Software Engineering Conference Russia

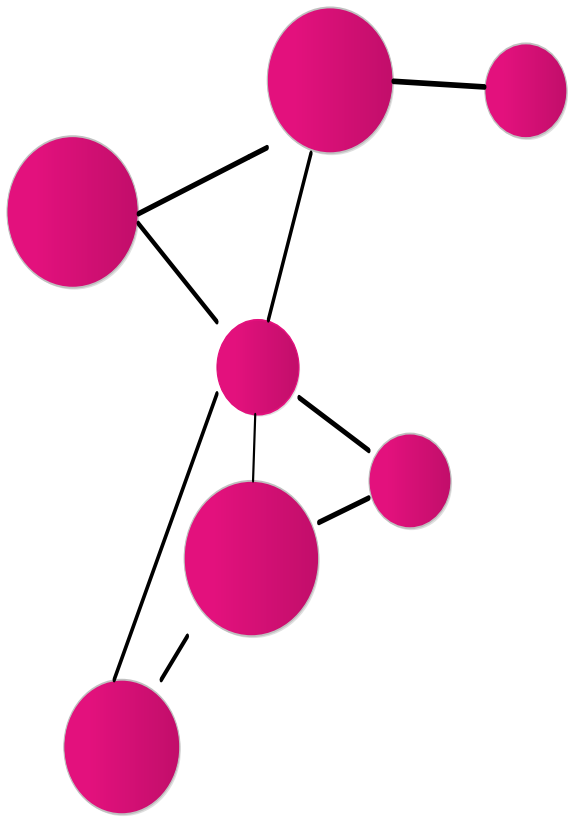
October 2017, St. Petersburg

Как сохранить работу распределенной IT-системы в эпоху бизнес-перемен?

Ольга Бенкен,

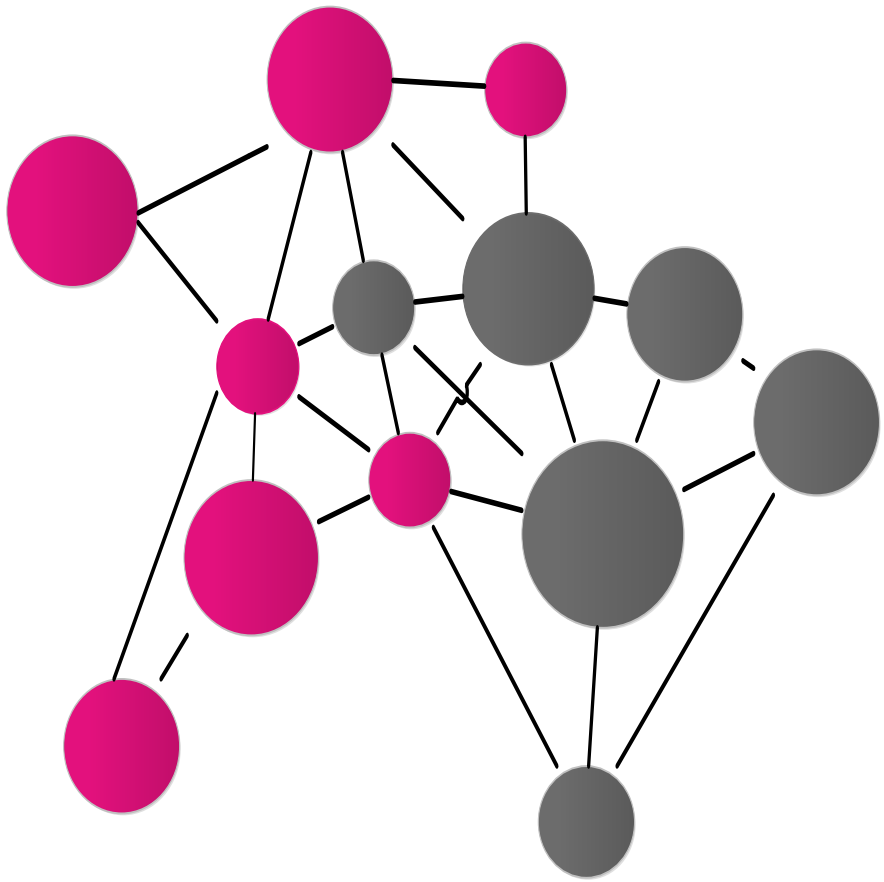
Ведущий бизнес-аналитик

Кто варил spaghetti?



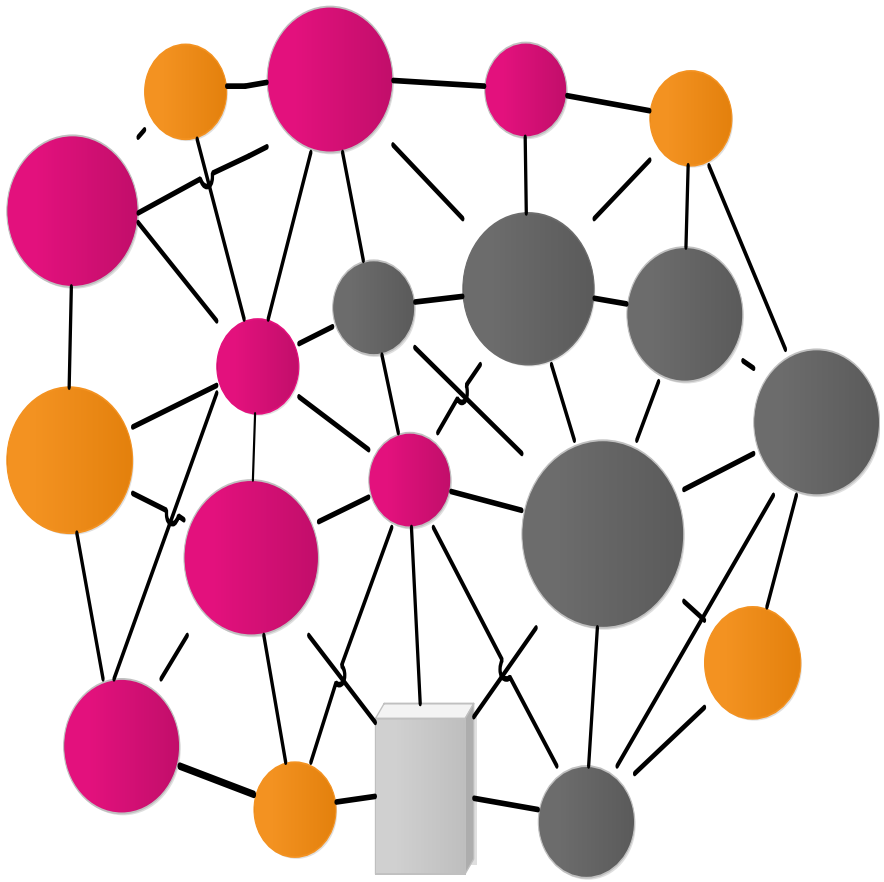
- >40 млн мобильных пользователей
- >13 млн пользователей DSL
- >100 систем в инфраструктуре технической поддержки
- крупнейший телекоммуникационный оператор Европы
- time-to-market не менее 9 месяцев
- динамично развивающаяся отрасль

Кто варил spaghetti?



- >40 млн мобильных пользователей
- >13 млн пользователей DSL
- >100 систем в инфраструктуре технической поддержки
- крупнейший телекоммуникационный оператор Европы
- time-to-market не менее 9 месяцев
- динамично развивающаяся отрасль

Кто варил spaghetti?

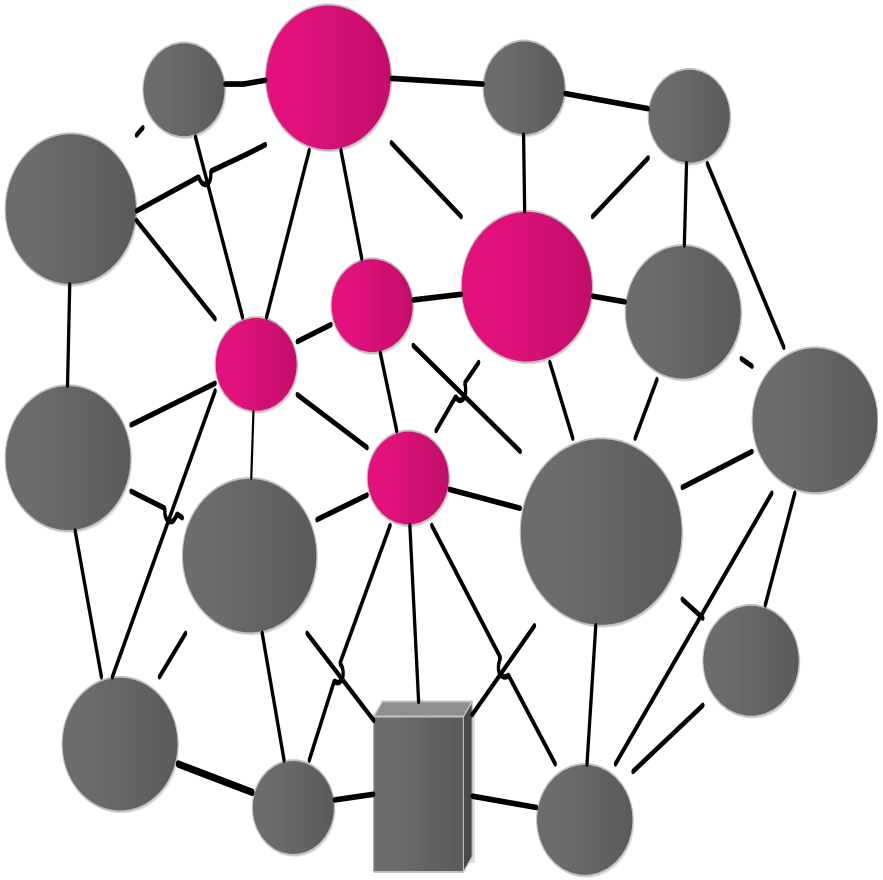


- >40 млн мобильных пользователей
- >13 млн пользователей DSL
- >100 систем в инфраструктуре технической поддержки
- крупнейший телекоммуникационный оператор Европы
- time-to-market не менее 9 месяцев
- динамично развивающаяся отрасль

