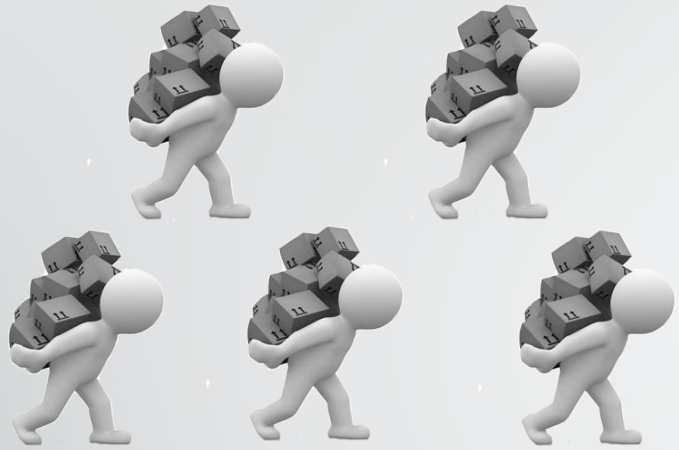
Синхронный MODX: как сделать настоящую синхронизацию и не застрелиться

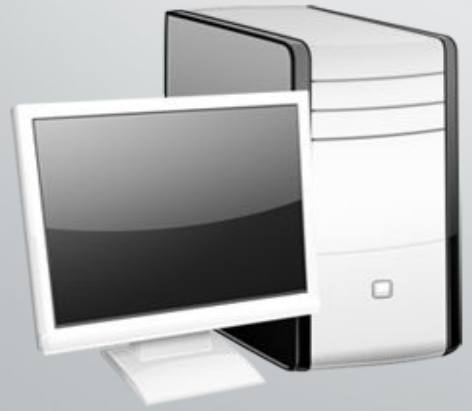
Воеводский Михаил Сергеевич

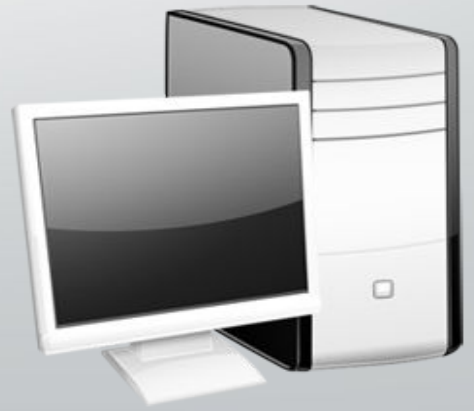
COMP AERO
Автоматизация бизнеса
и разработка ПО

С чего все началось?



Как автоматизировать?





Базовые требования к системе

- Централизация всех обрабатываемых данных
- Возможность автономной работы заведения при локальном отсутствии интернета
- Обмен данными в реальном времени

Почему MODX ?

