Антон Тюков

Применение ТРИЗ при проектировании архитектуры системы управления на основе данных в энергетике

ТРИЗ Саммит 2020

О спикере

Тюков Антон Павлович
(К.т.н., МАТРИЗ 1 ур. 2014, сертификат бережливого производства центра «Кайдзен»)

Руководить R&D ИТ компании (15 человек), которая разрабатывает алгоритмы управления в энергетике для Европейского рынка (Бельгия, Польша, Голландия, Франция и т.д.)

Модератор 30+ стратегических сессий

Цель презентации

Как Архитектор систем управления на основе данных я нашел практики, которые работают

Использовал свои знания в ТРИЗ, чтобы их сделать сильнее

Хочу поделиться практиками

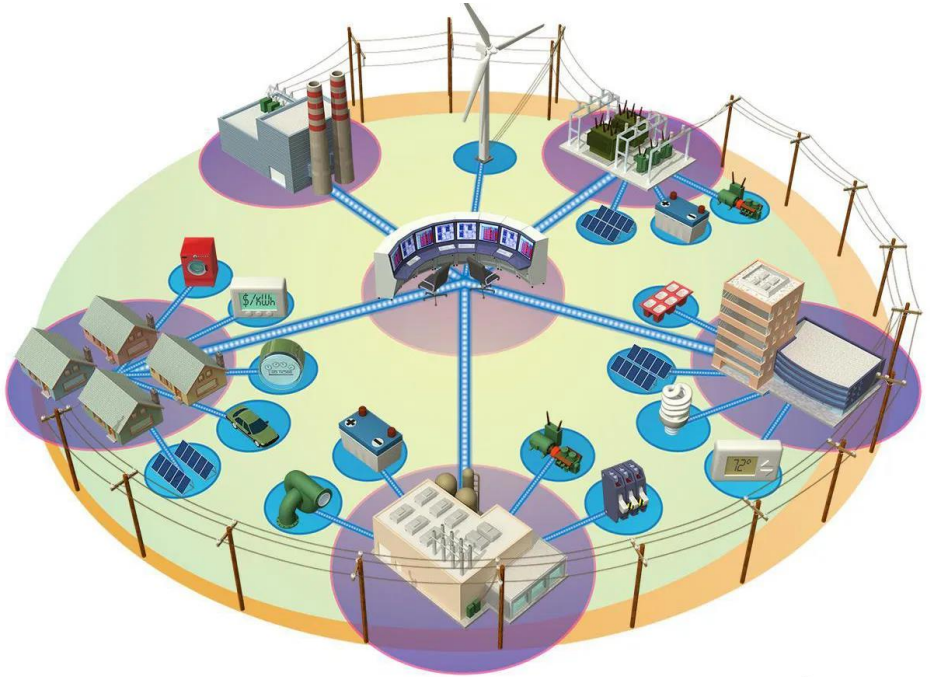
Сформулировал гипотезу на список недостатков-приемов, хочу получить обратную связь

О проекте и команде

Продукт: управление энергетическими потоками на основе данных

Проекты:

- управление отоплением с использованием прогноза погоды
- управление зарядкой электромобилей,
- управление энергетическими потоками в
- распределенных энергетических системах,
- технико-экономические обоснование для инвестиций



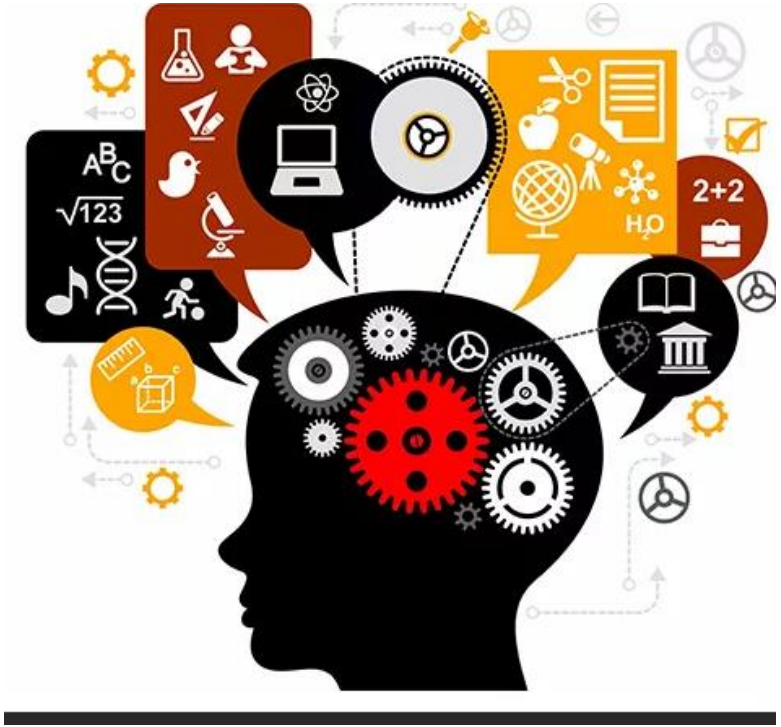
40+ разных разработчиков за 10 лет работы

Идея работы

Хотя в информационных технологиях разработан собственный технический аппарат к описанию и проектированию систем управления на основе данных, такие подходы ТРИЗ, как:

1) Функциональный анализ
2) Поиск обобщенных недостатков и приемов их устранения
3) Причинно-следственный анализ

Подходы ТРИЗ Позволяют усилить процесс проектирования систем управления на основе данных



Недостатки текущего процесса

1) Отсутствие глоссария
2) Разнообразный кругозор у разработчиков
3) Зоопарк инструментов проектирования

Что приводит

1) Барьеру максимальной сложности информационной системы
2) Большому техническому долгу экосистемы управления на основе данных
3) Внутренние споры относительно правильности использования инструментов
4) Недоступности к крупные тендеры

