МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

Направление подготовки 09.04.01 «Информатика и вычислительная техника»
Магистерская программа Вычислительные машины, комплексы, системы и сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

на тему

Система автоматической генерации функциональных блоков для автоматизации цифровых подстанций МЭК 61850

Студент 13.06.18 Войнов Артём Сергеевич
(подпись, дата)

Руководитель 13.06.18 Дубинин В.Н.
(подпись, дата)

Нормоконтролёр 13.06.18 Кучин А.В.
(подпись, дата)

Рецензент доцент каф. ВМиС ПензГТУ 13.06.18 Мартынюшкин А.И.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 25.05.2018 №38-09-12)
Заведующий кафедрой Пащенко Д.В.
(подпись)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 15.06.18 № 11

Секретарь ГЭК Гурин Е.И.
(подпись)

Пенза, 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	Вычислительная техника

Направление подготовки 09.04.01 «Информатика и вычислительная техника»
Магистерская программа Вычислительные машины, комплексы, системы и сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

на тему

Система автоматической генерации функциональных блоков для автоматизации цифровых подстанций МЭК 61850

Студент _____ Войнов Артём Сергеевич
(подпись, дата)

Руководитель _____ Дубинин В.Н.
(подпись, дата)

Нормоконтролёр _____ Кучин А.В.
(подпись, дата)

Рецензент _____ _____ _____
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 25.05.2018 №38-09-12)
Заведующий кафедрой _____ Пащенко Д.В.
(подпись)

Работа защищена с отметкой _____(протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Гурин Е.И.
(подпись)

Пенза, 2018

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»
заведующий кафедрой
_____Д. В. Пащенко
«__»_____2018 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
МАГИСТРА

1. Студент Войнов Артём Сергеевич гр. 16ВВм1 факультета ВТ
2. Тема работы: «Система автоматической генерации функциональных блоков для автоматизации цифровых подстанций МЭК 61850»
   Тема утверждена приказом ПГУ №_____от "____"_____ 2018_
3. Руководитель работы д.т.н., профессор Дубинин В.Н.
4. Задание на работу (назначение разработки, исходные данные и т. п.)

   Разработать методику автоматической генерации функционально-блочной модели системы защиты от перегрузок и методику автоматической генерации систем функциональных блоков стандарта МЭК 61499, реализующих человеко-машинный интерфейс, для цифровых подстанций МЭК 61850. На основе разработанных методик разработать инструментальную систему автоматической генерации функциональных блоков МЭК 61499 для автоматизации цифровых подстанций МЭК 61850.

   Перечень подлежащих разработке вопросов

   1) Обзор и анализ методов проектирования систем управления, коммуникации и релейной защиты современных систем электроснабжения.

   2) Разработка методики автоматической генерации функционально-блочной модели системы защиты от перегрузок для цифровых подстанций МЭК 61850 (Методика 1).

   3) Разработка методики автоматической генерации систем функциональных блоков стандарта МЭК 61499, реализующих человеко-машинный интерфейс для цифровых подстанций МЭК 61850 (Методика 2).

   4) Разработка инструментальной подсистемы для поддержки Методики 1.

   5) Разработка инструментальной подсистемы для поддержки Методики 2.

5. Календарный график по выполнению работы

Наименование этапов работы	**Обьем работы**	**Срок выполнения**	**Подпись руководителя, консультанта**
Обзор и анализ методов проектирования систем управления, коммуникации и релейной защиты современных систем электроснабжения	20%		
Разработка методики автоматической генерации функционально-блочной модели системы защиты от перегрузок для цифровых подстанций МЭК 61850 (Методика 1)	25%		
Разработка методики автоматической генерации систем функциональных блоков стандарта МЭК 61499, реализующих человеко-машинный интерфейс для цифровых подстанций МЭК 61850 (Методика 2)	25%		
Разработка инструментальной подсистемы для поддержки Методики 1	10%		
Разработка инструментальной подсистемы для поддержки Методики 2	15%		

Дата выдачи задания « »_____ 201 г.

Руководитель магистерской работы _____

(подпись, дата)

Задание к исполнению принял студент _____

(подпись, дата)

Работу к защите допустить

Декан факультета _____

(подпись, дата)

РЕФЕРАТ

Магистерская работа содержит 117 листов, 49 рисунков, 12 таблиц, 44 источника, 2 приложения.

Цель работы — разработка системы автоматической генерации функциональных блоков для цифровых подстанций МЭК 61850 на основе описания подстанции на языке Substation Configuration Language (SCL).

Основным предметом исследования являются методы и средства автоматизации проектирования систем управления на основе стандарта МЭК 61499 для цифровых подстанций МЭК 61850.

В результате проведенных исследований были получены следующие результаты: Предложена методика генерации ФБ, реализующих HMI для автоматизации ЦПС МЭК 61850, разработан модифицированный алгоритм Ли, предложена методика автоматической генерации функционально-блочной модели системы защиты от перегрузок для цифровых подстанций МЭК 61850, разработан автоматический транслятор, преобразующий исходное SCD-описание ЦПС в систему ФБ, реализующую HMI, разработан автоматический транслятор для получения Prolog-представления описания сети Smart Grid на основе SCD – описания, разработан автоматический транслятор FB-Prolog описания ЦПС в систему ФБ, реализующую HMI.

По теме диссертации опубликовано 10 научных работ в журналах, сборниках научных трудов и трудах конференций (из них 1 статья опубликована в изданиях, рекомендованных ВАК РФ, и 1 статья в зарубежном журнале, индексируемом в Scopus/Web of Science). Получены свидетельства о гос. регистрации программы для ЭВМ.

Ключевые слова: Smart Grid, схема защиты, электрическая сеть, функциональные блоки, МЭК 61499, МЭК 61850, nxtStudio, Prolog, транслятор, система управления.

Содержание

ВВЕДЕНИЕ.....7
1 ОБЗОР И АНАЛИЗ МЕТОДОВ ПРОЕКТИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ, КОММУНИКАЦИИ И РЕЛЕЙНОЙ ЗАЩИТЫ СОВРЕМЕННЫХ СИСТЕМ ЭЛЕКТРОСНАБЖЕНИЯ.....12
1.1 Краткий обзор литературы.....15
1.2 Разумные сети электроснабжения Smart Grid.....16
1.3 Проектирование на основе управления моделями (MDE).....17
1.4 Многоуровневая модель архитектуры Smart Grid.....18
1.5 Стандарт МЭК 61850 для спецификации цифровых подстанций (ЦПС).....21
1.5.1 Представление конфигурации ЦПС на языке SCL.....21
1.5.2 Описание устройств.....24
1.5.3 Описание логических узлов.....25
1.6 Стандарт МЭК 61499 для построения распределенных систем управления……27
1.6.1 Архитектура системы.....27
1.6.2 Функциональные блоки.....29
1.6.3 Инструментальная система NxtStudio.....30
1.7 Выводы.....32
2 МЕТОДИКА АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ ФУНКЦИОНАЛЬНО-БЛОЧНОЙ МОДЕЛИ СИСТЕМЫ ЗАЩИТЫ ОТ ПЕРЕГРУЗОК ДЛЯ ЦИФРОВЫХ ПОДСТАНЦИЙ МЭК 61850.....33
2.1 Схемы защиты от перегрузок на основе МЭК 61850.....33
2.2 Основные положения методики.....38
2.3 Разработка Prolog-представлений исходной и целевой моделей.....39
2.3.1 Prolog-представление конфигурации ЦПС.....39
2.3.2 Prolog-представление функциональных блоков МЭК 61850.....43
2.4 Разработка правил трансформации исходной модели в целевую модель.....46
2.4.1 Графовые правила трансформации.....46
2.4.2 Правила преобразования SCL-Prolog в FB-Prolog.....50
2.5 Разработка программного обеспечения для поддержки методики.....53
2.5.1 Общая структура программного обеспечения.....53
2.5.2 Транслятор SCL XML-описания в SCL Prolog-представление ЦПС.....54
2.5.3 Преобразователь SCL Prolog-представления в FB Prolog-представление.....59
2.5.4 Транслятор Prolog-представления функциональных блоков в их XML-описание для проектов nxtStudio.....62

2.5.5 Пример использования....63
2.6 Выводы....66
3. МЕТОДИКА АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ СИСТЕМ ФУНКЦИОНАЛЬНЫХ БЛОКОВ МЭК 61499, РЕАЛИЗУЮЩИХ ЧЕЛОВЕКО-МАШИННЫЙ ИНТЕРФЕЙС....68
3.1 Человеко-машинный интерфейс в системе автоматизации ЦПС....68
3.2 Основные положения методики....69
3.3 Создание библиотеки CAT-блоков для HMI....70
3.4 Разработка графовой модели....71
3.5 Размещение элементов HMI....73
3.6 Трассировка соединений на основе алгоритма Ли....74
3.7 Генерация системы функциональных блоков....80
3.8 Разработка транслятора SCD-описаний ЦПС в системы функциональных блоков
3.8.1 Структура транслятора....81
3.8.2 Алгоритмы работы и структуры данных....83
3.8.3 Примеры использования....102
3.9 Выводы....103
Заключение....105
Список использованных источников....108
Приложение А – Свидетельство о регистрации программы....113
Приложение Б – Грамоты за участие в конференциях....115

ВВЕДЕНИЕ

Актуальность работы.

Smart Grid – интеллектуальная сеть нового поколения. Основные её преимущества над традиционной системой электроснабжения:

- интеллектуализация сети

- способность самоконтроля и предоставление информации о любом участнике сети.

- высокая надежность сети. Даже при переключении источника, пользователь не заметит сбоев.

- повышение эффективности за счёт уменьшения потери в проводах и оптимального распределения нагрузки.

Современная Smart Grid представляет собой сложную киберфизическую систему включающую множество датчиков, исполнительных механизмов, котроллеров и сетей передачи данных. Киберфизические системы (КФС) являются атрибутом четвертой промышленной революции (Индустрия 4.0).

Внедрение Smart Grid также поставило задачу разработки устройств для управления электроэнергией безопасным и эффективным способом. Системы защиты играют важную роль в энергетических системах, так как они независимо контролируют безопасность и непрерывность энергоснабжения, а также качество энергии получаемое конечным потребителем.

Коммуникация между элементами сети Smart Grid и оперативным персоналом осуществляется посредством Human Machine Interface (HMI). Важность HMI сложно переоценить – независимо от степени автоматизации, именно операторы играют ключевую роль при принятии важных решений (особенно это актуально в чрезвычайных ситуациях).

Для контроля работы сети был введён стандарт МЭК 61850, в котором описаны правила совместимости протоколов систем автоматизации подстанции в виде множества объектно-ориентированного представления данных.

Одной из важнейших частей Smart Grid являются системы управления, представляющие кибернетическую часть КФС.

Стандарт МЭК 61499 описывает парадигму управления, на основе которой строится вся система управления. В качестве единицы управления выступают функциональные блоки.

Широкое внедрение интеллектуальных сетей электроснабжения нового поколения (сетей *Smart Grid*) и цифровых подстанций (ЦПС) определяет необходимость автоматизации большинства этапов их проектирования, включая разработку человеко-машинного интерфейса (HMI). Автоматизация проектирования позволит существенно снизить вероятность возникновения ошибок, а также сократить стоимость и сроки проектирования.

Исходя из вышесказанного, необходима разработка новых методов и средств для создания и проектирования систем управления сетями электроснабжения нового поколения на основе современных международных стандартов.

Объект исследования.

Интеллектуальные сети электроснабжения Smart Grid.

Предмет исследования.

Методы и средства автоматизации проектирования систем управления на основе стандарта МЭК 61499 для цифровых подстанций МЭК 61850.

Целью диссертационного исследования является разработка методик автоматической генерации функциональных блоков для автоматизации цифровых подстанций МЭК 61850.

Для достижения поставленной цели в диссертационной работе должны быть решены следующие задачи:

1) Обзор и анализ методов проектирования систем управления, коммуникации и релейной защиты современных систем электроснабжения.

2) Разработка методики автоматической генерации функционально-блочной модели системы защиты от перегрузок для цифровых подстанций МЭК 61850 (Методика 1).

3) Разработка методики автоматической генерации систем функциональных блоков стандарта МЭК 61499, реализующих человеко-машинный интерфейс для цифровых подстанций МЭК 61850 (Методика 2).

4) Разработка инструментальной подсистемы для поддержки Методики 1.

5) Разработка инструментальной подсистемы для поддержки Методики 2.

Научная новизна:

1) Предложена методика генерации ФБ, реализующих HMI для автоматизации ЦПС МЭК 61850, отличающаяся от существующих возможностью внесения изменений в структуру начальной сети для генерации оптимизированной системы управления, что позволяет значительно упрощать (или усложнять) системы управления, но только если это необходимо.

2) Разработан модифицированный алгоритм Ли, отличающийся от известных возможностью соединения не только узла с узлом, но и узла с уже проложенным соединением, причем в качестве исходной вершины для распространения волны всегда выбирается неподключенная вершина или вершина с наименьшей степенью связности, что позволяет ускорить процесс трассировки пути и уменьшить общее время работы алгоритма, а также повысить читабельность экранных представлений электрических схем в HMI;

3) Предложена методика автоматической генерации функционально-блочной модели системы защиты от перегрузок для цифровых подстанций

МЭК 61850, которая отличается выделением главного элемента защиты, отключение которого выводит из строя всю систему, что позволяет использовать алгоритмы достижения консенсуса вследствие чего система будет крайне устойчивой.

Практическая ценность:

1) Разработана библиотека ФБ, моделирующих и визуализирующих типы устройств на ЦПС, а также управляющих ими;

2) Разработан автоматический транслятор, преобразующий исходное SCD-описание ЦПС в систему ФБ, реализующую HMI, который в дальнейшем может интегрироваться в систему автоматизации ЦПС в целом. В рамках разработки данного транслятора:

а) Разработан способ представления описания ЦПС на языке Prolog. Данный метод является универсальным и достаточно легко адаптируемым в случае внесения изменений в стандарт МЭК 61850.

б) Разработан способ представления ФБ на языке Prolog

в) Предложен способ трансляции SCL-Prolog в FB-Prolog на основании правил преобразования.

3) Разработан автоматический транслятор для получения Prolog-представления описания сети Smart Grid на основе SCD - описания.

4) Разработан автоматический транслятор FB-Prolog описания ЦПС в систему ФБ, реализующую HMI. В рамках разработки данного транслятора:

а) Выведена формула для преобразования матрицы узлов в матрицу устройств. Данная формула позволяет размещать на плоскости графовую модель соединительных узлов, а затем превращать её в модель устройств, поскольку не из каждого описание сети можно сразу построить графовую модель устройств.

б) Предложен способ получения графовых моделей сети Smart Grid на основе её описания. Возможно получение 2 вариантов графа: граф соединительных узлов и граф устройств

в) Выведена формула для преобразования матрицы узлов в матрицу устройств. Данная формула позволяет размещать на плоскости графовую модель соединительных узлов, а затем превращать её в модель устройств, поскольку не из каждого описание сети можно сразу построить графовую модель устройств.

Публикации. По теме диссертации опубликовано 10 научных работ в журналах, сборниках научных трудов и трудах конференций (из них 1 статья опубликована в изданиях, рекомендованных ВАК РФ, и 1 статья в зарубежном журнале, индексируемом в Scopus/Web of Science). Получены 2 свидетельства о гос. регистрации программы для ЭВМ.

Апробация работы. Основные научные положения и результаты работы докладывались и обсуждались на следующих научно-технических конференциях: 15-й международной конференции «IEEE International Conference on Industrial Informatics (INDIN 2017)» (г. Эмден, Германия, 2017), 3-й Международной научно-технической и научно-методической конференции «Современные технологии в науке и образовании (СТНО)» (г. Рязань, 2018), 13-й и 14-й международных научно-технических конференциях «Новые информационные технологии и системы (НИТиС)» (г. Пенза 2016, 2017), международной научно-технической конференции “Современные информационные технологии (СИТ-2017)” (г. Пенза, 2017), 4-й и 5-й ежегодных межвузовских научно-практических конференциях «Информационные технологии в науке и образовании. Проблемы и перспективы» (г. Пенза 2017, 2018).

Структура работы. Работа состоит из введения, трёх глав, заключения, изложенных на 117 страницах, списка литературы из 44 наименований, 3 приложения и содержит 54 рисунка и 12 таблиц.

1 ОБЗОР И АНАЛИЗ МЕТОДОВ ПРОЕКТИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ, КОММУНИКАЦИИ И РЕЛЕЙНОЙ ЗАЩИТЫ СОВРЕМЕННЫХ СИСТЕМ ЭЛЕКТРОСНАБЖЕНИЯ

Электрическая сеть представляет собой сложную систему производителей и потребителей, которая генерирует, передает и распределяет электричество [11]. Пример традиционной электрической сети показан на рисунке 1.

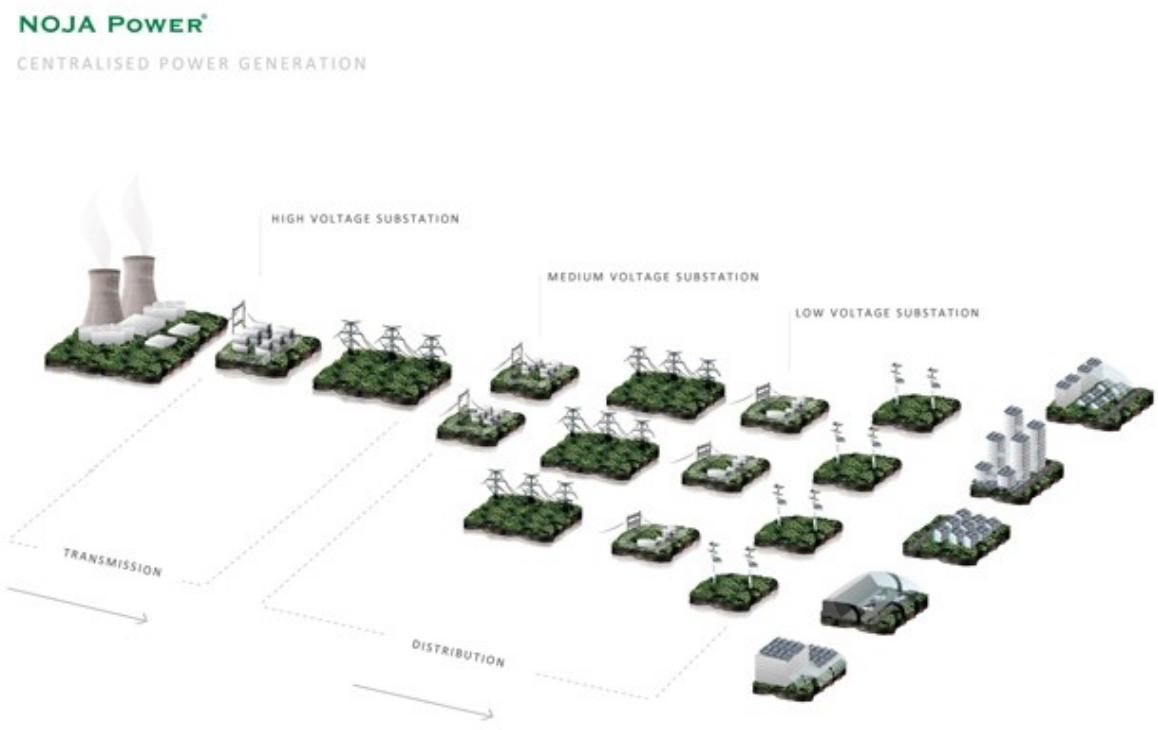


Рисунок 1 - Примет традиционной системы электроснабжения.

Электрическая энергия сначала генерируется на угольных, газовых заводах, гидроэлектростанций и т.д. До того, как передать сгенерированное электричество по линиям электропередач на дальние расстояния, его напряжение значительно повышается. После прихода на подстанцию, напряжение снижается до более низкого, которое распределяется по конечным пользователям (жилым и промышленным помещениям и др.). Представленный процесс передачи электроэнергии является однонаправленным, поскольку

производитель генерирует электричество, а потребитель его потребляет. Принцип работы данного процесса называется централизованной архитектурой.

Благодаря быстрому развитию информационно-коммуникационных технологий (ИКТ) технологий и интеграции ИКТ в существующую электрическую сеть, традиционную энергосеть становится «умной» с возможностью передачи информации в реальном времени с целью принятия разумного решения в целях контроля и защиты. Кроме того, внедрение возобновляемых энергетических ресурсов в Европе означает, что традиционная централизованная архитектура электрической сети постепенно смещается в сторону распределенной, где генерация энергии включает в себя также альтернативные источники энергии (ветровые или солнечные фермы), которые географически могут находиться на огромном расстоянии друг от друга. Затем необходимо рассмотреть возможность потребителей, особенно не промышленных, которые также могут быть источником небольшого количества энергии из возобновляемых источников энергии, таких как солнечные батареи, ветряные турбины и т. д. Можно заметить, что масштаб, в котором генерируется электричество, становится более распределенным. Потребитель теперь становится так называемым prosumer [12], поскольку теперь он может потреблять и производить электроэнергию, хотя и в меньших масштабах. Это означает, что существующие однонаправленные отношения между производителем и потребителем становятся двунаправленными, и потребители теперь также являются частью процессов производства электроэнергии. Концептуальный пример распределенной архитектуры показан на рисунке 2.

Массовые генераторы энергии все еще будут существовать. Однако там будут также альтернативные источники энергии, такие как солнечные фермы, ветряные электростанции и в меньших масштабах, солнечная энергия и энергия ветра от жилого уровня. По мере того, как архитектура электрической сети перемещается в сторону распределенной системы, система защиты и

управления автоматикой также должна следовать и развиваться в более распределенную структуру.

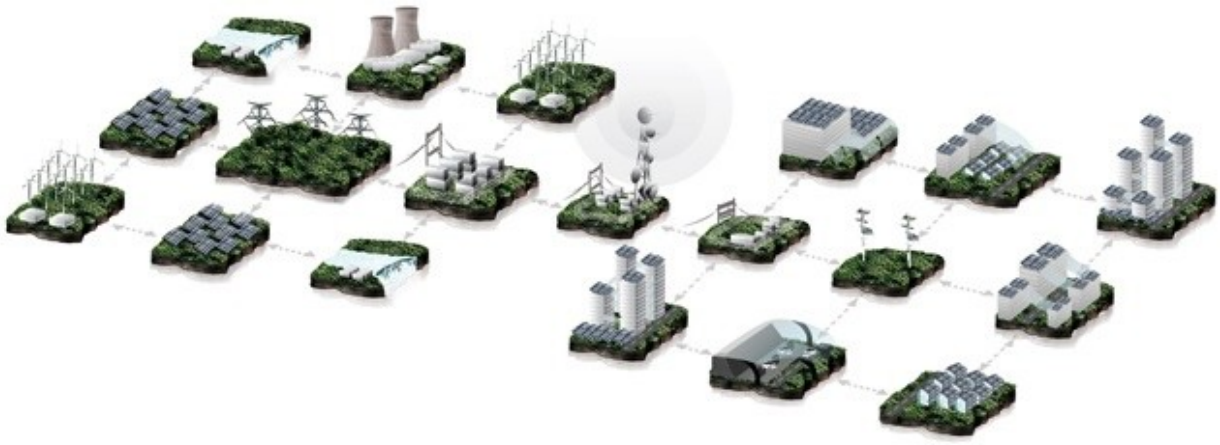


Рисунок 2 - Концептуальная модель сети электроснабжения Smart Grid представленная компанией NOJA power.

Определение Smart Grid в соответствии с Национальным институтом стандартов и технологий (NIST) [13] выглядит следующим образом:

Термин «Smart Grid» относится к модернизации системы подачи электроэнергии, поэтому он контролирует, защищает и автоматически оптимизирует работу своих взаимосвязанных элементов - от центрального и распределенного генератора через высоковольтную сеть передачи и систему распределения, до промышленных пользователей и систем автоматизации зданий а также хранилищ энергии и конечных потребителей и их бытовых приборов. Smart Grid будет характеризоваться двухсторонним потоком электроэнергии и информации для создания автоматизированной широко распределенной сети доставки энергии. Он объединяет в сеть распределенные системы вычислений и коммуникаций для доставки информации в режиме реального времени и обеспечения почти мгновенного баланса спроса и предложения на уровне устройства.

