Реализация логических алгоритмов для решения проблем диспетчерского контроля на железных дорогах

**Автор:** Кавин Борис Владимирович

Оглавление

[Глава 1. Проблемы диспетчерского контроля участков железной дороги 3](#_Toc430555432)

[Диспетчерский контроль на железных дорогах России 3](#_Toc430555433)

[Система диспетчерского контроля АПК-ДК 4](#_Toc430555434)

[Назначение системы 4](#_Toc430555435)

[Структура системы 5](#_Toc430555436)

[Проблемы диспетчерского контроля 7](#_Toc430555437)

[Логические алгоритмы для решения проблем 8](#_Toc430555438)

[Глава 2. Реализация логических алгоритмов 9](#_Toc430555439)

[Описание программы 9](#_Toc430555440)

[Схема базы данных 10](#_Toc430555441)

[Примеры методов 10](#_Toc430555442)

[Статически заданный метод 10](#_Toc430555443)

[Интеллектуальный метод 11](#_Toc430555444)

[Интерфейс программы 11](#_Toc430555445)

[Приложение 1. Реализация статически заданных методов. 15](#_Toc430555446)

# Глава 1. Проблемы диспетчерского контроля участков железной дороги

## **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:

Первая подсистема (подсистема нижнего уровня) состоит из специализированных контроллеров, обеспечивающих съём и первичную обработку информации, снимаемой с устройств ЖАТ.

Вторая подсистема (подсистема среднего уровня) состоит из концентраторов линейного поста (ЛП), собирающих информацию от подсистемы нижнего уровня и обеспечивающих обработку, хранение, архивацию и её передачу другим концентраторам, и концентраторов центрального поста (ЦП), которые кроме того обеспечивают передачу собранных данных на верхний уровень.

Третья подсистема (подсистема верхнего уровня) состоит из технических средств диспетчера дистанции сигнализации и связи и работников отделения дороги.

Конечные пользователи системы - диспетчеры, технологи, механики - могут видеть результаты мониторинга участков на своих автоматизированных рабочих местах (АРМах), машинах со специальным ПО верхнего уровня системы.

Наблюдая за системой на экране монитора, пользователь может выявить следующие проблемы на своем участке:

* Отказы / предотказы (например, перегорание красной лампы светофора).
* Технологические ситуации (например, переводы стрелок).
* И др.

## Система диспетчерского контроля АПК-ДК

Система диспетчерского контроля АПК-ДК разработана по заданию ЦШ МПС РФ кафедрой «Автоматика и телемеханика на ж. д.» Петербургского Государственного Университета Путей Сообщения (ПГУПС) в рамках отраслевой программы автоматизации хозяйства сигнализации и связи. Система рекомендована к применению на сети дорог Российской Федерации решением сетевой школы «Аппаратно-программный комплекс диспетчерского контроля (АПК-ДК)», утвержденным заместителем министра Путей Сообщения А.С. Мишариным 11 декабря 1999 года [4]. В 2001 году ПГУПС были выпущены методические указания по проектированию устройств автоматики, телемеханики и связи на железнодорожном транспорте И-352-01 «Система диспетчерского контроля АПК-ДК» [5].

### Назначение системы

Аппаратно-программный комплекс диспетчерского контроля АПК-ДК предназначен для централизованного контроля, диагностики и регистрации состояния устройств железнодорожной автоматики и телемеханики, диагностики их технического состояния, а также организации управления движением поездов в пределах диспетчерского круга. АПК-ДК позволяет осуществлять сбор, обработку, хранение и отображение информации о состоянии объектов контроля в реальном масштабе времени [5].

Комплекс образует вычислительную сеть для обеспечения оперативной информацией диспетчерского аппарата отделения дороги, управления дороги и линейных предприятий (например, дистанций сигнализации и связи).

Система АПК-ДК выполняет контроль и диагностику технического состояния СЖАТ на перегонах и станциях, в том числе позволяет собирать статистику, выявлять предотказные состояния, анализировать причины некачественной работы и автоматизировать поиск отказов устройств СЦБ, т.е. обеспечивает возможность перехода на ремонтно-восстановительную технологию обслуживания СЖАТ за счет диагностики и прогнозирования состояния устройств и учета ресурса приборов по их фактической наработке. Данная информация передаётся дежурному электромеханику, диспетчеру дистанции сигнализации и связи, техническому персоналу, ответственному за сбор и обработку статистики отказов, а также, при необходимости, другим пользователям локальной вычислительной сети дистанции, отделения или управления дороги.

Также АПК-ДК обеспечивает поездного диспетчера информацией о поездном положении в пределах круга диспетчерского управления: свободности/занятости блок-участков перегонов, главных и боковых приемоотправочных путей промежуточных станций, показаний входных и выходных светофоров, установленном направлении движения, состоянии переездов, температуре буксовых узлов и др.

### Структура системы

АПК-ДК включает в себя три подсистемы [6], реализуемые с использованием программируемых контроллеров, промышленных компьютеров и специального программного обеспечения (ПО), а также каналов связи между ними, позволяющих организовать вычислительную сеть и автоматизированные рабочие места (АРМ) пользователей (рисунок 2.1).

Первая подсистема (подсистема нижнего уровня) состоит из специализированных контроллеров, обеспечивающих съём и первичную обработку информации, снимаемой с устройств ЖАТ.

