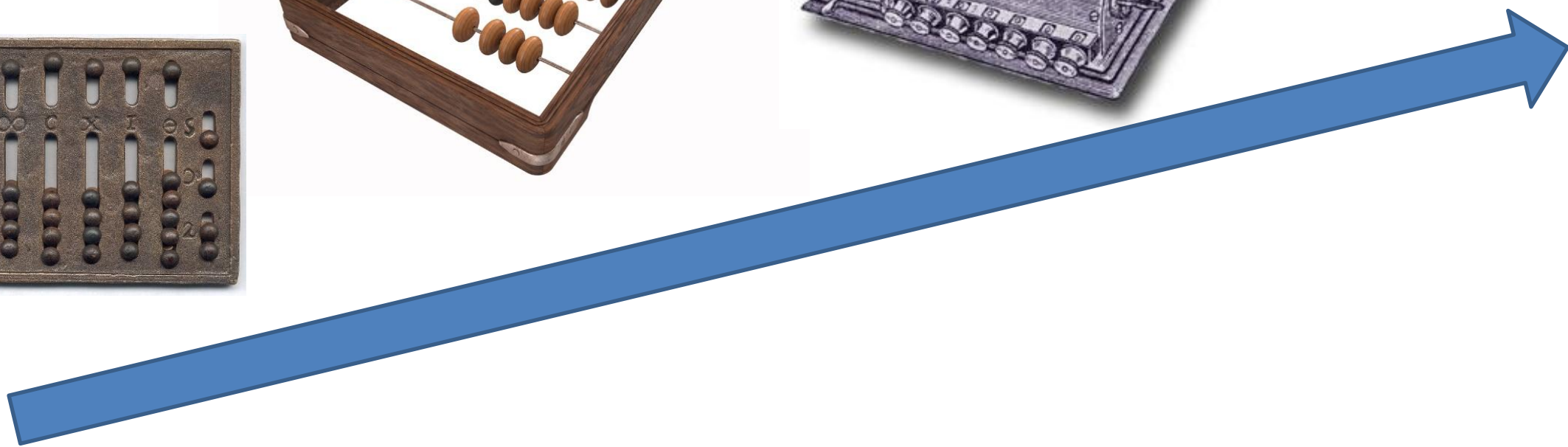
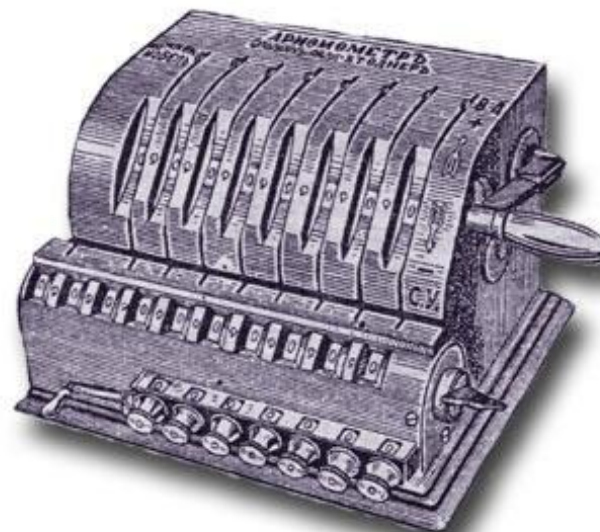
УПАКОВКА АТОМОВ И ПРОСТРАНСТВЕННАЯ РЕШЕТКА ПОВАРЕННОЙ СОЛИ