Требования к инструменту визуализации

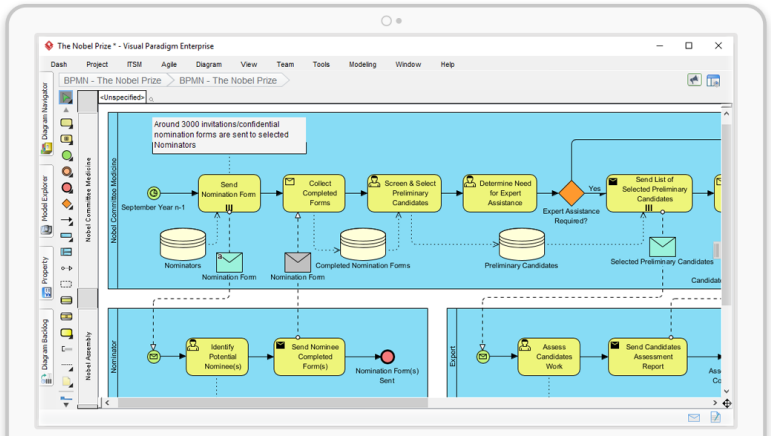
Перед началом работы с архитектурой приложения определены следующие требования:

i) программа должна моделировать систему на разных технических уровнях;

ii) повторно использовать описанные объекты,

iv) хранить все разработанные диаграммы в одном файле в репозитории онлайн;

v) поддерживать репозиторий и совместное хранение и редактирование архитектуры онлайн.



draw.io

FLYING LOGIC

Visual Paradigm

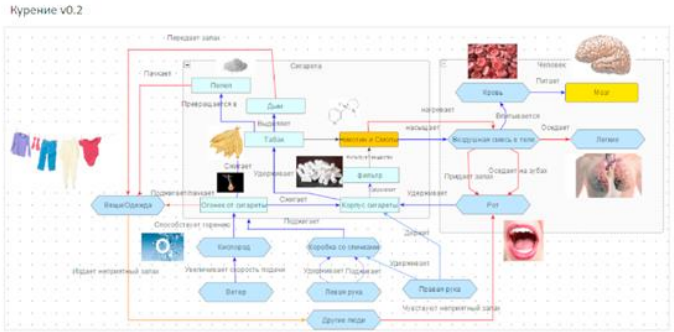
Прототип процесса

В качестве прототипа процесса проектирования архитектуры использовался процесс обучения Мастера ТРИЗ Юрия Даниловского по функциональному анализу (2016), состоящий из следующих этапов:

1) Проведение функционального анализа в нотации Майлза

2) Определение списка недостатков,

3) Проведение причинно-следственного анализа первопричин, проектирование списка недостающих компонент.



Функциональный анализ

Список недостатков

1. Сигарета создает опасное поле. Недостаток 1. Огонек сигареты может испортить одежду.

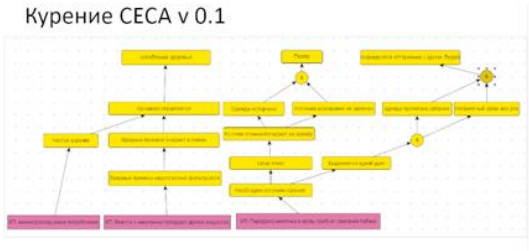
2. Сигарета создает опасное поле. Недостаток 2. Дым сигареты пропитывает одежду, она начинает издавать неприятных запах

3. Никотин добавляет вещество на зубы. Недостаток 3. Зубы меняют окраску.

4. Дым от сигареты создает опасное поле. Недостаток 4. У курильщика появляется неприятный запах изо рта, который чувствуют другие люди.

5. Дым от сигареты трансформирует вещество. Недостаток 5. Вредные вещества оседают в легких.

Формулизация недостатков



Проведение CECA

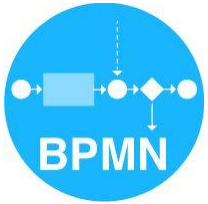
Описание решения

Польза от данной методики для команды:

- Структурирование процесса управления развитием управления на основе данных
- Снизить дублирование компонент
- Снизить транзакционные издержки при коммуникации кросс-дисциплинарных географически распределенных команд

Основные составляющие элементы методики:

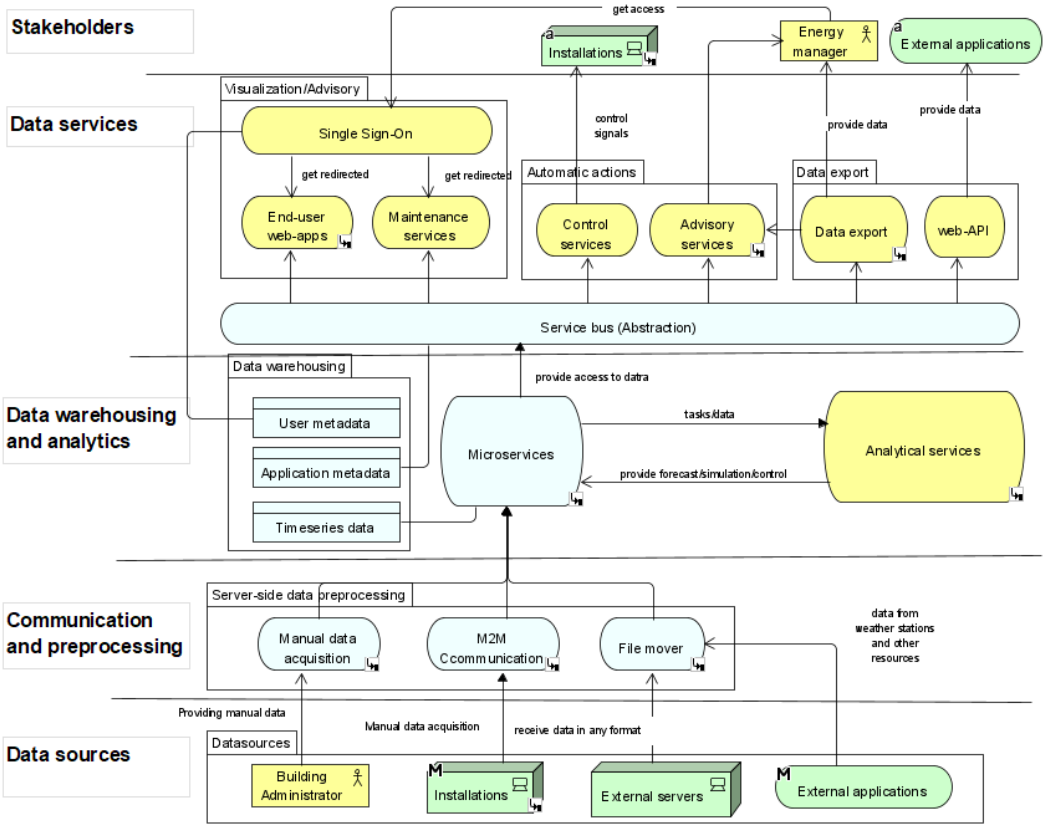
1) Использовании стандарта описания архитектуры организации TOGAF для определения аспектов внимания при проектировании архитектуры предприятия.
2) Визуализации отдельных аспектов системы на основе применения общепринятых стандартов визуального проектирования систем управления на основе данных:
    a) data flow diagram (DFD, аналог функционального анализа для информационных систем);
    b) BPMN для описания процессов, Archimate отображения архитектуры организации.
    c) Принципах функционального анализа технических систем в нотации Майлза перенесенных на архитектуры информационных систем и нотацию DFD.
    d) Процесс поиска недостатков на основе нотации ТРИЗ и приемов их устранения.