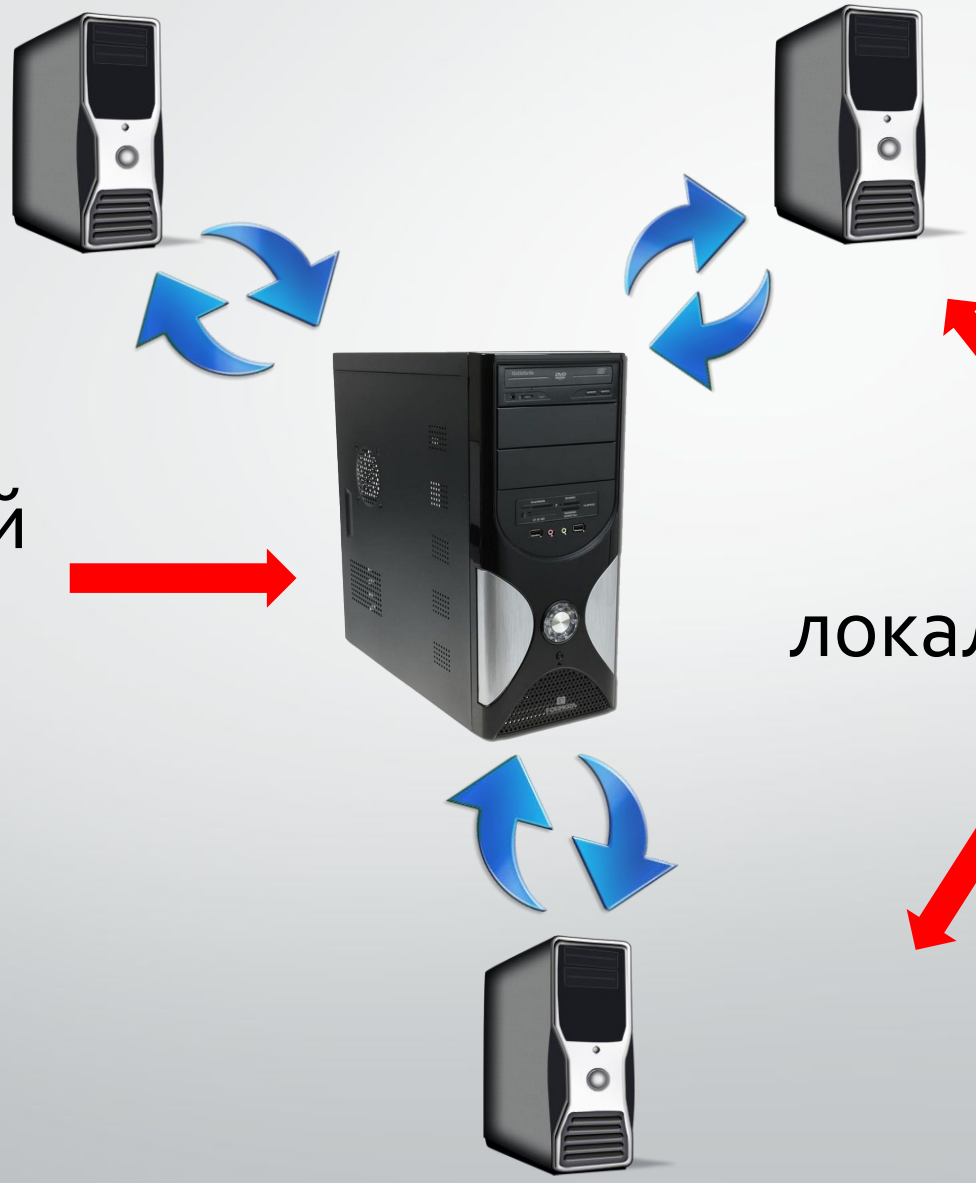
Основная причина

мое хорошее знание этого фреймворка и очень сжатые сроки до запуска первой версии проекта

Дополнительные аргументы в пользу MODX

- Модульность
- Механизм обновления компонентов через транспортные пакеты, возможность создать собственный репозиторий
- Наличие админки