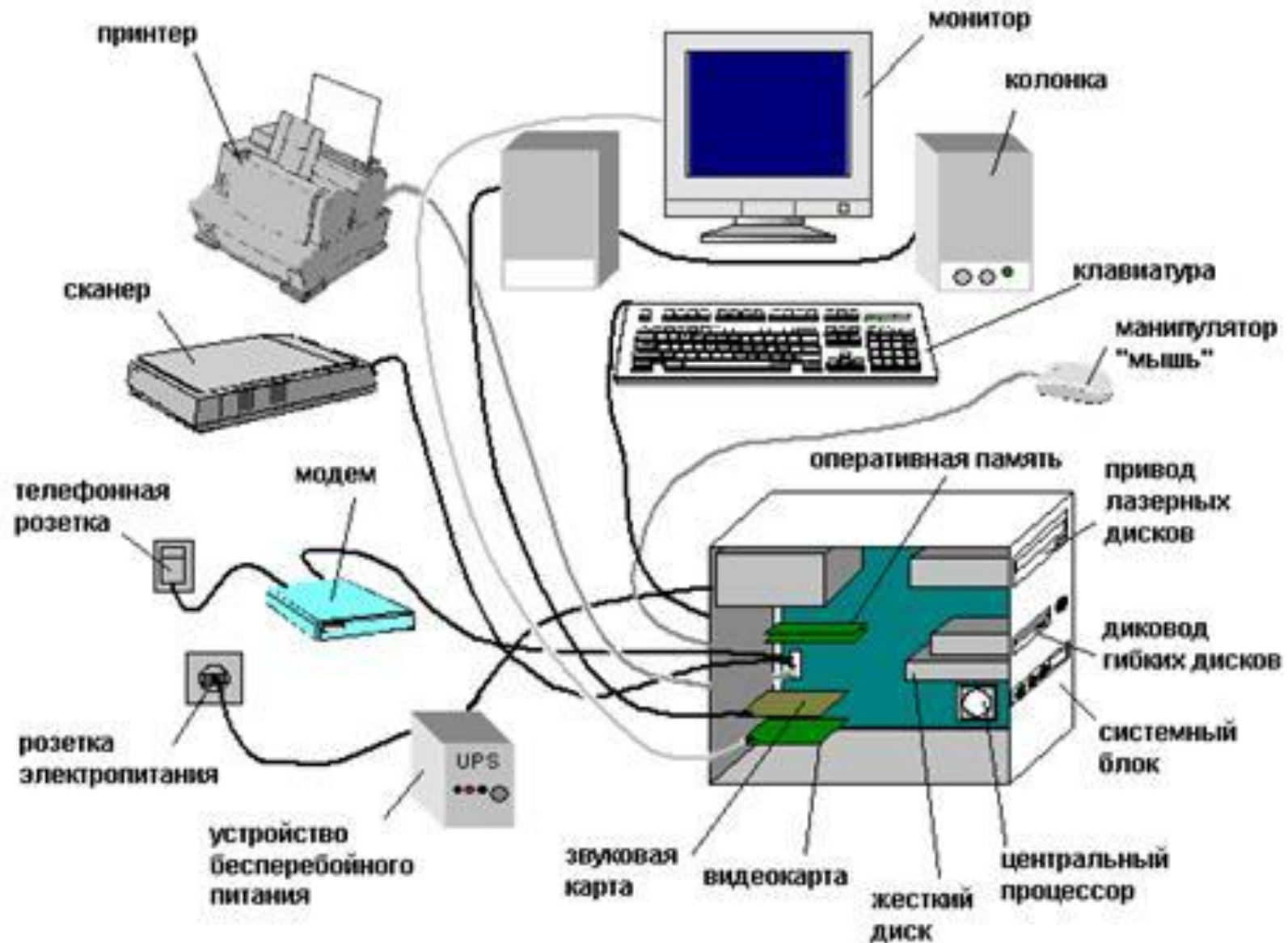
ПОВАРЕННАЯ СОЛЬ



5. Работающая сложная система является
результатом развития работавшей **простой**
системы