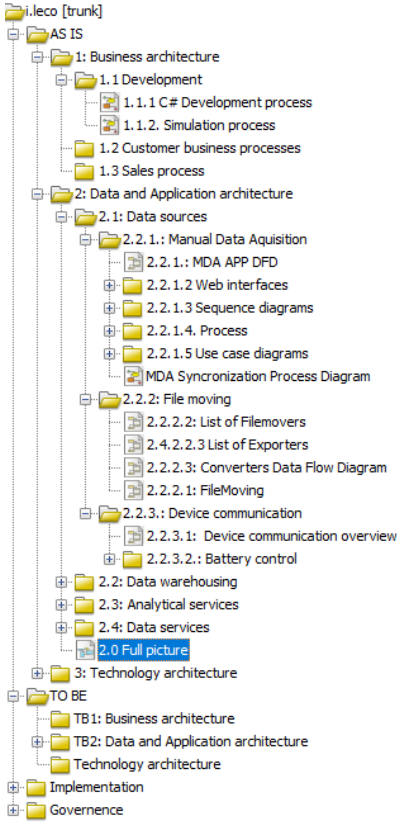


Задачи реализации решения

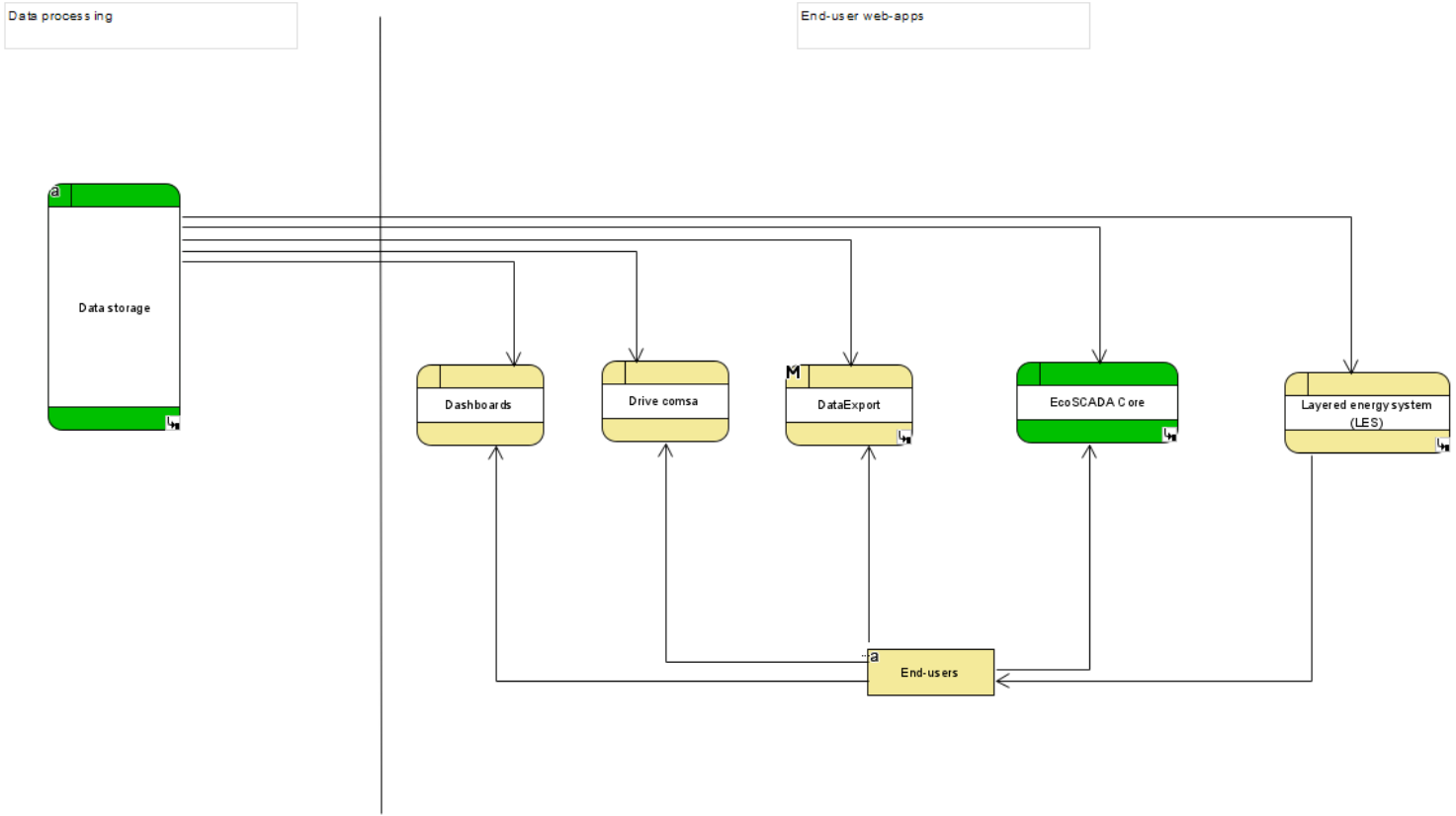
1) Формирование списка программного обеспечения, написанного и запущенного в экосистеме управления на основе данных, определение списка людей, ответственных за каждое разработанное программное обеспечение.
2) Формирование графика встреч с каждым из разработчиков для обсуждения элементов архитектуры системы. Для формирования полного представления требуется несколько итераций проработки используемого функционала с коллегами.
3) Создание синтезирующих диаграмм, позволяющих обобщить и сгруппировать созданные архитектурные описания.
4) Создание полностью единой, полностью синтетической диаграммы, описывающей концепцию системы управления на основе данных, которая позволяет: i) обзорно описать информационную систему управления на основе данных в понятном для нетехнологического специалиста формате ii) создать маршрутизацию доступа для диаграмм.
5) Проведение анализа всей экосистемы систем управления на основе данных для выделения недостатков, проведение причинно-следственного анализа для перехода к состоянию “Как должно быть”.
6) Составление группы диаграмм “Как должно быть” идеального состояния системы, на основании анализа текущего состояния.
7) Создание онлайн репозитория для коллективной работы с диаграммами.
8) Публикация результатов построения диаграмм в виде html файла в единую экосистему пользователя для вывода информации в экосистеме компании.
9) Обучение сотрудников работать с системой.