1.1Краткий обзор литературы

В настоящее время существует несколько исследовательских работ, включающих как МЭК 61850, так и МЭК 61499 с предлагаемой архитектурой и средой совместного выполнения. В работах [1,2] предлагается интеллектуальная архитектура логического узла, которая представляет каждый логический узел МЭК 61850 как МЭК 61499 функциональный блок. Более того, среды совместного моделирования были также разработаны в [3] с участием коммерческого МЭК 61499 на платформе разработки NxtStudio [4] и MATLAB Simulink. Также совместное моделирование с открытым исходным кодом было также разработано в [5], которое использует 4DIAC [6] в качестве МЭК 61499.

Model-Driven Engineering (MDE) широко используется в разработке программного обеспечения, поскольку она доказала свою эффективность при разработке приложений большого масштаба. MDE следует мантре «все это модель» [7], где парадигмы проектирования сосредоточены вокруг модели, будь то модели, зависящие от платформы модели или независимые. Недавно [8] MDE нашла свой путь в отрасли автоматизации. МЭК 61850 может быть рассмотрено как решение MDE для систем SAS с введением на подстанциях языка программирования SCL. Язык SCL представлен как XML-документ, и имеется в общей сложности 6 типов документов SCL, каждый документ описывающих различную стадию процесса проектирования MDE. Например, системная спецификация System Specication Description (SSD) используется на начальном этапе проектирования для определения формальной спецификаций системы, которая включает в себя функциональные и физические требования, представленные как логические узлы. Общая конструкция всей системы называется Substation Conguration Description (SCD), которое содержит общую конфигурацию системы, включая конфигурацию связи, логические узлы и IED.

Существуют исследовательские работы, в которых также используется язык SCL в качестве основы для автоматической генерации систем управления, которая является одним компонентом разработки MDE. Открытая платформа

для быстрой защиты прототипов и контроля схемы, основанные на системах МЭК 61850, были предложены в [9]. Документ SCD используется в этой работе как основа для преобразования и автоматического создания базы управления для С язык программирования. Архитектура SysGrid в [10] является MDE-ориентированной архитектурой которая использует SCL-конфигурации в качестве исходных моделей MDE для моделирования МЭК 61850/61499. Модели SCL, в частности, описание SCD используются для автоматически генерируемые коды управления ПЛК МЭК 61499 на основе архитектуры iLN.

1.2 Разумные сети электроснабжения Smart Grid

Smart Grid (интеллектуальная сеть) - это модернизированные сети электроснабжения, которые используют информационные и коммуникационные сети и технологии для сбора информации об энергопроизводстве и энергопотреблении, позволяющей автоматически повышать эффективность, надёжность, экономическую выгоду, а также устойчивость производства и распределения электроэнергии [14]. Основным её отличием является децентрализация системы управления, поскольку необходимо управление сетью в огромных масштабах (это необходимо потому, что возобновляемые источники энергии распределены на достаточно огромной территории, ввиду чего невозможно их централизованное управление). Проектированиие сети такого масштаба имеет множество проблем, таких как обеспечение безопасности передачи данных, надёжности, масштабируемости и технического обслуживания[15]. Основными элементами сети являются интеллектуальные электронные устройства (ИЭУ), которые располагаются на всех логических узлах сети. ИЭУ осуществляют контроль работы в сети а также взаимодействие друг с другом, благодаря чему происходит быстрый переход сети из одного состояния в другое (смена используемой линии передачи) при возникновении чрезвычайных ситуации (повышение напряжения, выход какого-либо оборудования из строя и др.)

Интеллектуальная сеть имеет проблемы не только при передаче данных и при взаимодействии отдельных компонентов сети, но и при обмене информацией между физической и кибернетической частью. Главной их причиной является высокая сложность и постоянно меняющаяся структура основных физических систем в течении жизненного цикла Smart Grid, что приводит к перестройке и повторному тестированию кибернетической части, которая является наиболее сложным элементом сети. Для контроля работы сети был введён стандарт МЭК 61850[16], в котором описаны правила совместимости протоколов систем автоматизации подстанции в виде множества объектно-ориентированного представления данных.

1.3 Проектирование на основе управления моделями (MDE)

Model-Driven Engineering - это парадигма разработки программного обеспечения, которая использует абстракции моделей на разных этапах процесса проектирования сложных программных систем. Целью MDE является повышение производительности и снижение затрат и времени на развития сложных программных систем путем подхода к задачам проектирования системы из перспективы домена, а не языка программирования или аппаратного обеспечения. MDE вводит несколько методологий, которые могут применяться для использование моделей абстракции и одной из основных методологий является автоматическая трансформация модели [17]. Идея MDE заключается в создании абстракции моделей технического процесса, а затем применение метода трансформации модели для преобразования одной модели к другой. Это один из наиболее важных элементов принятия MDE для инженерных программных системах в кибер-физической системе (CPS), поскольку эти абстракции моделей также являются средством официальной документации, формально документируя технический процесс через процесс проектирования. Эти абстракции моделей также обеспечивают доступность и возможность повторного использования что требуется в программной системе CPS, поскольку необходимо модифицировать только модель абстракции при

необходимости. Системное моделирование является одной из наиболее широко используемых методологий в разработке программного обеспечения при построении сложных программных систем и в качестве обособленной системы автоматизации Smart Grid становится более программным [18], необходимость в новых методологиях проектирования, моделирования и поддержки имеют первостепенное значение при переходе системы управления автоматизацией от централизованной парадигмы до распределенной парадигмы.

1.4 Многоуровневая модель архитектуры Smart Grid⁻

Модель архитектуры Smart Grid (Smart Grid Architecture Model) (SGAM) [19], показанная на рисунке 3, представляет собой среду разработки Smart Grid, выпущенных компанией NIST с целью обеспечения совместимости в стандартах и протоколах между устройствами и системами Smart Grid. SGAM показывает, что в Smart Grid большое взаимодействие между уровнями функциональной совместимости для выполнения всего контроля системы управления в реальном времени и услуги, которые Smart Grid обязуется предоставить в будущем. Цель и функциональность слоев следующие:

Первый слой представляет собой компонентный слой, который описывает физическое оборудование, которое используется в Smart Grid. Эти физические устройства могут быть как интеллектуальными электронными устройствами(IED) для защиты, так и оборудованием для физического переключения или автоматическими выключателями или переключателями на подстанции и в сетевом оборудовании, таким как маршрутизаторы и переключатели.

Второй слой описывает способ связи между физическими и логическими устройствами. Это может включать протоколы связи и механизм взаимодействия между полевыми устройствами.

Третий слой является информационным и представляет собой модель виртуальных данных, которые описывают кибернетическую систему(модель данных функций управления и защиты) и физическую систему (модель данных оборудования, такого как автоматические выключатели на подстанции) компонентов Smart Grid. Информационный слой обеспечивает стандартизованное представление Smart Grid компонентов, которые необходимы для взаимодействия.

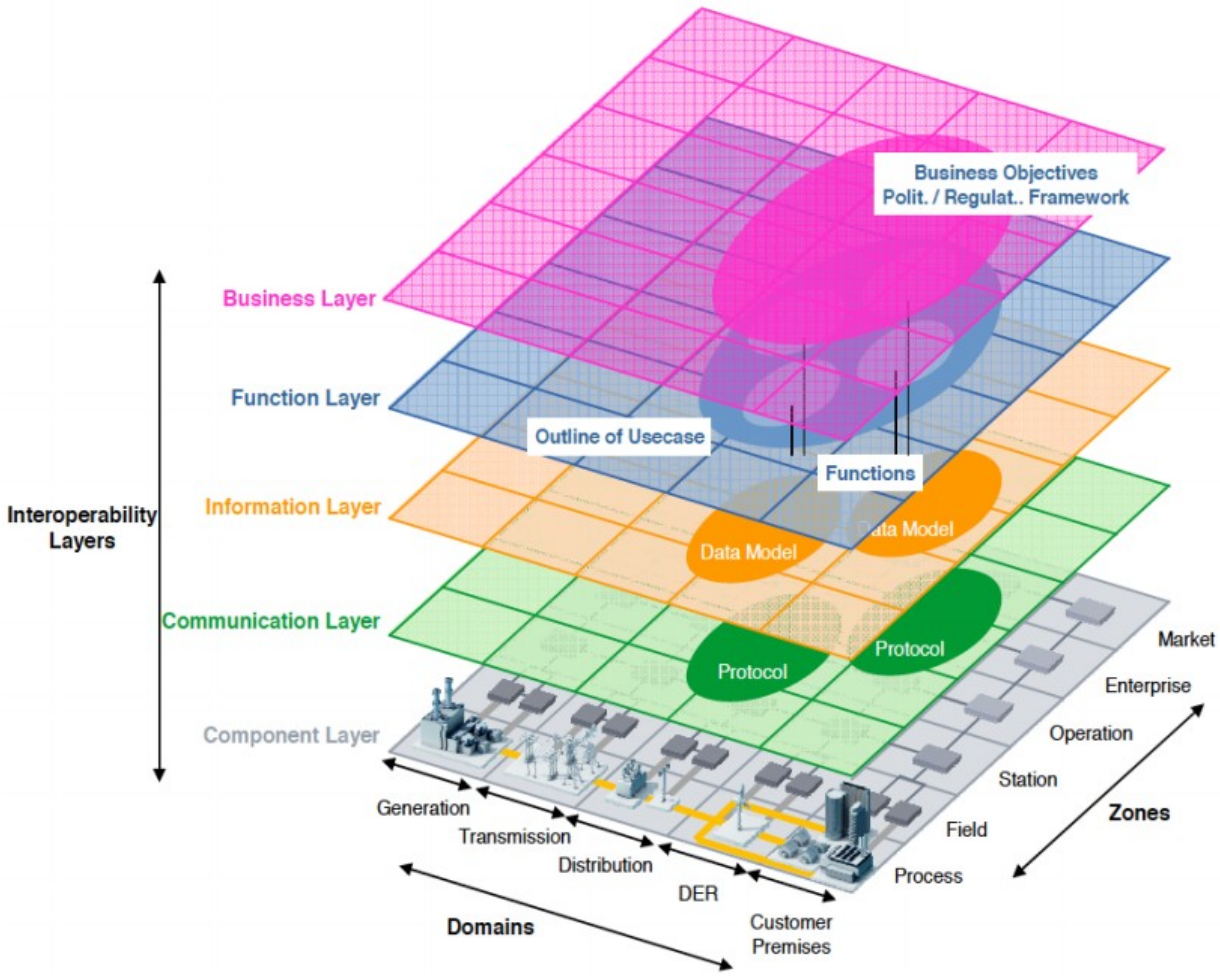


Рисунок 3 - Концептуальная модель сети Smart Grid представленная компанией компанией NIST

Четвёртый слой представляет собой функциональный уровень, который обеспечивает функциональную реализацию системы защиты и контроля Smart Grid и общее представление о том, как система реализована.

Пятый слой представляет собой бизнес-уровень, который включает бизнес-аспект Smart Grid. Этот слой может включать экономичные модели, которые влияют на решения управляющей компании относительно Smart Grid.

Существуют сотни стандартов, которые способствуют стандартизации Smart Grid. Из них выделяется два стандарта, которые наиболее легко удовлетворяют требованиям и дизайн устройств для распределённого взаимодействия. Во-вторых, внедрение системного инженерного подхода, который облегчает проектирование распределенных систем управления - промышленные стандарты МЭК 61850 [20] и МЭК 61499 [21]. В стандарте МЭК 61850 введен язык SCL (Substation Configuration description Language), который служит для описания подстанции. Что касается модели совместимости SGAM, то эти два стандарта способных преодолевать проблемы взаимодействия между коммуникационным, информационным и функциональным уровнем. МЭК 61850 - это стандарт автоматизации подстанции, который вводит стандартизированные протоколы связи и модели данных в системы подстанций для интероперабельности связи внутри подстанции. В частности, в МЭК 61850 определены компонентные модели артефактов и системной инженерии, которые основаны на использовании подхода сверху вниз, который позволяет создавать и конструировать автоматизацию связи подстанции распределенным способом. Два SGAM слоя, положение которых соответствуют стандарту МЭК 61850, являются уровнем связи и информационным слоем. МЭК 61499, с другой стороны, является эталонной архитектурой для разработки распределенных системы автоматизации управления и внедряет программные компоненты, называемые функциональными блоками для распределенных систем автоматизации. Слой, который МЭК 61499 вносит в SGAM - функциональный уровень. Согласование этих двух стандартов будет способствовать согласование трех уровней SGAM и, кроме того, содействовать разработке которые помогают в системе автоматизации проектной подстанции в распределенной путь.

1.5 Стандарт МЭК 61850 для спецификации цифровых подстанций (ЦПС)

1.5.1 Представление конфигурации ЦПС на языке SCL

В данном разделе рассмотрим конфигурацию файла SCL. Файл состоит из 4 основных блоков и блока заголовка (Рисунок 4).

Рисунок 4 - UML диаграмма классов SCL

Начнём рассмотрение блоков SCL с тега Substantion, который описывает первичное оборудование подстанции (Рисунок 5).

Рисунок 5- UML диаграмма блока Substation.

Представленная диаграмма UML определяет все классы из блока Substation, которые были транслированы в Prolog код.

UML диаграмма вторичного оборудования выглядит следующим образом (Рисунок 6).

Рисунок 6 - UML диаграмма блока Communication.

Вторичным оборудованием является сеть передачи ланных. Для обмена информацией используется адресация по протоколу TCP/IP, что позволяет реализовать данную сеть на базе сети Internet.

Самым главным блоком является блок IED (ИЭУ), поскольку именно в нём реализовано непосредственное управление всей подстанцией (Рисунок 7).

Рисунок 7 - UML диаграмма блока IED.

Связь между отдельными классами блока IED реализована с использованием вторичного оборудования, описанного ранее, а типы данных, передаваемых по этим линиям, определены в блоке DataTypeTemplates (Рисунок 8).

Рисунок 8 - UML диаграмма блока DataTypeTemplates.

1.5.2 Описание устройств

Каждое ИЭУ является физическим устройством. МЭК 61850 обеспечивает логический узел LPHD для моделирования физического устройства:

• Физическое (аппаратное) состояние.

• Проблемы связи со статистическими счетчиками.

• Состояние электропитания со статистическими счетчиками.

Поскольку все LN должны находиться в некотором логическом устройстве (LD), это относится также к вышеуказанным LN. Если у вас есть только один LD, то, естественно, они находятся внутри него, иначе они находятся в специально названном логическом устройстве LD0 вместе с LLN0, который содержит общие данные для всех LN LD. В этом втором случае рекомендуется зарезервировать LD0 только для этой цели, а также для определения всех наборов данных и блоков управления связью. Это оставляет другие LD полностью открытыми для функционального моделирования, и вам не нужно прикасаться к ним для коммуникации на уровне МЭК 61850 (пример на рисунке 9).

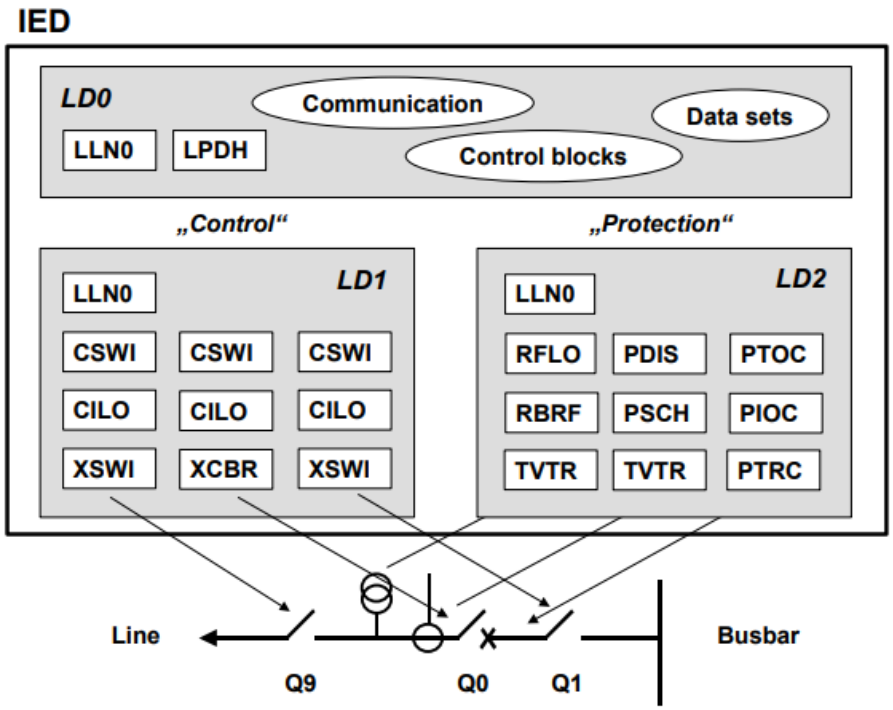


Рисунок 9 - Типичная модель для защиты и управления.

1.5.3 Описание логических узлов

Логическое устройство позволяет через свои логические узлы LLN0 использовать общие функции управления для всех содержащихся LN.

Можно выделить 2 функции:

• объект данных Mod, который позволяет одновременно блокировать или отключать все LN логического устройства

• объект данных Loc, который показывает локальное / удаленное состояние для всех элементов управления процессом в LD.

Обычно IED содержит под LD0 для физического устройства отдельные LD для защиты и управления согласно Рисунок 7, и очень часто, дополнительно для предотвращения помех. Если IED также управляет переключателями как трансформатор, то обычно далее происходит разделяется на разные LD. Простые IED, содержащие, например, просто защиту, могут иметь только один LD, который затем управляет всеми содержащимися LN. Если существует такая

групировка с указанием на функции, то рекомендуется указывать LD (под обязательным LD0) в соответствии с МЭК 61346.

Префикс позволяет производить функциональную группировку для генерации или присвоения имени в LD, без каких-либо общих функции управления. Представление при моделирования и проектировании устройств могут содержать предварительно сконструированные функциональные части в рамках IED выполняя одну общую цель. Типичная группировка уже упоминалась выше: все LN для распределённой защиты получают общий префикс от независимой резервной копии, затем функция перегрузки по току получает другой (или нет) префикс, чтобы отделить его. Префикс также может использоваться для различения и обозначают различных способов использования связанных с PTOC функции, например. первый или второй этап максимального тока, ток заземления защиты и т. д.

МЭК 61850 определяет семантику на LN и уровне объектов данных. Однако детальная функциональность LN также зависит от ее привязки к процессу. Типичным примером здесь является логика блокировки (LN CILO), которая выглядит разной для разъединителя шины и для заземляющего переключателя или линейного разъединителя. Один из способов показать специальную реализацию CILO в соответствии коммутатором на одной линейной диаграмме - использовать префиксы, как показано в таблице 1.

Таблица 1. Префиксы для CILO.

Prefix (acc. IEC 61346)	**Grouped LNs**
QA1 (circuit breaker)	CSWI1, CILO1, XCBR1
QB1 (busbar isolator BB1)	CSWI2, CILO2, XSWI2
QB2 (busbar isolator BB2)	CSWI3, CILO3, XSWI3
QE1 (earthing switch)	CSWI4, CILO4, XSWI4

Другой, более общий способ заключается в моделировании соответствующей части (например, отсека) устройства со ссылками на LN в разделе подстанции файла SCL для этого IED: этот SCL-файл просто содержит

описание типа IED с пустым именем IED, раздел IED и соответствующую часть раздела подстанции SCL.

1.6 Стандарт МЭК 61499 для построения распределенных систем управления

1.6.1 Архитектура системы

Управляющая система – это совокупностью соединенных друг с другом устройств в коммуникационной сети, состоящей из линий связи и сегментов. (рисунок 10).

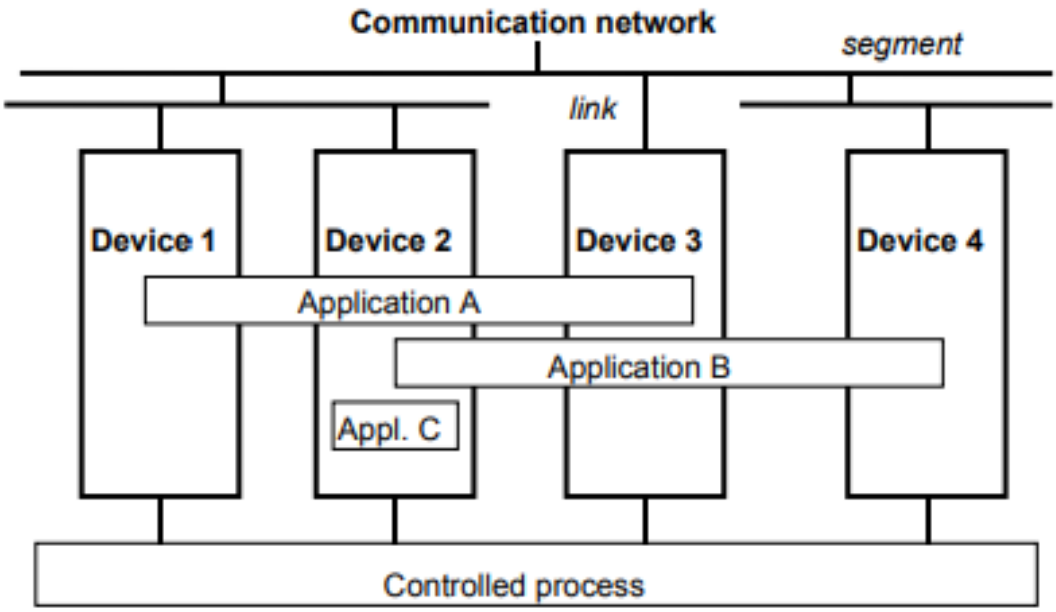


Рисунок 10 - Модель системы МЭК 61499 из [22].

Система управления выполняет функции, которые были описаны с использованием распределённых по одному или нескольким узлам приложений.

Устройство (device) - это физическая сущность, имеющая по меньшей мере один интерфейс, который может быть или коммуникационным интерфейсом или интерфейсом управляемого процесса, способная исполнять одну или множество функций. [22].

На практике в качестве устройства могут выступать программируемые логические контроллеры (ПЛК), программируемые контроллеры автоматизации (ПКА), промышленные компьютеры. Устройство состоит из одного или нескольких ресурсов.

Ресурс – это функциональная единица с независимой системой управление происходящих в ней операций (планирование и исполнение алгоритмов) [22]. Данная функциональная единица может быть создана, исполнена, настроена и удалена без воздействия на другие ресурсы. Основными функциями ресурса являются прием данных и событий а затем их обработка и выдача коммуникационной сети или управляемому процессу (в зависимости от конфигурации приложения, исполняемом на ресурсе). Самой главной функцией, исполняемой ресурсом, является планирование, поскольку оно определяет порядок исполнения ФБ и обмен данными между ними. В упрощенном виде ресурс можно охарактеризовать как операционную систему с некоторым исполняемым кодом.

Приложение – это особая функциональная единица, смысл которой в решении некоторой задачи в системе управления. Данная функциональная единица представляет собой сеть связанных ФБ и под-приложений, исполняющихся на разных устройствах системы.

Модель приложения представлена на рисунке 11.

