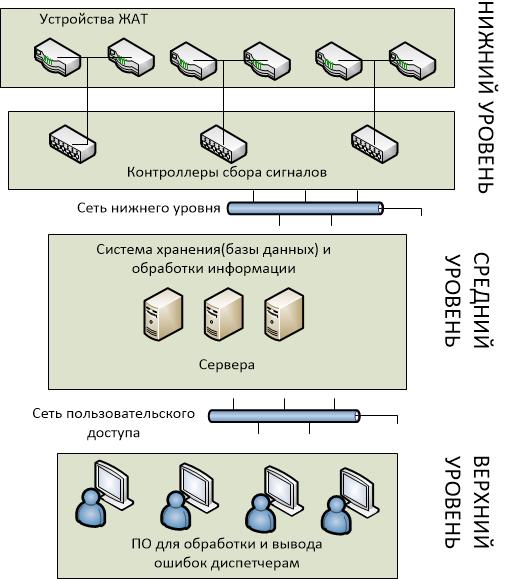
## **Диспетчерский контроль на железных дорогах России**

Для мониторинга состояния устройств железнодорожной автоматики и телемеханики (ЖАТ) на железных дорогах России используются системы аппаратно-программных комплексов диспетчерского контроля. Они состоят из трех подсистем:

1. Первая подсистема (подсистема нижнего уровня) - первичная обработка информации, снимаемой с устройств ЖАТ.

2. Вторая подсистема (подсистема среднего уровня) - сбор информации от подсистемы нижнего уровня, обработка, хранение, архивация и передача информации на верхний уровень.

3. Третья подсистема (подсистема верхнего уровня) - обработка и вывод информации о неисправности устройств ЖАТ.



## **Проблема диспетчерского контроля**

Чтобы отследить неисправность устройства ЖАТ, диспетчер должен привязать к соответствующему объекту в базе данных необходимую индикацию в диспетчерском ПО.

**Схема привязок**



На этапе проектирования станции диспетчера привязывают необходимые события к нужным индикациям в ПО, после этого как только произойдет перегорания красной лампы, диспетчер увидит на экране мигание красным цветом сигнализатора №1. И нажав на него, увидит, что на станции 150 перегорела лампа 3.

**Основная проблема**

Так как количество объектов, состояний и отказов очень велико, одной из главных проблем является человеческий фактор. Человек может не привязать нужное событие к индикации, привязать не к тому объекту или же не к той индикации. Таким образом, возвращаясь к нашему примеру, диспетчер не увидит, что на станции перегорела красная лампа светофора.

## **Автоматизация поиска ошибок**

Для решения проблемы была разработана программа, которая сканирует базу объектов, базу сигнализаторов и базу привязок и выявляет возможные ошибки.

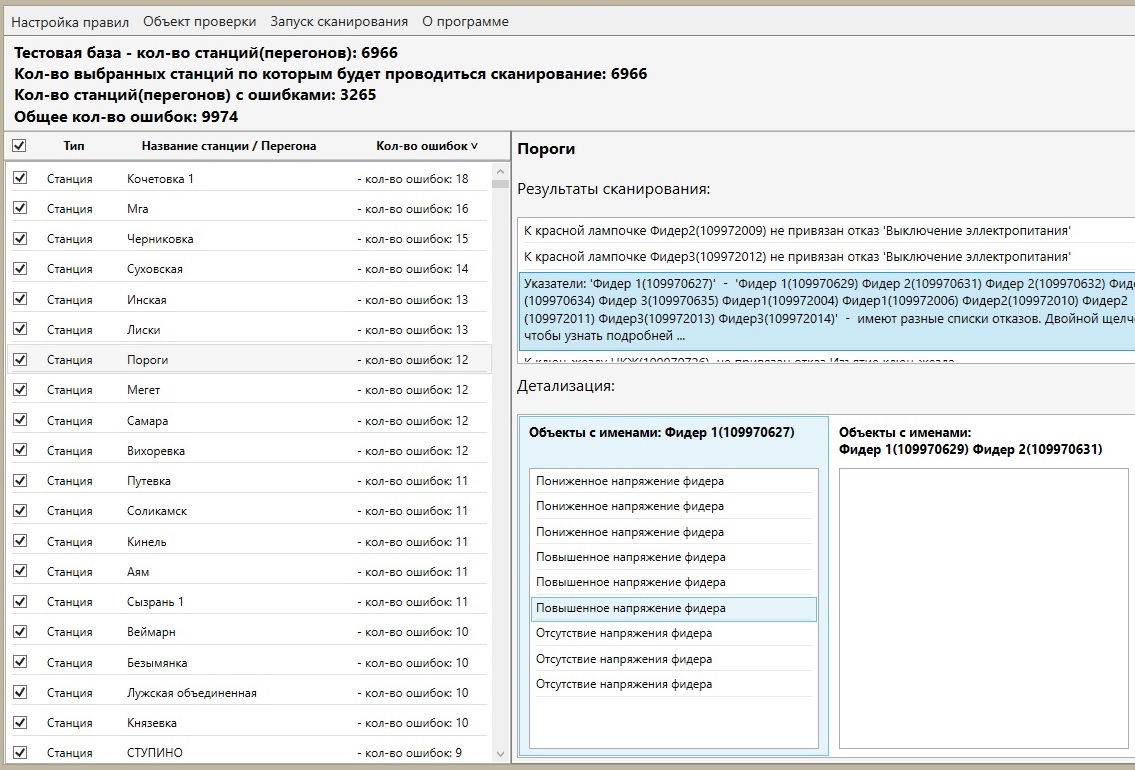
**Входные данные**: На вход в программу подаются базы данных и правила поиска ошибок, описанные в виде кода на языке С#, позволяющие выявить ошибки.