Пример: структура ПК



Промышленные программные системы

- **Большие системы**
терабайты и петабайты данных,
сотни и тысячи пользователей
- **Сложные системы**
тысячи файлов, десятки
подсистем, сотни компонент
- **Качественные системы**
надежность, безопасность,
отказоустойчивость, эргономика,
многократное использование,
документация, сопровождаемость
- **Сложная архитектура**
клиентская и серверная часть,
базы данных, веб-сервисы,
распределенность,
масштабируемость
- **Создаются командами
разработчиков**



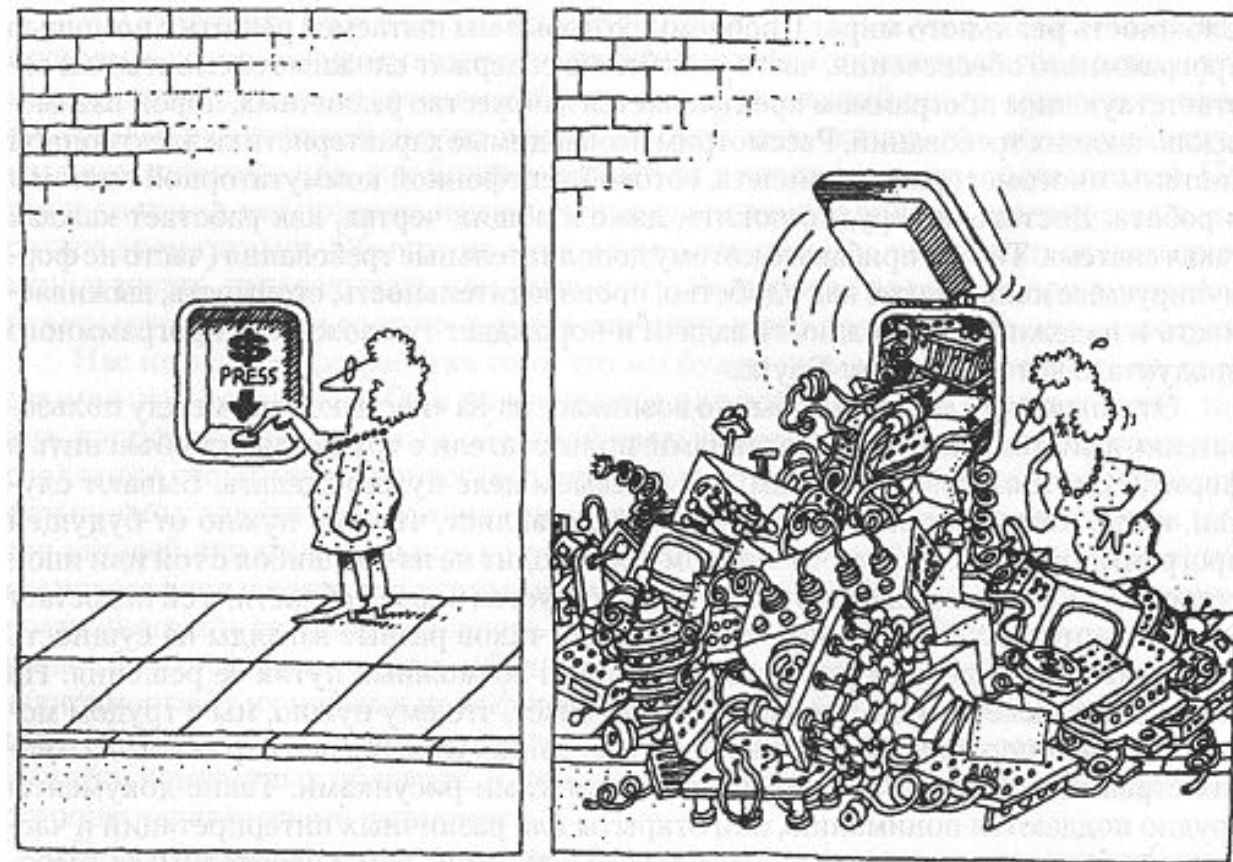
Причины сложности разработки программной системы

«Сложность программного обеспечения – отнюдь неслучайное его свойство»
(Ф. Брукс)

- Сложность **проблемы** (предметной области), **изменение требований**
- Сложность управления **процессом разработки**
- Сложность обеспечения **гибкости** конечного программного продукта
- Сложность **самой программной системы**, описания поведения отдельных подсистем

Задача разработчиков программной системы

Создать **иллюзию простоты** и защитить пользователей от **сложности** описываемого предмета или процесса
(Г. Буч)



Зачем бороться со сложностью?