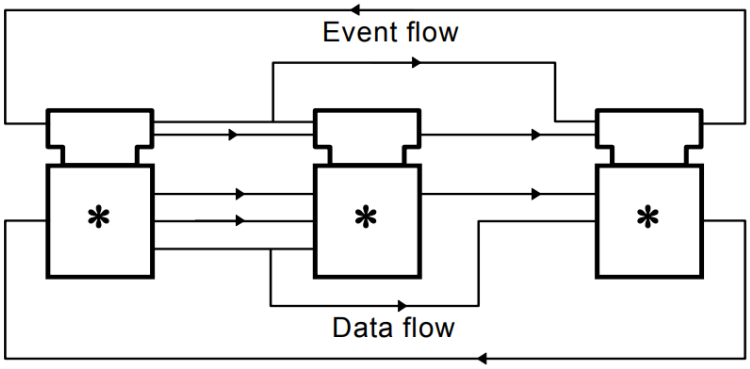


Рисунок 11 - Модель приложения МЭК 61499 из [23].

1.6.2 Функциональные блоки

Функциональный блок – основной артефакт проектирования в МЭК 61499. Данный артефакт представляет собой специальный компонент с интерфейсом. Интерфейс определяется совокупностью входов и выходов (событийных и информационных), посредством которых происходит общение с окружением. Информационные входы и выходы реализованы посредством переменных ФБ

Отдельные экземпляры ФБ содержат набор интерфейсов наследованных от типов ФБ, но имеются важные отличия:

1. В интерфейсе отдельного экземпляра могут отсутствовать некоторые событийные и/или информационные входы/выходы, поскольку этот интерфейс является подмножеством интерфейса типа ФБ.
2. В интерфейсе типа ФБ возможны специальные WITH ассоциации между информационными и событийными входами/выходами, которые используются для получения данных Интерфейс WITH ассоциации представляет собой соединение событийного входа/выхода с одним или несколькими информационными входами/выходами с которых необходимо будет произвести получение данных в случае прихода сигнала на соответствующий событийный вход.

Управление ФБ основано на событиях, при этом события являются активными, данные – пассивными.

Согласно данному стандарту существует 3 вида ФБ: базисный, составной и сервисный интерфейсный. Структуры базисного и составного показаны на рисунке 12.

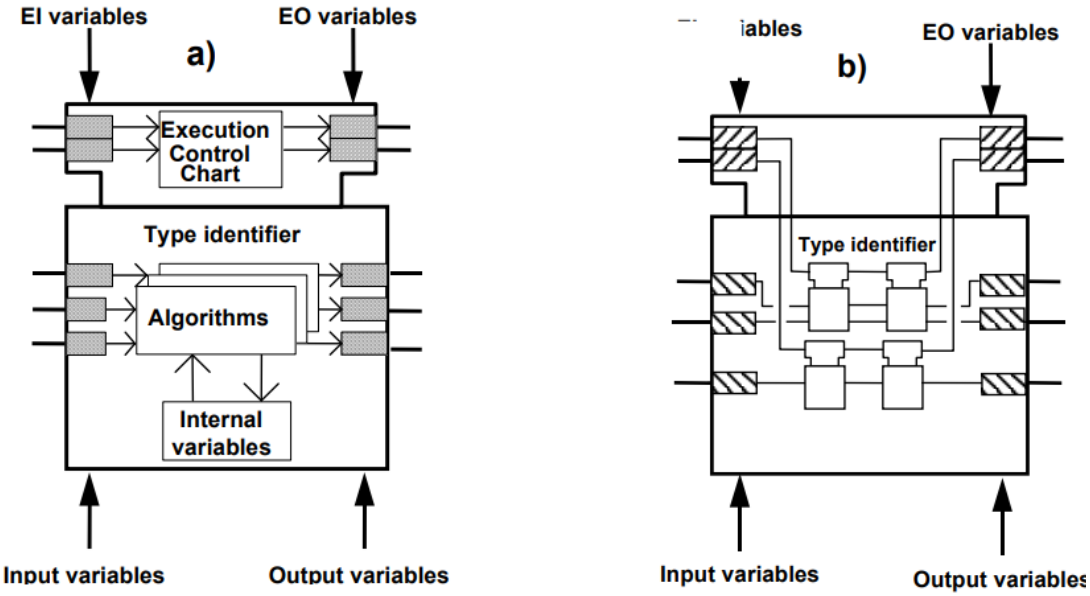


Рисунок 12 - Модель приложения МЭК 61499 из [22].

Принцип функционирования базисного ФБ определяется машиной состояний типа Автомата Мура. С состояниями автомата могут быть связаны алгоритмы и/или событийные выходы. Алгоритмы составляются с использованием одного из языков программируемых контроллеров стандарта МЭК 61131-3 [24]. Для активации базисного ФБ необходимо подать сигнал на некоторый событийный вход, после чего произойдет срабатывание автомата Мура и будут выданы выходные сигналы.

1.6.3 Инструментальная система NxtStudio

Инженерный инструмент nxtSTUDIO (Рисунок 13) предлагает наиболее эффективную разработку распределенных систем. Он бесшовно интегрирует все задачи автоматизации в один инструмент, предлагает готовые библиотеки программного обеспечения, обеспечивает аппаратно-независимое проектирование и использует стандарт МЭК 61499, а также МЭК 61131 в качестве своей парадигмы управления.

Инженеры nxtSTUDIO могут быстро и качественно создавать свои приложения. Дополнительная экономия средств создается путем разделения жизненных циклов аппаратного и программного обеспечения.

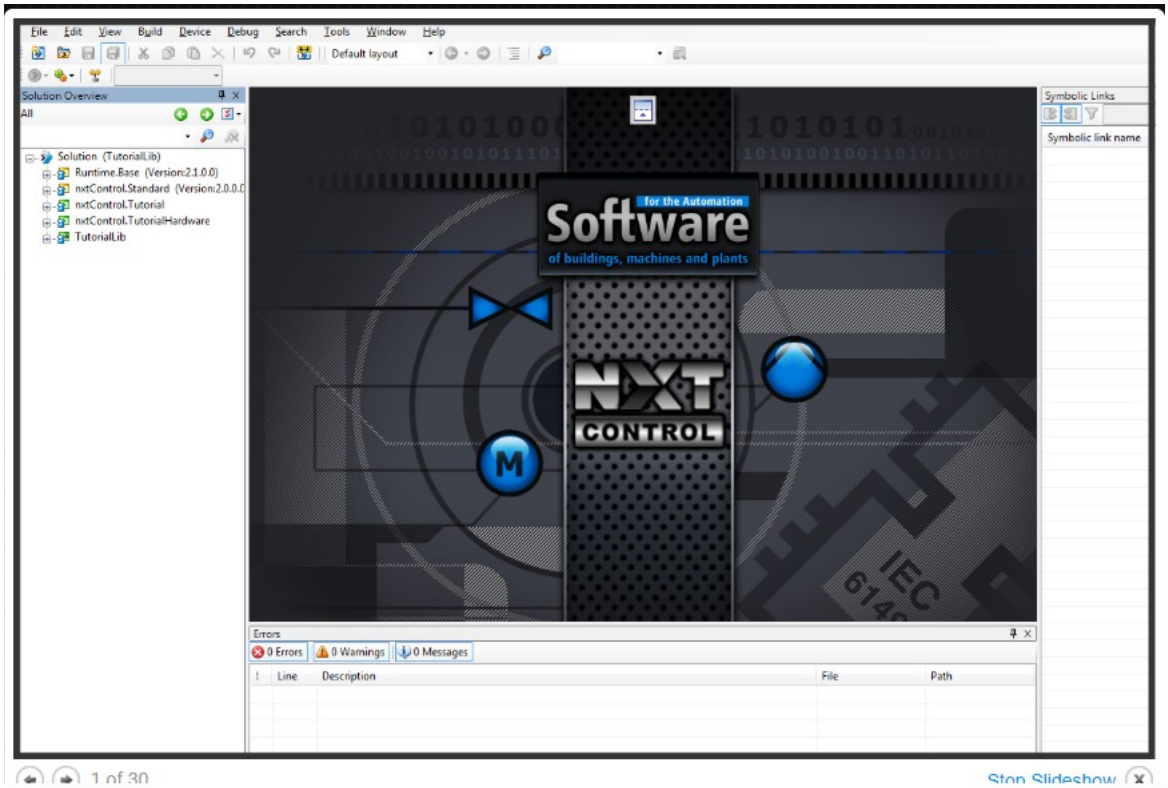


Рисунок 13 - Главное окно nxtStudio.

Возможны две основные процедуры:

1) С предварительно сознанными программными объектами / библиотеками программного обеспечения.

Разработка приложений с помощью CAT-блоков (готовых программных объектов) для быстрой реализации и высокого качества за счет применения повторно используемых разработанных ранее решений. Разработка приложений поддерживается многими функциями, направленными на уменьшение количества ошибок и облегчение работы.

2) Создание новых программных объектов.

Разработка приложений с созданными объектами программного обеспечения для максимальной гибкости и максимального индивидуального творчества.

Список функций и возможного применения:

- программирование управления, HMI / SCADA, документация, симуляция, тестирование в одном инструменте.

-автоматически строить пути связи между контроллерами в распределенной сети.

-автоматически строить пути связи между контроллерами и HMI

-автоматическое создание документации, включая диаграммы, с перекрестными ссылками.

-прямой обмен данными между контроллерами и базами данных SQL.

и многое другое.

1.7 Выводы

1. На основе анализа литературы определено, что будущие системы энергоснабжения будут построены на основе международного стандарта МЭК 61850.

2. Основной тенденцией в сфере промышленной автоматизации является переход от централизованных систем к распределенным системам. Ключом к построению современных распределенных систем управления промышленной автоматики является международный стандарт МЭК 61499.

3. Важным элементом в исследовании является разработка систем управления Smart Grid. Применение функциональных блоков позволяет сделать систему управления более гибкой и современной.

4. На основе анализа литературы определено, что в настоящее время существует несколько исследовательских работ, включающих как МЭК 61850, так и МЭК 61499 с предлагаемой архитектурой и средой совместного выполнения.

5. В качестве исходной модели можно выбрать язык описания подстанции SCL, поскольку он является XML-ориентированным языком, что позволяет при работе с ним использовать готовые парсеры. Данный язык описан в стандарте МЭК 61850.

2 МЕТОДИКА АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ ФУНКЦИОНАЛЬНО-БЛОЧНОЙ МОДЕЛИ СИСТЕМЫ ЗАЩИТЫ ОТ ПЕРЕГРУЗОК ДЛЯ ЦИФРОВЫХ ПОДСТАНЦИЙ МЭК 61850

По данной методике было опубликовано несколько работ [34-38].

2.1 Схемы защиты от перегрузок на основе МЭК 61850

Схемы защиты позволяют избежать остановки производственных или других процессов, вызванной критическим изменением напряжения (слишком высокое или низкое) или ошибками короткого замыкания. В работе [25] рассматриваются различные пути решения проблемы защиты от перегрузок по току. Выделяется два типа ошибок: ошибки на генераторе электроэнергии и ошибки на общей шине. Рассматриваются следующие схемы защиты: 1) защита распределительной шины; 2) выборочное отключение резервирования; 3) схема симпатического (индуцированного) отключения; 4) защита от отказа выключателя. Симпатическое отключение является феноменом в системах электроснабжения, когда устройство защиты в «здоровой» секции системы обрабатывает отказ в другой секции системы. Симпатическое отключение приводит к ненужному отключению электропитания работоспособного оборудования. В работе [25] приводится общее описание модели для тестирования, но не приводится конкретный пример. Это определяет актуальность разработки имитационных моделей для анализа и тестирования схем защиты.

В качестве основы для построения такой имитационной модели была принята работа [26]. В данной работе рассматривается система защиты от перегрузок на подстанции, реализующая сценарий распределенной защиты от симпатического отключения на основе стандартов МЭК 61850 и МЭК 61499. Основными артефактами проектирования в стандарте МЭК 61499 являются функциональные блоки (ФБ) [27]. В работе [26] реализация защиты на цифровой подстанции (ЦПС) МЭК 61850 выполнена на основе ФБ, однако

представлена только сеть ФБ верхнего уровня. Кроме того, не разработан визуальный человеко-машинный интерфейс (HMI) для тестирования схемы защиты.

Логическая структура схемы защиты от перегрузок (назовем ее элементом защиты) представляет собой комбинацию из четырех логических узлов: *TCTR* – датчик, который считывает текущее значение величины силы тока в измеряемом устройстве; *PIOC* – сравнение значения, полученного от датчика, с пороговым; *PTRC* – логический узел, который отвечает за рассылку сообщения об ошибке другим устройствам защиты в сети; *XCBR* – выключатель для разрыва соединения. Полная структура данной схемы защиты представлена в [24].

Кратко рассмотрим алгоритм ее работы. Датчик (*TCTR*) постоянно считывает текущее значение величины силы тока в заданной линии и выдает соответствующий сигнал *TCTR.AmpSV_instMag_f*. При поступлении этого значения в *PIOC* происходит его сравнение с некоторым внутренним значением, представляющим собой порог. Если текущее значение тока превышает пороговое, то выдаётся сигнал *PIOC.Op_general*. Следует заметить, что в *PIOC* также поступают внешние сигналы *PTRC.Tr_general*, которые свидетельствуют о том, что другими элементами защиты обнаружена ошибка. В этом случае также выдаётся сигнал *PIOC.Op_general*. Логический узел PTRC отвечает за проверку внутренних параметров при поступлении сигнала *PIOC.Op_general*. В работе принято, что внутренний параметр только один – время действия сигнала *PIOC.Op_general*. Если *PIOC.Op_general* слишком долго находится в активном состоянии, то происходит выдача сигнала *PTRC.Tr_general* (как на узел *XCBR*, так и наружу - другим внешним элементам защиты). Последним узлом структуры является узел *XCBR*. Данный логический узел производит разрыв цепи при поступлении на него сигнала *PTRC.Tr_general* (в том числе и извне).

Простая распределительная сеть с реализованной распределённой схемой защиты от симпатического отключения представлена на рисунке 14. В ней имеется один источник электроэнергии (сверху) и три потребителя (внизу). Всего в данной сети используется четыре интеллектуальных электронных устройства защиты (*PIED - Protective Intelligent Electronic Devices*), которые соединены между собой и могут обмениваться сообщениями.

Система защиты от перегрузок, включающая логические узлы *TCTR*, *PIOC*, *PTRC*, *XCBR*, была реализована в виде системы ФБ (рисунок 15), причем использовалась инструментальная система *NxtStudio* [28]. При тестировании системы защиты на вход *VALUE* блока *TCTR* подается случайное число в диапазоне от 10 до 100, определяющее условную силу тока. При моделировании приняты следующие допущения: 1) превышением порога считается сила тока более 90; 2) для обнаружения ошибки данное превышение должно сохраняться в течение 1,5 сек. (при условии измерения значения датчиком каждые 1 сек.). Данные условия позволяют пошагово, в визуальном режиме промоделировать работу схемы.

В ходе работы было выявлено два возможных случая срабатывания системы защиты:

1) если на одном элементе превышение порогового напряжения сохраняется более 1,5 сек.; 2) если последовательно на нескольких элементах выявлено превышение порогового напряжения и суммарная длительность превышения в системе более 1,5 сек.

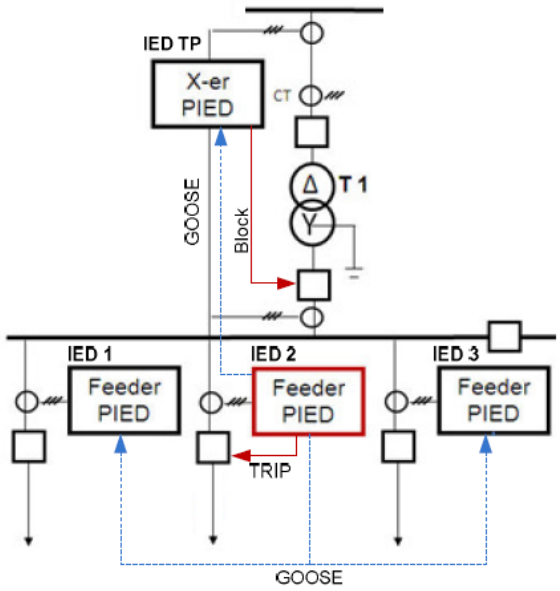


Рисунок 14 - Простая распределительная сеть с реализованной распределённой схемой защиты от симпатического отключения.

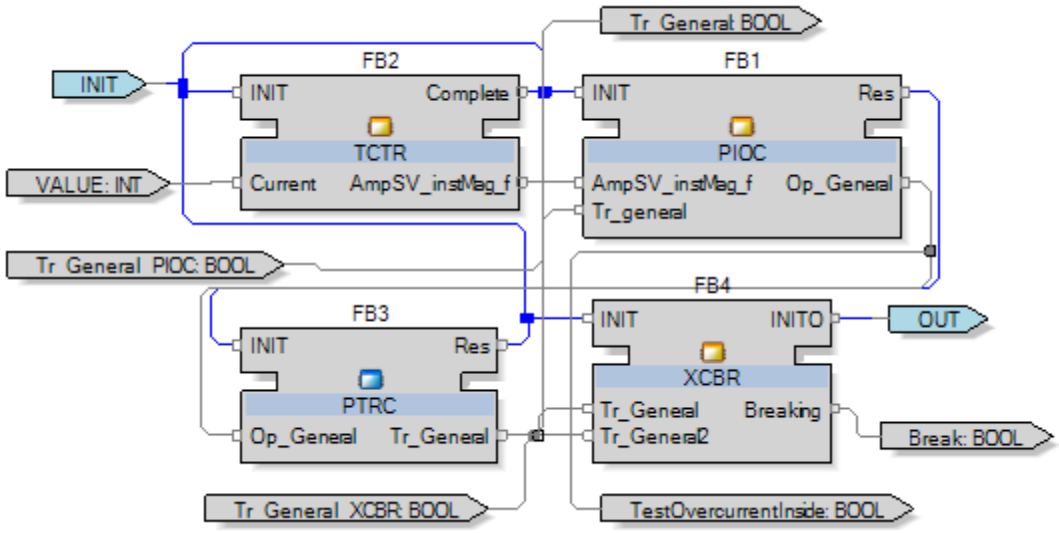


Рисунок 15 - Реализация одного элемента защиты на ФБ.

На рисунке 16 слева раскрыто содержимое блока *PTCR*. Соответствующая сеть ФБ включает блок *PTRC_Base* и блок временной задержки *E_DELAY* для измерения временных интервалов. Диаграмма управления выполнением (ECC) блока *PTRC_Base* приведена на рисунке 16 справа.

Рассмотрим работу блока *PTRC_Base*. При поступлении сигнала на событийный вход *INIT* происходит проверка значения на информационном

входе *Op_General*. Если оно равно *FALSE*, то происходит переход в состояние *S4*, в котором выдаётся выходной событийный сигнал *Out*, а на выход *Tr_generate* подаётся сигнал *Op_General* (который в данный момент равен *FALSE*). Далее происходит ожидание сигнала *INIT* с *Op_General*, равным *TRUE*, после чего происходит переход в состояние *S1*. Если оно равно *TRUE*, то происходит переход в состояние *S1*. В состоянии *S1* происходит подача сигнала на внешний таймер (выходной сигнал *Start_Time*), в котором происходит отсчёт времени.

Далее возможно два варианта развития событий. Если от таймера пришло событие *Time_end*, то на выход подаётся сигнал *Out* вместе с *Tr_generate*, который равен *Op_General* (который равен *TRUE*, или, если сигнал превышения напряжения внезапно пропал, то *FALSE*). Если в блок пришло событие *INIT* вместе с *Op_General*, равным FALSE (это обозначает, что напряжение вернулось в норму), то происходит переход в состояние *S2*, в котором подаётся сигнал останова на внешний таймер (событийный сигнал *Stop_Time*). Независимо от того, дошёл отсчёт таймера до конца или был прерван, происходит переход в начальное состояние.

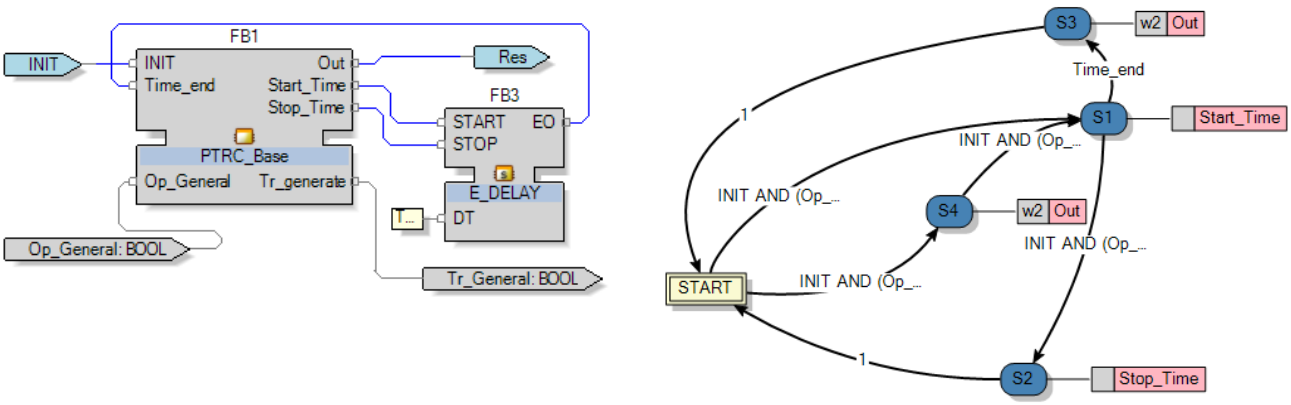


Рисунок 16 - Содержимое ФБ PTRC (слева) и диаграмма ECC ФБ PTRC_Base (справа).

На рисунке 17 приведен скриншот человеко-машинного интерфейса (HMI) имитационной модели распределенной системы защиты от перегрузок для распределительной сети, представленной на рис. 1. Как видно из рис. 4, отображается состояние каждого из четырех *PIED*, входящих в систему. Следует заметить, что для построения HMI использовались CAT-блоки системы NxtStudio.

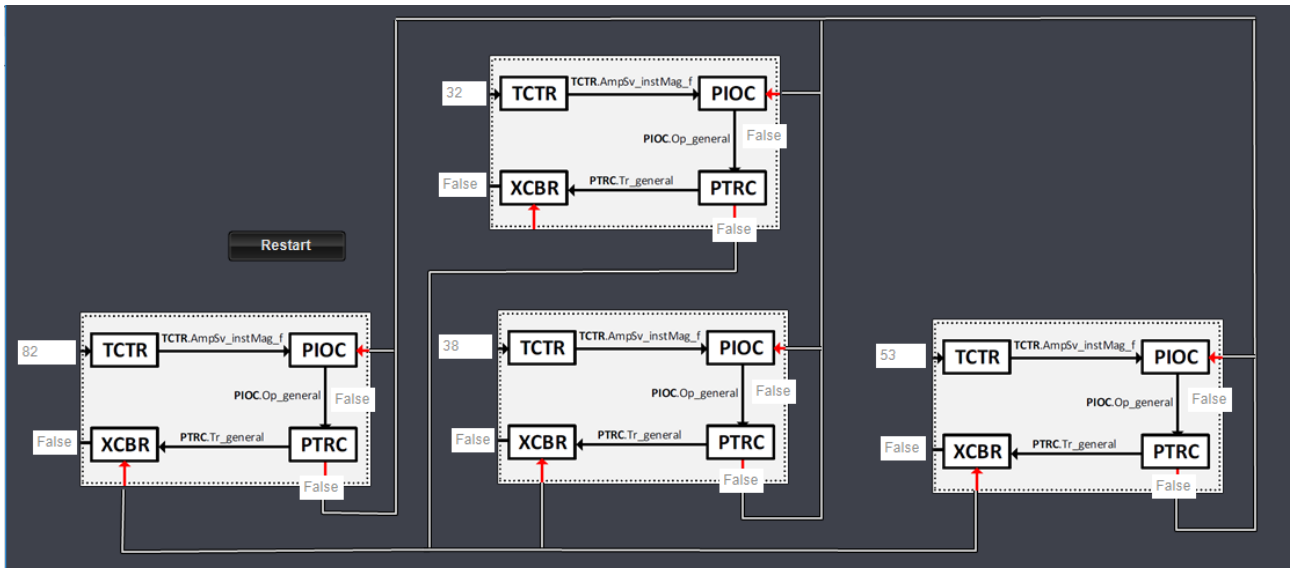


Рисунок 17 - Человеко-машинный интерфейс имитационной модели распределенной системы защиты от перегрузок для распределительной сети на рисунке 14 в системе NxtStudio

2.2 Основные положения методики

Методика автоматической генерации функционально-блочной модели системы защиты от перегрузок для цифровых подстанций МЭК 61850 базируется на концепции сетей Smart Grid.