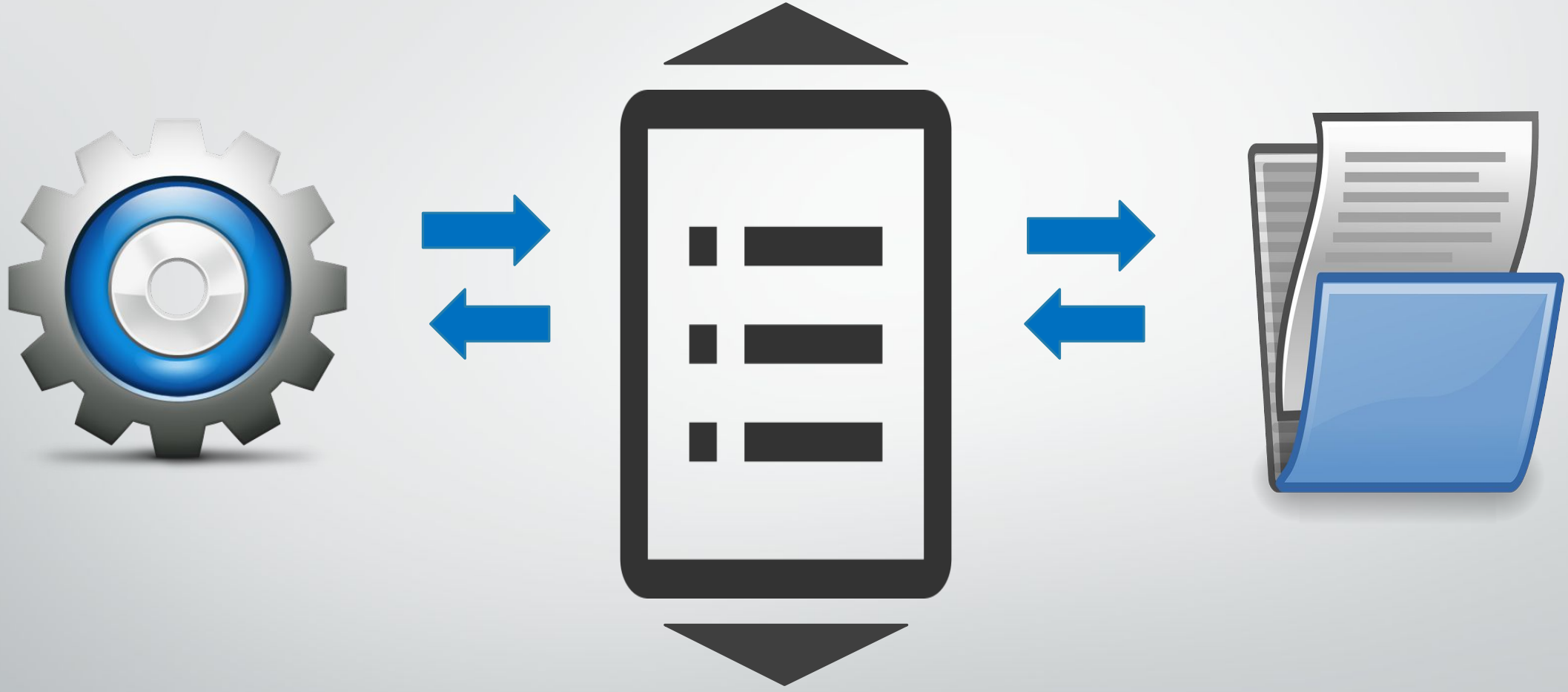
«Железная» структура



Центральный сервер (ЦС)

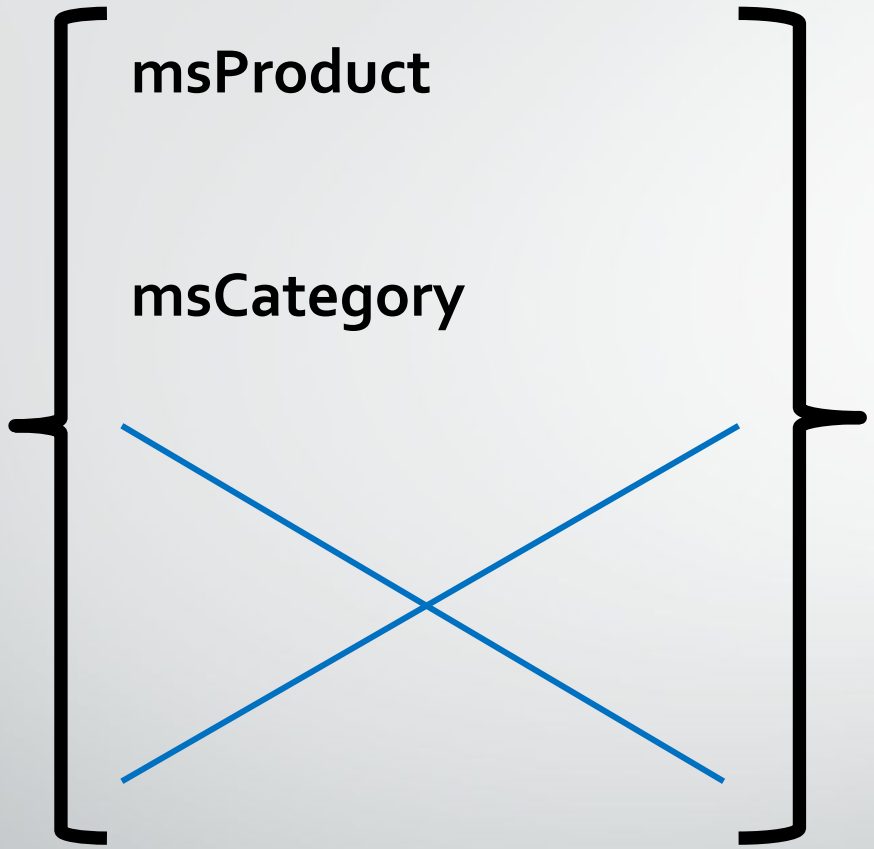
Физический локальный сервер (ЛС)