Вторая подсистема (подсистема среднего уровня) состоит из концентраторов линейного поста (ЛП), собирающих информацию от подсистемы нижнего уровня и обеспечивающих обработку, хранение, архивацию и её передачу другим концентраторам, и концентраторов центрального поста (ЦП), которые кроме того обеспечивают передачу собранных данных на верхний уровень. Для выполнения указанных задач концентраторы объединяются в сеть передачи данных (СПД).

Рис.1. Структура системы АПК-ДК

Третья подсистема (подсистема верхнего уровня) состоит из технических средств (АРМов) диспетчера дистанции сигнализации и связи и работников отделения дороги.

Структура системы АПК-ДК разрабатывается для каждого конкретного участка железной дороги с различным наполнением упомянутых подсистем источниками информации, устройствами сбора и передачи данных, концентраторами среднего уровня, с учетом количества и функционального назначения рабочих мест на верхнем уровне системы.

Информационное и программное обеспечение среднего уровня позволяет организовать сбор, обработку и передачу информации от низовых контроллеров, а также от других систем ЖАТ (микропроцессорных АБ, ЭЦ, ДЦ, автоведения поезда, контроля состояния подвижного состава и т.д.) на верхний уровень системы.

Информационное и программное обеспечение верхнего уровня позволяет реализовать выполнение специальных технологических функций и организовать различные виды АРМ: диспетчера дистанции сигнализации и связи (АРМ-ШЧД), поездного диспетчера (АРМ-ДНЦ) диспетчера железнодорожного узла (АРМ-ДНЦУ), вагонного оператора и т.д., а также обеспечивает обмен информацией с другими информационными системами (АСУ-Ш, АСОУП).

## Проблемы диспетчерского контроля

Подключение к станции такой системы означает следующее:

1. Установка и настройка специальных контроллеров.
2. Установка и настройка концентраторов, которые собирают информацию с контроллеров и передают эти данные на пользовательские АРМ.
3. Проектирование и привязка (в специальных редакторах) всевозможных отказов к устройствам ЖАТ для получения информации о них.

На последнем пункте остановимся подробней.

Если диспетчер или технолог хочет получить информацию о том, что на светофоре с номером 1 перегорела красная лампочка, то на уровне проектирования и привязки необходимо добавить нужный объект (в данном случае светофор с номером 1), привязать к этому объекту необходимый отказ (для нашего примера «перегорание красной лампы») и выбрать для отказа соответствующее при такой ситуации состояние объекта (например, мигание красной лампы светофора).

В таком случае, когда на светофоре 1 перегорит красная лампочка, в тот же момент специальный контроллер передаст информацию об этом на АРМ диспетчера, который увидит, что объект «Светофор 1» мигает красным и для него завелась ситуация «перегорание красной лампы».

Таким образом, проектирование и привязка отказов сводится к тому, чтобы добавить в систему мониторинга все необходимые объекты, привязать к ним соответствующие отказы и для каждого из отказов выбрать нужное состояние индикации.

Процесс уникален для каждой станции, поэтому автоматизировать его невозможно. Привязкой отказов занимается человек.

Так как количество объектов, состояний и отказов очень велико, одной из главных проблем является человеческий фактор. Человек может не привязать нужный отказ к объекту, 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, если к объекту «Светофор 1» не привязан отказ «перегорание красной лампы» или привязан не к той индикации.

## Логические алгоритмы для решения проблем

Для решения данной проблемы было предложено воспользоваться средствами кластеризации и классификации.

Необходимо автоматически просканировать станцию и выявить группы объектов, разбитых по типу, по набору отказов, по состоянию индикации и другим параметрам.

Проанализировав группу кластеров, можно будет выявить ошибочные объекты, опираясь на логику привязок большинства однотипных объектов в рамках станции.

Таким образом, воспользовавшись необходимыми алгоритмами, можно будет просканировать станцию и вывести сообщения о возможных ошибках в привязке отказов, что позволит избежать проблемы в обнаружении неисправностей на станции.

# Глава 2. Реализация логических алгоритмов

## Описание программы

Программный продукт реализует логические алгоритмы k-means (метод k-средних), которые используются для кластерного анализа большого объема данных. Программа предназначена для интеллектуального поиска и выявления ошибочных или пропущенных полей в базе данных отказов аппаратуры АПК-ДК (Аппаратно-программный комплекс диспетчерского контроля системы железнодорожной автоматики и телемеханики). Платформа .NET делает возможным использовать программу в виде настольного приложения диспетчера или оператора.

Алгоритм поиска ошибок анализирует одинаковые группы объектов в базе данных и выявляет возможные ошибки. Определение ошибки бывает интеллектуальное (использует метод k-means) и статически заданное.

Интеллектуальный метод сканирует огромное количество полей базы данных (примерно 1.5 гб текстовой информации) и выявляет общие закономерности для однотипных объектов. Закономерности типа: максимальные и минимальные пороги числовых значений, наличие и типы связей с другими объектами, присутствие или отсутствие обязательных параметров у объекта, ВОЗМОЖНО ПРОДОЛЖИТЬ. На основании выявленных закономерностей алгоритм проходит все объекты еще раз и строит таблицу возможных ошибок. В таблице содержится информация о возможной ошибке и процент вероятности, который рассчитывается как отношения объектов с выходящими за рамки закономерностей значений к «нормальным» объектам (с входящими в рамки закономерностей значениями).

Статически заданный метод опирается на установленные вручную для каждой группы объектов математические правила. Правила бывают следующих видов: высчитываемые для каждого объекта пороги числовых значений, соответствие текстовых значений и наименований, наличие или отсутствие обязательных связей с другими объектами, присутствие или отсутствие обязательных параметров, возможно продолжить.