Трансляция с языка SCL на язык описания функциональных блоков невозможна напрямую, поскольку файл SCL содержит информацию только о структуре подстанции, но нет информации о её управлении. Для этой цели необходимо создание специальной программы-преобразователя, которую можно разработать на основе языка искусственного интеллекта Prolog [29].

Язык Prolog был выбран не случайно, а ввиду его важной особенности: возможности самомодификации программы в процессе исполнения. Таким образом получается, что естественным первым шагом является разработка транслятора из SCL в Prolog. Следующим шагом является разработка преобразователя SCL Prolog в FB Prolog на основе заранее определённых общих правил. Заключительным этапом является перевод результирующего файла на языке Prolog в описания FB на языке XML. После работы всех трех трансляторов и преобразоватедей, полученный файл можно загрузить в одну из систем программирования на основе ФБ МЭК 61499 (например, nxtStudio), с целью проверки его работы и отладки. При успешной проверке, информация из файла может быть загружена на ПЛК.

2.3 Разработка Prolog-представлений исходной и целевой моделей

2.3.1 Prolog-представление конфигурации ЦПС

После работы транслятора исходное представление Smart Grid на языке SCL преобразуется в набор фактов языка Prolog.

Каждый факт может быть одного из двух типов:

1. Экземпляр класса с уникальным именем

Поскольку может быть множество экземпляров одного класса (например класса IED), то необходимо обеспечитъ каждому экземпляру уникальное имя. Для этого, в самом простом случае, можно просто нумеровать по порядку все всреченные реализации класса. В качестве примера возьмём класс EnumVal:

'EnumVal'(a1:CBOpCap_1).

'EnumVal'(a1:CBOpCap_2).

'EnumVal'(a1:CBOpCap_3).

'EnumVal'(a1:CBOpCap_4).

'EnumVal'(a1:CBOpCap_5).

'EnumVal'(a1:CBOpCap_6).

'EnumVal'(a1:CBOpCap_7).

Как можно заметить, все экземпляры отличаются друг от друга только порядковым номером (в каком порядке они встречались в исходном SCL файле).. Основным недостатком данного подхода является то, что невозможно сказать, какому классу более высшего порядка принадлежит данный экземпляр, если просто смотреть только на него.

Более практичным и удобным способом идентификации элементов классов является задание имени на основе наиболее важных атрибутов высших классов (экземпляром подклассов которых является данный объект) и уникального атрибута данного экземпляра. В качестве примера возьмём класс DO:

'DO'(a1:LN0_Beh).

'DO'(a1:LN0_Health).

'DO'(a1:LN0_Mod).

'DO'(a1:LN0_NamPlt).

…

'DO'(a1:LPHDa_Beh).

'DO'(a1:LPHDa_PhyHealth).

'DO'(a1:LPHDa_PhyNam).

…

Уникальное имя каждого экземпляра класса состоит из имени самого высшего класса (LN0 и LPHDa) и уникального имени объекта в пространстве имён этого класса.

2. Соответствие атрибутов объектам класса

Каждый класс (за редким исключением) в SCL имеет набор атрибутов, которые в языке Prolog представляются в виде факта с двумя объектами, построенного в соответствии с рисунком 18.

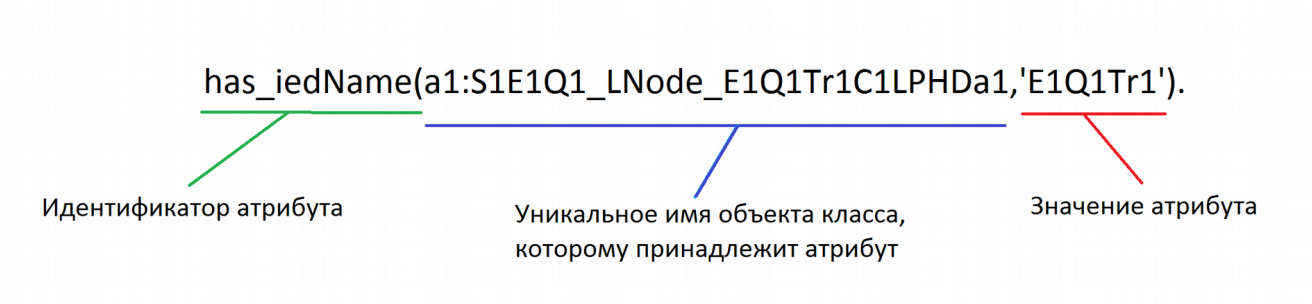


Рисунок 18 - Правило построения соответствия атрибутов и объектов класса.

3. Определение класса, как подкласса другого класса.

Поскольку между классами в SCL установлены отношения, то необходимо их сохранить и в Prolog представлении.

Определение одного класса в качестве подкласса другого происходит аналогично соответствию атрибутов объектам класса. Только вместо Значения атрибута указывается имя подкласса без скобок. Идентификатор атрибута также заменяется на идентификатор подкласса.

Приведём общую таблицу правил преобразования для классов и их атрибутов (Таблица 2).

Таблица 2. Соответствие SCL XML-описаний Prolog представлениям.

SCL	Prolog
<IED name="E1Q1Tr1">	'IED'(a1:E1Q1Tr1).
<Services>	has_name(a1:E1Q1Tr1,'E1Q1Tr1').
<AccessPoint name="S1">	'AccessPoint'(a1:E1Q1Tr1_S1).

</AccessPoint> </IED>	has_name(a1:E1Q1Tr1_S1,'S1'). has_AccessPoint(a1:E1Q1Tr1,a1:E1Q1Tr1_S1) .

В таблице 2 представлена последовательность преобразований при конвертации из SCL в Prolog. Рассмотрим их поподробнее:

1. Первое, что встречается в SCL файле это тег IED. Поскольку все IED устройства в рамках одной сети всегда имеют уникальные имена, то можно создать объект в Prolog типа IED и значением уникального имени, которое необходимо сохранить для использования при идентификации атрибута name. Что и делается в первой строчке Prolog. Также мы сохраняем это значение.

2. Созданный объект не имеет ничего. Встретив атрибут name у тега IED, записываем вторую строчку в Prolog файл, которая свяжет созданный объект с значением атрибута name (которое ранее было сохранено).

3. В данном примере тег Services не имеет ни атрибутов, ни дочерних тегов, поэтому он не транслируется в Prolog.

4. Далее идёт тег AccessPoint. При его трансляции в Prolog также используется уникальное имя полученное из соединения имени высшего тега (IED) и значения атрибута name у самого тега AccessPoint. Поскольку в рамках одного IED не может быть 2 тега AccessPoint с одинаковым значением атрибута name, такой способ создания имени объекта однозначно идентифицирует его в сети Smart Grid.

5. Четвёртая строка добавляет имя к созданному объекту, аналогично тому, как это происходило в теге IED.

6. Последней строчкой Prolog представления в данном примере является факт, который связывает ранее созданные экземпляры IED и AccessPoint (объект AccessPoint является подклассом обекта IED).

7. Последними в SCL файле идут закрывающие теги, которые при трансляции игнорируются.

2.3.2 Prolog-представление функциональных блоков МЭК 61850

Prolog-представление ФБ есть набор фактов (и правил) языка Prolog, которые определяют отдельные элементы ФБ (например, событийные входы и выходы, входные и выходные переменные, WITH-связи, состояния и переходы диаграмм ECC, привязку EC-акций к EC-состояниям, алгоритмы, компонентные ФБ в составном ФБ, событийные и информационные связи). При описании фактов будут использоваться следующие обозначения частей ФБ:

ei – событийный вход (имя);

eo – событийный выход (имя);

di – входная переменная (имя);

do – выходная переменная (имя);

v – внутренняя переменная (имя);

dtype – тип переменной (целый, булевский, строковый и т.д.)

s – состояние диаграммы ECC (имя);

gc – сторожевое условие EC-перехода (идентификатор);

fbt –тип ФБ, в котором находятся описываемые артефакты (имя);

fbc – компонентный ФБ внутри составного ФБ (имя);

code – исходный код сторожевого условия или алгоритма (обычно представляется в виде строки на языке ST).

Интерфейс ФБ-типа fbt представляется в виде следующих фактов языка Prolog:

event_input (fbt, ei1). /* Событийный вход ei1 */

event_output (fbt, eo1). /* Событийный выход eo1 */

data_input (fbt, di1, dtype1). /* Входная переменная di1 типа dtype1 */

data_output (fbt, do1, dtype1). /* Выходная переменная do1 типа dtype1 */

with_in (fbt ,ei1, di1). /* WITH-связь во входном интерфейсе ФБ */

with_out (fbt, eo1, do1). /* WITH-связь в выходном интерфейсе ФБ */

Сеть ФБ внутри составного ФБ-типа fbt представляется в виде следующих фактов языка Prolog:

fb_component (fbt, fbc, fbct). /* Компонентный ФБ fbc типа fbct */

event_conn (fbt, fb1, ev1, fb2, ev2). /* Событийная связь, имеющая источник (fb1, ev1) и приемник (fb2, ev2), где fb1 и fb2 – это компонентный ФБ или «оболочка» составного ФБ; ev1 и ev2 – это событийные входы или выходы компонентного ФБ или «оболочки» */

data_conn (fbt, fb1, d1, fb2, d2). /* Информационная связь, имеющая источник (fb1, d1) и приемник (fb2, d2), где d1 и d2 – это информационные входы или выходы компонентного ФБ или «оболочки» */

Для описания сервисных интерфейсных ФБ предлагаются собственный набор фактов (пример ниже):

service (fbt1,int2,int3). /* Точки назначения int1, int2 сервисных последовательностей */

sequence (fbt1,seq4). /* Последовательность seq4 в ФБ типа fbt1 */

primitive (fbt1,seq4,2,in,int3,ei5,[arg6]). /* Транзакция #2 из последовательности seq4: входной сигнал ei5 с параметром arg6 на интерфейс int3 */

На рисунке 19 приведено графическое представление стандартного базисного ФБ “D-триггер”: интерфейс (слева) и диаграмма ECC (справа), входящего в библиотеку инструментальной системы проектирования FBDK [6]:

Рисунок 19 - Пример ФБ D-триггера.

Ниже приведено Prolog-представление диаграммы ECC.

ec_state(e_d_ff, q0). /* Начальное состояние qo */

ec_state(e_d_ff, set). /* Состояние set, в котором производится установка */

ec_state(e_d_ff, reset). /* Состояние reset, в котором производится сброс */

ec_tran(e_d_ff, q0, clk, gc1, set). /* Переход q0->set */

ec_tran(e_d_ff, set, clk, gc2, reset). /* Переход set->reset */

ec_tran(e_d_ff, reset, clk, gc1, set). /* Переход reset->set */

guard_cond(e_d_ff,gc1,[d]). /* Сторожевое условие gc1 */

guard_cond(e_d_ff,gc2,[d]). /* Сторожевое условие gc2 */

algorithm(e_d_ff, latch, [d], [q]). /* Алгоритм latch */

ec_action(e_d_ff, set, 1, latch, eo). /* (Первая) EC-акция в состоянии set */

ec_action(e_d_ff, reset, 1, latch, eo). /*(Первая) EC-акция в состоянии reset*/

Для представления системной конфигурации, приложений, устройств и ресурсов предлагаются следующие предикаты: system, fb_netwiork, device_type, has_argument_d, darg_value, has_argument_r, rarg_value, device, application, resource_type, has_dev_FB, has_app_FB, has_res_FB, refers_ad, refers_ar.

2.4 Разработка правил трансформации исходной модели в целевую модель

2.4.1 Графовые правила трансформации

В данном разделе будут с помощью графов описаны правила преобразования из SCL-Prolog в FB-Prolog. Количество правил может быть любым, в зависимости от поставленной задачи. В данной работе это генерация системы защиты, поэтому будут генерироваться 4 типа ФБ:

- Для трансформатора (Current transformer)

- Для прерывателя (Circuit breaker)

- Для силового трансформатора (Power transformer)

-Для системы защиты (Protection)

На рисунке 20 показаны правила для силового трансформатора.

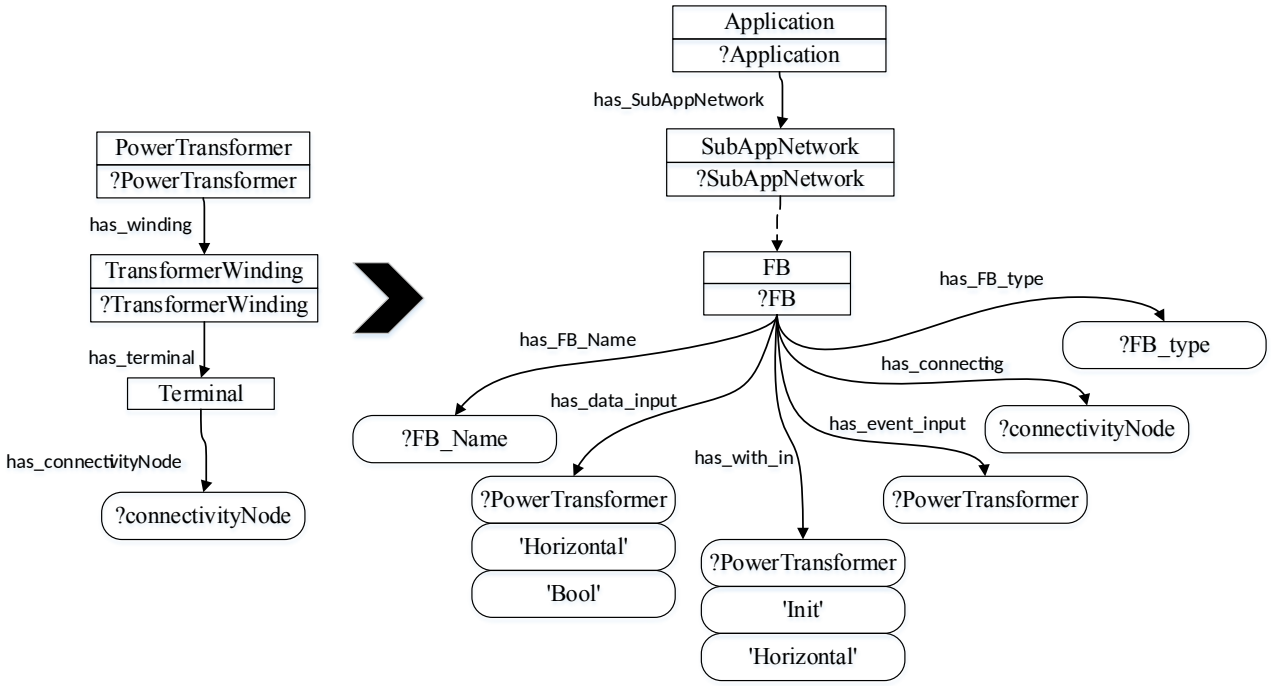


Рисунок 20 - Абстрактное представление правила преобразования для Power transformer.

В левой части приведено представление SCL-Prolog в виде графа. Верхняя часть отображает название факта Prolog, а нижняя – уникальное имя, которое представляет собой объект факта (оно представлено с знаком ? в начале, что обозначает что это переменная). На дугах написано название правил, которые реализованы для определения взаимоотношения между объектами.

В правой части показано графовое представление сгенерированных правил. Значения FB_Name и FB_type определяются в ходе выполнения программы и представляют собой параметры базисного ФБ. В факте has_data_input перечисляются все параметры нового факта, а именно – уникальное имя ФБ, совпадающее с именем в левой стороне, название данного информационного входа и тип данных на нём. В факте has_with_in определяется также имя ФБ, а затем названия информационного и событийного входов, которые необходимо объединить. Факт has_event_input представляет собой название входного события. Фатов has_connecting будет минимум 2, поскольку любое устройства в сети имеет соединения – для входа и выхода

электроэнергии. Значение connectivityNode также совпадат с соответствующим значением в левой части.

На рисунке 21 показаны правила для трансформатора.

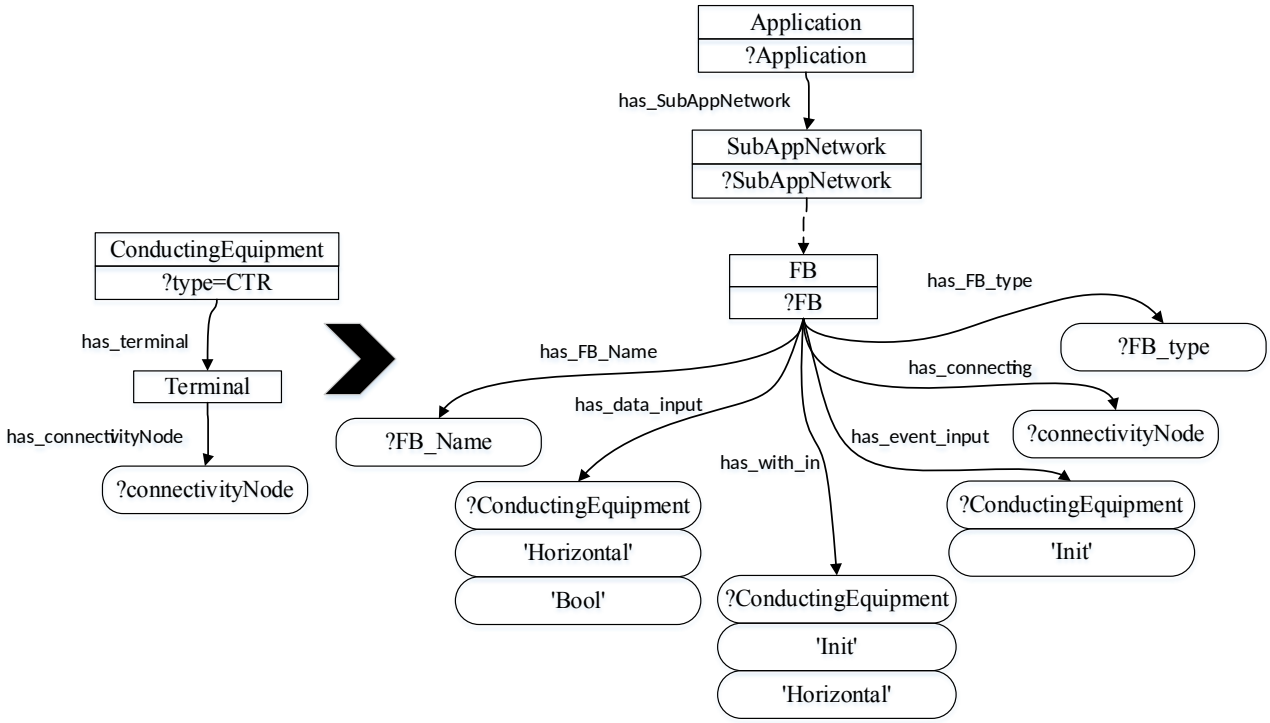


Рисунок 21 - Абстрактное представление правила преобразования для Current transformer.

Принцип построения правил такой-же, как и у Power transformer (в левой части исходное представление Prolog, а с правой – сгенерированное представление на основе правил)

На рисунке 22 показаны правила для прерывателя.

Данное правило немного отличается от представленных ранее ввиду того, что 2 событийных входа, 2 информационных входа и 2 соединения между информационным и событийным входом. Данные, относящиеся к разным входам, разделены жирной горизонтальной чертой.

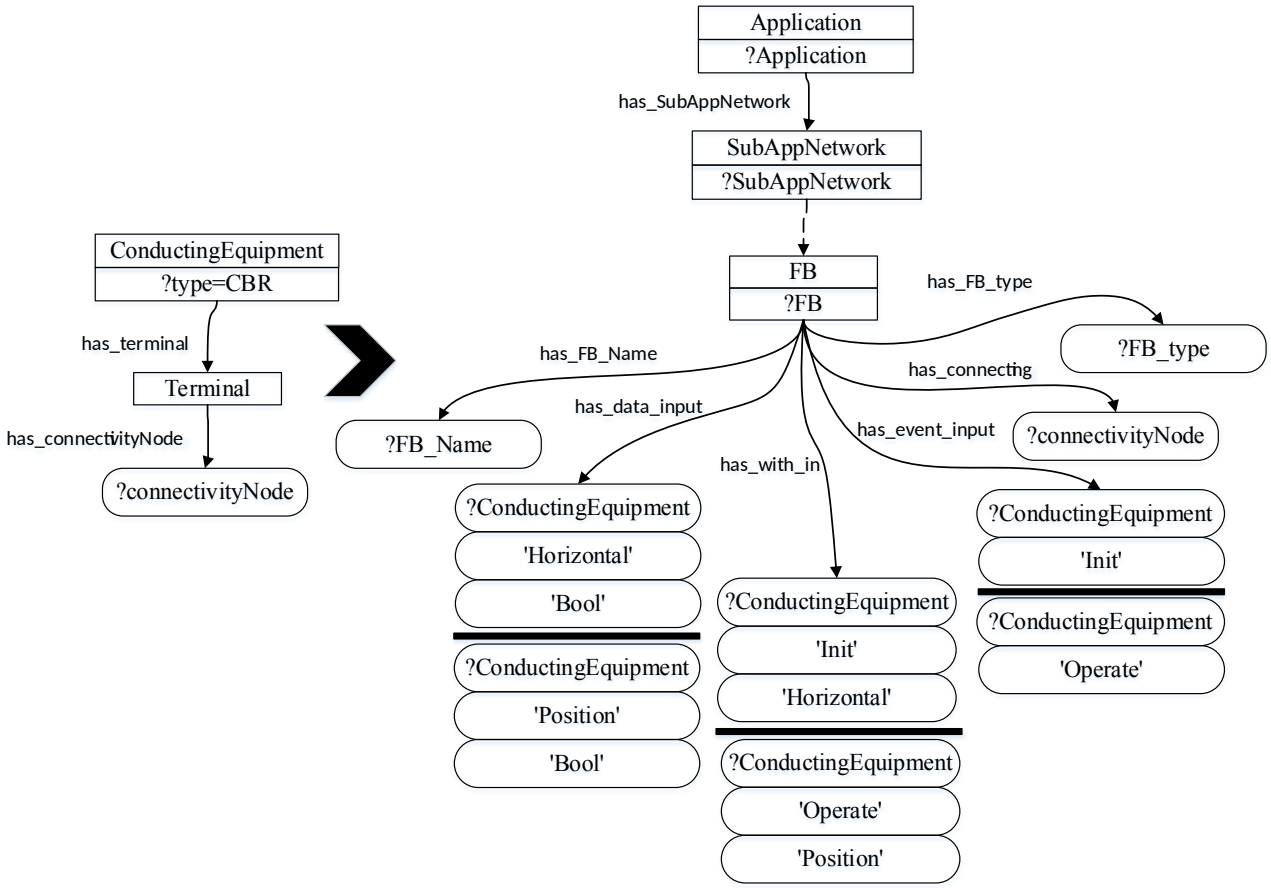


Рисунок 22 - Абстрактное представление правила преобразования для Circuit breaker.

На рисунке 23 показаны правила для системы защиты.

Данное правило отличается по типу построения только наличие помимо входных данных и событий также и выходные.