Результаты внедрения

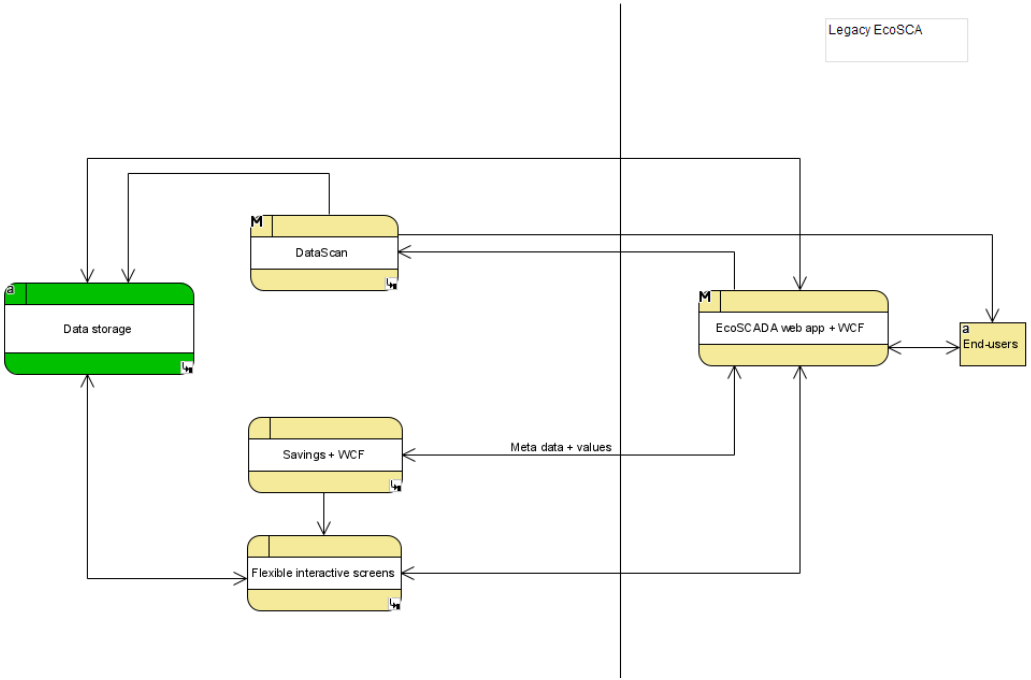




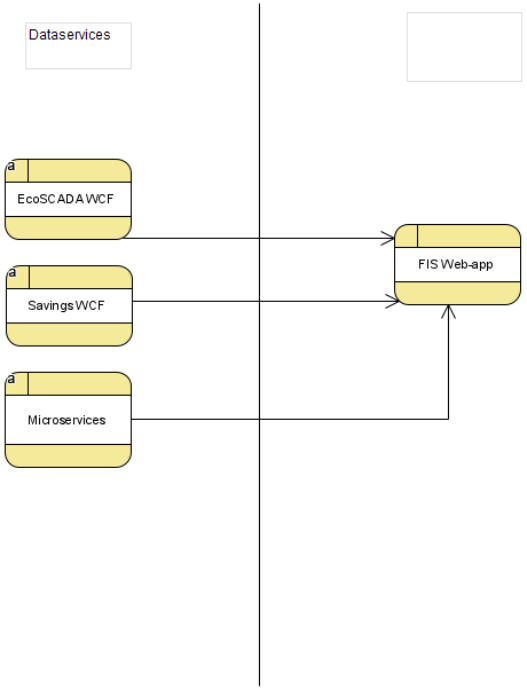
Приложения конечного потребителя (ур 1)



Декомпозиция EcoSCADA Core (Ур 2)

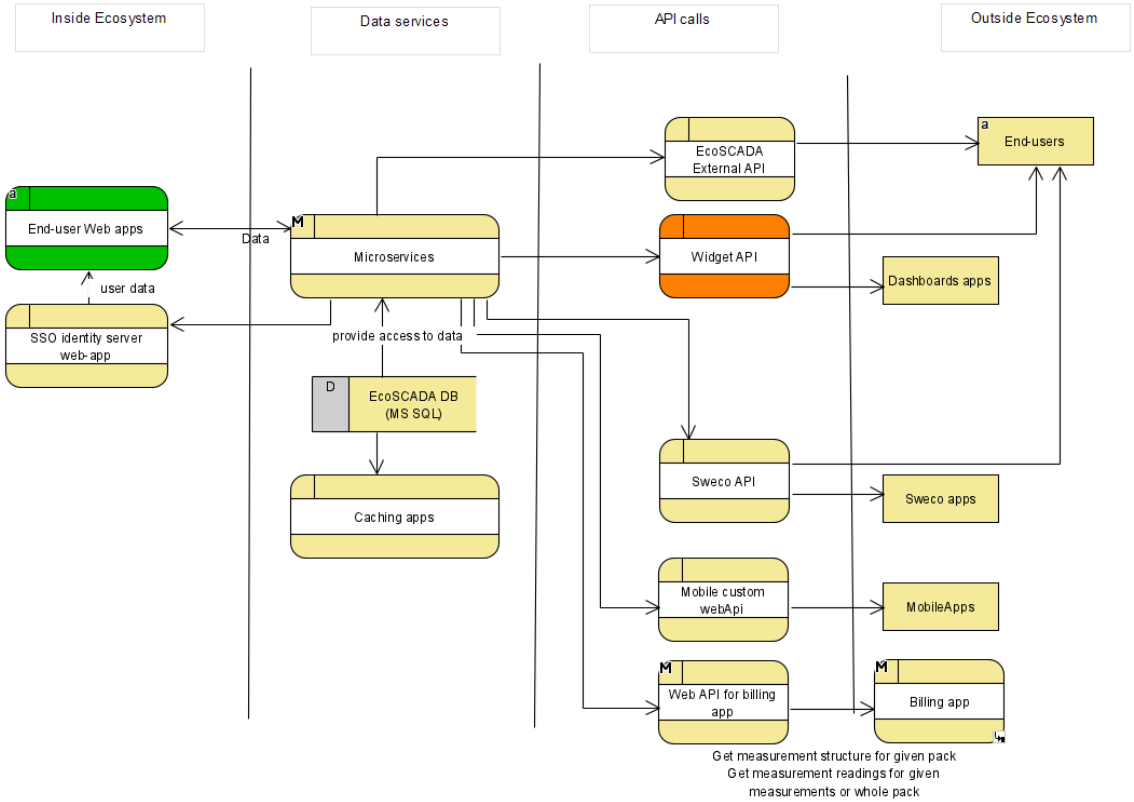


Веб-приложения EcoSCADA core 2 (2 уровень)