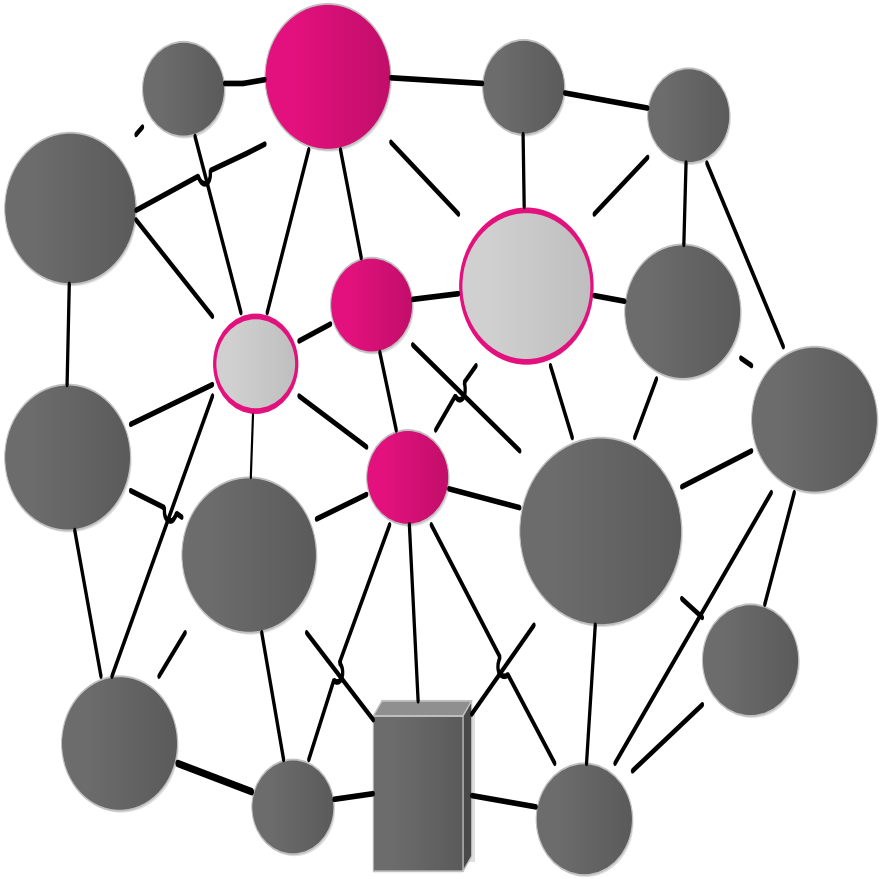
Переписать нельзя поддерживать

Контракты	Нужно поддерживать старые контракты и услуги
Hardware	Update старых систем по требованию data centers
Конкуренты	Нужно предлагать новые услуги, чтобы оставаться конкурентноспособными
Клиенты	Сервис текоммуникационного оператора должен быть доступен 24/7
Бюджет и ресурсы	Не безграничны