Главная причина проблем: физическая ограниченность возможностей человека при работе со сложными системами

Человеческий мозг может **одновременно** следить за **7 ± 2** объектами



Внесение порядка в хаос

Способы преодоления сложности:

Декомпозиция

Абстракция

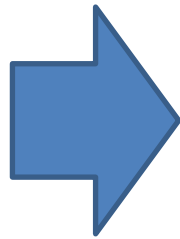
Иерархия



Декомпозиция

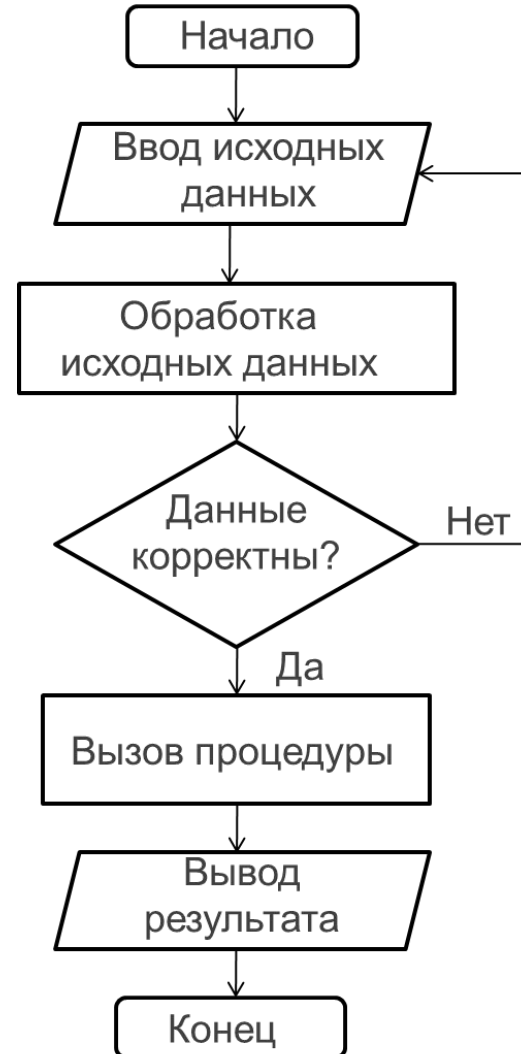
Роль декомпозиции: разделение сложной системы на простые части