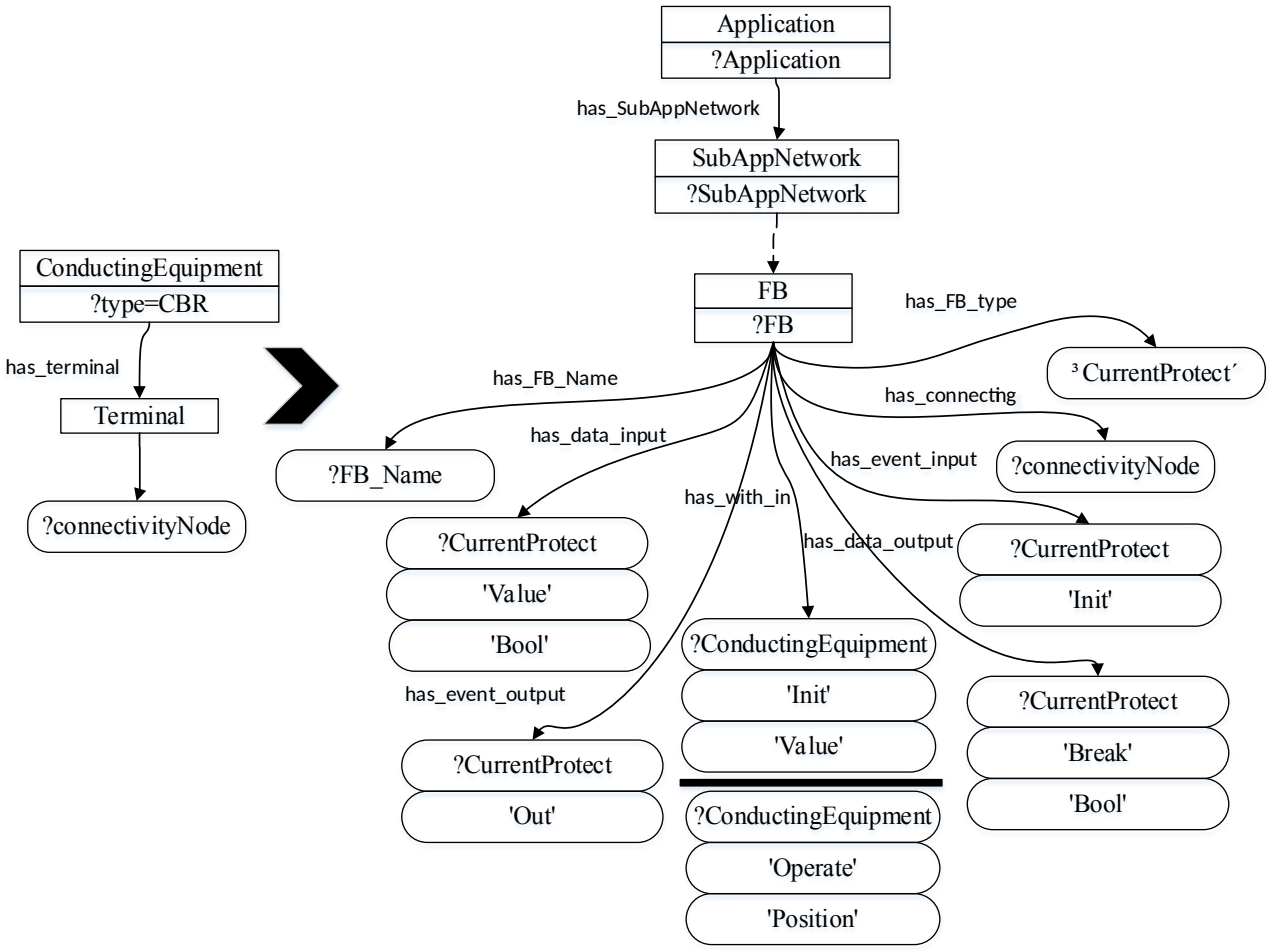


Рисунок 23 - Абстрактное представление правила преобразования для схемы защиты

2.4.2 Правила преобразования SCL-Prolog в FB-Prolog

Описанные ранее правила преобразования SCL-Prolog в FB-Prolog представлены на Prolog в следующем виде:

rule_1:-'ConductingEquipment'(Name), has_class(Name,'CTR'), assert(event_input(Name, 'Init')), assert(data_input(Name, 'Horizontal', 'bool')), assert(with_in(Name,'Init','Horizontal')), fail.

rule_1.

rule_2:-'PowerTransformer'(Pt), has_name(Pt,Name),assert(event_input(Name, 'Init')), assert(data_input(Name, 'Horizontal', 'Bool')), assert(with_in(Name,'Init','Horizontal')), fail.

rule_2.

rule_3:-'ConductingEquipment'(Name), has_class(Name,'CBR'),assert(event_input(Name, 'Init')), assert(event_input(Name, 'Operate')), assert(data_input(Name, 'Position', 'bool')), assert(data_input(Name, 'Horizontal', 'bool')), assert(with_in(Name,'Init','Horizontal')), assert(with_in(Name,'Operate','Position')), fail.

rule_3.

rule_4:-'ConductingEquipment'(Name), has_class(Name,'CTR'), has_Terminal(Name,Terminal), has_connectivityNode(Terminal,Connode),assert(connecting(Name, Connode)), fail.

rule_4.

rule_5:-'ConductingEquipment'(Name), has_class(Name,'CBR'), has_Terminal(Name,Terminal), has_connectivityNode(Terminal,Connode),assert(connecting(Name, Connode)), fail.

rule_5.

rule_6:-'PowerTransformer'(Pt),has_TransformerWinding(Pt,Wind),has_Terminal(Wind,Terminal),has_connectivityNode(Terminal,Connode),assert(connecting(Pt, Connode)), fail.

rule_6.

rule_7:-'ConductingEquipment'(Name), has_class(Name,'CBR'),assert(event_input('S', Name, 'Init')), assert(data_input('S', Name, 'Value')), assert(with_in('S', Name,'Init','Value')), assert(data_output('S', Name, 'Break')),assert(event_output('S', Name, 'Out')) assert(with_in('S', Name,'Break','Out')))fail.

rule_7.

rule_0:-rule_1,rule_2,rule_3,rule_4,rule_5,rule_6, rule_7, tell('Complete.txt'), listing(connecting), listing(has_class), listing('PowerTransformer'), listing(event_input), listing(data_input), listing(data_ output), listing(event _ output), listing(with_in),told.

rule_0.

Правило rule_1 служит для добавления экземпляра ФБ типа Current transformer.

Правило rule_2 служит для добавления экземпляра ФБ типа Power transformer.

Правило rule_3 служит для добавления экземпляра ФБ типа Circuit breaker.

Правила rule_4 - rule_6 служит для установление, с какими узлами должен быть связан данный ФБ.

Правило rule_7 Реализует защиту. Исходные факты точно такие-же, как и у Circuit breaker, но в результирующих добавлена константа ‘S’ что-бы их различать.

Правило rule_0 является самым главным правилом, поскольку оно поочерёдно вызывает все остальные правила, а затем создаёт новый файл, в который будет записываться результат. Функция listing принимает в качестве аргумента название факта, который должен быть выведен, поэтому она вызывается несколько раз. Последним этапом является закрытие файла.

Каждое правило заканчивается командой fail, которая запрещает Prolog возвращаться на предыдущий шаг. Она необходима, чтобы программа работала линейно в данном случае.

2.5 Разработка программного обеспечения для поддержки методики

2.5.1 Общая структура программного обеспечения

Для автоматизации генерации функционально-блочной модели системы защиты от перегрузок для ЦПС предлагается технологическая цепочка проектирования, приведенная на рисунке 24. В отличие от [30] эта цепочка упрощена, поскольку в ней отсутствует онтологическое представление системы и преобразования, связанные с ним. В качестве внутреннего рабочего представления используется язык Prolog. Как видно из рис. 5 исходное описание ЦПС представлено на языке SCL. В дальнейшем это описание транслируется в Prolog-базированное описание ЦПС (уже включающее схему защиты), которое, в свою очередь, преобразуется в описание управляющих ФБ на языке Prolog. Для преобразования используется набор правил, также написанных на языке Prolog.

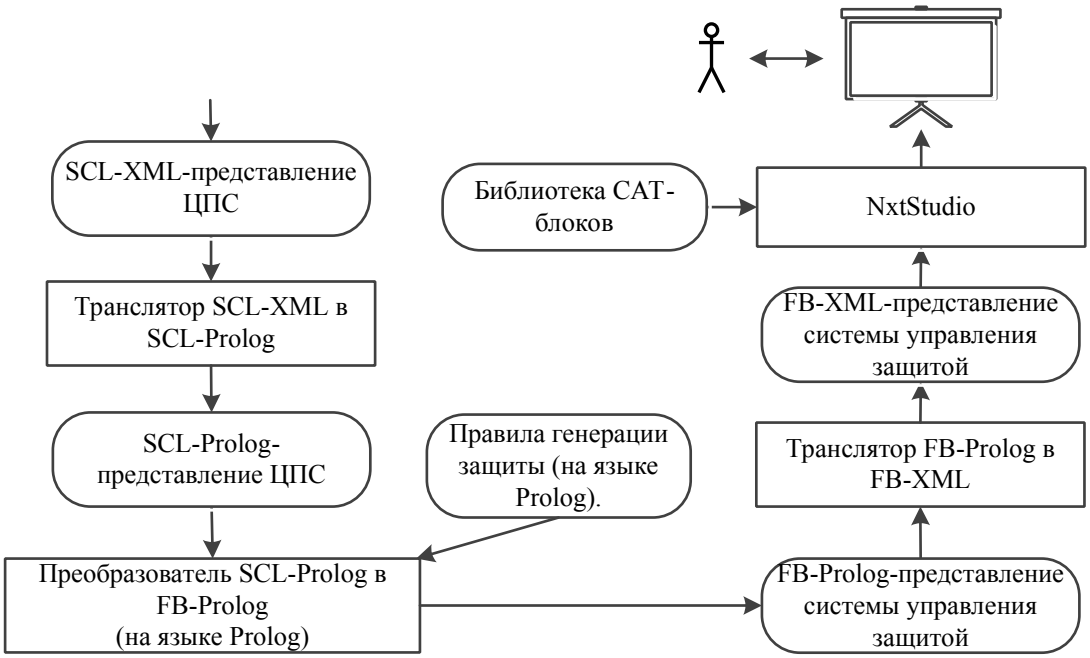


Рисунок 24 - Структура программного обеспечения для генерации функционально-блочных моделей защиты от перегрузок

2.5.2 Транслятор SCL XML-описания в SCL Prolog-представление ЦПС

Транслятор состоит из 3 основных частей:

1. Парсер XML документа

В качестве парсера был использован встроенный в c# парсер XmlNode.

2. Функция распознавания текущего тега и записи соответствующей информации в Prolog файл.

Данная функция является основной, поскольку именно в ней происходит трансляция из SCL в Prolog.

3. Сортировщик

Сортировщик является дополнительным приложением к транслятору и работает (сортирует) уже полностью готовый Prolog файл. Он нужен для

структуризации данных в выходном файле, поскольку без этого будет трудно найти нужную информацию в Prolog представлении.

В коде транслятора данные структурируются особым образом:

1. Имеется отдельная строковая переменная для каждого из возможных тегов, которая хранит в себе имя последнего созданного в Prolog объекта данного тега.

2. Имеется одельный счётчик для подсчёта количества встреченных тегов одного типа. Счётчик количества экземпляров любого тега сбрасывается, если встречается тег (класс) более высшего порядка.

3. В трансляторе также используются переменные для хранения промежуточных значений (атрибутов). Эти переменные являются универсальными и могут быть использованы при трансляции любого класса.

Полная структура транслятора и все его модули показаны на Uml диаграмме классов (Рисунок 25).

Рисунок 25 - UML диаграмма транслятора SCLtoProlog.

При запуске транслятора первым работает класс Global. В нём происходит инициализация путей до исходного SCL файла и до желаемого Prolog файла.

Класс DisplayTree является парсером, который из потока данных считывает новый тег. В данном классе реализован обход в глубину (т.е. сначала происходит проход по одной ветви от “корня” дерева тегов до “листа”, затем возврат к последней развилке и проход до другого “листа” и т.д.).

В классе Format происходит определение типа поступившего тега и запись его представления в Prolog файл.

Разработанный транслятор является легко масштабируемым ввиду того, что список тегов и их определение подгружаются из базы тегов BaseTag, которую достаточно легко модифицировать с целью расширения или уменьшения возможностей транслятора.

Класс SpecialVariable содержит переменные, которые хранят промежуточные значения и могут быть использованы любым тегом как для записи, так и для чтения.

При составлении имени нового объекта в Prolog представлении часто возникает ситуация, когда нужно использовать сокращения от названия классов SCL. Это необходимо для уменьшения длины строки в Prolog файле. Для этой цели используется класс DeleteWord, который не является жизненно важным для работы транслятора, как и два последних класса: DeleteAllData и Sort (вместе с подклассом SortRule).

Класс DeleteAllData позволяет очистить значения всех переменных, используемых в трансляторе, т.е. транслятор начнёт работу с нуля.

Класс Sort является вспомогательным и работает автономно от остальной части программы. Он необходим для сортировки выходного Prolog файла. Алгоритм сортировки записан в классе SortRule и может быть изменён программистом в любой момент. Работу всего модуля Sort может отключить и сам пользователь с главного окна программы (Рисунок 26).

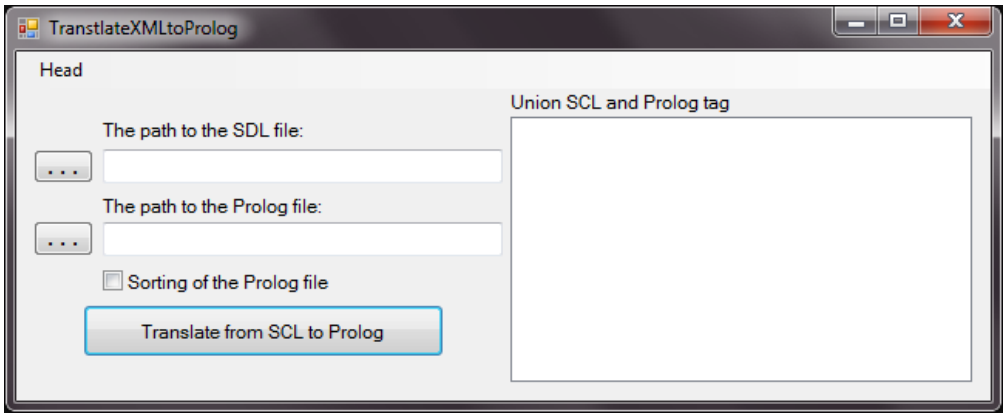


Рисунок 26 - Главное окно программы.

Транслятор имеет простой человеко-машинный интерфейс:

1) Поле для ввода пути до исходного SCL файла.

2) Ввод пути до места, где должен быть сохранён созданный Prolog файл.

3) Галочка, определяющая, нужно ли проводить сортировку выходного файла (использовать модуль Sort).

4) Кнопка для запуска работы транслятора

После окончания работы программы в окошке справа появится дерево двойных тегов (сначала записано имя тега в SCL файле, а через тире – представление этого объекта класса в Prolog файле). Пример дерева соответсвия на рисунке 27.

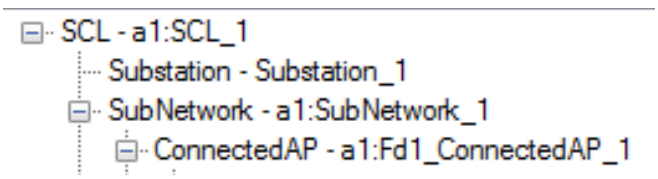


Рисунок 27 - Дерево соответсвия исходных тегов объектам в Prolog файле.

2.5.3 Преобразователь SCL Prolog-представления в FB Prolog-представление

Для проведения данного преобразования необходимо загрузить в Prolog программу внешние правила. Количество правил неограниченно. Главное, что бы в результате получилось FB-Prolog представление, а не просто набор фактов. Далее представлены примеры более сложных правил, которые можно использовать в при работе с большими сетями Smart Grid

```
    /* RULE_0_1 - Creating DataType-EnumeratedType using EnumType
*/

    rule_0_1:-

    'EnumType'(Enumt),

    has_id(Enumt, EnumtypeID),

    /* -> */

    createInstance('DataType', EnumtypeID),

    createLink(associate, Enumt, EnumtypeID),

    createLink(has_DataType_Name, EnumtypeID, EnumtypeID),

    atom_concat(EnumtypeID, '_EnumType' , Etype),

    createInstance('EnumeratedType', Etype),

    createLink(has_EnumeratedType, EnumtypeID, Etype),

    fail.

    rule_0_1.


    /* RULE_0_2 - Creating EnumeratedValue using EnumVal-elements
*/
```

```
rule_0_2:-
'EnumType'(Enumtype),
'DataType'(Datatype),
associate(Enumtype, Datatype),
'EnumeratedType'(Etype),
has_EnumeratedType(Datatype, Etype),
'EnumVal'(Enumval),
has_EnumVal(Enumtype, Enumval),
has_ord(Enumval, Enumvalord),
has_Value(Enumval, Enumvalue),
/* -> */
atom_concat(Enumvalord, '_', N1),
atom_concat(N1, Enumvalue, Eval),
createInstance('EnumeratedValue', Eval),
createLink(has_EnumeratedValue, Etype, Eval),
createLink(has_EnumeratedValue_Name, Eval, Enumvalord),
createLink(has_EnumeratedValue_Comment, Eval, Enumvalue),
fail.
rule_0_2.
/* RULE_1_1 - Creating StructureTypes using DATypes */
rule_1_1:-
'DAType'(Datype),
has_id(Datype, DatypeName),
/* -> */
```

```
createInstance('DataType', DatypeName),
createLink(associate, Datype, DatypeName),
createLink(has_DataType_Name, DatypeName, DatypeName),
atom_concat(DatypeName, '_StructureType', St),
createInstance('StructureType', St),
createLink(has_StructureType, DatypeName, St),
fail.
rule_1_1.
```

2.5.4 Транслятор Prolog-представления функциональных блоков в их XML-описание для проектов nxtStudio

Главное окно транслятора представлено на рисунке 28.

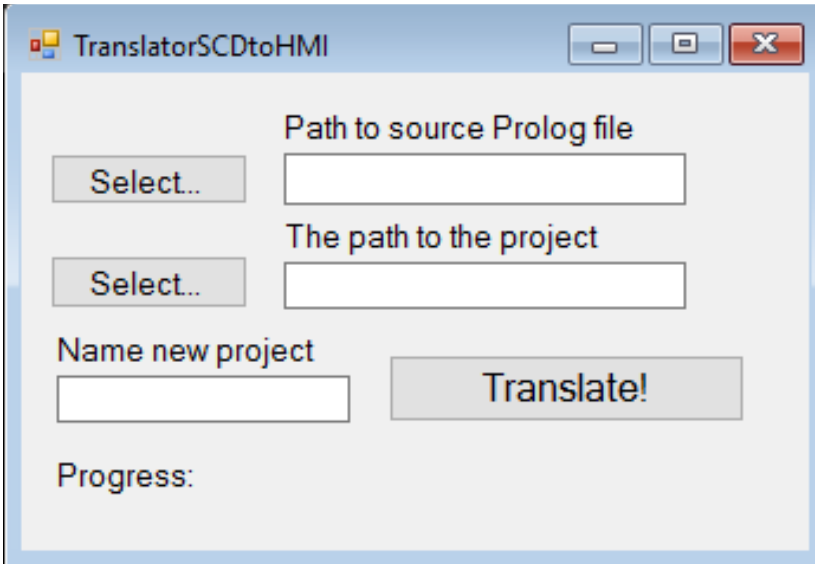


Рисунок 28 - Главное окно транслятора.

Трансляция происходит в несколько этапов:

1) Чтение исходного Prolog-представления ФБ и перевод его во внутреннее представление транслятора.

2) Открытия проекта nxtStudio запись полученных ФБ в приложение и на устройство. Поскольку данный транслятор на подразумевает получение HMI, то прорисовка на canvas не нужна.

3) Соединение ФБ между собой в проекте и подключение их к блоку инициализации.

Для упрощения работы с транслятором, шаблон пустого проекта для nxtStudio уже вшит в него. Таким образом необходимо только ввести имя будущего проекта и транслятор создаст все необходимые файла для его запуска после трансляции.

2.5.5 Пример использования

В качестве примера возьмем Smart Grid, показанную на рисунке 29.

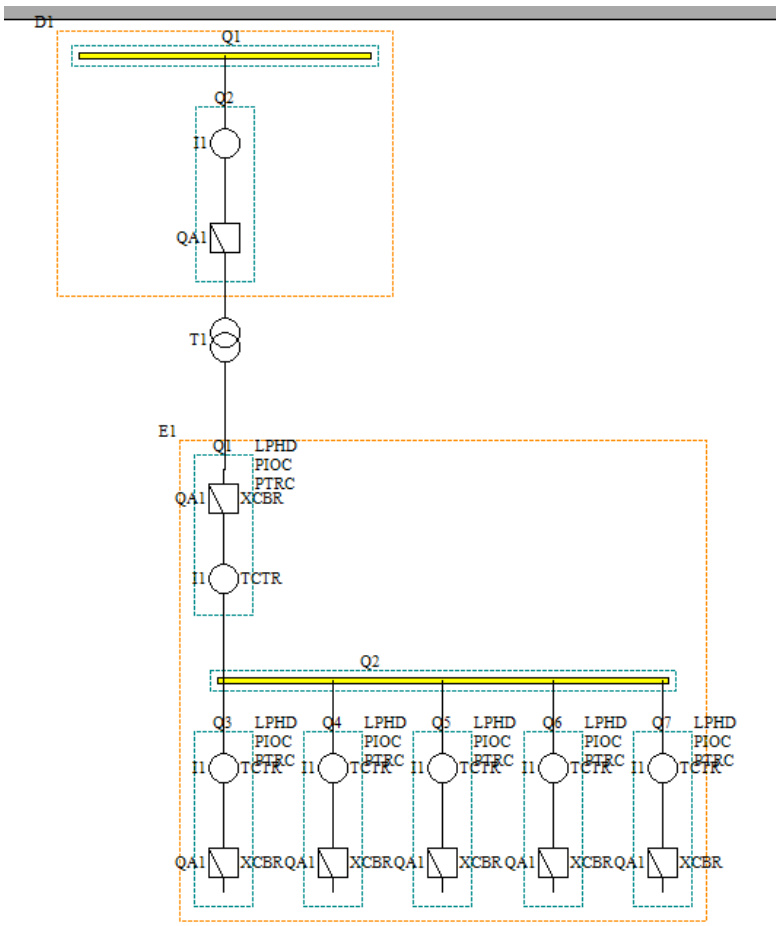


Рисунок 29 - Демонстранионная сеть Smart Grid.

Представленная сеть состоит из зоны D1 с напряжением 220В и зоны E1 с напряжением 132В, а также трансформатора напряжения T1.

Электричество поступает в зону D1, на линию Q1, затем проходит блок защиты Q2 и идёт на трансформатор напряжения.

После трансформатора, электрический ток попадает в зону E1 на блок защиты Q3.

Далее происходит распределение электричества на 5 отдельных линий с прохождением проверки на допустимое напряжения в каждой из них.

Описания данной сети на языке SCL в SCD файле выглядит следующим образом (Рисунок 30)

```
<?xml version="1.0" encoding="utf-8"?>
<!--This file is generated using Kalkitech SCL Manager IEC61850 Configurati
<SCL version="2007" revision="A" xmlns="http://www.iec.ch/61850/2003/SCL">
  <Header id="" version="4.2.0" revision="" toolID="Kalkitech SCL Manager" 
  <Substation name="S1">
    <PowerTransformer sxy:x="198" sxy:y="200" name="T1" xmlns:sxy="http://w
    <VoltageLevel sxy:x="10" sxy:y="10" name="D1" xmlns:sxy="http://www.iec
      <Voltage unit="V" multiplier="k">220</Voltage>
      <Bay sxy:x="10" sxy:y="10" name="Q1">
        <ConnectivityNode name="B1" pathName="S1/D1/Q1/B1" />
      </Bay>
      <Bay sxy:x="10" sxy:y="45" name="Q2">
        <ConductingEquipment sxy:x="178" sxy:y="15" name="I1" type="CTR">
        <ConductingEquipment sxy:x="178" sxy:y="75" name="QA1" type="CBR">
        <ConnectivityNode name="L1" pathName="S1/D1/Q2/L1" />
        <ConnectivityNode name="L2" pathName="S1/D1/Q2/L2" />
      </Bay>
    </VoltageLevel>
    <VoltageLevel sxy:x="9" sxy:y="284" name="E1" xmlns:sxy="http://www.iec
      <Voltage unit="V" multiplier="k">132</Voltage>
      <Bay sxy:x="10" sxy:y="10" name="Q1">
      <Bay name="Q2">
        <Bay name="Q3">
```

Рисунок 30 - Описания сети Smart Grid на языке SCL.