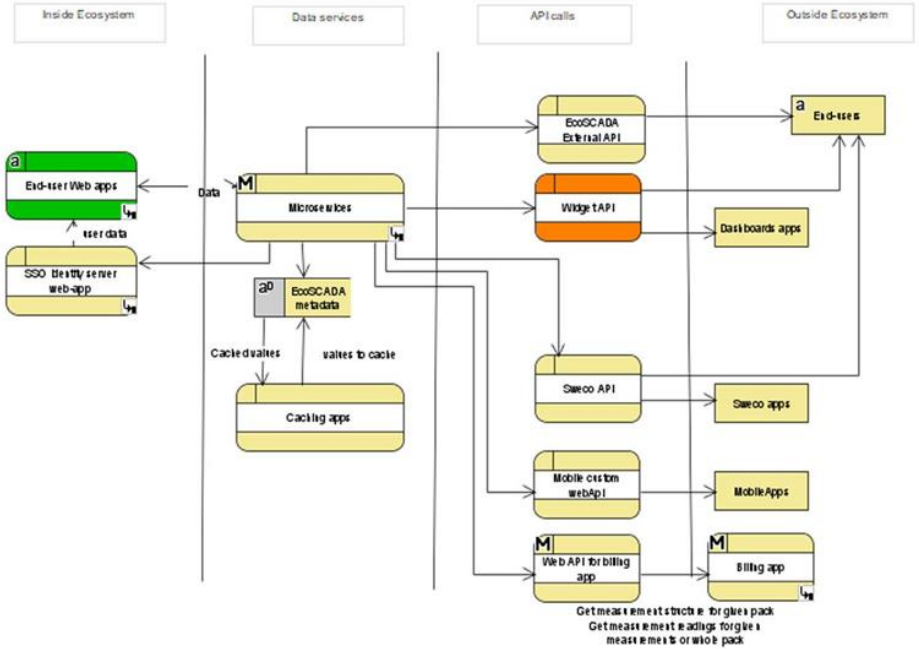


Веб-приложение Flexible Information Screens (3 уровень)

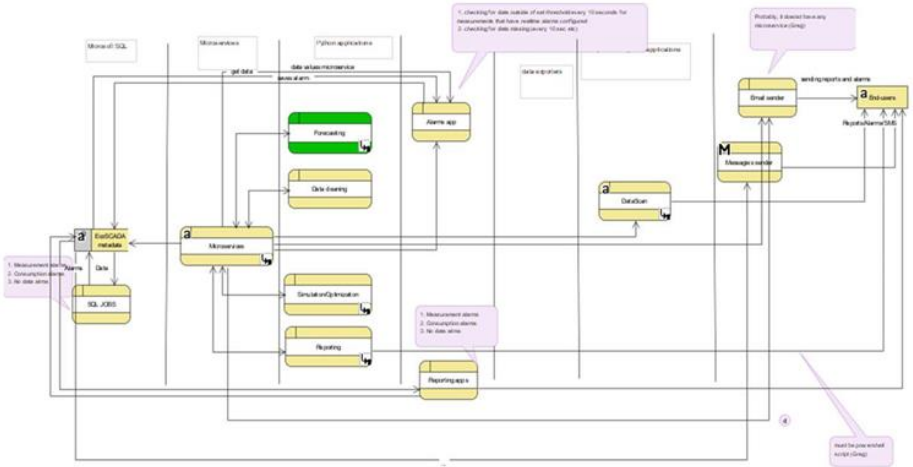
Архитектура доступа к данным



Логическое представление архитектуры данных в системе

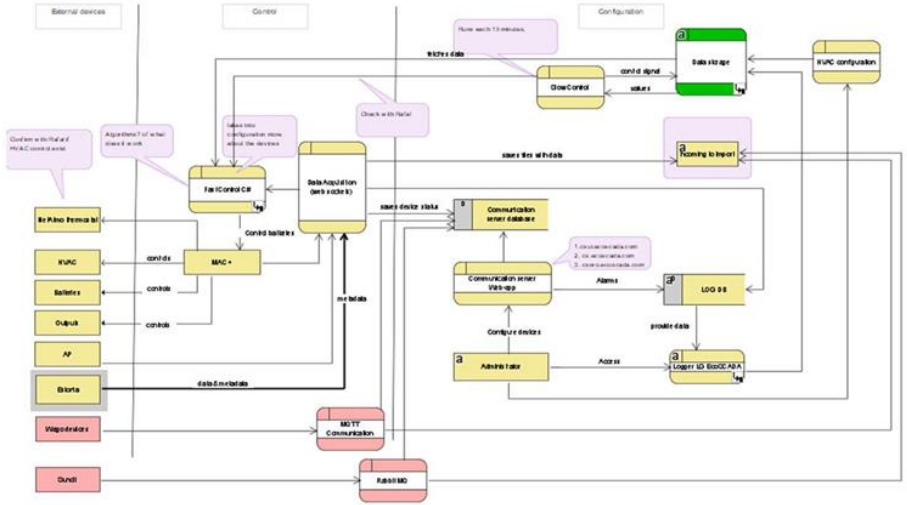


Хранение, кэширование и выгрузка данных

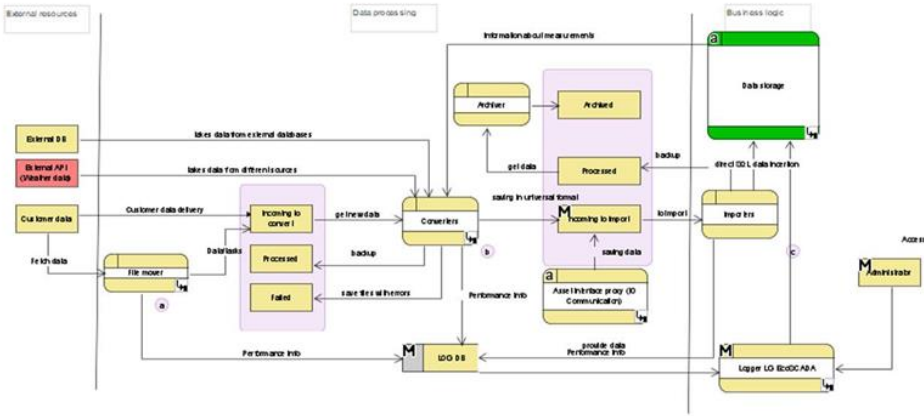


Управление алгоритмами управления на основе данных

Визуальное представление архитектуры информационного обмена между оборудованием, установленном в зданиях и сервером баз данных (2 уровень)