- Алгоритмическая декомпозиция
- **Объектно-ориентированная декомпозиция**



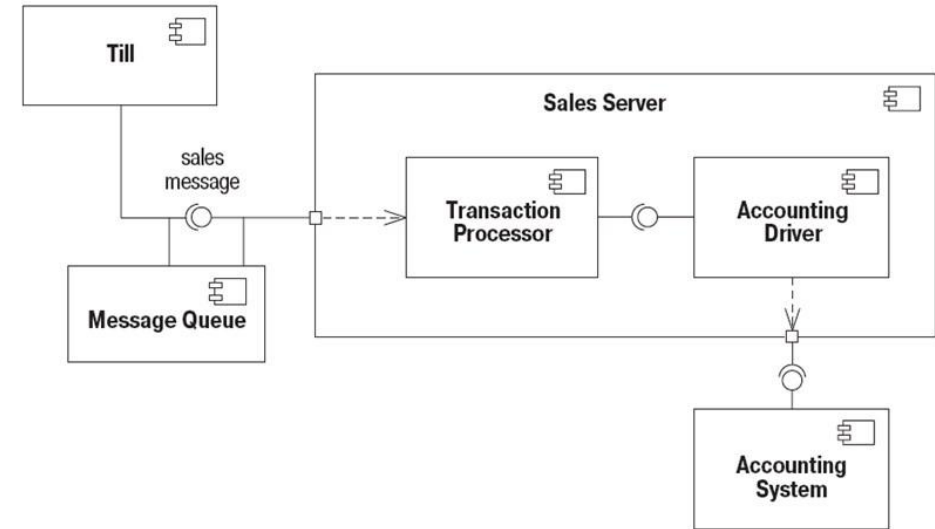
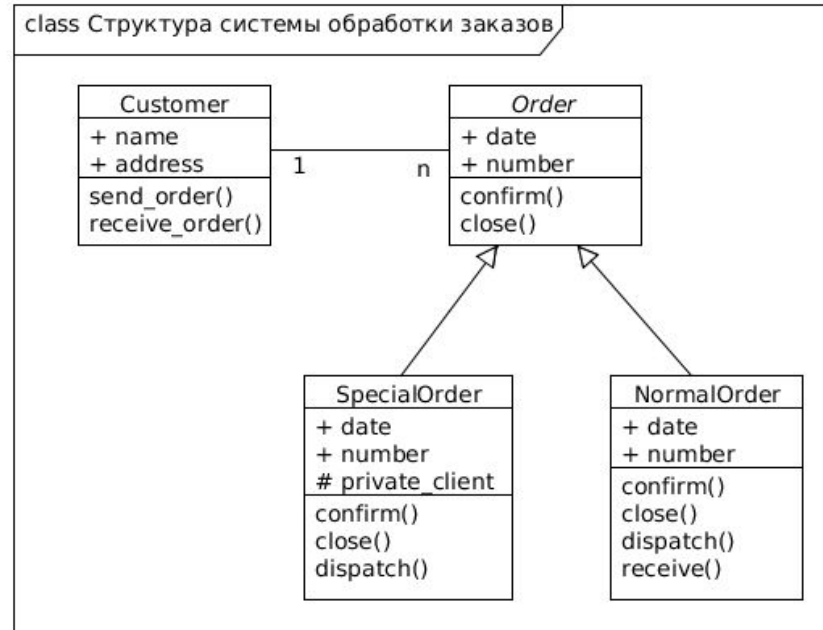
Виды декомпозиции: процедурная

- Действия
- Процессы
- Процедуры
- Алгоритмы
- Функции
- Этапы/Блок
и



Виды декомпозиции: объектно-ориентированная

- Объекты
- Классы
- Модули
- Компоненты
- Подсистемы



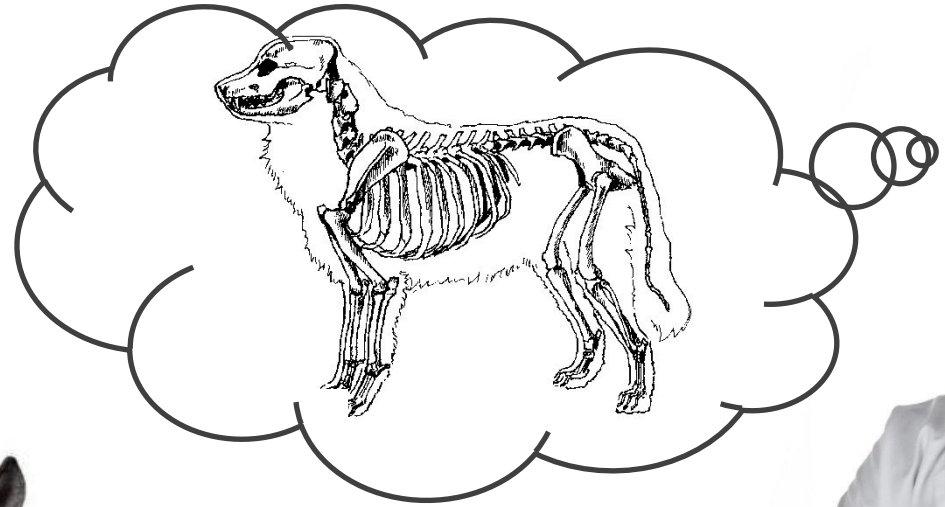
существительные

Какая декомпозиция лучше: алгоритмическая или объектно-ориентированная?