Классическая проблема point-to-point



Классическая проблема point-to-point

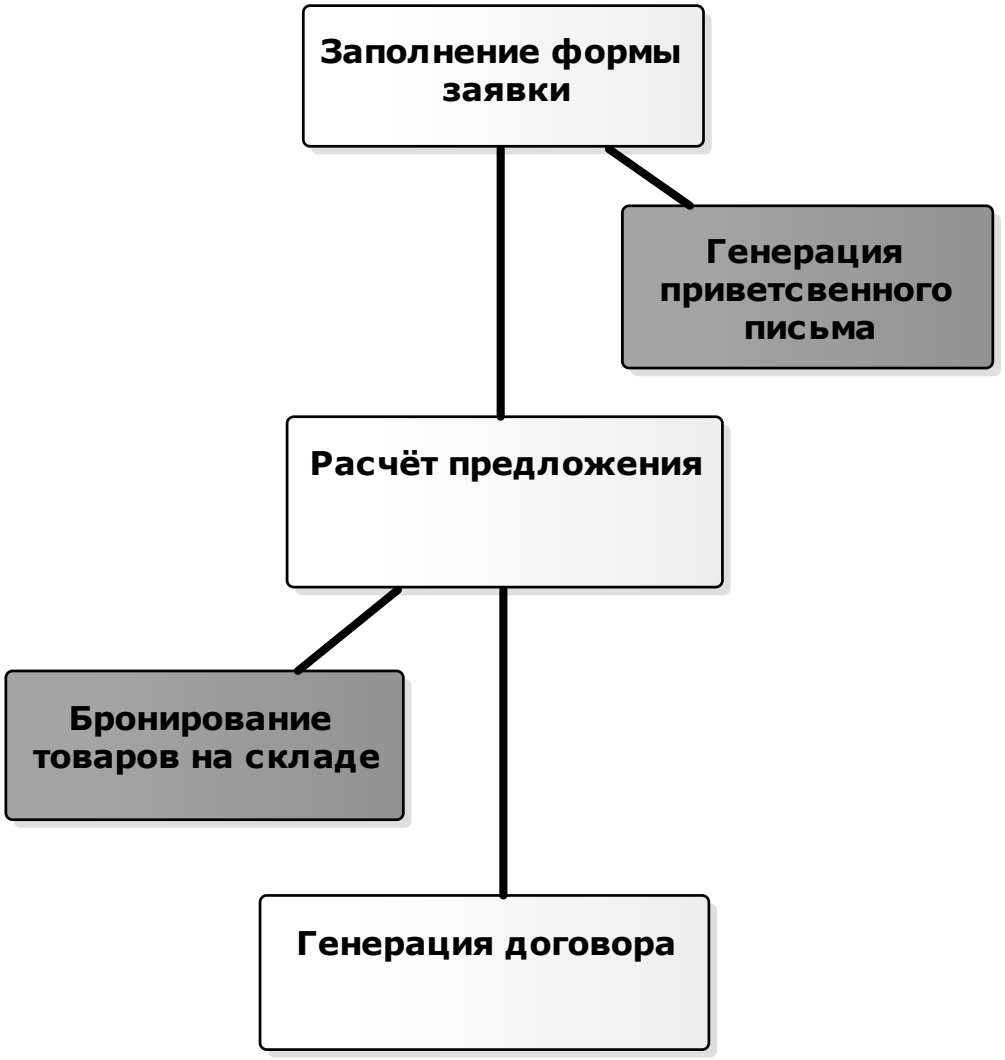


Зависимости делятся на

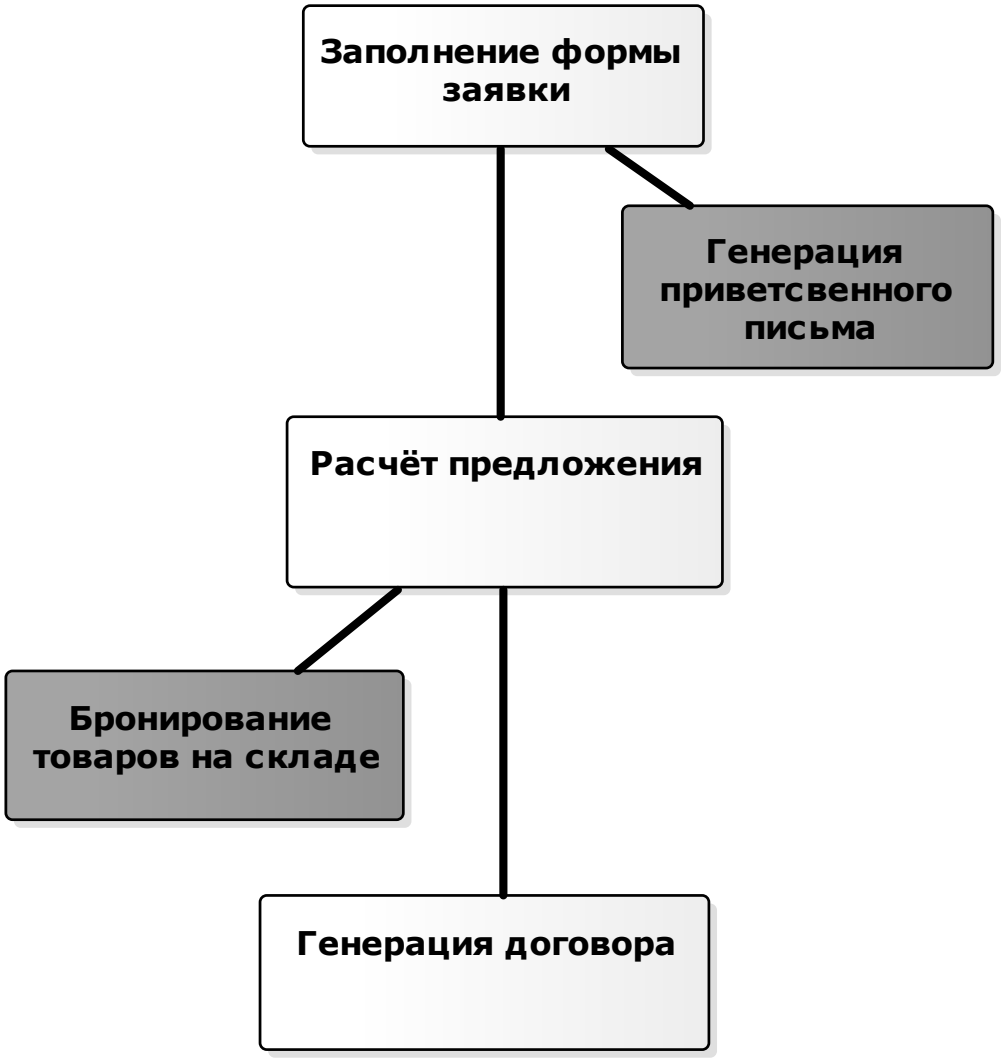
критичные

опциональные

Что такое каскадная зависимость? Пример



Что такое каскадная зависимость? Пример



Бизнес-требование

Организовать работу с юридическими лицами

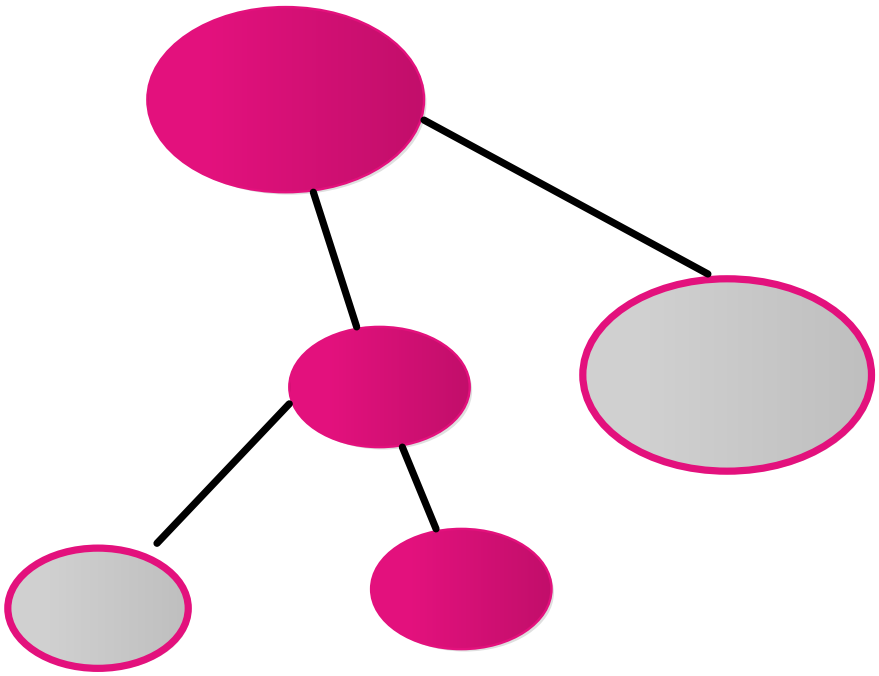
Критичные компоненты

- Заполнение формы заявки
- Расчет предложения
- Генерация договора

Опциональные компоненты

- Бронирование товаров на складе
- Генерация приветственного письма

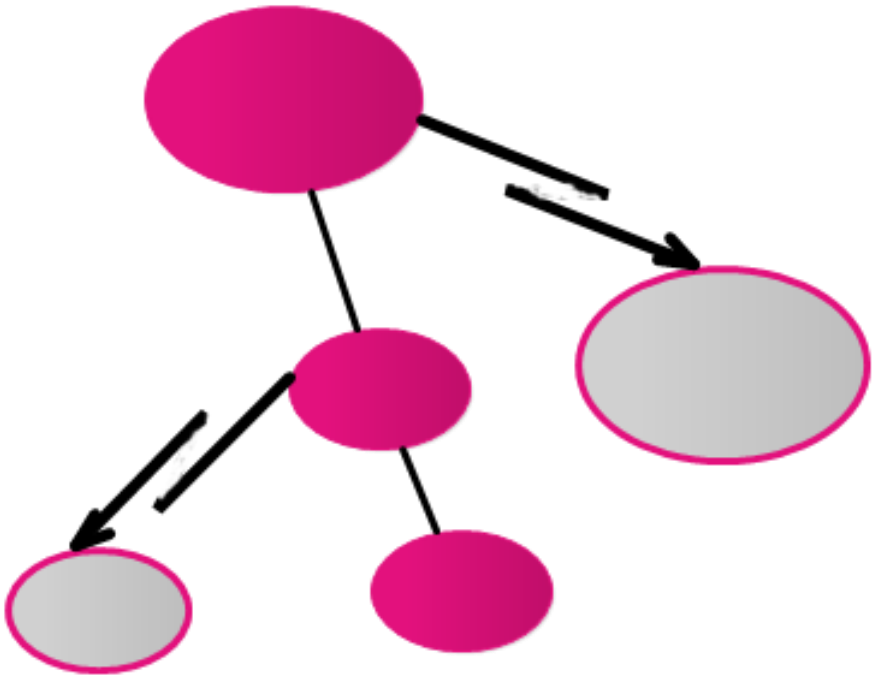
Сколько стоит поддержка?



- Затраты на обновление критичных и опциональных компонентов сопоставимы
- Обновление опциональных компонентов практически не расширяет функциональность системы

разрываем зависимости

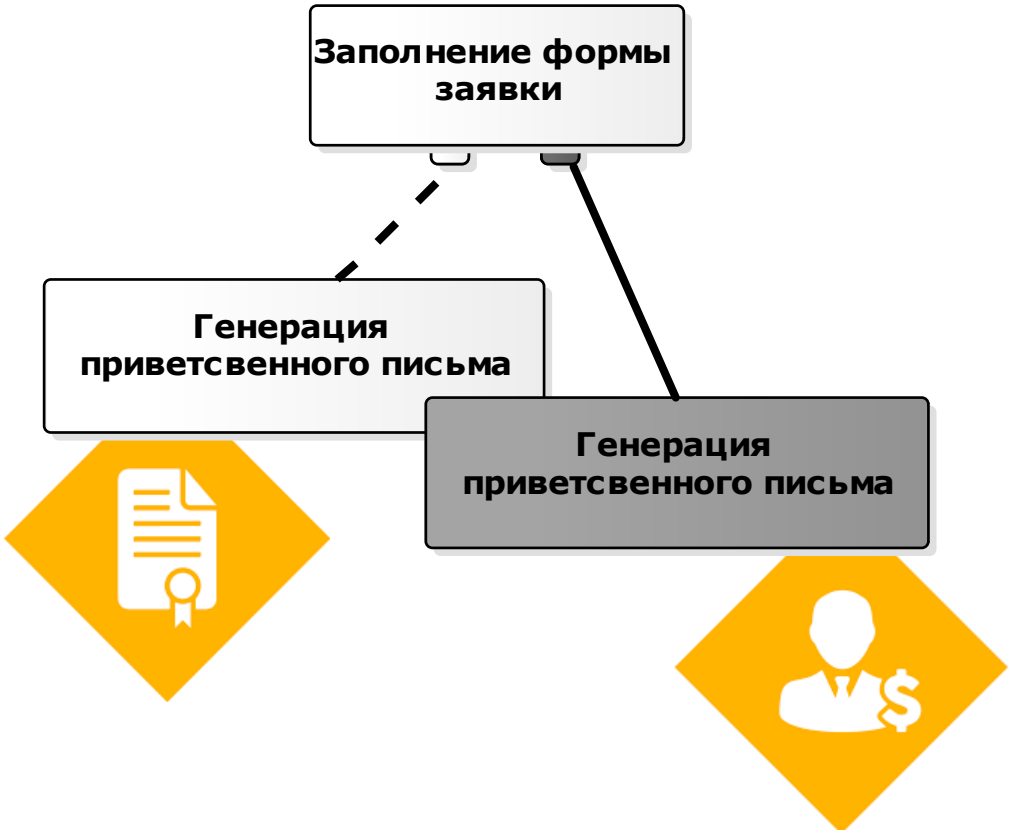
Сколько стоит поддержка? - Дорого!



Разрыв зависимостей

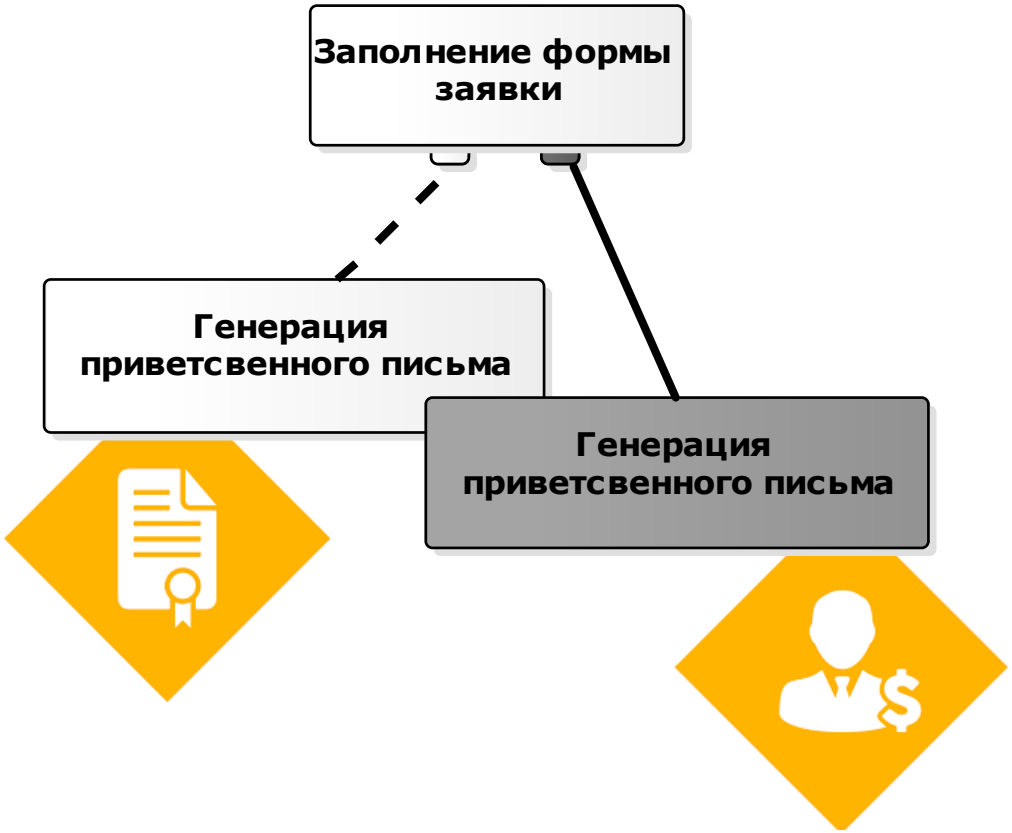
Проблема «стыковки» систем с разными версиями интерфейсов

Разрываем зависимости. Ключи



- Обновленный компонент работает с двумя версиями зависимости
- Ключ определяет версию зависимой системы

Разрываем зависимости. Ключи

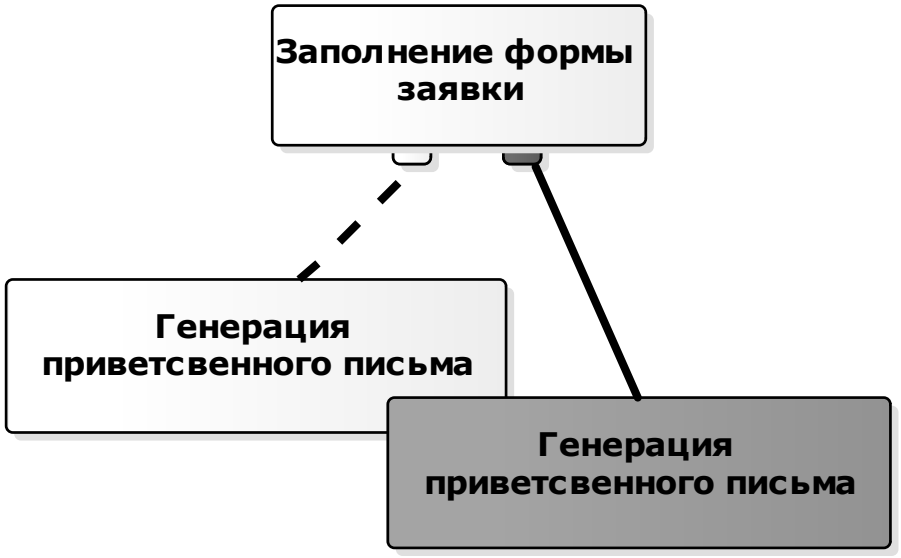


- Обновленный компонент работает с двумя версиями зависимости
- Ключ определяет версию зависимой системы

Компонент «Заполнение формы заявки» умеет посылать данные юр. лица

- в форме ФИО для физического лица (старый интерфейс)
- в новом объекте для юридического лица (новый интерфейс)