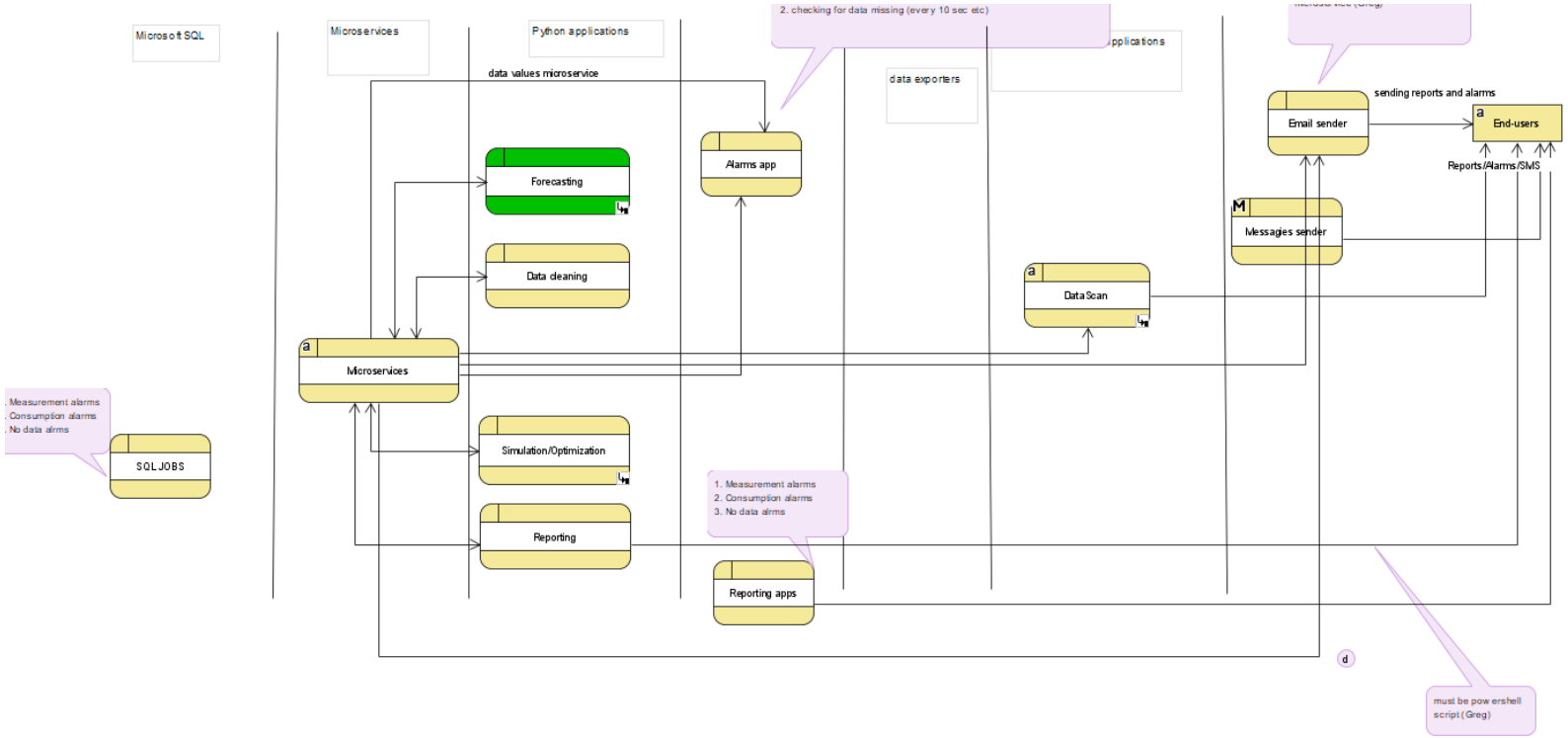


Процес выгрузки и загрузки данных

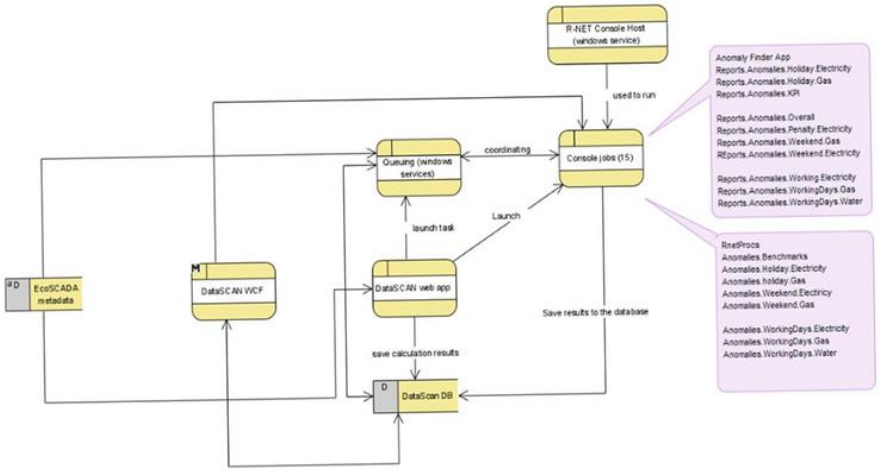


Процессы управления в управления распределеной системы управления на основе данных

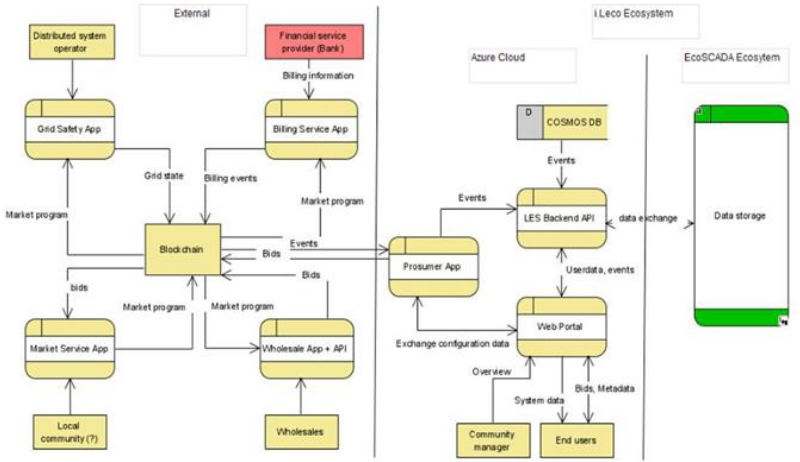
Сервисы Аналитики



Визуальное представление архитектуры информационного обмена между оборудованием, установленном в зданиях и сервером баз данных