Синхронизатор: важное требование



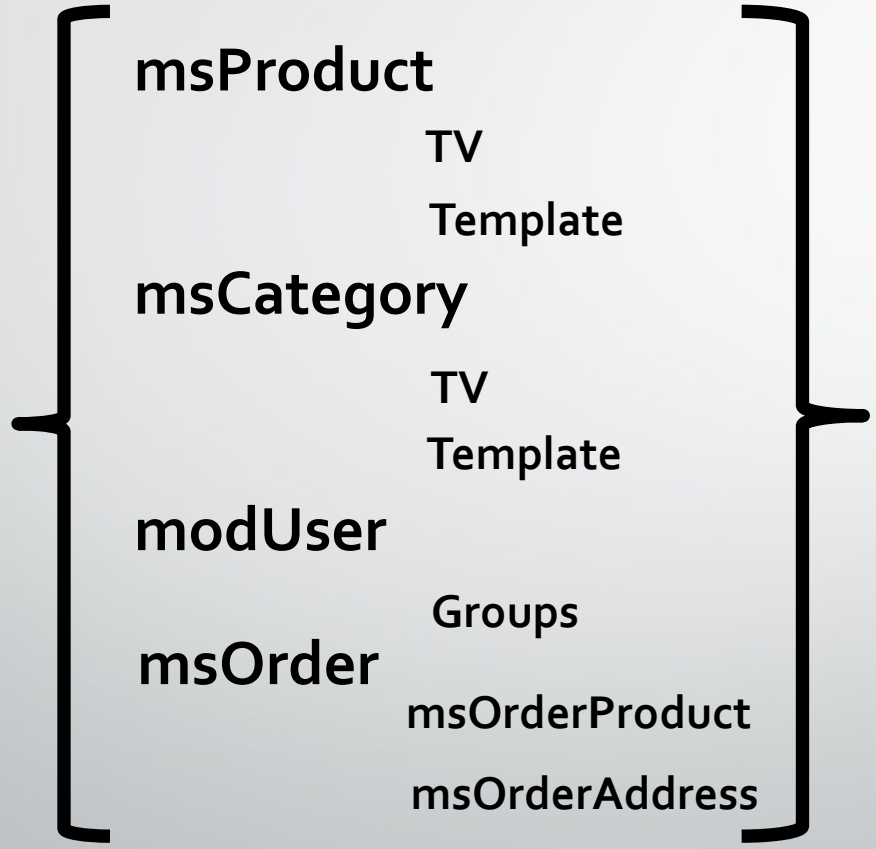
Динамическая поддержка любых объектов

Синхронизатор: первый шаг. Товары и категории



- Всего два типа объектов
- Минимальная обработка ошибок
- Недостаточная стабильность

Синхронизатор: развитие.
Улучшенная обработка ошибок + заказы



- Корректная обработка всех известных ошибок
- Обработка сопутствующих объектов
- Значительно расширенный список обрабатываемых объектов