Опыт показывает, что полезнее начинать с объектной декомпозиции:

- объектная декомпозиция уменьшает размер программных систем за счет **повторного использования общих механизмов**
- объектно-ориентированные системы **более гибки и проще эволюционируют** со временем
- объектная декомпозиция помогает нам разобраться в сложной программной системе

Абстракция



Абстракция

Роль абстракции: выделение важных деталей сложной системы

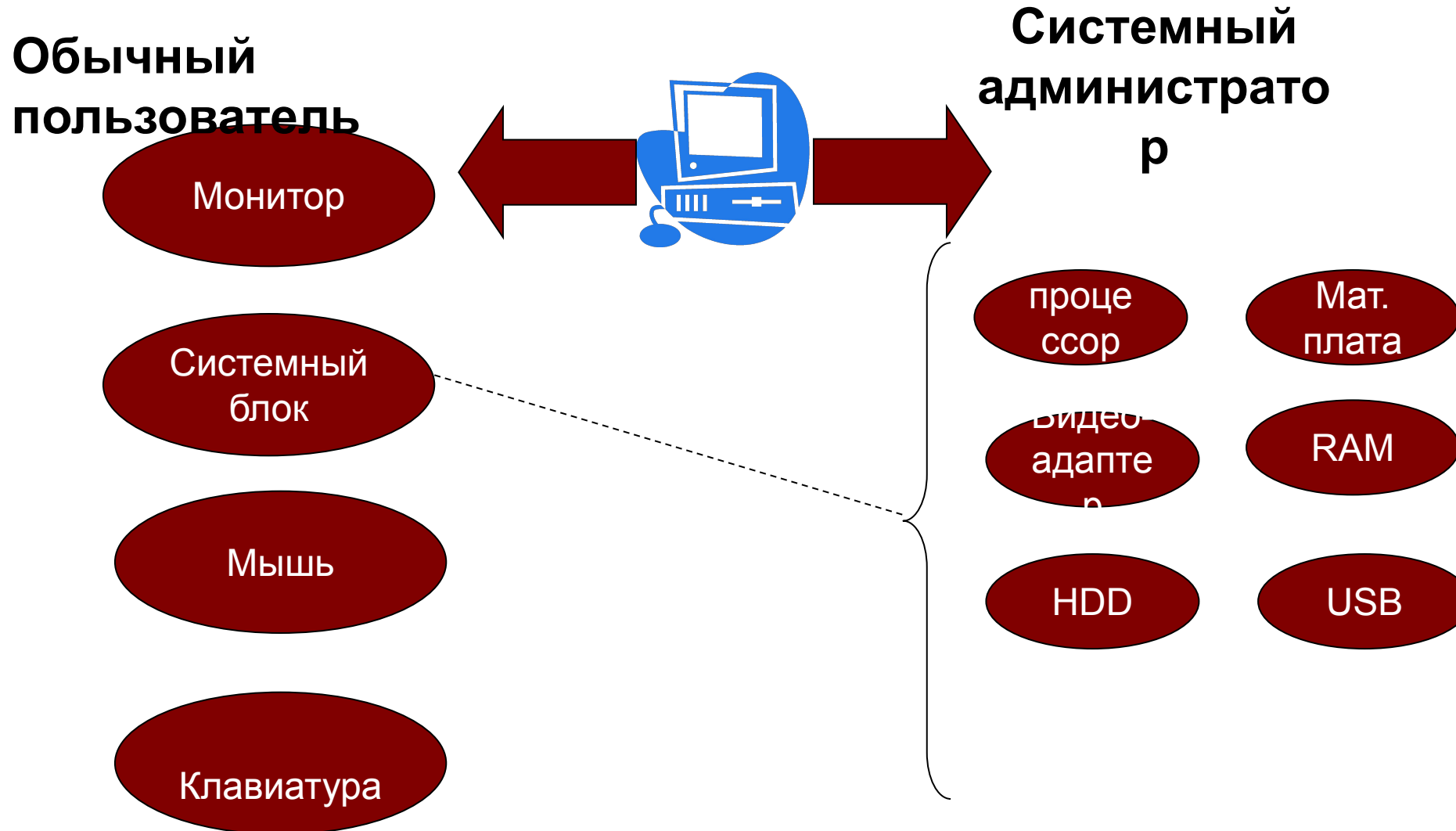
Абстракция (от лат. abstractio отвлечение) — форма познания, основанная на мысленном выделении существенных свойств и связей предмета и отвлечении от других, частных его свойств и связей