Разрываем зависимости. Ключи



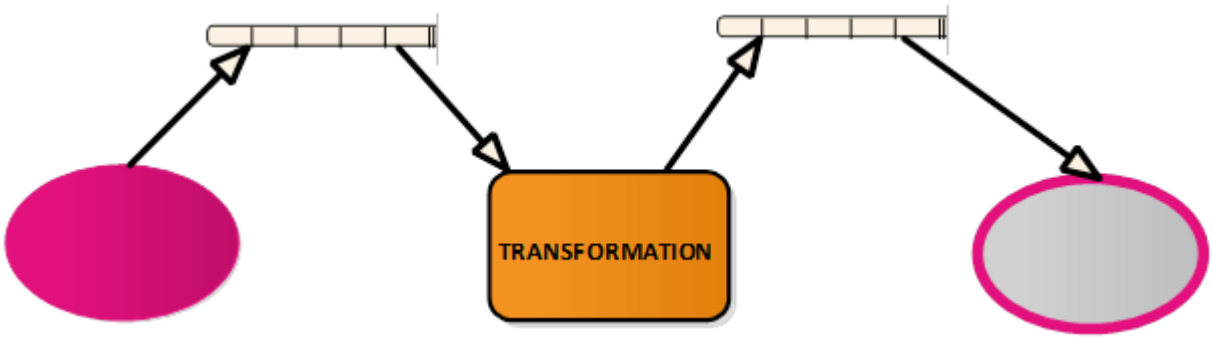
Плюсы

- скорость выхода новых функций на рынок
- обновление зависимостей за несколько релизов
- рабочее состояние системы независимо от версий

Минусы

- реализация ключей в коде компонентов
- двойная работа по обслуживанию ключа из-за необходимости его создания и удаления

Разрываем зависимости. MQ-маршрутизатор



Разрыв зависимости на уровне транспорта

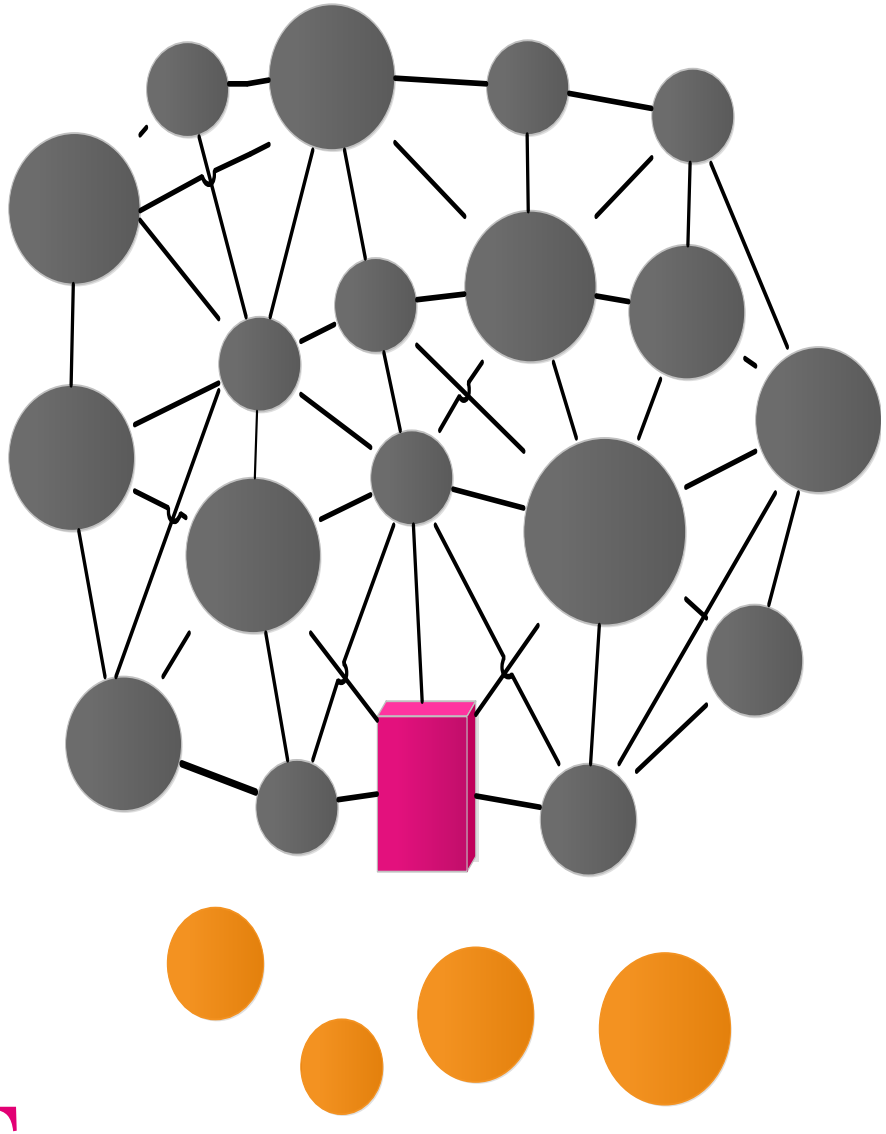
Плюсы

- Код компонентов не затронут

Минусы

- Тяжеловесное решение, привязанное к платформе IBM
- Работает с одним видом транспорта
- Изменения проходят через все фазы подготовки релиза

Сколько стоит развитие? – Очень дорого!

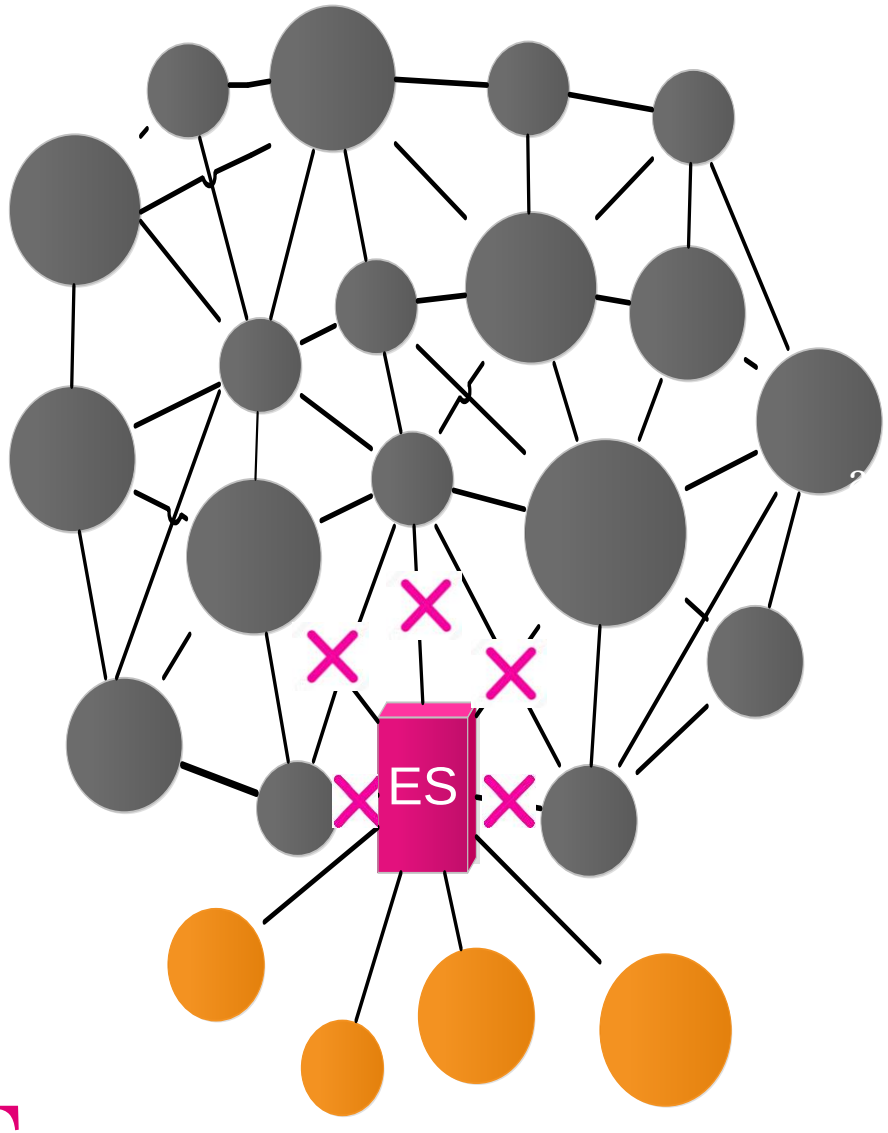


Для развития системы необходимо

- Расширение числа компонентов
- Новые требования к инфраструктуре
  - стабильность
  - возможность частой модификации
  - увеличение скорости взаимодействия

LIFE IS FOR SHARING.

Required: new transport



Стабильность

+

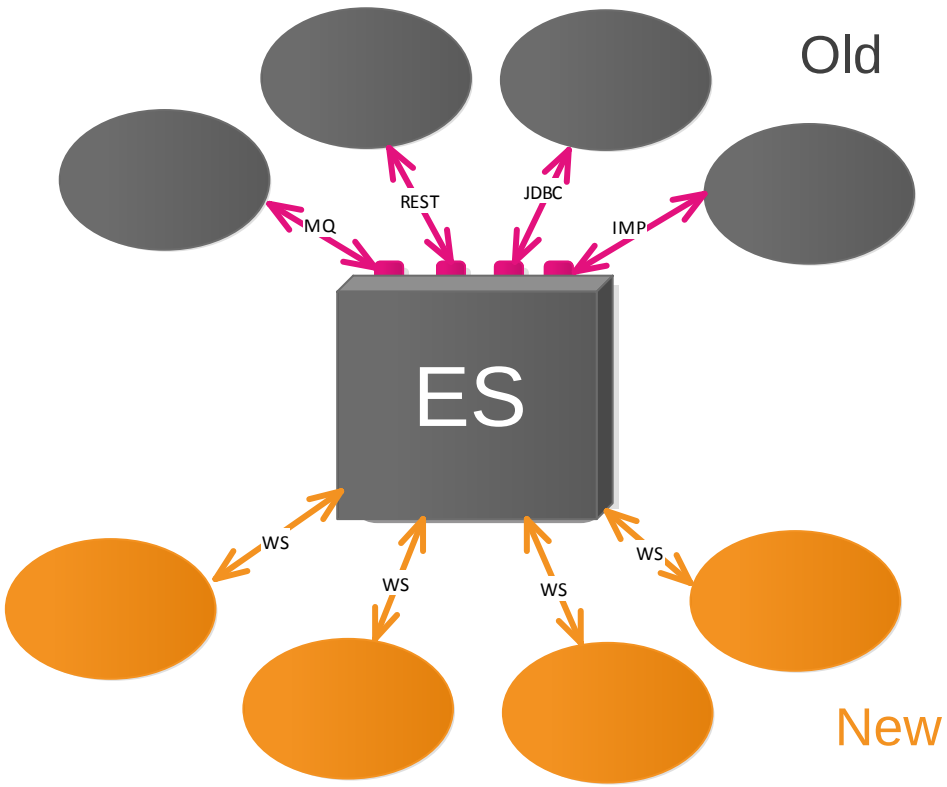
Увеличение скорости взаимодействия

Новый транспортный протокол

Ограниченная поддержка старого протокола

LIFE IS FOR SHARING.