Показатели на данный момент

База данных

- **400** товаров
- **3500** пользователей
- **500 000** заказов
- **1 000 000** товаров в заказах

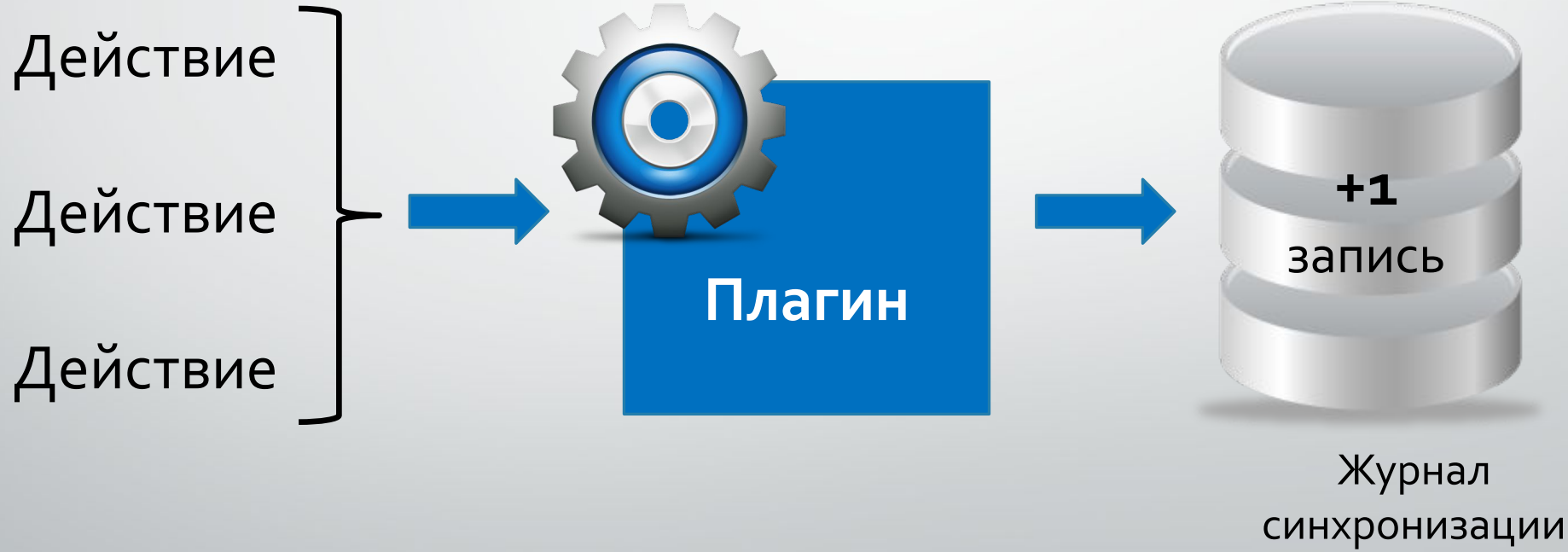
Серверы

- **1** центральный
- **14** локальных

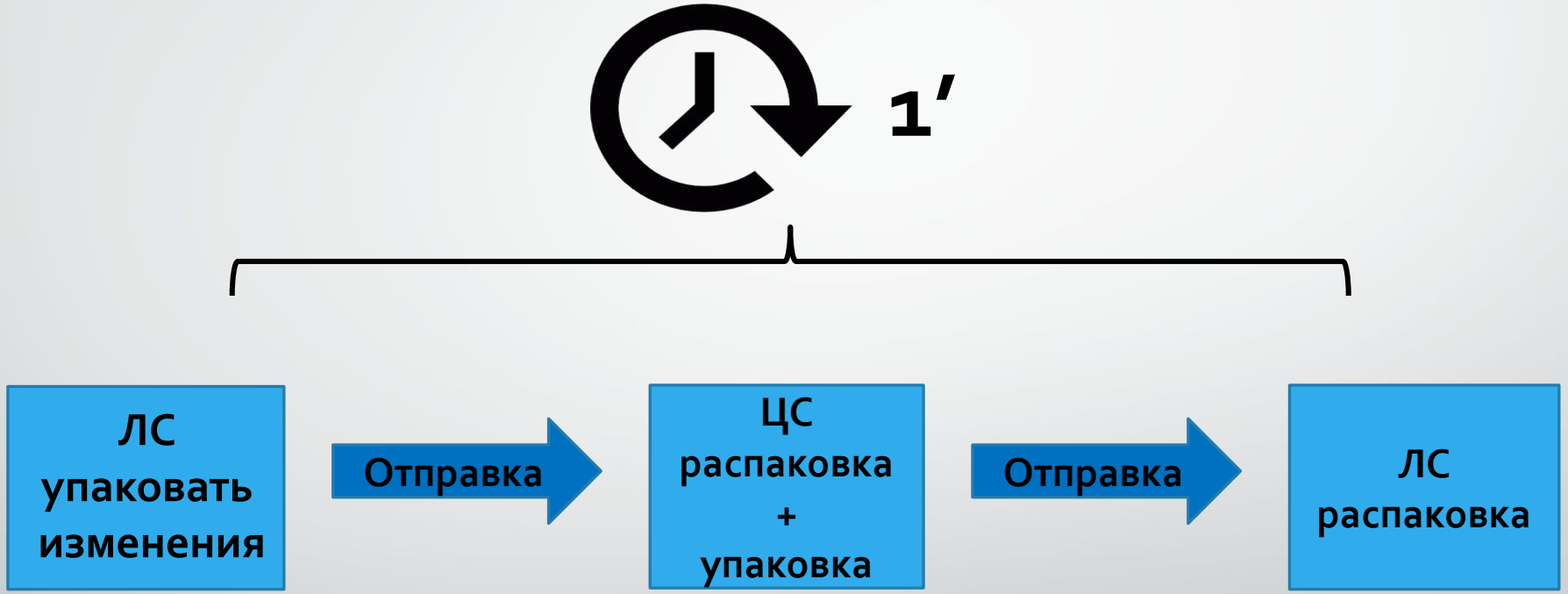
+ 20 типов обрабатываемых объектов, без учета сопутствующих

Принцип работы синхронизатора

- Отслеживание изменений



- Обмен данными между серверами



Правила синхронизации

```
class chsrmodUser extends chsrRule {
    // class code here
}

class chsrGuest extends chsrmodUser {
    // class code here
}
```

```
class chsrmodResource extends chsrRule {
    // class code here
}

class chsrmsProduct extends chsrmodResource {
    // class code here
}
```

Специальные классы, название которых состоит из специального префикса и названия оригинального класса объекта

Главные методы

Упаковать

sync::pack()

- получает из БД запись журнала об измененном объекте
- запускает **pack()** из соответствующего Правила

Распаковать

sync::unpack()

- получает весь массив данных для распаковки
- для каждого элемента запускает **unpack()** соответствующего Правила

```
class chSync {

   public function pack($stock_id = 0, $piece = '') {
      // function code here
   }

   public function unpack($package = array()) {
      // function code here
   }
}
```