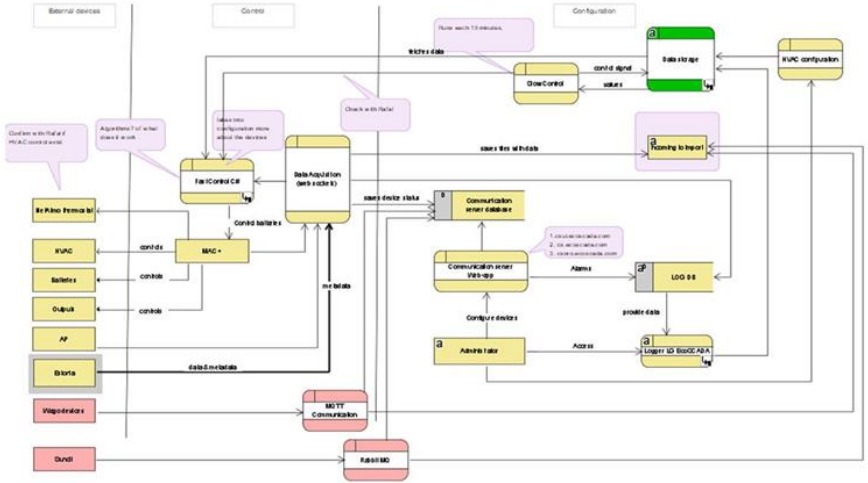


Автоматическая система создания отчечтов на основе анализа данных (3 уровень)

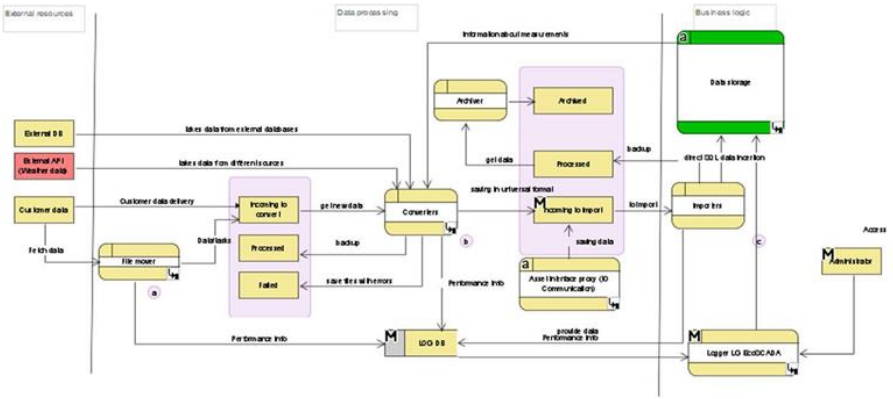


Архитектура управления распределенной энергетической системой (3 уровень)

Процесс выгрузки, загрузки данных и управление распределенным оборудованием

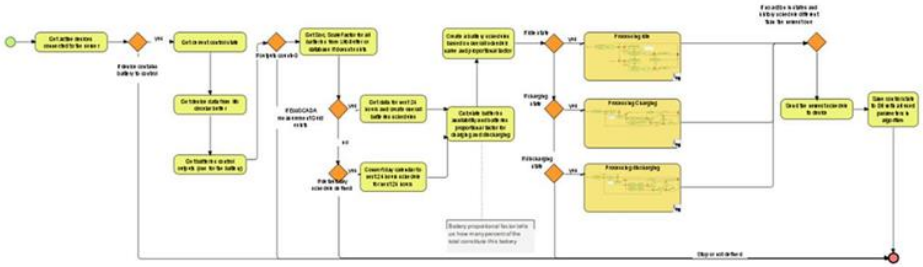


Процес выгрузки и загрузки данных

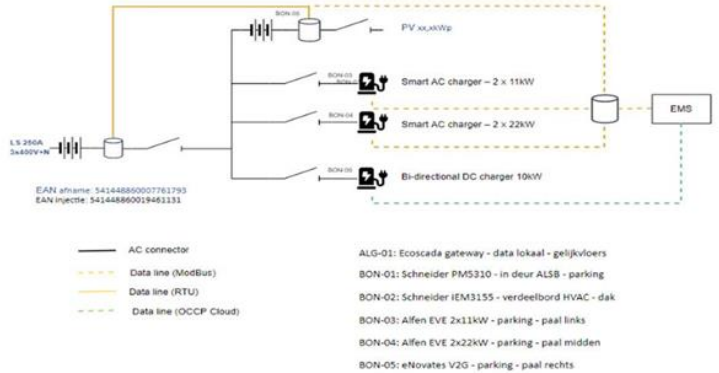


Процессы управления в управления распределеной системы управления на основе данных

Примеры описания алгоритмов управления и диаграммы подключения



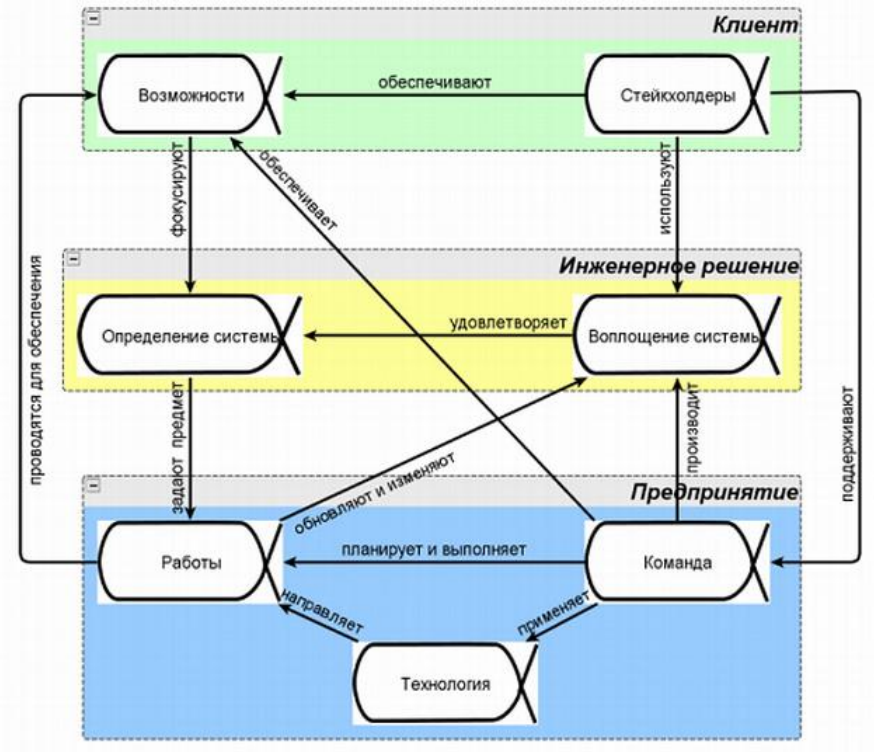
Описание алгоритмов управления батареей
(4 уровень)