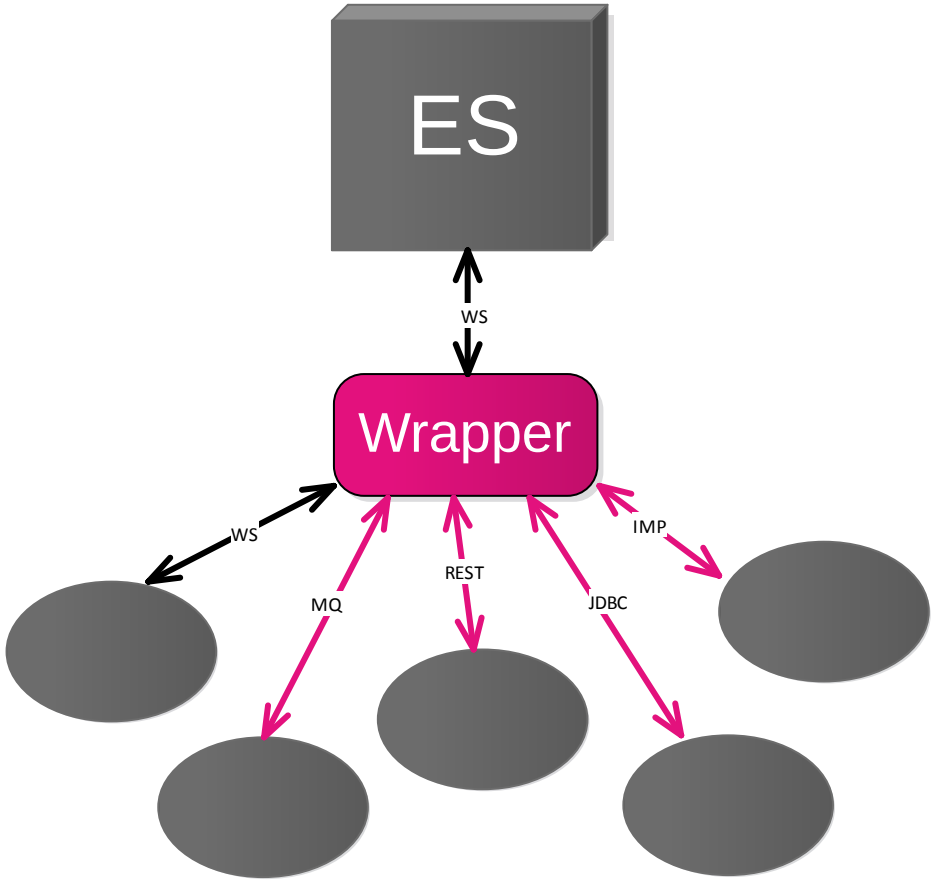
Required: new transport



Для старых зависимостей доступны два варианта обновления:

- Платная кастомизация выходного модуля коробочного продукта
- Перевод системы на новый транспортный протокол

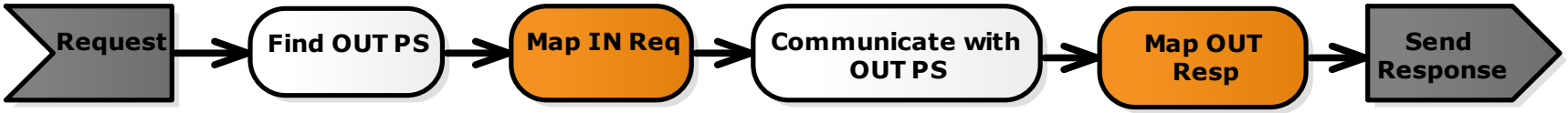
Wanted: адаптер с поддержкой транспортов



Требования

- Поддержка разных транспортных протоколов
- Трансформация на уровне конфигурации (без повторной сборки)
- Отсутствие требований к формату сообщений
- Нагрузка 200 000 запросов в день

Wrapper. Схема работы



In-Out communication

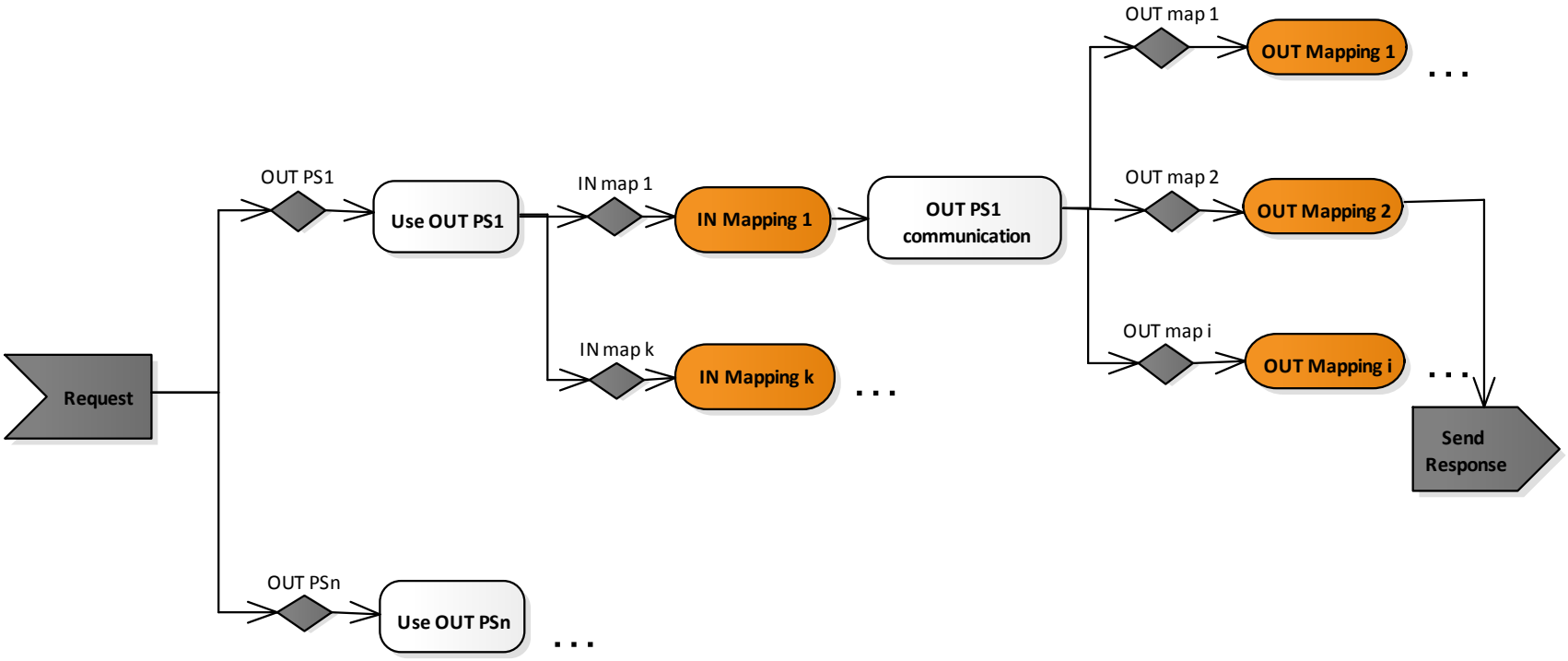
- Поддержка транспортных протоколов (WS, MQ, REST, DB, IMP)

Transformation

- Собственный DSL для
  - парсинга входящих и
  - формирования новых XML-сообщений на основе XSD
  - вычисления выражений

Wrapper. Схема работы

Поддержка условий позволяет сделать процесс обработки запроса гибким и формировать разные сценарии обработки в зависимости от контекста.



Wrapper. Результаты внедрения

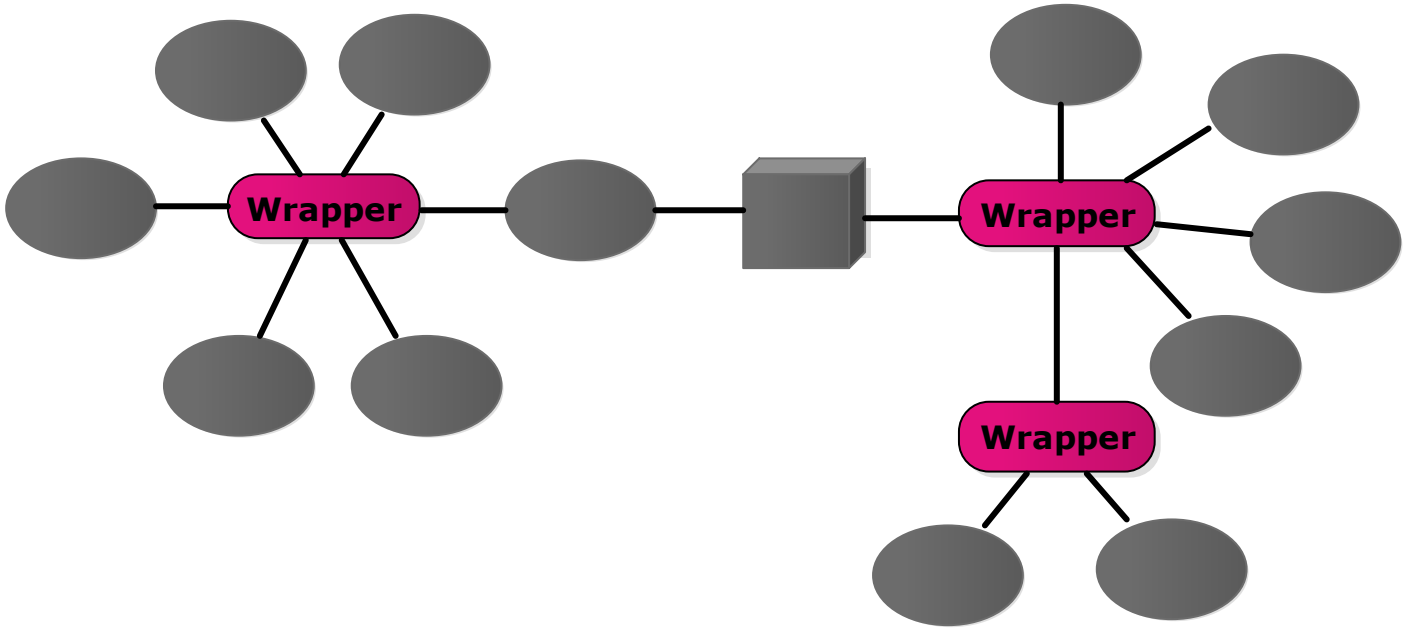
Вынесение согласования интерфейсов и транспортов на уровень конфигурации позволило