После проведения трансляции конечный Prolog файл выглядит следующим образом (без сортировки рисунок 31, с сортировкой – рисунок 32).

```
13  has_Substation('a1:SCL','Substation_1').
14  has_name('a1:SCL','Substation_1').
15  'Communication'('a1:Communication_1').
16  has_Communication('a1:SCL','a1:Communication_1').
17  'SubNetwork'('a1:SubNetwork_1').
18  has_SubNetwork('a1:Communication_1','a1:SubNetwork_1').
19  has_name('a1:SubNetwork_1','subnetwork1').
20  has_type('a1:SubNetwork_1','8-MMS').
21  has_Text('a1:SubNetwork_1','Station bus').
22  'BitRate'('a1:BitRate_1').
23  has_BitRate('a1:SubNetwork_1','a1:BitRate_1').
24  has_unit('a1:BitRate_1','b/s').
25  has_Value('a1:BitRate_1','10').
26  'ConnectedAP'('a1:E1Q3Fd1_ConnectedAP_1').
27  has_ConnectedAP('a1:SubNetwork_1','a1:E1Q3Fd1_ConnectedAP_1').
28  has_iedName('a1:E1Q3Fd1_ConnectedAP_1','E1Q3Fd1').
29  has_apName('a1:E1Q3Fd1_ConnectedAP_1','S1').
30  'Address'('a1:E1Q3Fd1_ConnectedAP_1_Address_1').
31  has_Address('a1:E1Q3Fd1_ConnectedAP_1','a1:E1Q3Fd1_ConnectedAP_1_Address_1').
32  'P'('a1:E1Q3Fd1_Address_1_P_1').
33  has_P('a1:E1Q3Fd1_ConnectedAP_1_Address_1','a1:E1Q3Fd1_Address_1_P_1').
34  has_type('a1:E1Q3Fd1_Address_1_P_1','IP').
35  has_Value('a1:E1Q3Fd1_Address_1_P_1','10.0.0.3').
36  'P'('a1:E1Q3Fd1_Address_1_P_2').
```

Рисунок 31 - Выходной Prolog файл без использования сортировки.

```
'AccessPoint'(a1:E1Q1Tr1_S1).
'AccessPoint'(a1:E1Q3Fd1_S1).
'AccessPoint'(a1:E1Q4Fd2_S1).
'AccessPoint'(a1:E1Q5Fd3_S1).
'AccessPoint'(a1:E1Q6Fd4_S1).
'AccessPoint'(a1:E1Q7Fd5_S1).
'Address'(a1:E1Q1Tr1_ConnectedAP_6_Address_1).
'Address'(a1:E1Q1Tr1_ConnectedAP_6_GSE_1_Address_1).
'Address'(a1:E1Q1Tr1_ConnectedAP_6_GSE_2_Address_1).
'Address'(a1:E1Q1Tr1_ConnectedAP_6_GSE_3_Address_1).
'Address'(a1:E1Q1Tr1_ConnectedAP_6_GSE_4_Address_1).
'Address'(a1:E1Q1Tr1_ConnectedAP_6_GSE_5_Address_1).
'Address'(a1:E1Q1Tr1_ConnectedAP_6_GSE_6_Address_1).
'Address'(a1:E1Q3Fd1_ConnectedAP_1_Address_1).
'Address'(a1:E1Q3Fd1_ConnectedAP_1_GSE_1_Address_1).
'Address'(a1:E1Q4Fd2_ConnectedAP_2_Address_1).
'Address'(a1:E1Q4Fd2_ConnectedAP_2_GSE_1_Address_1).
'Address'(a1:E1Q5Fd3_ConnectedAP_3_Address_1).
'Address'(a1:E1Q5Fd3_ConnectedAP_3_GSE_1_Address_1).
'Address'(a1:E1Q6Fd4_ConnectedAP_4_Address_1).
'Address'(a1:E1Q6Fd4_ConnectedAP_4_GSE_1_Address_1).
'Address'(a1:E1Q7Fd5_ConnectedAP_5_Address_1).
'Address'(a1:E1Q7Fd5_ConnectedAP_5_GSE_1_Address_1).
'BDA'(a1:AnalogueValue_1_f).
'Bay'(a1:S1D1Q1).
'Bay'(a1:S1D1Q2).
```

Рисунок 32 - Выходной Prolog файл после проведения сортировки.

Следующим этапом является трансляция в FB-Prolog представление. На рисунке 33 показан результат трансляции в сокращенном виде, поскольку в полном варианте весь экран занимает набор фактов connection.

```
:- dynamic connecting/2.

connecting('S1_D1_Bay_2_ConductingEquipment_1', 'S1/D1/Q1/B1').
connecting('S1_D1_Bay_2_ConductingEquipment_1', 'S1/D1/Q2/L1').
connecting('S1_E1_Bay_1_ConductingEquipment_2', 'S1/E1/Q1/L1').
connecting('S1_E1_Bay_1_ConductingEquipment_2', 'S1/E1/Q2/B2').


connecting('S1_PowerTransformer_1', 'S1/D1/Q2/L2').
connecting('S1_PowerTransformer_1', 'S1/E1/Q1/L3').

has_class('S1_D1_Bay_2_ConductingEquipment_1', 'CTR').
has_class('S1_D1_Bay_2_ConductingEquipment_2', 'CBR').
has_class('S1_E1_Bay_1_ConductingEquipment_1', 'CBR').
has_class('a1:E1Q1Tr1_Address_1_P_1', 'IP').
has_class('a1:E1Q1Tr1_Address_1_P_2', 'IP-SUBNET').
has_class('a1:E1Q1Tr1_Address_1_P_3', 'IP-GATEWAY').
has_class('a1:E1Q1Tr1_Address_1_P_4', 'OSI-TSEL').

has_class('a1:LLN01_1_DO1', myMod).
has_class('a1:LLN01_1_DO2', myBeh).
has_class('a1:LLN01_1_DO3', myHealth).
has_class('a1:LLN01_1_DO4', myNamPlt).
has_class('a1:LLN01_2_DO1', myBeh).
has_class('a1:LLN01_2_DO2', myPhyNam).
has_class('a1:LLN01_2_DO3', myPhyHealth).
```

Рисунок 33 - Выходной Prolog файл после проведения трансляции в FB-Prollog.

Последним этапом является трансляция полученного представления в сеть ФБ. Результат трансляции на рисунке 34.

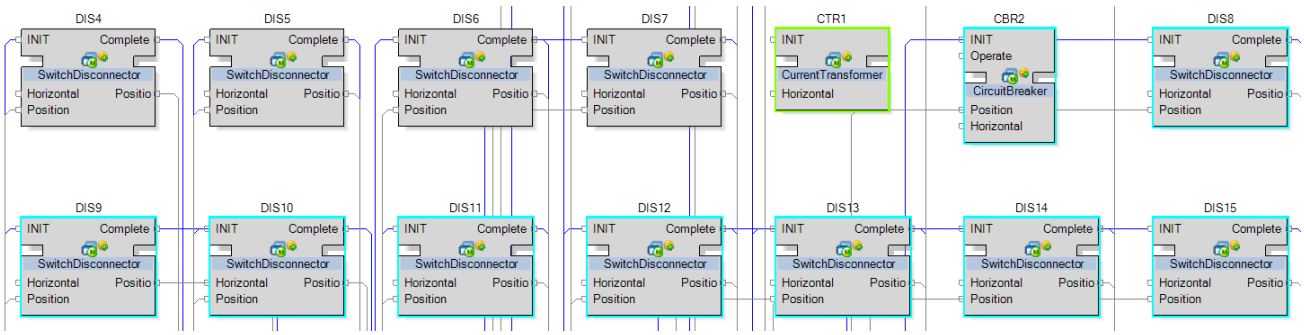


Рисунок 34 - Результирующая сеть ФБ.

2.6 Выводы

1) Предложена методика автоматической генерации функционально-блочной модели системы защиты от перегрузок для цифровых подстанций МЭК 61850, которая отличается выделением главного элемента защиты, отключение которого выводит из строя всю систему, что позволяет

использовать алгоритмы достижения консенсуса вследствие чего система будет крайне устойчивой.

2) Разработана библиотека ФБ, моделирующих типы устройств на ЦПС, а также управляющих ими;

3) Разработан автоматический транслятор, преобразующий исходное SCD-описание ЦПС в систему ФБ, который в дальнейшем может интегрироваться в систему автоматизации ЦПС в целом.

4) Разработан способ представления описания ЦПС на языке Prolog. Данный метод является универсальным и достаточно легко адаптируемым в случае внесения изменений в стандарт МЭК 61850.

5) Разработан способ представления ФБ на языке Prolog

6) Предложен способ трансляции SCL-Prolog в FB-Prolog на основании правил преобразования.

7) В ходе работы над второй главой в Роспатенте были зарегистрированы следующие программы для ЭВМ:

Транслятор SCL-описаний цифровых подстанций МЭК 61850 в набор фактов языка Пролог. №2018613272 от 19.01.2018

Функционально-блочная модель защиты от перегрузок цифровых подстанций МЭК 61850 (в процессе регистрации).

3. МЕТОДИКА АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ СИСТЕМ ФУНКЦИОНАЛЬНЫХ БЛОКОВ МЭК 61499, РЕАЛИЗУЮЩИХ ЧЕЛОВЕКО-МАШИННЫЙ ИНТЕРФЕЙС

По данной методике было опубликовано несколько работ [39-44].

3.1 Человеко-машинный интерфейс в системе автоматизации ЦПС

Система автоматизации ЦПС, спроектированная на основе шаблона MVC, представляет собой трехслойную структуру (рисунок 35).

Рисунок 35 - Архитектура системы автоматизации ЦПС

На нижнем слое (слое процесса) располагается интерфейс взаимодействия с физическим оборудованием (электромеханическими устройствами). Слой процесса транслирует сигнал из этого интерфейса в HMI, что позволяет производить мониторинг и защиту системы, а в обратном направлении осуществляется передача команд из HMI и управляющего слоя. Средний слой представляет распределенную систему управления. Управляющее приложение взаимодействует как со слоем процесса, так и со слоем HMI. На самом верхнем слое располагается HMI, который организует взаимодействие с пользователем. Здесь отображается статус всех элементов сети. Предоставляется возможность управлять системой вручную. Например, при нажатии на определенный символ можно изменить состояние устройства, с которым этот символ связан. В ходе такого взаимодействия происходит влияние оператора на слои процесса и

управления.

3.2Основные положения методики

Ниже предлагается и подробно рассматривается методика генерации ФБ, реализующих HMI для автоматизации ЦПС МЭК 61850, включающая следующая основные шаги:

1. создание библиотеки (типов) CAT-блоков для HMI;

2. разработка графовой модели электрической схемы ЦПС с отображением соединительных узлов в виде рёбер и устройств в виде вершин;

3. размещение узлов графовой модели на плоскости (в виде матрицы) с правильным расположением (ориентацией) вершин графа для применения в дальнейшем алгоритма трассировки;

4. применение алгоритма трассировки для нахождения всех путей между вершинами графа;

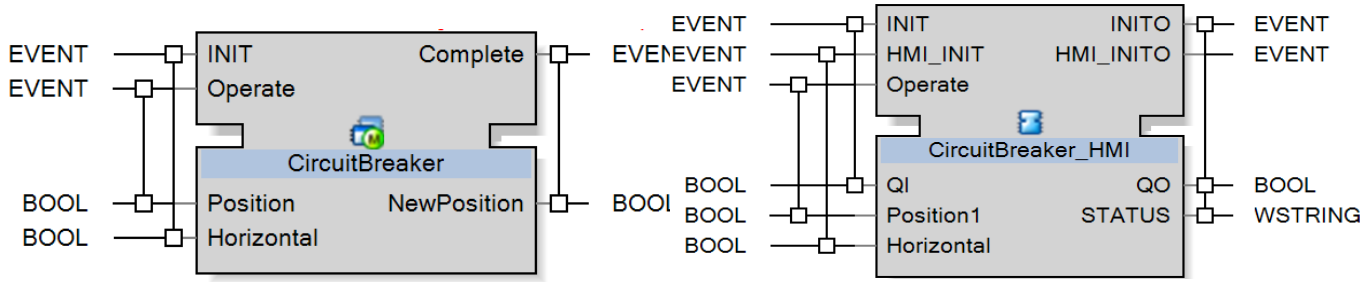
5. создание необходимых экземпляров CAT-блоков в приложении и их настройка на визуализацию электрической схемы в соответствии с п.п. 3 и 4 на экране HMI.

В статье также рассматривается программная поддержка предложенной методики в виде автоматического транслятора, преобразующего начальное SCD-описание ЦПС в систему ФБ, представляющую HMI системы SCADA, который в дальнейшем может интегрироваться в систему автоматизации ЦПС в целом. Следует отметить, что в данном случае имеется ввиду не традиционная система SCADA в форме какого-то отдельного пакета, а скорее ее отдельные функции, которые могут быть реализованы с использованием CAT-блоков в системе *NxtStudio* [28].

3.3 Создание библиотеки CAT-блоков для HMI

CAT-блок – это особый тип ФБ, который используется в системе *NxtStudio* для создания HMI. Данный ФБ основан на стандарте МЭК 61499, но объединяет в себе логическую и визуальную составляющую. Фактически, CAT-блок, также как и составной ФБ, является контейнером для сети ФБ, но в ней выделяется ФБ с суффиксом HMI (например: *Line_HMI*), который и реализует визуальные изменения в ответ на входные события и данные. Логическая составляющая представлена остальными внутренними ФБ сети (хотя она может и отсутствовать).

Пример модели выключателя в виде CAT-блока приведен на рисунке 36.

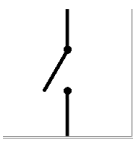
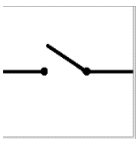


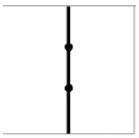
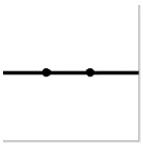
а) б) Рисунок 36 - Интерфейс CAT-блока (а) и его внутреннего HMI ФБ (б).

Для этого ФБ определены два параметра. Параметр *Position* имеет тип *Boolean* и указывает на то, находится ли выключатель во включенном или выключенном состоянии (0 - выключен, 1 - включен). Второй параметр – *Horizontal* имеет также тип *Boolean* и определяет пространственную ориентацию выключателя (0 - вертикальная, 1 - горизонтальная). Сигнал событийного входа *Operate* изменяет состояние выключателя в зависимости от соответствующего значения параметра *Position*. Выходное событие *Complete* указывает на изменение состояния выключателя пользователем, который щелкнул по его символу на экране HMI.

Все возможные значения комбинаций входных параметров для выключателя приведены в таблице 1 вместе с соответствующими графическими представлениями CAT-блока *CircuitBreaker*.

Таблица 3. Графическое представление комбинаций входных параметров.

	Horizontal = 0	Horizontal = 1
Position = 0		

Position = 1		

3.4 Разработка графовой модели

Исходными данными второго этапа методики (по разработке графовой модели электрической схемы ЦПС) является SCD-описание ЦПС на языке SCL. Пример SCD-описания простейшей сети, состоящей из двух устройств и одного соединительного узла, приведен на рисунке 37.

Substation – с данного тега начинается описание одной отдельно взятой подстанции. В одном файле SCD может быть описано несколько подстанций.

VoltageLevel и *PowerTransformer* – внутри тега *VoltageLevel* описывается все элементы подстанции, которые работают с одним напряжением. Для соединения устройств из разных *VoltageLevel* используются трансформаторы, каждый из которые описывается в теге *PowerTransformer*. Поэтому эти теги находятся на одном уровне.

Bay – данный тег описывает группу связанных устройств, которые образуют законченный элемент ЦПС. Также в этом теге имеется ссылка на интеллектуальные электронные устройства для управления этим элементом.

ConductingEquipment – этот тег является основным, поскольку в нём описан непосредственно элемент сети (генератор, резистор, прерыватель и т.д.)

```
 <Substation name="S1">
                    <VoltageLevel sxy:x="155" sxy:y="52" name="D1"
xmlns:sxy="http://www.МЭК.ch/61850/2003/SCLcoordinates">
    <Voltage unit="V" multiplier="k">220</Voltage>
```

```
    <Bay sxy:x="10" sxy:y="45" name="Q2">
      <ConductingEquipment sxy:x="178" sxy:y="15" name="I1" type="CTR">
                <Terminal connectivityNode="S1/D1/Q2/L1" substationName="S1"
voltageLevelName="D1" bayName="Q2" cNodeName="L1" />
      </ConductingEquipment>
      <ConductingEquipment sxy:x="178" sxy:y="75" name="QA1" type="CBR">
                <Terminal connectivityNode="S1/D1/Q2/L1" substationName="S1"
voltageLevelName="D1" bayName="Q2" cNodeName="L1" />
      </ConductingEquipment>
      <ConnectivityNode name="L1" pathName="S1/D1/Q2/L1" />
      <ConnectivityNode name="L2" pathName="S1/D1/Q2/L2" />
    </Bay>
  </VoltageLevel>
 </Substation>
```

Рисунок 37 - Пример SCD-описания

Рассмотрим основные теги, которые содержатся в любом SCD-описании:

Terminal – этот тег описывает соединительный узел, к которому подключено устройство. В основном, все устройства подключены к двум соединительным узлам (вход и выхода).

Модель электрической схемы, содержащейся в SCD-описании, может быть представлена в виде неориентированного графа:

$$G = (V, E),$$

где V – множество вершин, представляющих устройства;

E – множество рёбер, представляющих соединительные узлы.

Построение графа по SCD-описанию начинается с определения всех его вершин. Вершины определяются по тегам *ConductingEquipment*. Из SCD-описания на рисунке 37 можно выделить две вершины (типов CTR и CBR). Для

определения ребер используются теги *Terminal*, которые находятся внутри тегов *ConductingEquipment*. Если в разных тегах *ConductingEquipment* содержатся теги *Terminal* с одинаковым именем, то между соответствующими вершинами графа имеется связь. В SCD-описании на рисунке 37 имеется только один тег *Terminal*, представляющий соединительный узел S1/D1/Q2/L1, к которому подключены две упомянутые выше вершины. Таким образом, по приведенному на рисунке 37 SCD-описанию можно построить простой граф с двумя вершинами и одним ребром.

Для визуализации электрической схемы исходный SCD-файл может быть открыт в специальном редакторе *VisualSCL* [31] (рисунок 38,*а*).

3.5 Размещение элементов HMI

Третьим шагом методики является размещение элементов исходной электрической схемы на плоскости, разбитой на квадраты (называемой в дальнейшем матрицей размещения). Для этого может быть использован любой подходящий алгоритм размещения [32], например, гамма-алгоритм. Он позволяет разместить планарную часть любого графа. Если граф непланарен, то рёбра, которые нужно удалить для превращения графа в планарный, не учитываются.

В простейшем случае, когда исходный граф является древовидным, для размещения схемы на плоскости могут использоваться обычные алгоритмы обхода графа. На рисунке 38 представлен пример древовидной электрической схемы и результат размещения ее элементов на матрице трассировки.

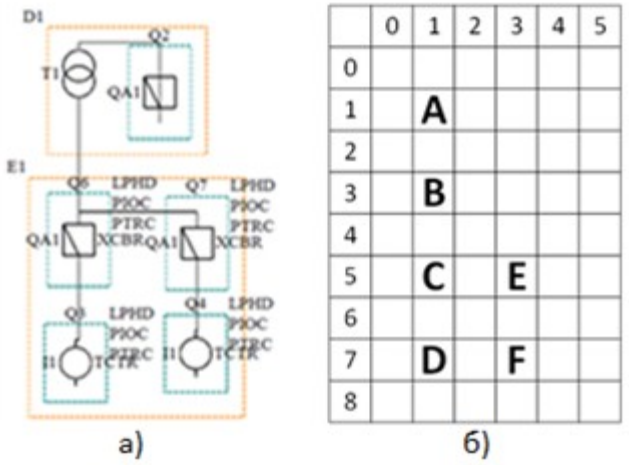


Рисунок 38 - Пример электрической схемы (а) и размещение ее элементов на плоскости (б)

3.6 Трассировка соединений на основе алгоритма Ли

Алгоритм Ли[33] – это алгоритма поиска кратчайшего пути в лабиринте. В данной работе его модификация будет использоваться для нахождения кратчайшего пути между двумя объектами сети Smart Grid, представленными на плоскости.

Основные этапы работы традиционного алгоритма Ли:

1. Из исходной точки в двумерном пространстве по четырем направлениям начинает распространяться волна, увеличивая свою мощность на единицу каждую итерацию. Значение волны записывается при первом прохождении через данную точку и в дальнейшем не изменяется, что позволяет представить область волны с одинаковым значением мощности в виде круга.

2. При достижении в ходе распространения волны конечной точки в пространстве, происходит её остановка.

3. Последним этапом является определение пути между начальной и конечной точкой в пространстве. Для этого начинает распространяться обратная волна из конечной точки, но она проходит только по точкам, в которых имеется значение первой волны. Эта волна имеет начальную

мощность, равную конечной мощности первой волны, и движется только в одном направлении – по убыванию значения мощности первой волны, пока не достигнет исходной точки.

Работа традиционного алгоритма Ли представлена на рисунке 39.

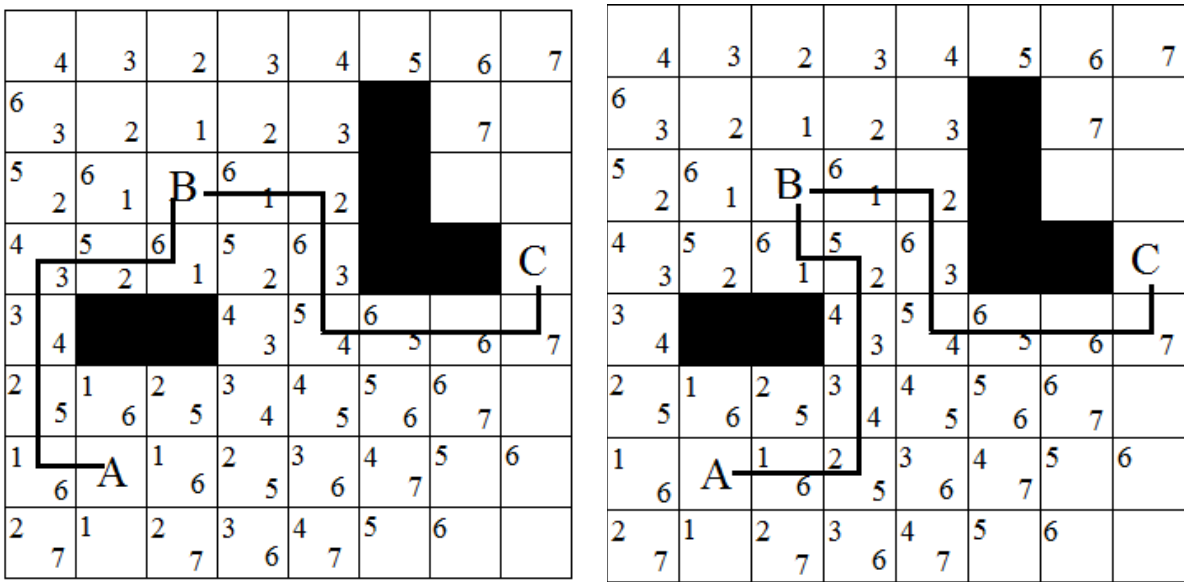


Рисунок 39 - Пример нахождения пути между точками A и B и точками B и C.