## Модель данных

## Примеры методов

Примеры использования статического и интеллектуального метод.

### Статически заданный метод

Статически заданный метод для выявления потенциальных ошибок в базе данных использует математически заданные правила. Примером может быть правило сформулированное следующим образом:

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

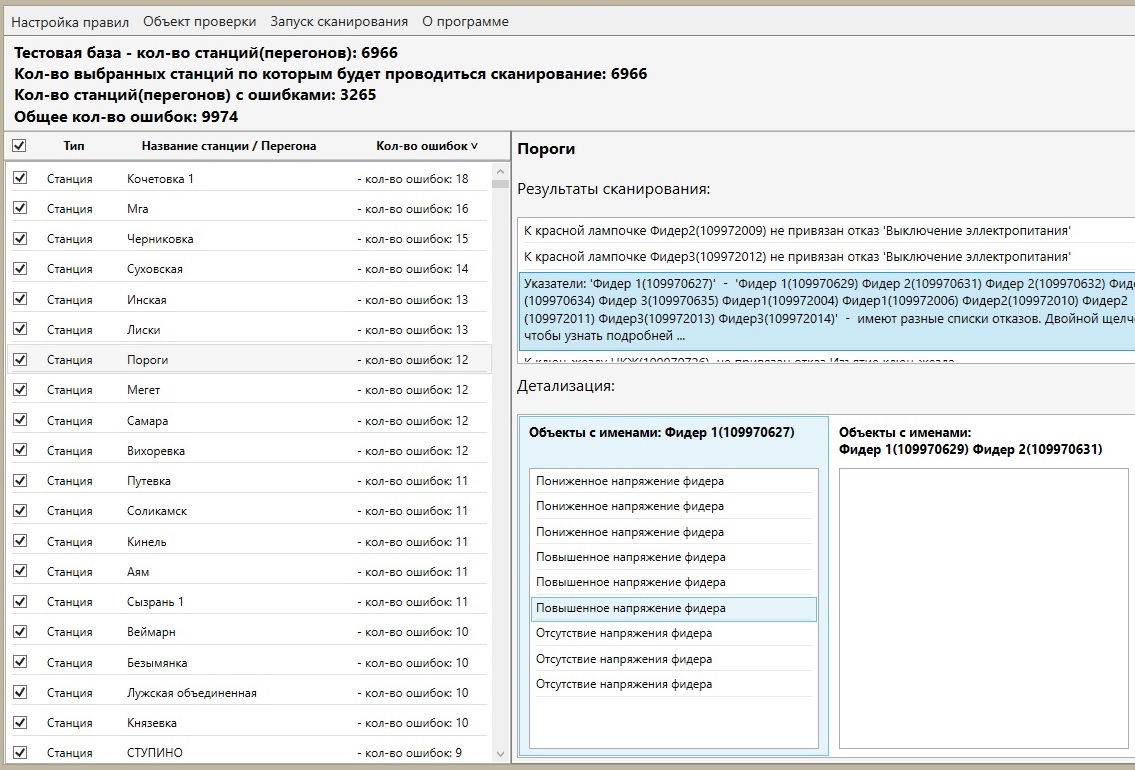
*2. Все остальные светофоры должны иметь одинаковый список отказов.*

Правило имеет следующую математическую форму:

ЗАПИСАТЬ В МАТЕМАТИЧЕСКОЙ ФОРМЕ.

Реализация данного метода в ***Приложение 1.***

Результаты выводятся следующим образом:

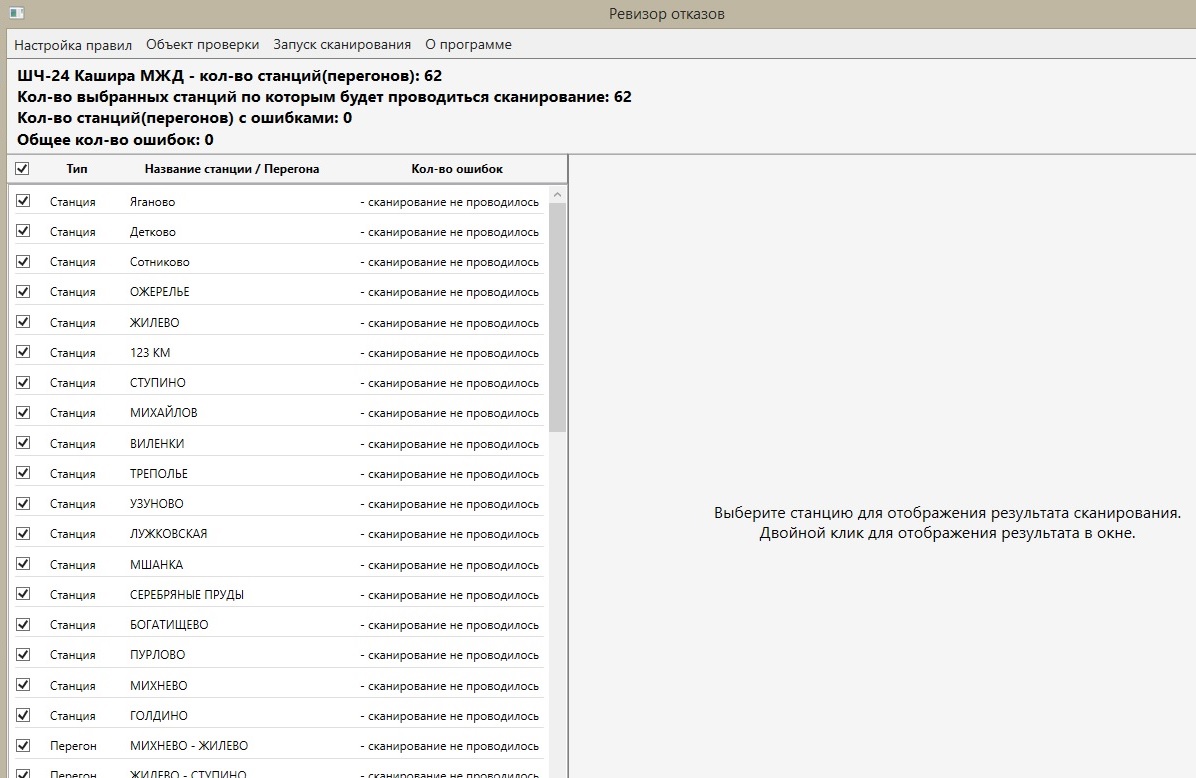


### Интеллектуальный метод

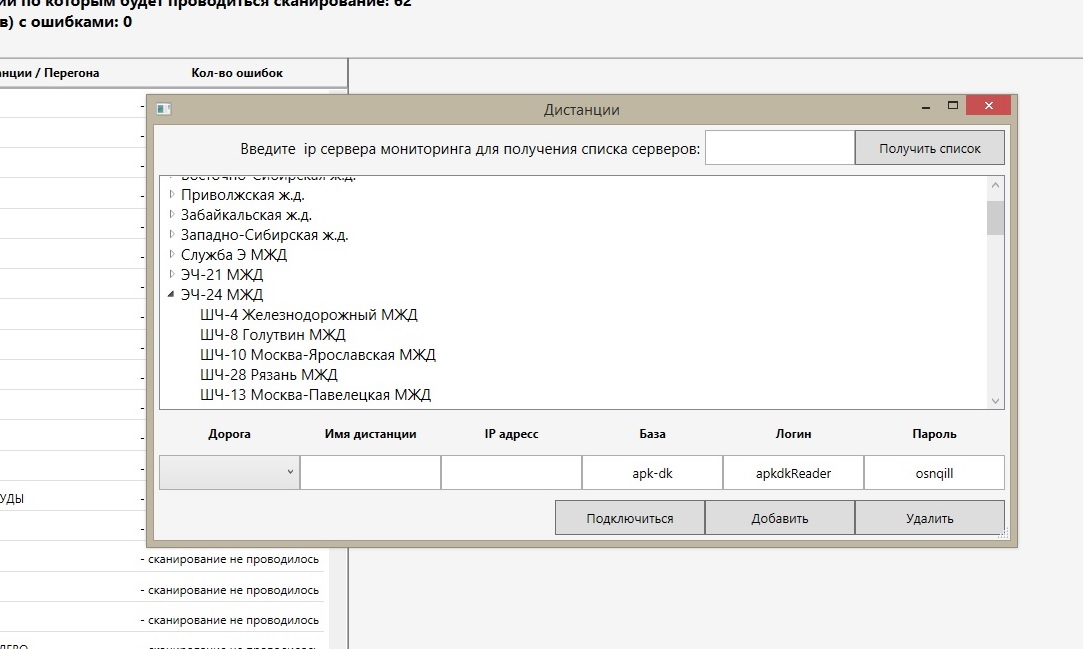
ПРИМЕР ПОКА НЕ ГОТОВ

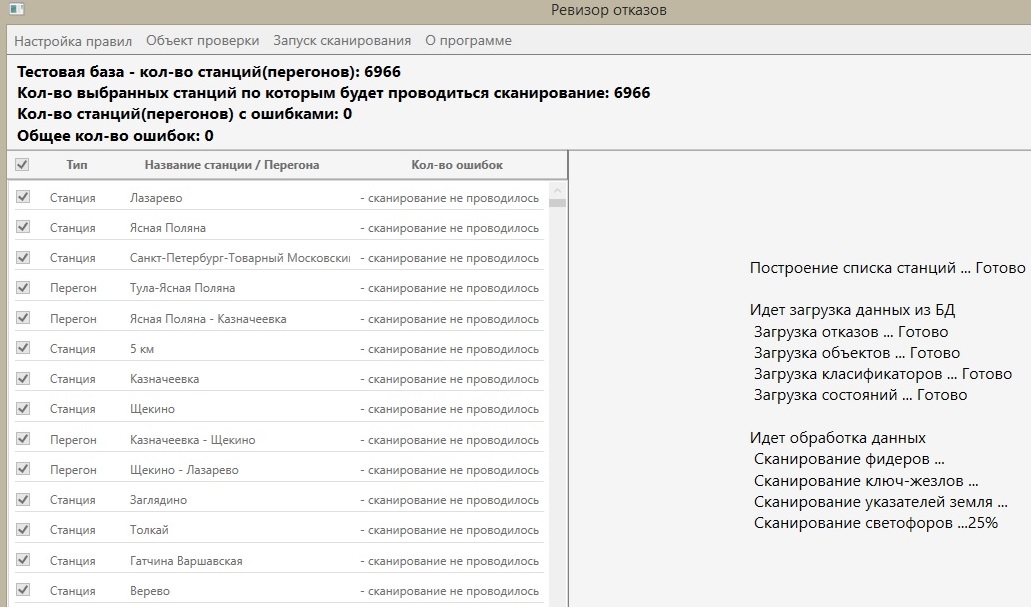
## Интерфейс программы

Окно выбора станции для сканирования. Сканирование на наличие ошибок проходит в рамках станции, перед запуском необходимо выбрать одну, несколько или все станции.