- до 4х раз сократить время и расходы на поддержку каскадных зависимостей
- гибко изменять системный ландшафт
- удвоить число опрашиваемых компонентов (9 -> 18)
- внедрить модули статистики и анонимизации

Wrapper. Движение к Process Engine

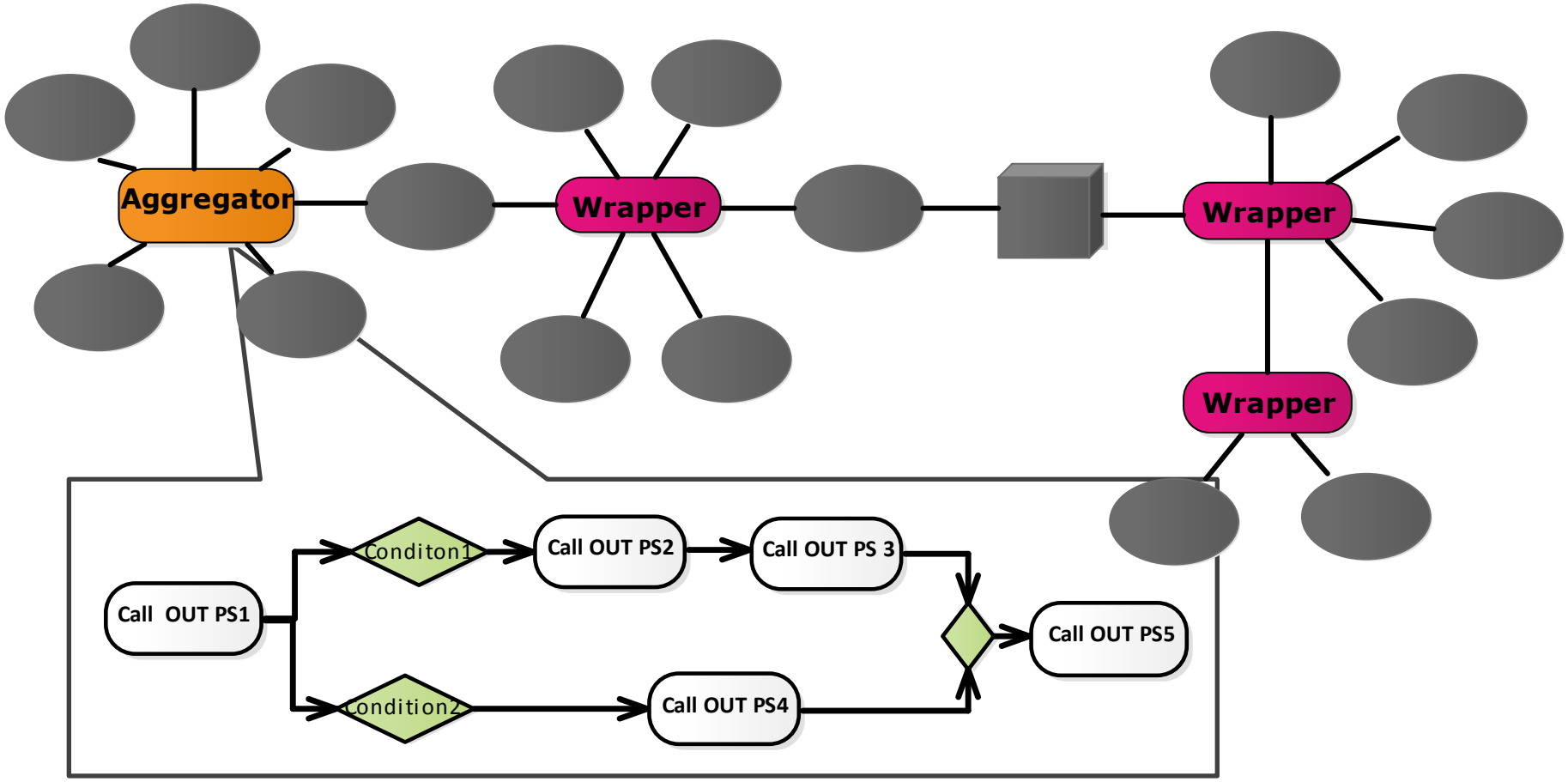
- Использование в трех сегментах сети приводит к тому, что запрос может проходить через Wrapper более одного раза