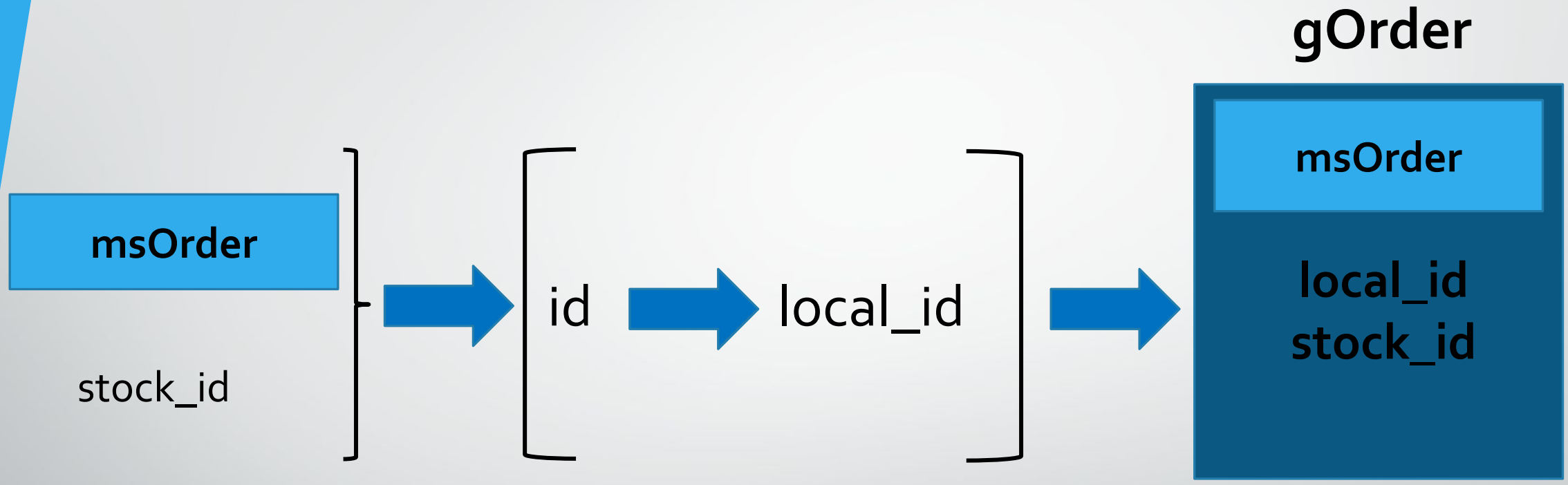
```
class chsrmodUser extends chsrRule {

   public function pack($journalEntry) {
      // function code here
   }

   public function unpack($package = array()) {
      // function code here
   }
}
```

Уникальные ключи

Уникализация msOrder



Обработка ошибок

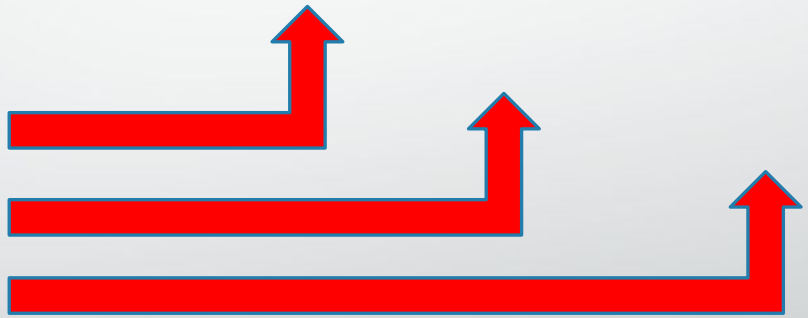
Прогнозируемые

- Заранее определяется критичность ошибки. Важно?
- Нет: запись пропускается, в журнале сохраняется статус ошибки.
- Да: Если обработать и исправить ошибку важно в ручном режиме, должно отправляться уведомление разработчику.