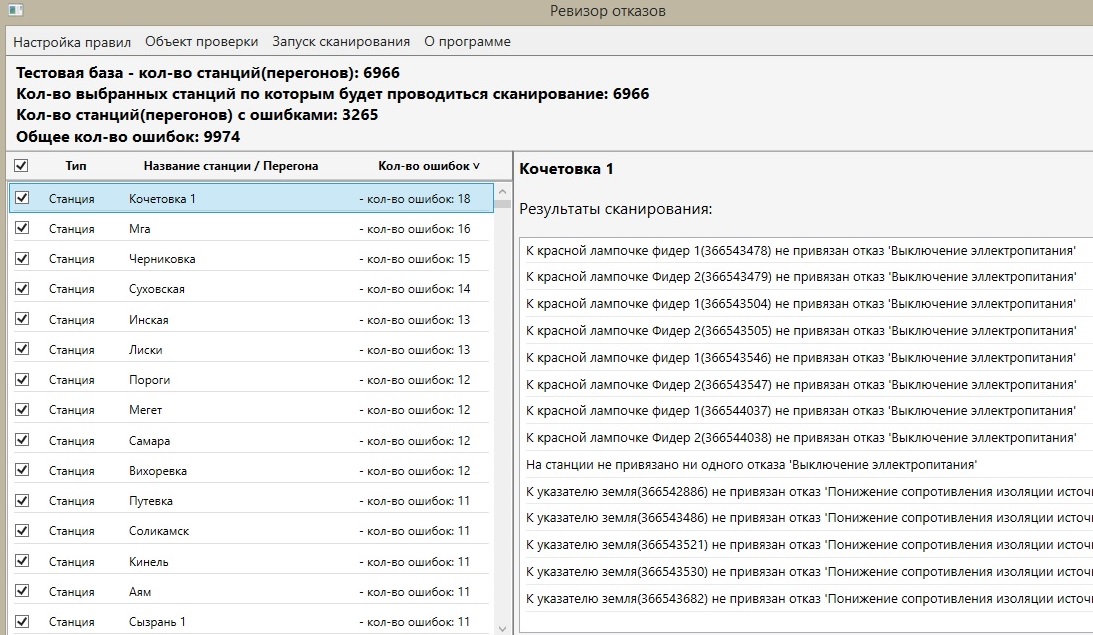


Окно выбора дистанции. Каждая дистанция содержит от 50 до 300 станций.

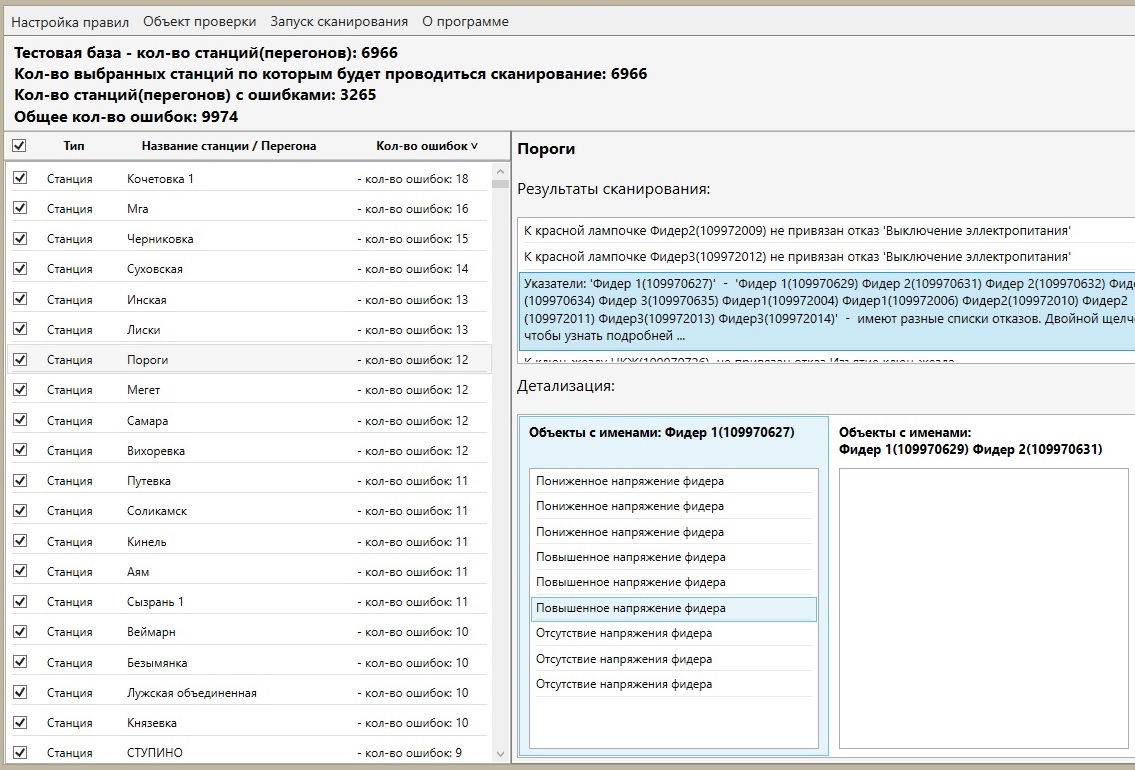
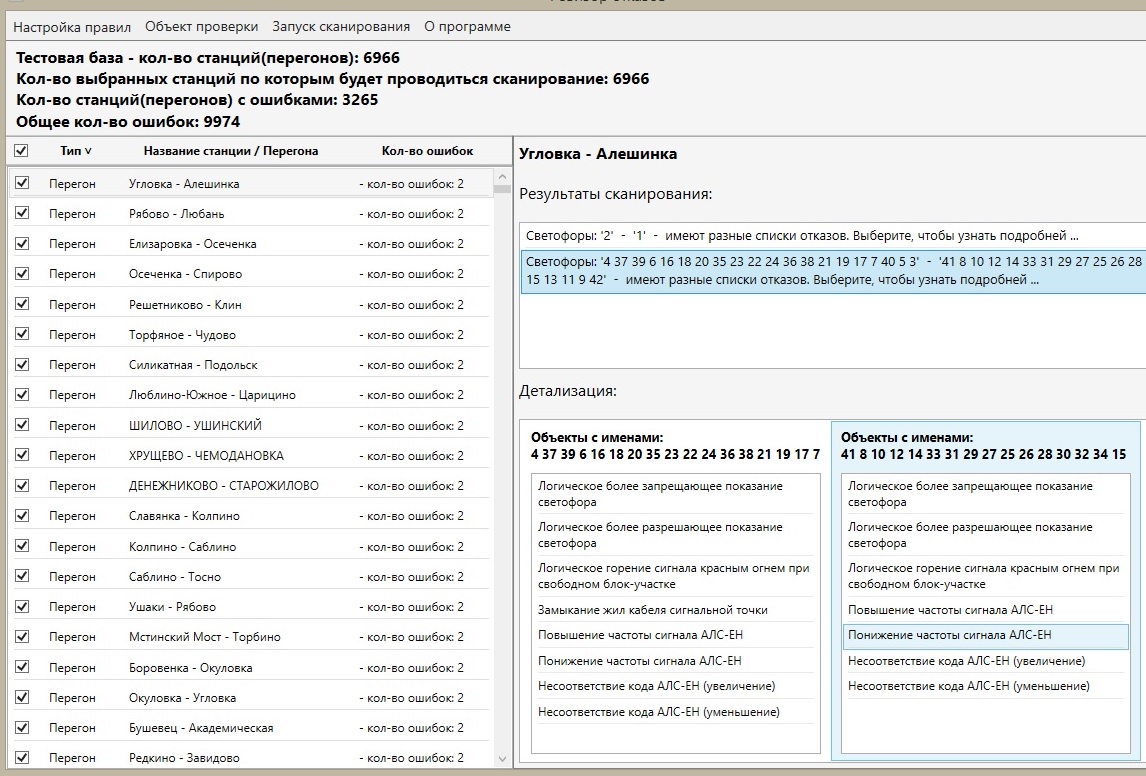