Wrapper влияет на работу не двух компонентов, а целой цепи



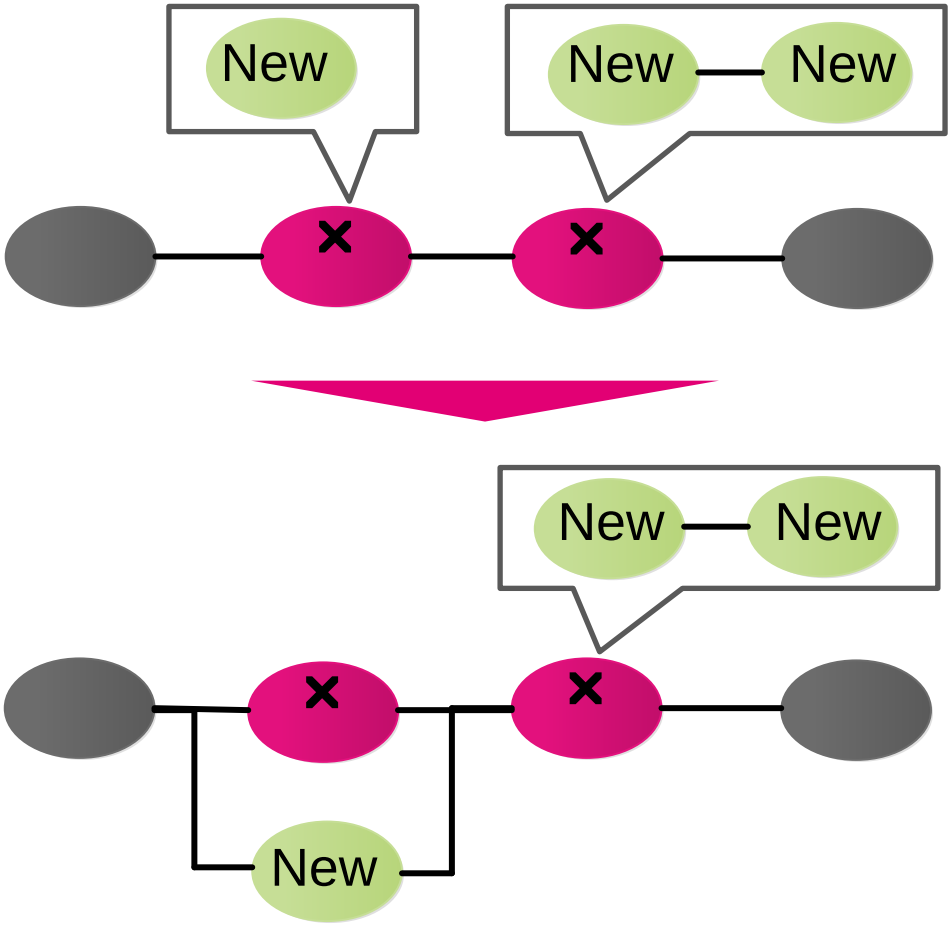
Wrapper. Движение к Process Engine

- Исключение систем-агрегаторов из топологии системы



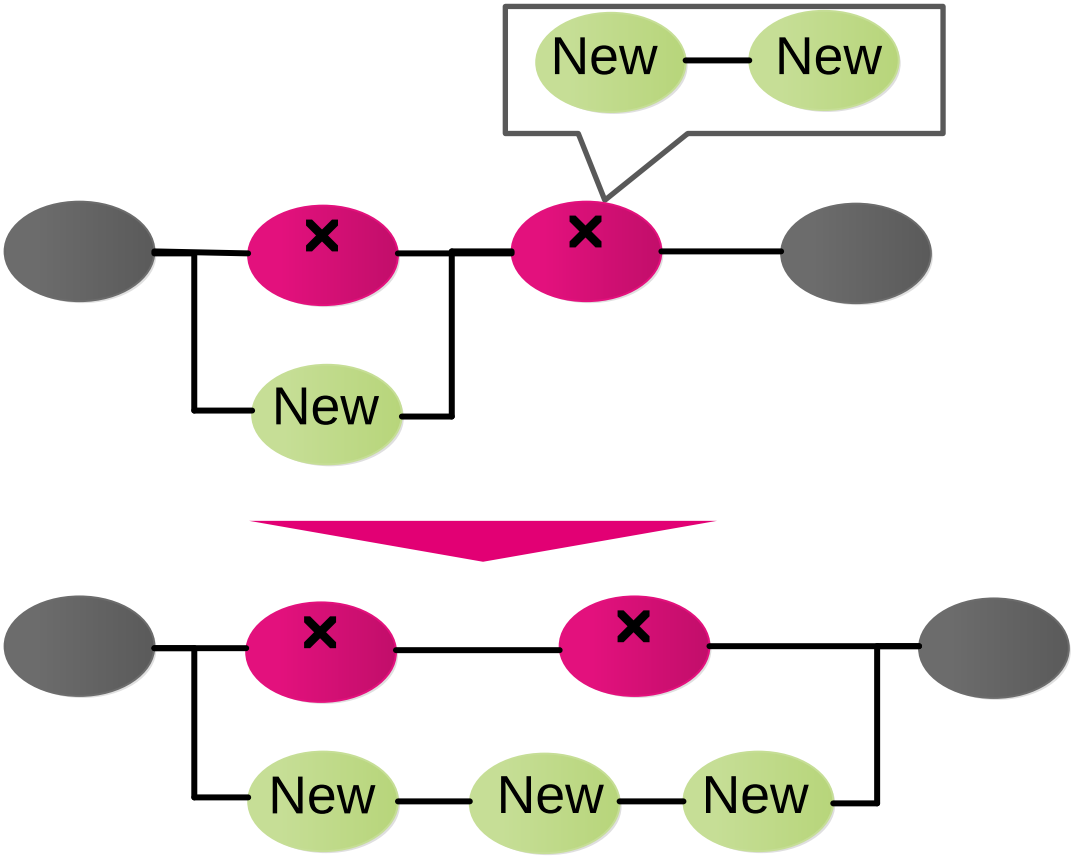
Wrapper. Движение к Process Engine

- Гибкое обновление компонентов системы, проходящее в несколько этапов



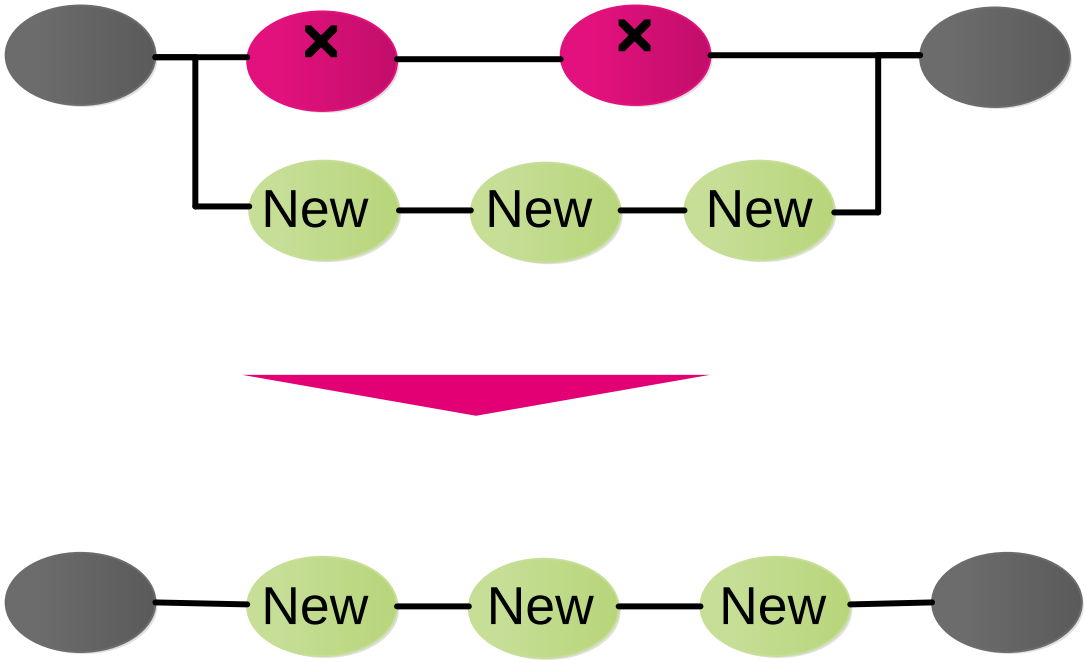
Wrapper. Движение к Process Engine

- Гибкое обновление компонентов системы, проходящее в несколько этапов



Wrapper. Движение к Process Engine

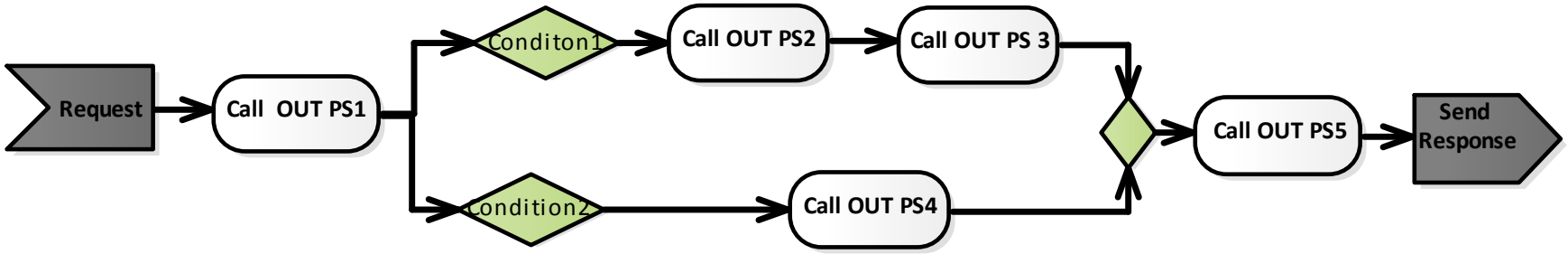
- Гибкое обновление компонентов системы, проходящее в несколько этапов



Wrapper. Process Engine

Основная идея следующей версии Wrapper-а –

перейти от вызова одной системы к выполнению бизнес-процесса.

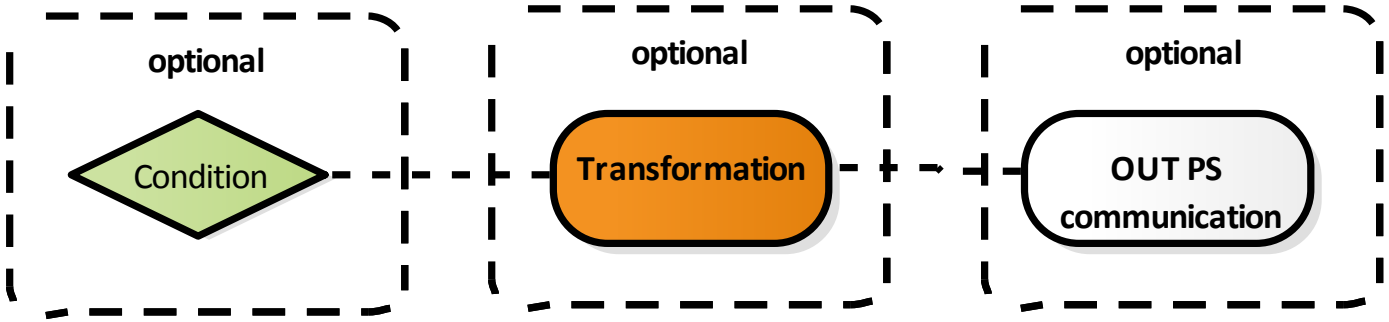


Wrapper. Process Engine

Бизнес-процесс разбит на шаги, состоящие из пары опциональных активностей

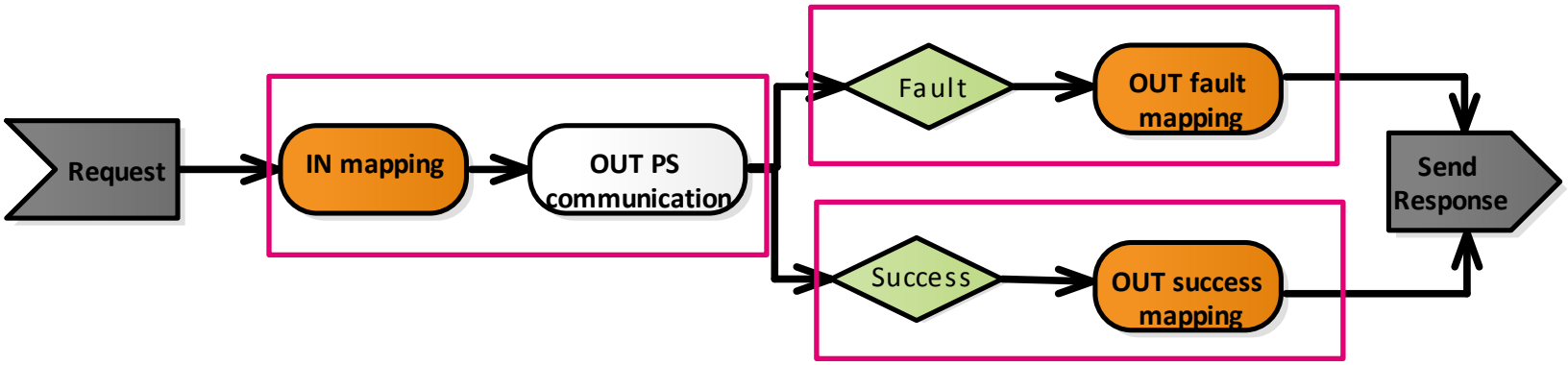
- трансформация XML-сообщения
- коммуникация со сторонней системой,

выполнение которых происходит в случае удовлетворения заданных условий.



Wrapper. Process Engine

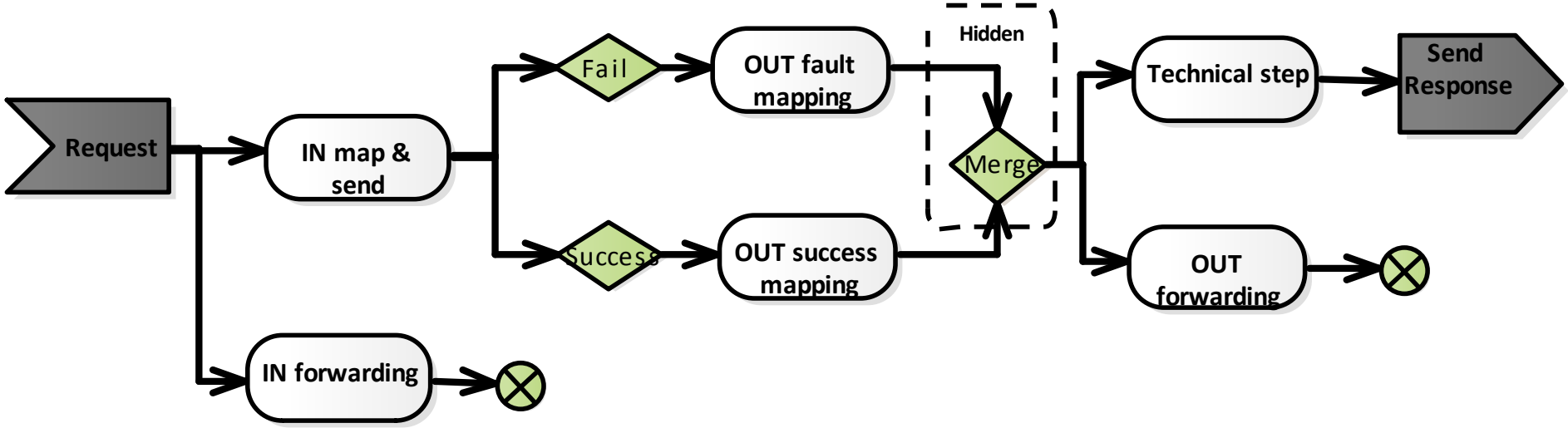
Простейший сценарий обработки запроса может быть представлен в виде трех шагов.