В левом верхнем углу обозначены значения волны для пути между A и B, а в правом нижнем углу – между точками B и C. Черные квадраты обозначают препятствие. Рисунки справа и слева отображают различные варианты пути между точками A и B.

Как видно из рисунка 1, существует несколько возможных кратчайших путей между точками A и B, из которых в традиционном алгоритме Ли путь выбирается случайным образом. Если рассмотреть пути на рисунке 1 справа внимательнее, то можно заметить, что два различных пути можно заменить на один путь с разветвлением (рисунок 2). Это особенно актуально при прокладке соединений между компонентами электрической сети, поскольку позволяет экономить на проводниках и обеспечивает уменьшение паразитных токов. Для этой цели была разработана модификация алгоритма Ли.

Основные модификации алгоритма Ли:

1. Наличие приоритета по осям.

Движение второй волны всегда приоритетнее по оси Y. Это позволяет убрать путаницу, возникшую на рисунке 40.

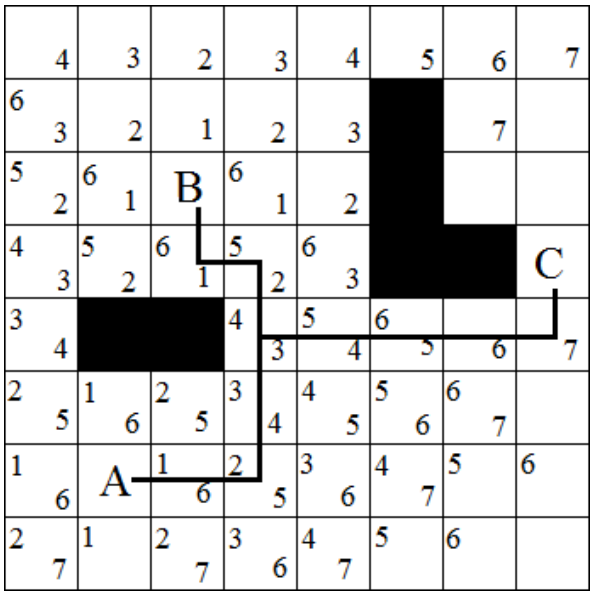


Рисунок 40 - Схема соединения между точками A,B,C используя модифицированный алгоритм Ли.

2. Приоритет по наименьшему числу связей.

Это обозначает, что начинать первую волну следует с точки, которая имеет меньше соединений. Например, необходимо соединить точку A с B, а точку B с C. Если путь между A и B уже существует, то соединение B с C следует делать начиная первую волну с точки C, потому что точка B, как и точка A, имеет уже одно соединение.

3. Наличие буфера памяти у каждой точки в пространстве, через которую может быть проложен путь.

Этот буфер используется только в том случае, если через данную точку проходит какой-либо путь. В этом случае в буфере храниться список точек,

которые соединяет данный путь (две точки в простейшем случае). Каждая точка пути, показанного на рисунке 2, содержит в своём буфере три точки (A, B и C). При работе модифицированного алгоритма Ли в дальнейшем будет проверятся не только достижение волной конечной точки, но и достижения точки, в буфере которой записана вершина, которою может достичь волна.

Рассмотрим пример применения модифицированного алгоритма Ли. Начальный файл описания сети Smart Grid (SLD-файл) был считан транслятором и все элементы, которые описываются в этом файле сети, были расположены на плоскости (рисунок 3). Координаты элементов на плоскости вводятся также из SLD файла. Для удобства записи элементов в буфер ячейки при их соединении закодируем каждый из них уникальной буквой.

Кроме того, из файла выбирается следующая информация о соединениях (для лучшего понимания приводится в формате координат с указанием стороны соединения (п-право, л- лево, в- верх, н- низ):

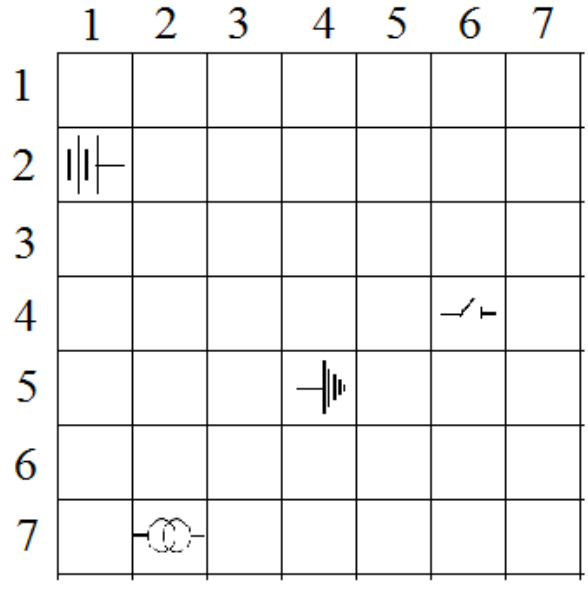
1)1.2п – 6.4п

2)1.2п – 2.7п

3)2.7л – 4.5л

4)4.5л – 6.5л

Ход работы модифицированного алгоритма Ли для данного примера следующий:



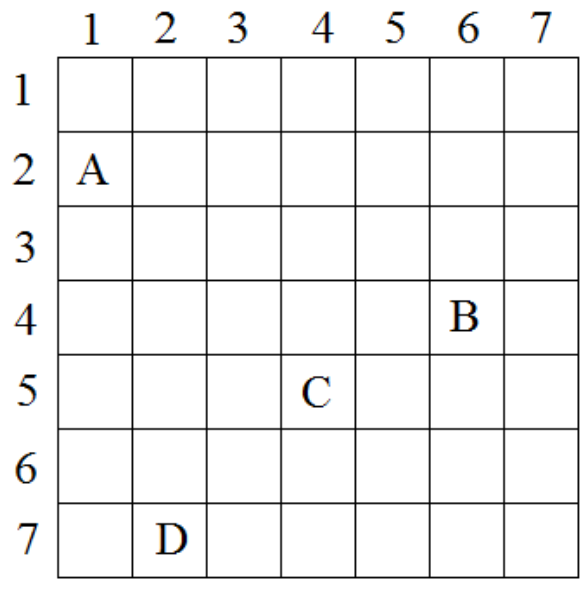


Рисунок 41 - Схема расположения элементов на плоскости и их кодировка перед началом работы модифицированного алгоритма Ли.

1. Проводим первый путь между элементами сети и заносим в буфер каждой пройденной ячейки значение начального и конечного элемента сети.

2. Следующее соединение необходимо провести между точками A и D, но точка A уже имеет одно соединение, поэтому за начальную точку принимается точка D.

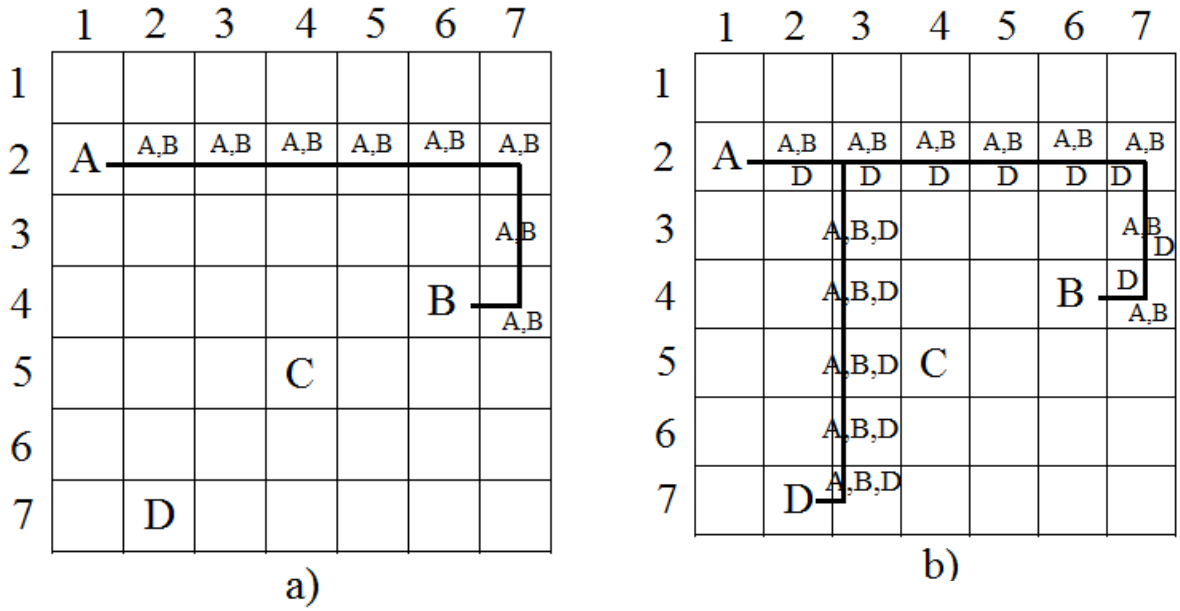


Рисунок 42 - Схема соединения между элементами A,B,D.

Распространение волны происходит до координаты (2,2), поскольку она является ближайшей согласно алгоритму Ли. Хотя эта точка и не элемент сети, но в её буфере содержится информация, что она подключена к искомой ячейке. Поэтому эта ячейка принимается за конечную, и буфер всего пути изменяется (рисунок 4б), поскольку к текущему пути добавлен новый элемент.

3. Остальные соединения находятся аналогичным образом.

Следует отметить, что модифицированный алгоритм Ли был реализован программно на языке C/C# и был использован при построении человеко-машинного интерфейса системы трансляции и визуализации схем

электроподстанций, построенных в соответствии со стандартом МЭК 61850, в виде функциональных блоков в системе NxtStudio.

В ходе выполнения данной работы был разработан модифицированный алгоритм Ли, описаны его отличия и представлен пример его работы. В дальнейшем планируется использование этого алгоритма при разработке реальной системы Smart Grid.

3.7 Генерация системы функциональных блоков

Далее необходимо произвести преобразование матрицы (поскольку была построена матрицы соединительных узлов. А для отображения на экране HMI необходима матрица устройств). Для преобразования матрицы используется следующая формула:

$$(x_j; y_j) = (x_i; y_i) \vee \left\{ \frac{argmax}{i}(x_i); \frac{argmax}{i}(y_i) \right\}, i = f(j) \quad (1)$$

где,

$x_j; y_j$ – координаты j-го устройства,

$x_i; y_i$ - координаты соединительных узлов, к которым подключено оборудование,

f(j) – массив вершин, соединенных с j-вершиной графа.

Согласно формуле, сначала берутся координаты двух соседних соединительных узлов и вычисляются максимумы по x и по y. По этим координатам будет располагаться оборудование, которое находится между этими соединительными узлами в графе.

Результатом этапов 2-4 методики является матрица трассировки, пример которой можно найти на рисунке 43. Последним этапом является генерация системы ФБ в соответствии с этой матрицей или, иными словами, создание полного HMI с использованием библиотек CAT-блоков. Как можно понять из

рис. 9, всего будет сгенерировано десять ФБ - шесть ФБ, отвечающих за устройства ЦПС, и четыре ФБ - за отрисовку линий соединения (рисунок 43). Служебный блок типа E_RESTART служит для запуска системы.

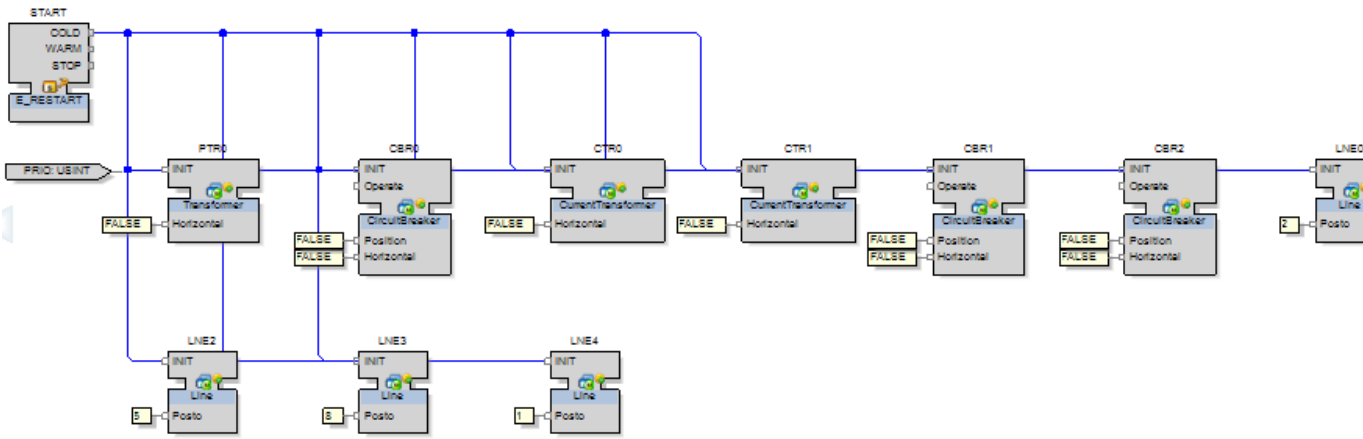


Рисунок 43 - Система ФБ, сгенерированная на основе исходного SCD-описания

Важно отметить, что было сгенерировано четыре экземпляра ФБ для линий, а не шесть, ввиду того, что некоторые ячейки (например, [1,1], [1,6] и [3,6]) содержат одинаковый тип линии, для отрисовки которых может использоваться только один тип ФБ. Это стало возможным благодаря тому, что линии на HMI никак не связаны с реальными устройствами сети, что позволяет создать только один экземпляр ФБ для одного типа линии. Таким образом, в проекте может быть максимум одиннадцать ФБ, отвечающих за рисование соединительной линии на HMI (рис. 8). Выполнение системы ФБ из рис. 10 в системе *nxtStudio* приводит к генерации HMI и его визуализации на экране монитора (рис. 11). По сути дела, это интерактивная схема электрической сети ЦПС. Как можно заметить, она полностью соответствует исходной сети, что может служить доказательством правильности преобразования.

3.8 Разработка транслятора SCD-описаний ЦПС в системы функциональных блоков

3.8.1 Структура транслятора

На рисунок 44 представлена структура программного обеспечения, предназначенного для генерации и работы с HMI ЦПС.

Центральное место в ней занимает автоматический транслятор, преобразующий исходное SCD-описание ЦПС в систему ФБ, представляющую HMI (обведен штриховым контуром). Основные модули этого транслятора реализуют шаги предложенной выше методики. Сгенерированный HMI может отображаться и интерпретироваться в NxtStudio - инструментальной системе проектирования распределенных систем управления на основе. стандарта МЭК 61499. Для работы в рамках реальной ЦПС необходимо использовать среду выполнения nxtRT61499F. Кроме того, в этом случае в виде систем ФБ должна быть представлена управляющая логика, например, реализующая алгоритмы защиты ЦПС.

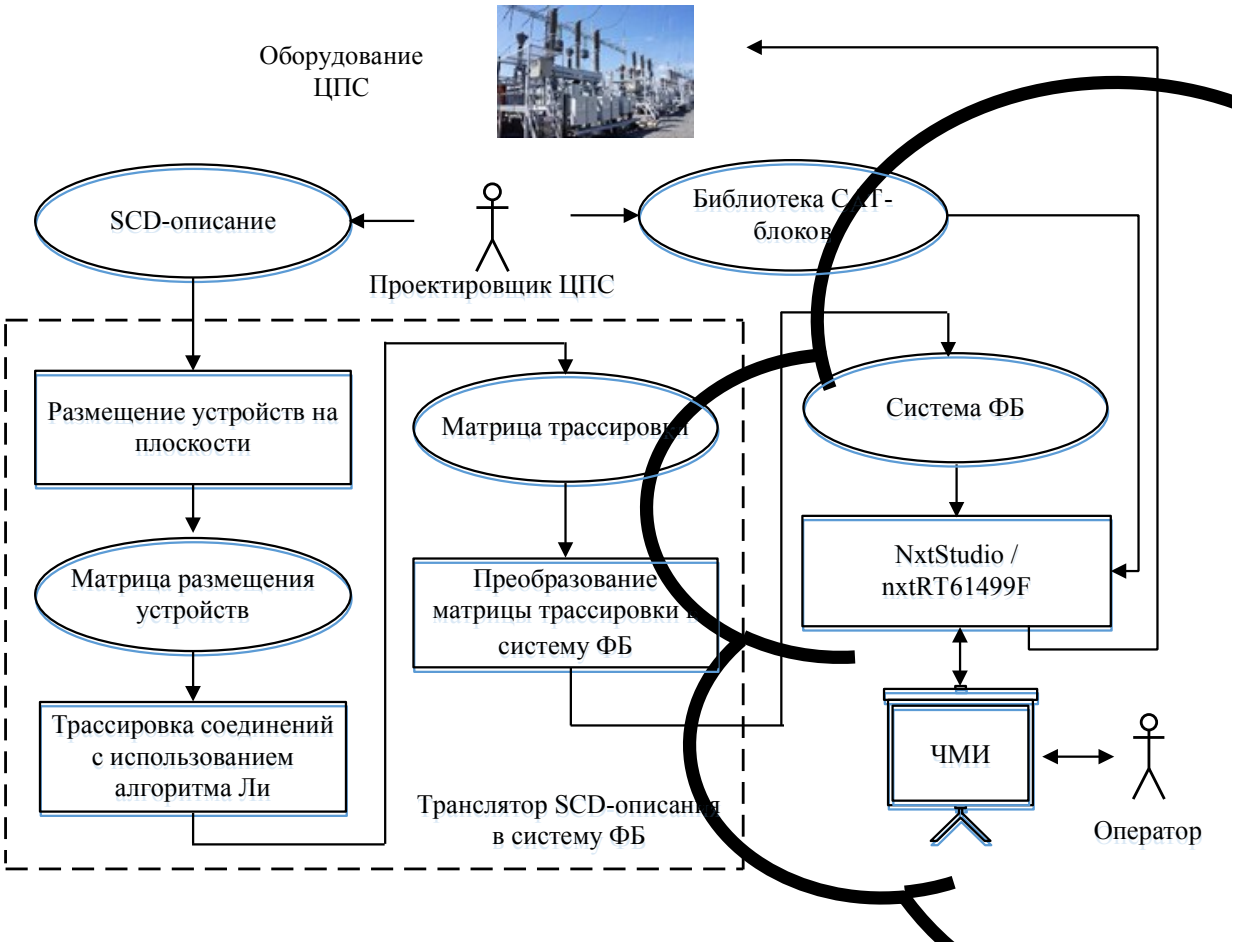


Рисунок 44 - Структура программного обеспечения для генерации и работы с HMI ЦПС.

3.8.2 Алгоритмы работы и структуры данных

Алгоритм работы состоит из нескольких шагов:

1) Выбор файла для трансляции и места сохранения результирующего проекта.

2) Открываем файл System.deploy.sys и при помощи парсера определяем пути отображения по ID и по Name, которые сохраняем в глобальных переменных MapDevice и NameDevice.

3) Открываем файл System.sys и определяем Имя и ID Application, которые сохраняем в глобальные переменные APPName и APPID

4) Инициализируем новый контейнер устройств DEVELOP и новый объект класса Eqiupment а также объект класса INS, в котором храниться соответствие между ID Application и ID Inst для одного CAT блока

5) Заполнение контейнера DEVELOP и массива ConnectionNode (используя XmlDocument)

6) Создание матрицы смежности для графа Connection Node. Мост между двумя вершинами существует, если существует Terminals содержащий эти вершины. Пример графа показан на рисунке 45. Его матрица смежности показана в таблице 4. Соответствие номера строки/столбца матрицы с именем вершины осуществляется через массив ConnectionNode.

7) Алгоритм расстановки элементов на плоскости:

7.1) Определяем вершину у которой только один мост. Принимаем её за Begginer.

7.2) Выделяем куб размером 2*количество вершин X 2*количество вершин количество вершин.

Первые 2 атрибута – x и y координаты на плоскости, а третий атрибут – номер вершины(на Z = номеру вершины ставиться 1).

7.3) Объявляем переменные Multx = 1, Multy = 1.

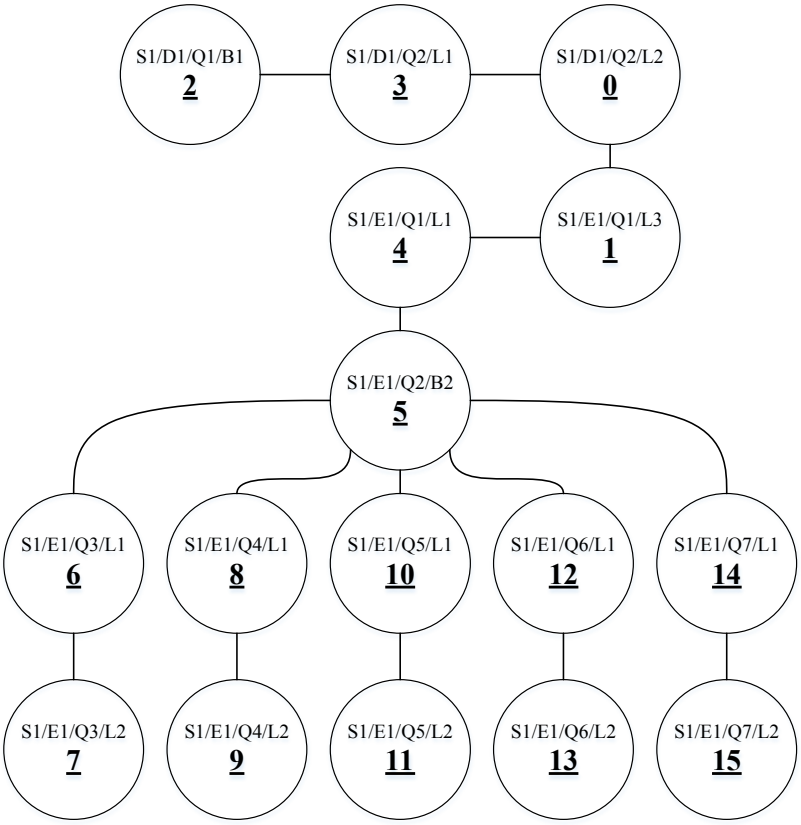


Рисунок 45 - Граф соединительных узлов.

7.4) Находим значение 1 в матрице смежности на пересечении с Begginer. Запоминаем её как x1.

7.5) В кубе по координатам [Multx, Multy, Begginer] ставим значение 1.

7.6) Увеличиваем значение Multy на 2.

7.7) Находим 1 в матрице смежности на пересечении с Begginer. Запоминаем её как x1. Если не нашли, то переход на пункт 7.11

7.8) В кубе по координатам [Multx, Multy, x1] ставим 1. Также добавляем новый экземпляр в контейнер CoordinateEqiupment.

7.9) Увеличиваем значение Multy на 2.

7.10) Присваиваем Begginer значение x1.

7.11) Повторяем пункты 7.6 – 7.9 до тех пор пока не дойдём до конца ветки (нет 1 в матрице смежности на пересечении с Begginer).

Таблица 4. Матрица смежности для графа Connection Node.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		1		1												
1	1				1											
2				1												
3	1		1													
4		1				1										
5					1		1		1		1		1		1	
6						1		1								
7							1									
8						1				1						
9									1							
10						1						1				
11											1					
12						1								1		
13													1			
14						1										1
15															1	

7.12) Вычитаем из Multy значение 2, Если Multy стал меньше 0, то переход на пункт 7.14.

7.13) Значение Multx увеличиваем на 2, но при условии, что в данном цикле была найдена хотя-бы одна смежная вершина в пунктах 7.6 – 7.9.

7.14) Постепенно проходя по всему кубу, уменьшаем координату X, находим вершину, которая является предком Begginer. Запоминаем её как Begginer. Переход на пункт 7.6.

7.15) Алгоритм окончен.