- Любая система (окружающий мир) у человека представляется в виде некоторой абстракции



Абстракция – существенные характеристики некоторого объекта, которые отличают его от других видов объектов и, таким образом, четко определяют особенности данного объекта с точки зрения дальнейшего рассмотрения и анализа (Г. Буч).

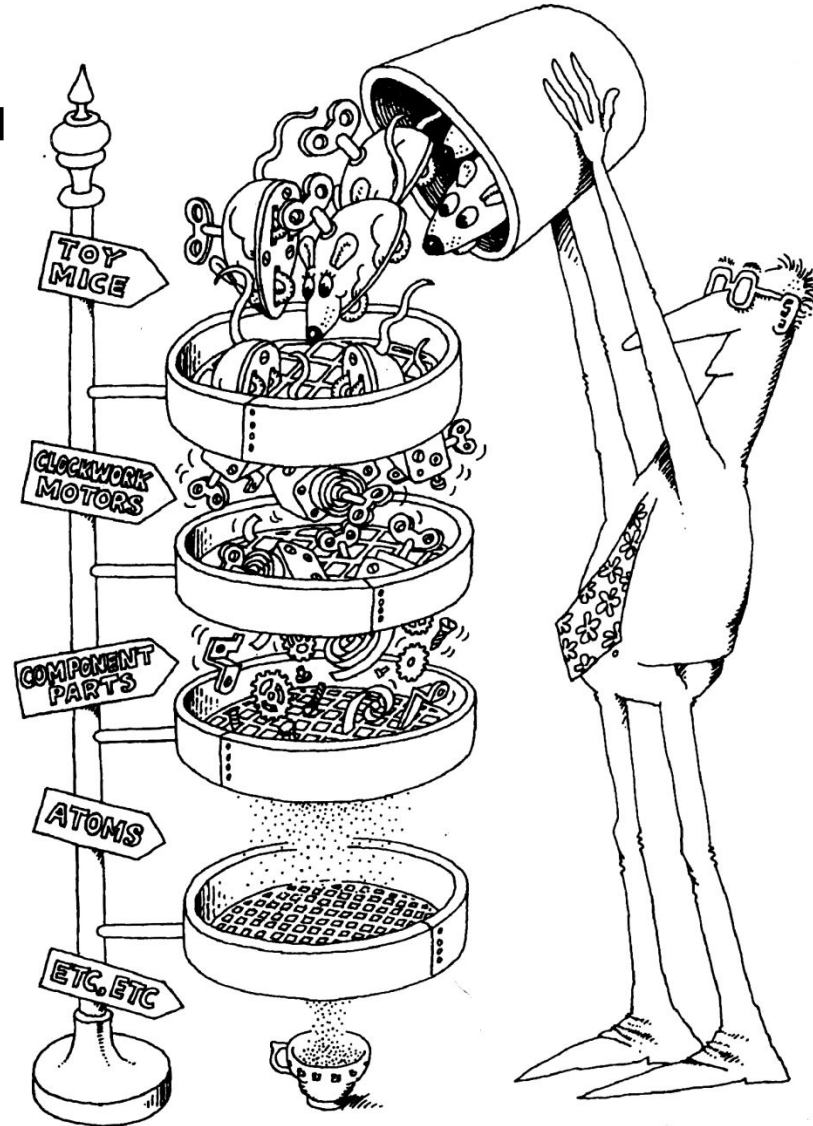
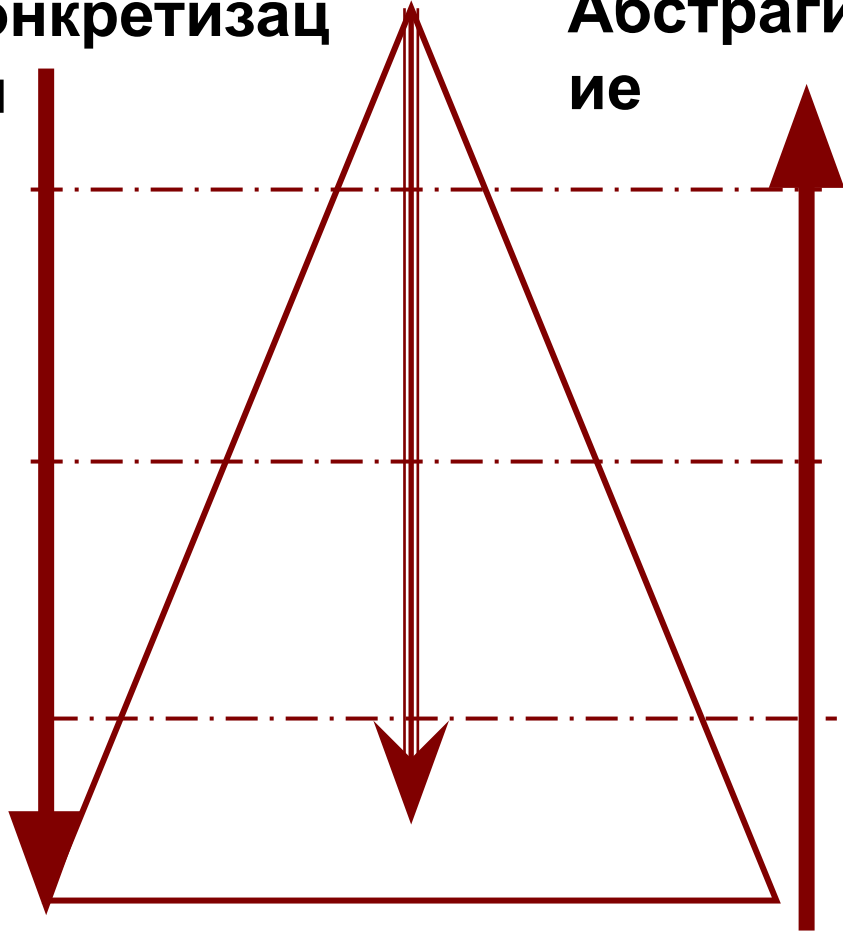
Примеры абстракций



Уровни абстракции

Конкретизация

Абстрагирование



Иерархия

Роль иерархии: упорядочение родственных абстракций по

ур



Резюме: рассмотренные вопросы

1. Какие общие признаки можно выделить у сложных систем?
2. В чем заключается сложность разработки программного обеспечения?
3. Какие существуют способы борьбы со сложностью?
4. В чем заключается роль абстракции, декомпозиции и иерархии в борьбе со сложностью?
5. В чем отличие процедурной и объектно-ориентированной декомпозиции?