Wrapper. Process Engine. Control flow

Все варианты gateways, предусмотренные в BPMN 2.0 могут быть реализованы в Process Engine, в том числе

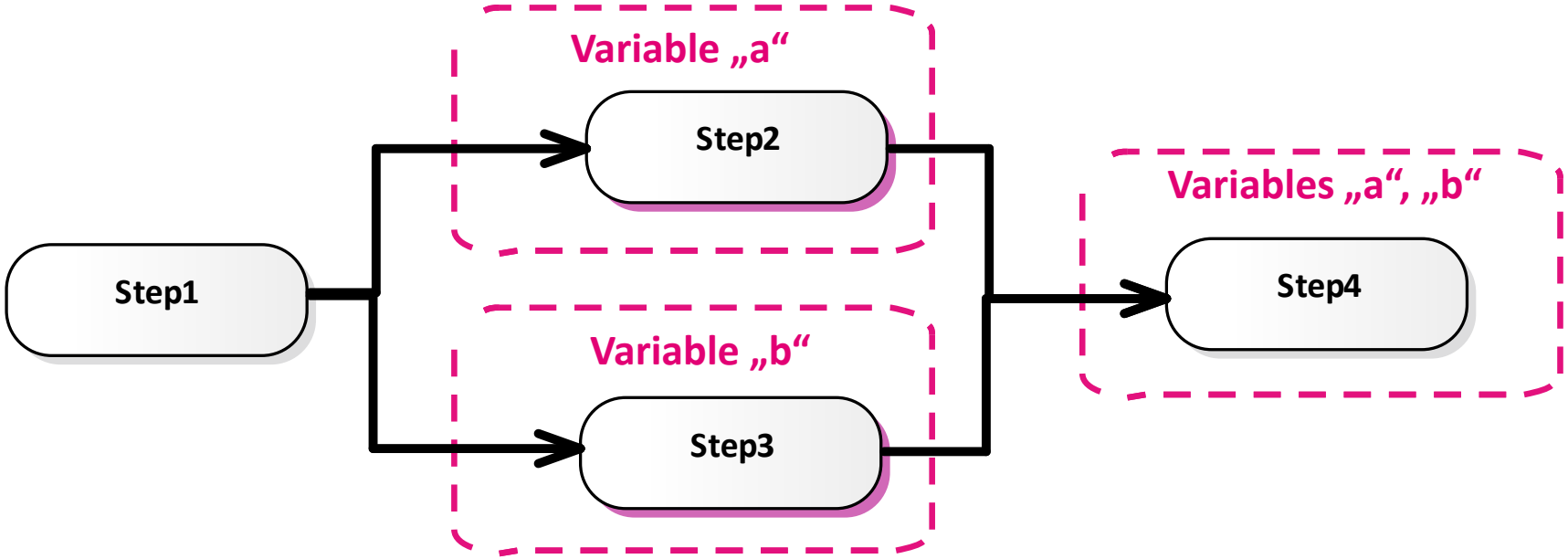
- process Fork/Join
- process Branch/Merge



Wrapper. Process Engine. Data flow

Поток данных проходит через каждый шаг процесса, даже если данные внутри него не изменяются. Причем,

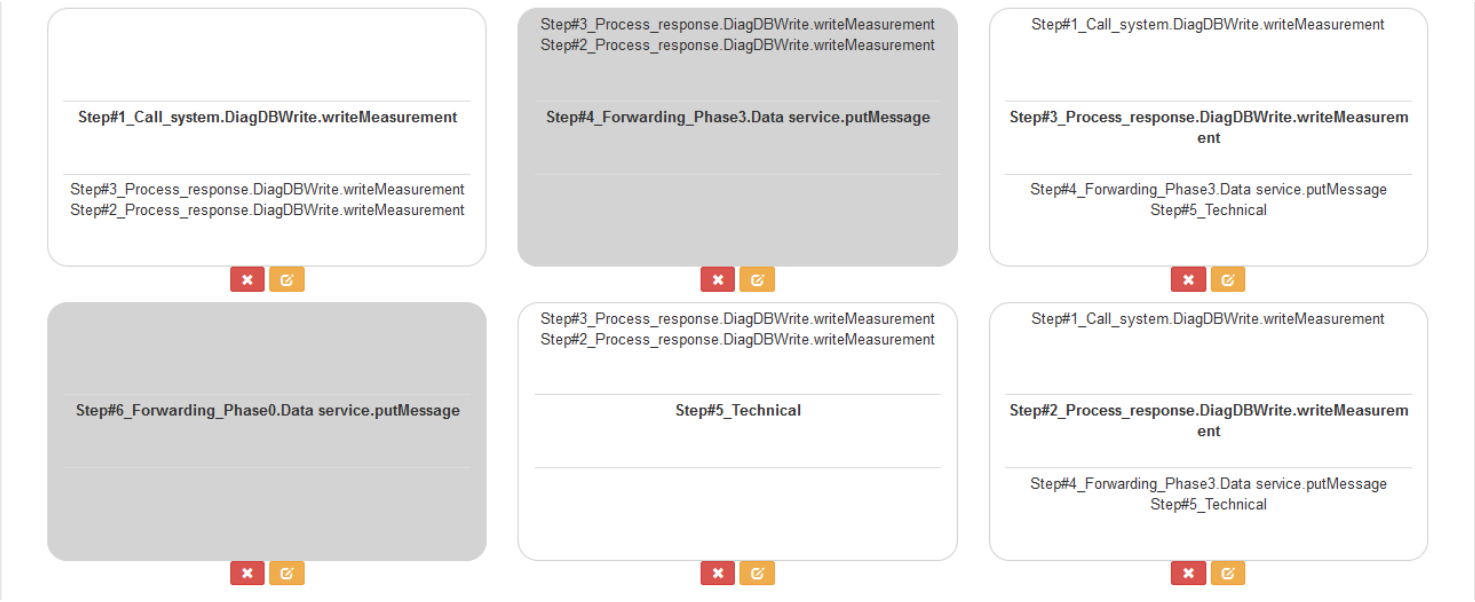
- данные каждого шага изолированы
- шаги могут обмениваться данными через переменные ограниченной области видимости.



Wrapper. Process Engine. Optional features

В сравнении с промышленными BPM решениями Wrapper-PE не содержит

- графического редактора процессов
- модуля статистики и анализа
- расширенных функций модуля мониторинга



Результаты одним словом

Гибкость

Результаты чуть более подробно

Вынесение маршрутизации и согласования интерфейсов на уровень конфигурации позволяет

- уйти от жестких связей
- сократить time-to-market до 4х раз
- исключить из системы компоненты, не реализующие бизнес-функции
- обеспечить поэтапное внедрение новых систем в разные сегменты сети

Спасибо!

Wrapper. Process Engine. Linkages

Linkages configuration

Method

Solvatio > SKS-WMS > SKS-SST

Available processes

Solvatio.SKS-WMS.SKS-SST_mmap_id_2

Condition: `element_exists(query("Envelope/Body/skswms"))`

+ New

Handled exceptions

INTERNAL

+ New

Upload ZIP | ← Go back | Download linkages for the method

Wrapper. Process Engine. Process view

Edit process

Name

Solvatio.DiagDB.writeMeasurement_mmap_id_11

Timeout

0

In Methods

Solvatio -> DiagDB -> writeMeasurement

Description

Process Description

Steps

Step#1_Call_system.DiagDBWrite.writeMeasurement

Step#3_Process_response.DiagDBWrite.writeMeasurement
Step#2_Process_response.DiagDBWrite.writeMeasurement

Step#3_Process_response.DiagDBWrite.writeMeasurement
Step#2_Process_response.DiagDBWrite.writeMeasurement

Step#4_Forwarding_Phase3.Data service.putMessage

Step#1_Call_system.DiagDBWrite.writeMeasurement

Step#3_Process_response.DiagDBWrite.writeMeasurement

Step#4_Forwarding_Phase3.Data service.putMessage
Step#5_Technical

Step#6_Forwarding_Phase0.Data service.putMessage

Step#3_Process_response.DiagDBWrite.writeMeasurement
Step#2_Process_response.DiagDBWrite.writeMeasurement

Step#5_Technical

Step#1_Call_system.DiagDBWrite.writeMeasurement

Step#2_Process_response.DiagDBWrite.writeMeasurement

Step#4_Forwarding_Phase3.Data service.putMessage
Step#5_Technical

Wrapper. Process Engine. Step view

Edit step

Step name

Name: Step#1_Call_system.OpDiNG_DiagnosticsManagementConsumer_v02.00.executeResolution

Step description

Description:

Previous steps

No previous steps

Next steps

Step#3_Process_response.OpDiNG_DiagnosticsManagement...

Step#4_Process_response.OpDiNG_DiagnosticsManagement...

Step#2_Process_response.OpDiNG_DiagnosticsManagement

Active

Step condition

Condition: element_exists(query("Envelope/Body/isi_remove_password_suspension_query"))

Step condition description

Condition description:

Transformation

```
    <csdg:from>{{ __properties["csdg_from"] }}</csdg:from>
    <csdg:routingInfo>{{ __properties["csdg_routing_info"] }}</csdg:routingInfo>
    <csdg:messageId>{{ __properties["async_message_id"] }}</csdg:messageId>
    <csdg:currentSenderTimestampUTC>{{ format_date(date(), "yyyy-MM-dd'T'HH:mm:ss.SSSU") }}</csdg:currentSenderTimestampUTC>
    <csdg:expiryOffsetInMillis>{{ __properties["async_timeout"] }}</csdg:expiryOffsetInMillis>
    </csdg:technicalContext>
  <csdg:businessContext>
    <csdg:processId>{{ uuid() }}</csdg:processId>
    <csdg:processTypeId>executeResolution</csdg:processTypeId>
   </csdg:businessContext>
</typ:context>
```

✖ Hide and clear ✖ Clear

Out method

Partner system: OpDiNG | Service: OpDiNG_DiagnosticsManagementConsumer_v02.00 | Method: executeResolution

✖ Hide and clear

✔ Save ✖ Cancel

Wrapper. Process Engine. Step statuses

An executed step can be in one of the following statuses

- STARTED – initial status of a step being executed
- CANCELLED – condition of the step or its ancestors was not met
- FINISHED – transformation and/or call of OUT PS were finished successfully
- FAILED – exception occurred by execution the step or its ancestor

Wrapper. Process Engine. Steps order

Sequence 1 2 3

Step#1_Call_system.DiagDBWrite.writeMeasurement

Step#3_Process_response.DiagDBWrite.writeMeasurement
Step#2_Process_response.DiagDBWrite.writeMeasurement

Step#1_Call_system.DiagDBWrite.writeMeasurement

Step#3_Process_response.DiagDBWrite.writeMeasurement

Step#4_Forwarding_Phase3.Data service.putMessage
Step#5_Technical

Step#3_Process_response.DiagDBWrite.writeMeasurement
Step#2_Process_response.DiagDBWrite.writeMeasurement

Step#5_Technical

Step#1_Call_system.DiagDBWrite.writeMeasurement

Step#2_Process_response.DiagDBWrite.writeMeasurement

Step#4_Forwarding_Phase3.Data service.putMessage
Step#5_Technical

Step#3_Process_response.DiagDBWrite.writeMeasurement
Step#2_Process_response.DiagDBWrite.writeMeasurement

Step#4_Forwarding_Phase3.Data service.putMessage

Fail steps

Step#6_Forwarding_Phase0.Data service.putMessage

Step#7_ExceptionHandler

NOT IMPLEMENTED