Отображение процесса сканирования. При выборе большого кол-ва станций алгоритм может обрабатывать около пятидесяти миллиардов полей, что может занять несколько десятков минут. Многопоточное отображение результатов помогает следить за процессом. 

Отображение результатов сканирования для каждой станции.



Детализация для каждой ошибки.



# Приложение 1. Реализация статически заданных методов.

**Файл:** TrafficScan.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data;

using System.Windows;

namespace FailCheck

{

public class TrafficScan : IRules

{

// 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.

public Stance stance { get; set; }

public DataTable dt\_objects { get; set; }

public DataTable dt\_situations { get; set; }

public DataTable dt\_types { get; set; }

public TrafficScan(Stance st, DataTable dt\_o, DataTable dt\_s, DataTable dt\_t)

{

stance = st;

dt\_objects = dt\_o;

dt\_situations = dt\_s;

dt\_types = dt\_t;

}

public TrafficScan(DataTable dt\_o, DataTable dt\_s, DataTable dt\_t)

{

dt\_objects = dt\_o;

dt\_situations = dt\_s;

dt\_types = dt\_t;

}

public void get\_stance\_errors() {}

public void get\_peregon\_errors()

{

List<Situation> enters = new List<Situation>();

List<Situation> lights = new List<Situation>();

foreach (DataRow traffic in dt\_objects.Select("ids =" + stance.id + " and type = 2"))

{

if ((string)traffic["name"] == "2" || (string)traffic["name"] == "1") // Предвходные светофары

{

enters.Add(get\_situation(traffic)); //Получаем список ситуаций по предвходным светофорам

}

else

{

lights.Add(get\_situation(traffic)); // Получаем список отказов по проходным светофорам

}

}

enters = compare(enters); //Комбинируем список по одинаковым отказам

lights = compare(lights);

set\_errors(enters); // Формируем ошибку

set\_errors(lights);

}

private Situation get\_situation(DataRow traffic)

{

List<int> situations = new List<int>();

foreach (DataRow enter in dt\_situations.Select("ids =" + stance.id + " and ido\_failure =" + traffic["ido"]))

{

if ((int)enter["situation\_id"] != 0) //Исключение для АКСТ

{

situations.Add((int)enter["situation\_id"]);

}

}

return new Situation((int)traffic["ido"], (string)traffic["name"], situations);

}

private List<Situation> compare(List<Situation> list)

{

List<Situation> compared\_list = new List<Situation>();

foreach (Situation st in list)

{

bool is\_exist = false;

foreach (Situation compared\_situation in compared\_list)

{

bool equals = true;

if (compared\_situation.situation\_list.Count == st.situation\_list.Count)

{

foreach (int situation\_id in st.situation\_list)