Диаграма подключений при заправки электромобилей
(4 уровень)

Обобщенные недостатки и приемы в ИТ (обобщено 16)

В рамках исследований разработан глоссарий системы, который позволяет прийти к единому языку описания системы, построена и визуализирована архитектура. Наличие диаграмм снизило трудозатраты при объяснении сложных задач, распределении труда разработчиков, формировании структуры отделов компании. В рамках анализа была предложена гипотеза о списке недостатков в архитектурах управления на основе данных. В качестве классификации недостатков были выбраны 7 альф системной инженерии[2]: i) стейкхолдеры, ii) возможности, iii) определение системы, iv) воплощение системы, v) команда, vi) работа, vii) технология. Информация с описанием предложенных недостатков и приемов их устранения представлена в Таблице 1.



Гипотезы обобщенных недостатков и приемов их устранения

Наименование недостатка	Причина возникновения	Прием устранения
Низкая оперативность передачи данных (Воплощение системы)	Возникает из-за слишком большого количества компонент- посредников в системе, низкой пропускной способности информационного канала, особенности технологического оборудования	Убрать количество компонент-посредников, передать функции в надсистему, реализовать систему кэширования
Наличие посредников-компонент обработки данных (Воплощение системы)	Возникает, когда разработчики не синхронизируют свою работу	Согласовать деятельность разработчиков (общая архитектура системы)
Дублирование компонент (Воплощение системы)	Возникает, когда в экосистеме присутствует несколько компонент, делающих одно и тоже	Согласовать с разработчиками, перестать поддерживать одну из компонент
Низкая согласованность технологий (Технологии)	Возникает тогда, когда один и тот же функционал реализован различными компонентами	Создание документа Architecture Definition в компании
Дублирование функций в различных приложениях (Определение системы)	Несколько приложений повторяет полностью один и тот же функционал	Перераспределение функций между компонентами, проектирование системы таким образом, чтобы она входила в одну функциональную группу
Низкая стандартизация программного обеспечения на уровне технологий (Технологии)	Приложения одного и того же типа разрабатыватываются на разных языках. Возникают проблемы с обучением, передачи кода и т.д.	Создание документа Architecture Definition в компании
Несогласованность функций в экосистеме (Определение системы)	Несколько приложений частично содержат схожие функции. За полным функционалом пользователю необходимо использовать несколько приложений	Перераспределение функций между компонентами в соответствии с функциональными группами

Недостаточная прозрачность при принятии решения (Стейкхолдеры)	Приложение показывает какую-то важную информацию, но пользователю непонятно, что с этим делать	Перейти от поддержки принятия решений к управлению, добавить в систему модуль пояснений.
Избыточные трудозатраты при получении информации (Стейкхолдеры)	Для своевременного получения информации требуется слишком много усилий	Организовать доставку информации по наиболее доступным для пользователя каналам: в мобильный телефон, на почту.
Низкая вовлеченность пользователя в работу приложения (Стейкхолдеры)	Возникает тогда, когда пользователей не понимает собственных целей работы в системе	Разработать систему мотиваций для отслеживания изменений в системе. Формировать контур обратной связи с помощью предоставления статистик и игрофикации.
Долгое время подготовки данных перед анализом (Команда)	Для подготовки данных требуется слишком много времени	Стандартизировать формат получаемых данных, создать систему автоматической предобработки данных перед анализом
Отсутствие прозрачности при контроле за техническими процессами (Команда)	Неизвестно, какие компоненты системы работают, а какие нет.	Создание связки клиент-сервер-данные-качество данных для постоянного отслеживания фактического достижения договоренностей с пользователем.

Избыточная связность между приложениями (Стейкхолдеры)	Приложения при своей работе вызывают друг друга без объективной необходимости, это приводит к снижению устойчивости системы	Провести рефакторинг, уменьшить связность между компонентами в системе
Недостаточная осведомленность руководителя относительно текущей архитектуры (Стейкхолдеры)	Происходит тогда, когда слишком высокая скорость изменений, низкое качество описания системы, отсутствует система проектирования и планирования	Сформировать стандарты описания компонент системы
Недостаточный уровень качества работоспособности компонент (Воплощение системы)	Возникает запрос на поиск альтернативных решений при разработке программного обеспечения	Разработать инструмент оценки качества работоспособности компонент
Низкая скорость заказа на изменения	Возникает в случае медлительности менеджмента при проектировании изменений, приводит к дублированию компонент	Увеличивать осведомленность людей, поддерживать стандарты качества, быстро реагировать на изменения.

Результаты проекта

1) Предложен способ анализа и улучшения систем управления на основе данных с использованием ТРИЗ
2) Сформулировали обобщенные недостатки архитектур систем управления на основе данных и предложили приемы для их устранения.
3) Авторами предполагается, что дальнейшая работа по структурированию недостатков и приемов позволит развить ТРИЗ для проектирования архитектур управления на основе данных, что в потенциале позволит увеличить аналитическую мощность архитекторов.

Развитие исследования

1) Переход от обобщенных недостатков к обобщенным противоречиям
2) Расширение описания системы на бизнес-процессы пользователя, объединение описаний бизнес-процессов пользователя с функциональным приложения, возможно использования статистик пользователей,
3) разработка онтологии развития архитектуры управления на основе данных, в рамках которого он получает ценность при использовании сервиса компании,
4) Дальнейшее структурирование структурирование недостатков и приемов.

Спасибо за Внимание!

anton.tyukov@gmail.com