Непредвиденные

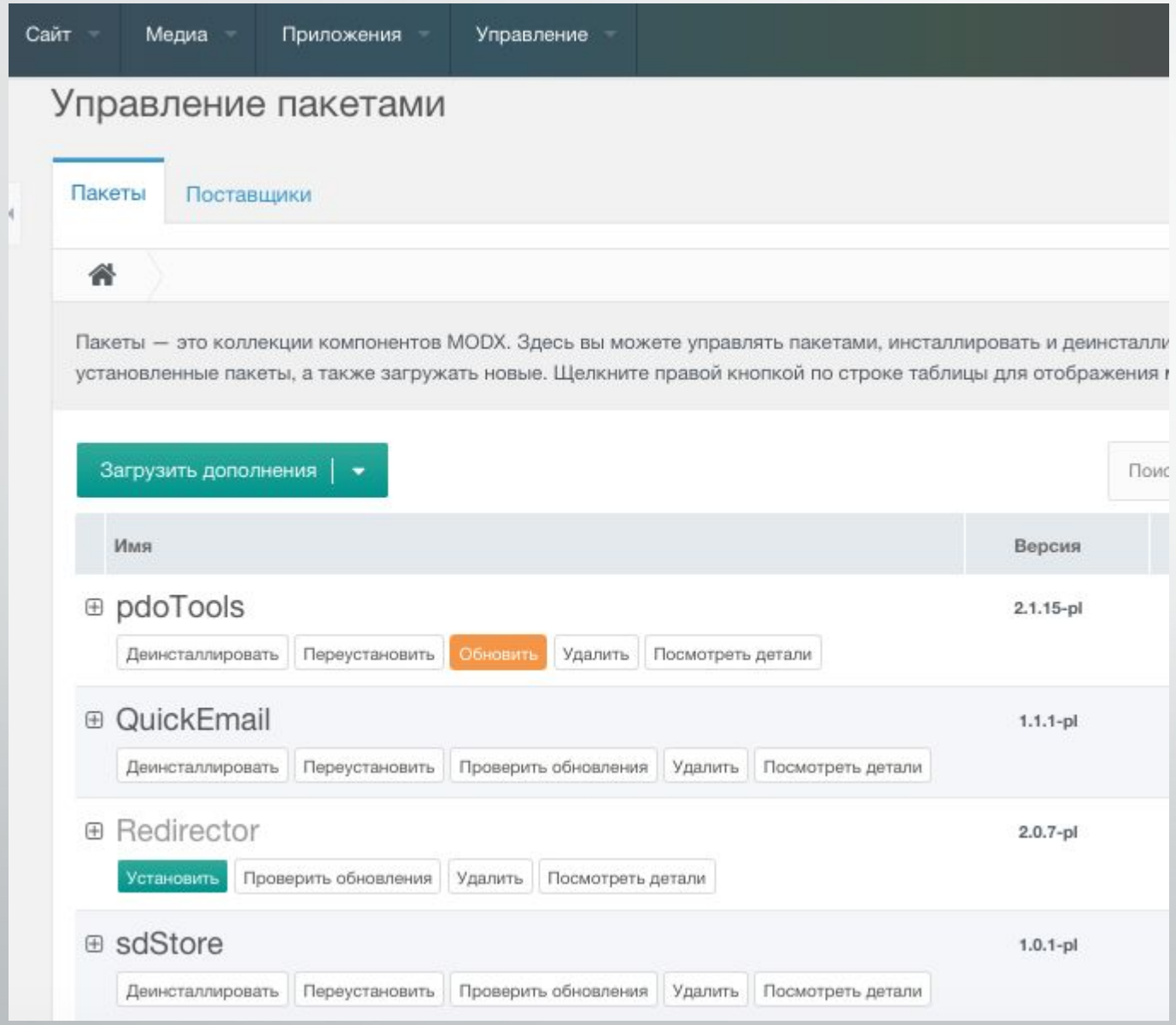
- Здесь все зависит от конкретной ситуации и времени обнаружения ошибки. В большинстве случаев все ограничиваетя остановкой синхронизации на некоторое время.

Обслуживание серверов



Выполнение сервисных команд

Обновление компонентов системы



Используется штатный механизм обновления пакетов, обновления происходят в автоматическом режиме после получения команды с ЦС

Перспективы развития

- Устранение старого кода в правилах, переход на максимальное использование принципов ООП и процессоров MODX
- Администрирование серверов:
  - Выполнение сервисных команд/задач
  - Выполнение небольших кусков кода
  - Мониторинг состояния

Заключение

Потенциал

Вывод

MODX

безграничен!

COMP AERO
Автоматизация бизнеса
и разработка ПО