{

if (compared\_situation.situation\_list.Contains(situation\_id) == false)

{

equals = false;

}

}

}

else

{

equals = false;

}

if (equals)

{

compared\_situation.name += " " + st.name;

is\_exist = true;

break;

}

}

if (is\_exist == false)

{

compared\_list.Add(st);

}

}

return compared\_list;

}

private void set\_errors(List<Situation> list)

{

if (list.Count > 1)

{

Error error = new Error();

string error\_str = "Светофоры: ";

List<int> situations\_id = new List<int>();

foreach(Situation situation in list)

{

Error\_object error\_object = new Error\_object(situation.name);

error\_str += "'" + situation.name + "' - ";

List<Fail> fail\_list = new List<Fail>();

foreach (int situation\_id in situation.situation\_list)

{

fail\_list.Add(new Fail(situation\_id));

if (situations\_id.Contains (situation\_id) == false)

situations\_id.Add(situation\_id);

}

error\_object.fail\_list = fail\_list;

error.list\_error\_object.Add(error\_object);

}

error\_str += "имеют разные списки отказов. Выберите, чтобы узнать подробней ...";

error.error\_text = error\_str;

change\_error(error, situations\_id);

stance.error\_list.Add(error);

}

}

private void change\_error(Error er, List<int> ids)

{

foreach (int id in ids)

{

string prim = "Не удалось определить описание отказа";

DataRow[] drs = dt\_types.Select("id = " + id);

if (drs.Length == 1)

{

prim = drs[0]["prim"].ToString();

}

else

{

prim = "Не удалось определить описание для отказа " + id;

}

foreach (Error\_object eo in er.list\_error\_object )

foreach (Fail f in eo.fail\_list)

{

if (f.id == id)

{

f.name = prim;

}

}

}

}

}

}

**Файл:** Rule1.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data;

namespace FailCheck

{

public class Rule1:IRules

{

// rule\_1 - Проверка фидеров

// 1. - На станции должен быть хотя бы один отказ "Выключение эллектропитания".

// 2. - Все указатели красного цвета с именим \*фидер\* должны иметь отказ "Выключение эллектропитания".

// 3. - Список отказов остальных лампочек фидеров должен быть одинаковым.

private const int exception1 = 20083; // Код исключения - увеличенное время работы фидеров

private const int fail1 = 101; // Код отказа - Выключение эллектропитания

public Stance st { get; set; }

public DataTable dt\_objects { get; set; }

public DataTable dt\_situations { get; set; }

public DataTable dt\_types { get; set; }

public List<int> vids { get; set; }

public List<int> objects\_fail = new List<int>();

public List<int> list\_fail = new List<int>();

public List<Situation> no\_red\_pointers = new List<Situation>();

public Rule1(Stance s, DataTable dt\_o, DataTable dt\_s, DataTable dt\_t, List<int> v)

{

st = s;

dt\_objects = dt\_o;

dt\_situations = dt\_s;

dt\_types = dt\_t;

vids = v;

}

public void get\_stance\_errors()

{

get\_situations(objects\_fail); // Получение списка отказов Выключение эллектропитания

foreach (DataRow pointer in dt\_objects.Select("IDS = " + st.id + " and Type = 6 "))

{

string str = pointer["name"].ToString();

str = str.ToLower();

if (str.Contains("фидер")) // ЗАМЕЧАНИЕ! ПРОПУСКАЕТ УКАЗАТЕЛЬ "Выкл фид."

{

if (vids.Contains(Convert.ToInt32(pointer["vid"]))) // Красная лампочка

{

if (objects\_fail.Contains((int)pointer["ido"]) == false) // Проверка привязан ли к красной лампе выключение эллектропитания

{

st.error\_list.Add(new Error("К красной лампочке " +

pointer["name"] + "(" + (int)pointer["ido"] + ") не привязан отказ 'Выключение эллектропитания'"));

}

}

else // Не красная лампочка

{

no\_red\_pointers.Add(get\_situation(pointer));

}

}

}

no\_red\_pointers = compare(no\_red\_pointers);

set\_errors(no\_red\_pointers);

if (objects\_fail.Count == 0)

{

st.error\_list.Add(new Error("На станции не привязано ни одного отказа 'Выключение эллектропитания'"));

}

}

public void get\_peregon\_errors(){ }

private void get\_situations(List<int> objects\_fail)

{

foreach (DataRow dr in dt\_situations.Select("ids = " + st.id))

{

if (fail1 == (int)dr["situation\_id"]) // Получение списка

{

objects\_fail.Add((int)dr["IDO\_Failure"]); // Выключение эллектропитания

}

}

}

private Situation get\_situation(DataRow traffic)

{

List<int> situations = new List<int>();

foreach (DataRow enter in dt\_situations.Select("ids =" + st.id + " and ido\_failure =" + traffic["ido"]+ "and disabled = 0" ))

{

if ((int)enter["situation\_id"] != exception1) //Исключение

{

situations.Add((int)enter["situation\_id"]);

}

}

return new Situation((int)traffic["ido"], (string)traffic["name"], situations);

}

private List<Situation> compare(List<Situation> list)

{

List<Situation> compared\_list = new List<Situation>();

foreach (Situation st in list)

{

bool is\_exist = false;

foreach (Situation compared\_situation in compared\_list)

{

bool equals = true;

if (compared\_situation.situation\_list.Count == st.situation\_list.Count)

{

foreach (int situation\_id in st.situation\_list)