**Выходные данные:** Результатом работы программы являются таблицы с информацией о кол-ве ошибок на каждой станции и детализацией каждой ошибки.

**Схема работы программы**

****

**Результаты выводятся следующим образом:**



**Правила поиска ошибок**

Правила поиска объединены в библиотеку правил. Каждое правило имеет текстовое описание и алгоритм работы. Все правила реализуют общие интерфейсы и способы без дополнительного кода встраиваться в библиотеку и исполняемый код. На данный момент в библиотеке доступны пять правил.

**Пример правила:**

*// Сканирование светофоров  
// 1. Все объекты с типом 2 (светофоры) и именем '1' или '2' (предвходные) должны иметь одинаковый список отказов.  
// 2. Все остальные светофоры должны иметь одинаковый список отказов.*

foreach (DataRow traffic in dt\_objects.Select("ids =" + stance.id + " and type = 2"))  
{  
 if ((string)traffic["name"] == "2" || (string)traffic["name"] == "1")   
{  
enters.Add(get\_situation(traffic));   
}else  
{  
lights.Add(get\_situation(traffic));  
}}

**Выводы**

Применение приложения автоматизации поиска ошибок на реальных дорогах дало следующие результаты:

**Результаты для 5 правил:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Кол-во станций | 6996 | 4010 | 2110 | 800 | 400 |
| Кол-во устройств | 9094800 | 5213000 | 2743000 | 1040000 | 520000 |
| Кол-во привязок | 63663600 | 36491000 | 19201000 | 7280000 | 3640000 |
| Кол-во найденных ошибок | 3265 | 1265 | 982 | 300 | 112 |
| Процент правильно найденных ошибок | 15 | 20 | 20 | 40 | 45 |

Анализ результатов выявил следующие проблемы:

1. Процент правильно найденных ошибок увеличивается с уменьшением кол-ва сканируемых станций. Проблема связана с тем, что к разным группам станций нужно писать разные правила и чем меньше групп мы сканируем, тем больше процент корректности результатов.
2. Кол-во найденных ошибок слишком мало и на практике мы смогли покрыть только 5-10% от общего числа содержащихся ошибок.

**Применение логических алгоритмов**

Анализ проблем говорит о том, что вручную написанные правила поиска ошибок не дают необходимой эффективности.

Существует два варианта решения этих проблем:

1. Постоянное пополнение, изменение и расширение библиотеки правил, что является трудоемкой ручной работой и в конечном счете также может содержать ошибки.
2. Применение логических алгоритмов для автоматического построения библиотеки правил.

//ДОПИСАТЬ ПРО ТО КАК ЛОГИЧЕСКИЕ АЛГОРИТМЫ УПРОСТЯТ ЖИЗНЬ И ПРО ТО КАК ИХ МОЖНО ИСПОЛЬЗОВАТЬ