{

if (compared\_situation.situation\_list.Contains(situation\_id) == false)

{

equals = false;

}

}

}

else

{

equals = false;

}

if (equals)

{

compared\_situation.name += " " + st.name + "(" + st.id + ")";

is\_exist = true;

break;

}

}

if (is\_exist == false)

{

compared\_list.Add(new Situation (st.id, st.name + "(" + st.id + ")", st.situation\_list));

}

}

return compared\_list;

}

private void set\_errors(List<Situation> list)

{

if (list.Count > 1)

{

Error error = new Error();

string error\_str = "Указатели: ";

List<int> situations\_id = new List<int>();

foreach (Situation situation in list)

{

Error\_object error\_object = new Error\_object(situation.name);

error\_str += "'" + situation.name + "' - ";

List<Fail> fail\_list = new List<Fail>();

foreach (int situation\_id in situation.situation\_list)

{

fail\_list.Add(new Fail(situation\_id));

if (situations\_id.Contains(situation\_id) == false)

situations\_id.Add(situation\_id);

}

error\_object.fail\_list = fail\_list;

error.list\_error\_object.Add(error\_object);

}

error\_str += "имеют разные списки отказов. Двойной щелчок, чтобы узнать подробней ...";

error.error\_text = error\_str;

change\_error(error, situations\_id);

st.error\_list.Add(error);

}

}

private void change\_error(Error er, List<int> ids)

{

foreach (int id in ids)

{

string prim = "Не удалось определить описание отказа";

DataRow[] drs = dt\_types.Select("id = " + id);

if (drs.Length == 1)

{

prim = drs[0]["prim"].ToString();

}

else

{

prim = "Не удалось определить описание для отказа " + id;

}

foreach (Error\_object eo in er.list\_error\_object)

foreach (Fail f in eo.fail\_list)

{

if (f.id == id)

{

f.name = prim;

}

}

}

}

}

}

**Файл** Rule3.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data;

namespace FailCheck

{

public class Rule3 : IRules

{

// rule\_3 - Проверка указателей земля

// 1. - К каждому указателю с названием ("Земля","ГСЗК", "СЗК", "ГСЗ") привязан отказ

// "Понижение сопротивления изоляции источника питания" к состоянию горение или мигание красным.

private const int fail3 = 6; //Код отказа - Понижение сопротивления изоляции источника питания

Stance st ;

DataTable dt\_situations;

DataTable dt\_objects;

DataTable dt\_states;

DataTable dt\_colors;

public Rule3(Stance s, DataTable dt\_o, DataTable dt\_s,

DataTable dt\_st, DataTable dt\_c)

{

st = s;

dt\_situations = dt\_s;

dt\_objects = dt\_o;

dt\_states = dt\_st;

dt\_colors = dt\_c;

}

public void get\_peregon\_errors() { }

public void get\_stance\_errors()

{

foreach (DataRow pointer in dt\_objects.Select("IDS = " + st.id + " and Type = 6 "))

{

string str = pointer["name"].ToString();

str = str.ToLower();

int fail\_cout = 0;

if (str.Contains("земля") || str.Contains("гсзк") // Находим все объекты "земля"

|| str.Contains("сзк") || str.Contains("гсз"))

{

foreach (DataRow situation in dt\_situations.Select("ids = " + st.id

+ " and ido\_failure = " + pointer["ido"] + " and situation\_id = " + fail3)) // Получаем нужные отказы

{

fail\_cout++;

bool elment\_state = false; // Привязан к нужному состоянию

//Получаем состояние отказа

foreach (DataRow dr in dt\_states.Select("situation\_object\_id = " + situation["situation\_object\_id"]))

{

int state = (byte)dr["state\_object"];

int type = Convert.ToInt16(pointer["type"]);

int vid = Convert.ToInt16(pointer["vid"]);

if (check\_state(state, type, vid))

{

elment\_state = true;

}

}

if (elment\_state == false)

{

st.error\_list.Add(new Error("Отказ 'Понижение сопротивления изоляции источника питания' указателя " +

pointer["name"].ToString() + "(" + pointer["ido"] + ") должен быть привязан к состоянию горения или мигания красным"));

}

}

if (fail\_cout == 0) // Нет ни одного отказа

{

st.error\_list.Add(new Error("К указателю " + pointer["name"].ToString() +

"(" + pointer["ido"] + ") не привязан отказ 'Понижение сопротивления изоляции источника питания'"));

}

}

}

}

bool check\_state(int state, int type, int vid)

{

foreach (DataRow dr in dt\_colors.Select("type = " + type + " and vid =" + vid))

{

int color\_count = (byte)dr["colorcount"];

switch (color\_count)

{

case 1:

if (((int)dr["colorpassive"] == 255 || (int)dr["ColorActive1"] == 255)

&& (state == 1 || state == 2))

{

return true;

}

break;

case 2:

if (((int)dr["colorpassive"] == 255 || (int)dr["ColorActive1"] == 255)

&& (state == 1 || state == 2))

{

return true;

}

if (((int)dr["ColorActive2"] == 255)

&& (state == 4 || state == 8))

{

return true;

}

break;

case 3:

if (((int)dr["colorpassive"] == 255 || (int)dr["ColorActive1"] == 255)

&& (state == 1 || state == 2))

{

return true;

}

if (((int)dr["ColorActive2"] == 255)

&& (state == 4 || state == 8))

{

return true;

}

if (((int)dr["ColorActive3"] == 255)

&& (state == 10 || state == 20))

{

return true;

}

break;

}

}

return false;

}

}

}