8) Используя информацию из контейнера CoordinateEqiupment (x,y,Code), заполняем матрицу расстановки компонентов GrathLine (Таблица 5.), где на координате, имеющей Eqiupment, ставим значение -1.

Таблица 5. Матрица расстановки компонентов.

	0	1	2	3	4	5	6	7	8	9
0										
1		-1								
2										
3		-1								
4										
5		-1								
6										
7		-1								
8										
9		-1								
10										
11		-1								
12										
13		-1		-1		-1		-1		-1
1										

4										
15		-1		-1		-1		-1		-1
16										
17										

9)Алгоритм нахождения пути между 2 вершинами (модифицированный алгоритм Ли):

9.1) Находим в матрице значение -1 (не подсоединенная вершина). Сохраняем её координаты в X и Y, а её номер в CodeIsk. Если в матрице нет значений -1, то переход на 9.15.

9.2) Уменьшаем количество мостов у данной вершины на 1.

9.3) Если у вершины не осталось мостов, то удаляем её из GrathLine.

9.4) Копируем матрицу GrathLine в матрицу GrathTemp.

9.5) Присваиваем GrathTemp [X,Y] = 1 (длина волны до этой координаты равна 1). Принимаем значение Step = 1 (текущая мощность волны).

9.6) Распространение волны (Таблица 6).

9.6.1) Проверяем все координаты матрицы GrathTemp на значение Step.

9.6.2) Проверяем все соседние координаты матрицы GrathTemp с координатами элемента матрицы со значением Step на значение -1.

9.6.3) По координатам каждой найденной ячейки с -1 получаем номер вершины, находящейся по этой координате.

9.6.4) Номер полученной вершины и CodeIsk проверяем на наличие пути

9.6.3.1) Получаем их строковое значение в виде kode1 и kode2.

9.6.3.2) Проверяем наличие Terminals соединяющего эти вершины в контейнере DEVELOP.

9.6.3.3) Ищем в контейнере CoordinateEqiupment экземпляр Coordinate в котором одно из подключений будет соответствовать номеру вершины kode1 или номеру вершины kode2, но если в одном Coordinate имеется обе вершины, то путь уже проложен и нового пути не существует.

9.6.4) Если имеется новый путь между полученной вершиной и CodeIsk, то сохраняем координаты полученной вершины в X2,Y2 и переходим к пункту 9.7

Таблица 6. Пример того, как может выглядеть матрица GrathTemp.

	0	1	2	3	4	5	6	7	8	9
0										
1		0								
2										
3		0								
4										
5		0								
6										
7		0								
8	5		5							
9	4	0	4	5						
10	3	2	3	4	5					
11	2	1	2	3	4	5				
12	3	2	3	4	5					
13	4	-1	4	-1		-1		-1		-1
14	5		5							
15		-1		-1		-1		-1		-1

16										
17										

9.6.5) Увеличиваем Step на 1.

9.6.6) Все соседние координаты матрицы GrathTemp с координатами элемента матрицы со значением Step проверяем на 0. Если равно 0, то записываем по этой координате значение Step.

9.6.7) Переход к пункту 9.6.1.

9.7) Уменьшаем количество мостов у вершины X2,Y2 на 1.

9.8) Если у вершины не осталось мостов, то удаляем её из GrathLine.

9.9) Сохраняем коды начальной и конечной вершины в переменные Beginner и Ender.

9.10) Начиная с вершины X2,Y2 ищем в соседних элементах матрицы по оси Y значение Step – 1.

9.11) Координаты найденного элемента матрицы со значением Step – 1 заносим в контейнер CoordinLine вместе с Beginner и Ender.

9.12) Присваиваем X2,Y2 значения найденного элемента матрицы.

9.13) Пока Y2 не равен Y – 1, переход на пункт 9.10.

9.14) Аналогичные шаги 9.10 - 9.12 проводим для оси X.

9.15) Переход к пункту 9.1.

9.16) Алгоритм окончен.

10) После шага 9 в CoordinLine имеются все координаты для всех путей, которые можно представить на матрице (для удобства значение -1 заменил на номер вершины а в скобках показан один элемент контейнера)(Таблицы 7,8,9).

Далее необходимо оптимизировать контейнер (объединить все элементами с одинаковыми координатами).

10.1 Объединяем все множества линий в одной координате. После этого количество элементов в контейнере резко сократиться и можно представить на матрице.

10.2 Объединяем соседние по координатам множества, если имеется разница хотя-бы в 1 вершину подключения. После этого количество элементов в контейнере не сократиться, но измениться количество подключений каждой ячейки матрицы.

Таблица 7. Матрица после нахождения всех путей.

	0	1	2	3	4	5	6	7	8	9
0										
1		0								
2		(0;1)								
3		1								
4		(1;0)								
5		2								
6		(2;3)								
7		3								
8		(3;4)								
9		4								
10		(4;5)								
11		5								
12		(5;6) (5;8) (5;10)	(5;8) (5;10) (5;12	(5;8) (5;10) (5;12	(5;10) (5;12)	(5;10) (5;12) (5;14)	(5;12) (5;14)	(5;12) (5;14)	(5;14)	(5;14)

		(5;12)(5;14)	)(5;14)	)(5;14)	(5;14)					
13		6		8		10		12		14
14		(6;7)		(8;9)		(10;11		(12;13)		(14;15)
15		7		9		11		13		15
16										
17										

Таблица 8. Матрица после слияние дублирующих вершин в одной ячейке.

	0	1	2	3	4	5	6	7	8	9
0										
1		0								
2		(0;1)								
3		1								
4		(1;0)								
5		2								
6		(2;3)								
7		3								
8		(3;4)								
9		4								
10		(4;5)								
11		5								
12		(5;6;8;10;12;	(5;8;10;12;14)	(5;8;10;12;14)	(5;10;12;14)	(5;10;12;14)	(5;12;14)	(5;12;14)	(5;14)	(5;14)

		14)								
13		6		8		10		12		14
14		(6;7)		(8;9)		(10;11		(12;13)		(14;15)
15		7		9		11		13		15
16										
17										

Таблица 9. Матрица после оптимизации.

	0	1	2	3	4	5	6	7	8	9
0										
1		0								
2		(0;1)								
3		1								
4		(1;0)								
5		2								
6		(2;3)								
7		3								
8		(3;4)								
9		4								
10		(4;5)								
11		5								
12		(5;6; 8; 10;1 2; 14)	(5;6; 8; 10;1 2; 14)	(5;6;8 ; 10;12 ; 14)	(5;6; 8; 10;1 2; 14)	(5;6; 8; 10;12 ; 14)	(5;6; 8; 10;1 2; 14)	(5;6;8; 10;12; 14)	(5;6; 8; 10;1 2; 14)	(5;6;8; 10;12; 14)
13		6		8		10		12		14
14		(6;7)		(8;9)		(10;1 1		(12;13)		(14;15)
15		7		9		11		13		15
16										
17										

11) Определяем тип каждого из элементов контейнера CoordinLine Для этого используем 4 флага (1-флаг – одна сторона) и проверяем их совокупность в конце.

11.1) Берём значения из текущей ячейки (определяется по x,y из элемента контейнера CoordinLine) и сравниваем с соседней.

11.2) Если соседняя ячейка матрицы – вершина, то проверяем наличие её номера в списке подключения текущей ячейки. Если имеется то, выставляем соответствующий флаг.

11.3) Если соседняя ячейка матрицы – тоже линия, то сравниваем их списки подключения, и если они равны, то выставляем флаг.

11.4) По комбинации флагов выставляем тип линии (Таблица 10).

Таблица 10. Соответствие комбинаций флагов типу линии.

f1	f2	f3	f4	Тип линии
1	0	1	0	1
0	1	0	1	2
1	0	1	1	3
1	1	1	0	4
0	1	1	1	5
1	1	0	1	6
1	1	1	1	7
1	0	0	1	8
1	1	0	0	9
0	0	1	1	10
0	1	1	0	11

12) Работа с контейнером DEVELOP.

12.1) Берём Terminals из контейнера DEVELOP и определяем координаты, входящих в него Terminal (используя контейнер CoordinateEqiupment). Оборудование будет нарисовано по максимальным X и Y.

12.2) Определяем уникальное название оборудования на основе счётчиков и полного имени, полученного из DEVELOP.

12.3) Задаём значения по умолчанию для отображения.

12.4) Добавляем полностью готовый элемент в контейнер CoordCAT.

12.5) Повторяем шаги 12.1-12.4 для всех элементов контейнера DEVELOP.

12.6) Для каждого из элементов контейнера CoordinLine.

12.6.1) Получаем уникальное имя CAT блока (LNEX), где X – уникальный номер.

12.6.2) Значения координат умножаем на 50 и генерируем уникальный ID.

12.6.3) Добавляем полностью готовый элемент в контейнер CoordCAT.

Далее происходит работа с проектом nxtStudio и рисование на Canvas всех элементов CoordCAT. Все дальнейшие шаги выполняются для всех элементов контейнера CoordCAT.

13) Запись в проект nxtStudio.

13.1) Открываем файл System.cfg и добавляем в него тег Inst с атрибутами из CoordCAT.

13.2) Открываем файл System.sys.

13.2.1) Создаём запись о FB в приложении и на устройстве. Рисуем на плоскости их по 4 штуки в ряд.

13.2.2) Создаём Mapping между FB устройства и приложения.

13.2.3) Начальная инициализация CAT блока (START.COLD и значения по умолчанию).

14) Рисование CAT блоков на Canvas. Для этого модифицируем файл \\ HMI\\Canvas1.cnv.Designer.cs.

14.1) Добавляем информацию об элементе перед строкой //Canvas1.

14.2) Добавляем элемент на Canvas после строки private void InitializeComponent().

14.3) Добавляем элемент в качестве компонента после строки this.Shapes.AddRange(new System.ComponentModel.IComponent[] {.

14.4) Добавляем строку, о том что элемент существует в данном пространстве перед #endregion.

15) Записываем в файл \\HMI\\Canvas1.cnv.resx метаданные объекта, чтобы программа могла его использовать.

Описание работы с XmlDocument

Парсер обходит документ в глубину, изучая каждый тег. У него на входе 3 аргумента: Логический узел, тип запроса и элемент CoordCAT, который может быть не задан.

Логический узел позволяет перемещаться по дереву тегов, а CoordCAT хранит информацию о блоке, который нужно создать и записать в проект.

Тип запроса определяет то, что нужно сделать при очередном обходе всех тегов документа. Это необходимо ввиду того, что в разных документах могут быть одинаковые теги, в таком случае необходимо точно определить с каким

документом мы работаем и что нам нужно. Расшифровка всех типов запроса представлена в таблице 11.

Таблица 11. Описание модифицируемых файлов проекта nxtStudio.

Тип запроса	Файл для работы	Описание запроса
1	\\МЭК61499\\bin\\Deploy\\ System.deploy.sys	Получение пути отображения на ресурс устройства
3	\\МЭК61499\\System\\ System.cfg	Создаёт новый CATType с заданным именем и экземпляр Inst или просто добавляет новый Inst в существующий CATType
4	\\МЭК61499\\System\\ System.cfg	Проверка на существование CATType с определённым именем
5	\\МЭК61499\\System\\ System.sys	Создание FB в приложении и в устройстве (а также их расстановка по 4 в ряд с интервалом в 550). Добавление отображения для соответствия между FB в приложении и в устройстве.
6	\\МЭК61499\\System\\ System.sys	Проверка на наличие EventConnections
7	\\МЭК61499\\System\\ System.sys	Соединение CAT FB с START.COLD (инициализация CAT FB)
8	\\МЭК61499\\System\\ System.sys	Получение имени и id application

Таблица 11. Продолжение.

9	Путь до scd файла	Получение всех ConnectionNode и запись их в массив ConnectionNode
10	Путь до scd файла	Заполнение контейнера DEVELOP
11	\\МЭК61499\\System\\ System.sys	Инициализация входных параметров для FB

Описание контейнера DEVELOP

Внутри контейнера находятся экземпляры класса Eqiupment, который может быть двух типов: по умолчанию или Transformer. Полная структура контейнера представлена на рисунке 46.

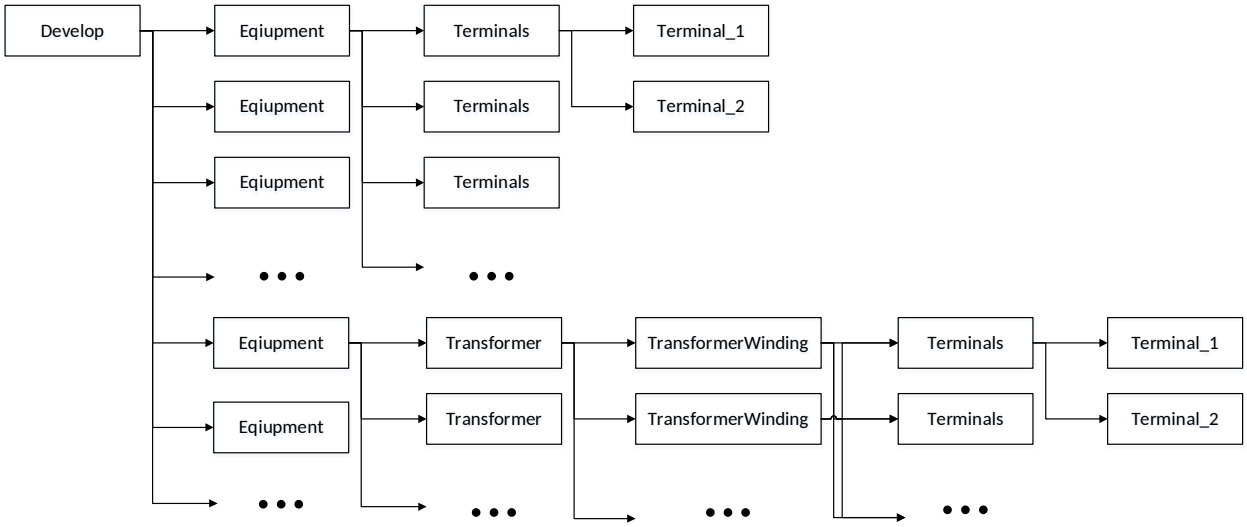


Рисунок 46 - Логическая структура контейнера DEVELOP

Режим текущего Eqiupment определяется после функции SetFullName(Name), в которую передаётся название типа устройства. Это определяет поведение функции SetTerminal(Name), поскольку при режиме Transformer каждый Terminal текущей Winding должен быть связан со всеми другими Terminals всех других Windings в контексте текущего Transformer. Также в режиме Transformer можно использовать функцию SetWinding(), которая инициализирует новую Windings.

Рассмотрим методы класса Eqiupment

1. GetDoubleTerminal()

   Возвращет количество объектов Terminals у данного Eqiupment.

2. GetNumTerminal(int NumTerm, int Num)

Возвращает имя Terminal_1 или Terminal_2 (в зависимости от параметра Num) находящееся в Terminals номер NumTerm.

3. SetWinding()

Добавляет новый Winding у Transformer по умолчанию

4. AddTerminal (string connectivityNode1, string connectivityNode2)

Добавляет новый объект Terminals с Terminal_1 равным connectivityNode1 и Terminal_2 равным connectivityNode2

5. SetTerminal (string connectivityNode)

Добавляет новый Terminal (автоматически определяя, Terminal_1 или Terminal_2, если оба уже созданы, то создаёт новый Terminals)

6. SetFullName(string name)

Определяет тип устройства

7. GetFullName()

Возвращает тип устройства

Описание контейнера CoordinateEqiupment

Контейнер содержит в себе экземпляры класса Coordinate. Структура котейнера представлена на рисунке 47.

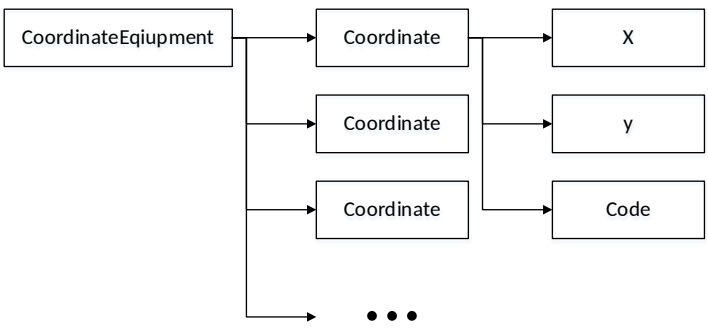


Рисунок 47 - Структура контейнера CoordinateEqiupment

Класс Coordinate содержит 3 переменных: координаты X и Y на Canvas и код вершины графа (ConnectivityNode), который равен индексу в массиве ConnectionNode (Таблица 12).

Таблица 12. Структура массива ConnectionNode.

0	S1/D1/Q1/ B1
1	S1/D1/Q2/L1
2	S1/E1/Q1/L1
…	…

Инициализация элементов класса Coordinate происходит только 1 раз в конструкторе.

Класс Coordinate имеет 3 метода для получения каждого из параметров: X, Y, Code.

Описание контейнера CoordinLine

Контейнер (Рисунок 48) содержит в себе экземпляры класса CoordinateLine, который наследуется от класса Coordinate. Структура контейнера показана на рисунке.

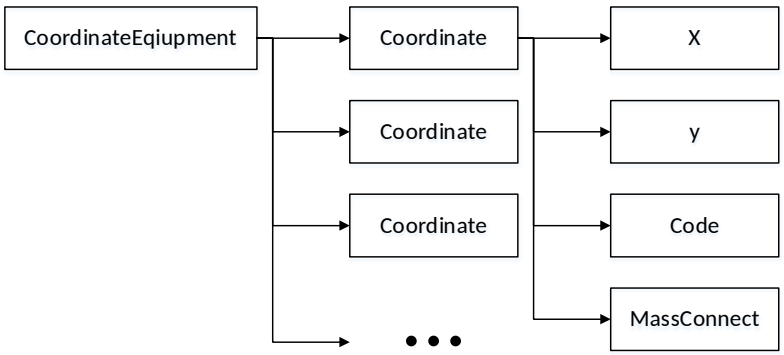


Рисунок 48 - Контейнер устройств.

Разница заключается в изменении значения переменной Code (рисунок 53) и добавления методов работы с списком MassConnect (список вершин, которые подсоединены к данной координате).

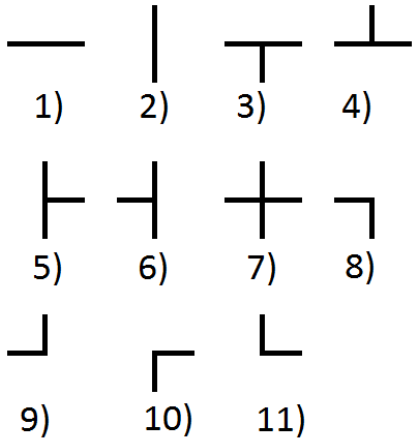


Рисунок 53 - Графическое представление значения переменной Code.

Дополнительные метода класса CoordinateLine:

1. SetNewConnect(int NameConnect)

   Добавляет новое подключение (входной параметр – код из ConnectionNode) к данной координате.
2. GetCountConnect()

   Возвращает количество подключений к данной координате.
3. GetConnect(int Num)

   Возвращает код подключенной вершины по входному индексу.
4. SetType(int Num)

   Устанавливает тип изображения в координате (в соответствии с рисунком 4).

Описание контейнера MassCoordCAT

Контейнер содержит в себе экземпляры класса CoordCAT, который содержит всю необходимую информацию о координате на Canvas (Рисунок 49)

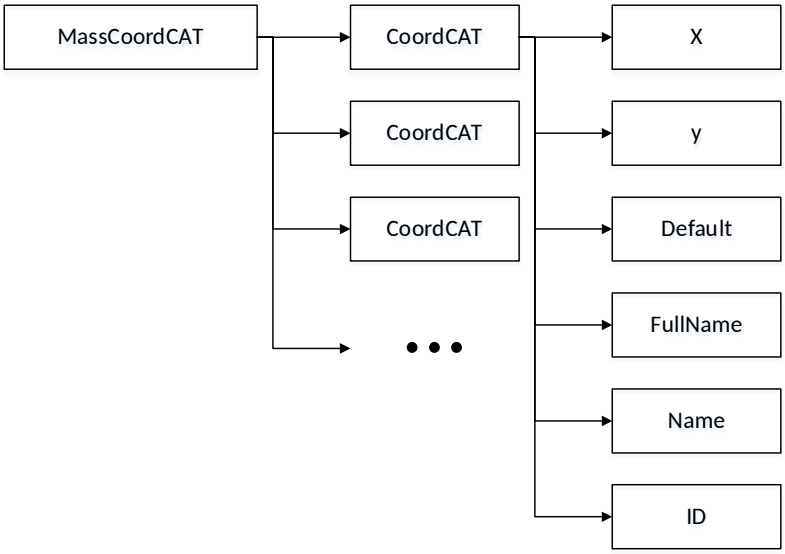


Рисунок 49 - Структура контейнера MassCoordCAT.

Класс CoordCAT содержит только конструктор для единичной инициализации всех элементов. Также имеются 6 методов для получения значения каждого из 6 элементов класса.

X – координата x

Y – координата y

Default – значения входных переменных, разделённые пробелом (FALSE TRUE 4, …)

FullName – тип оборудования (Transformer, CircuitBreaker, …)

Name – имя данного CAT блока (CB1, LNE4, CT3, …)

ID – уникальный идентификатор данного CAT блока

3.8.3 Примеры использования

Практическое применение данная работа нашла в электрической сети на базе лаборатории AIC3 в Техническом университете Лулео, Швеция. Лаборатория представляет собой реконфигурируемую фабрику с несколькими потребителями энергии: различные производственные машины, роботы, системы управления окнами и др. SLD – представление системы

электроснабжения AIC3 представлено на рисунке 11. Оно включает в себя 3 распределителя, управляемых автоматическими выключателями. Также каждый распределитель управляет 8 переключателями ретрансляционного устройства KMTronic UDP (рисунок 12).

Каждое устройство электрической сети адресуется уникальным IP адресом, а управление 8 переключателями осуществляется посредством шестнадцатеричных команд. Для трансляции команд в шестнадцатеричный формат и обратно был разработан специальный ФБ ConnectToTrigger (рисунок 13).

Два информационных входа (IP и PORT) однозначно определяют устройство для управления. Информационные входы w1-w8 являются индивидуальными для каждого устройство с булевским выходным сигналом, что однозначно позволяет определить каждого устройства в любой момент времени. Логическое значение TRUE обозначает включение переключателя, а FALSE – выключение. Булевское значение автоматически преобразуется в соответствующую шестнадцатеричную команду для отдельного переключателя и выводиться через информационный выход String. ConnectToTrigger подключен к ФБ NETIO, который используется для общения по сети интернет с электронными устройствами. NETIO FB является сетевым ФБ, который как принимает, так и отправляет данные в сеть. Выход SD отправляет сформированный пакет в сеть, в то время как RD-вход принимает ответный пакет из сети.

3.9 Выводы

1) Предложена методика генерации ФБ, реализующих HMI для автоматизации ЦПС МЭК 61850, отличающаяся от существующих возможностью внесения изменений в структуру начальной сети для генерации оптимизированной системы управления, что позволяет значительно упрощать (или усложнять) системы управления, но только если это необходимо.

2) Разработан модифицированный алгоритм Ли, отличающийся от известных возможностью соединения не только узла с узлом, но и узла с уже проложенным соединением, причем в качестве исходной вершины для распространения волны всегда выбирается неподключенная вершина или вершина с наименьшей степенью связности, что позволяет ускорить процесс трассировки пути и уменьшить общее время работы алгоритма, а также повысить читабельность экранных представлений электрических схем в HMI;

3) Разработана библиотека ФБ, моделирующих и визуализирующих типы устройств на ЦПС, а также управляющих ими;

4) Разработан автоматический транслятор FB-Prolog описания ЦПС в систему ФБ, реализующую HMI.

5) Выведена формула для преобразования матрицы узлов в матрицу устройств. Данная формула позволяет размещать на плоскости графовую модель соединительных узлов, а затем превращать её в модель устройств, поскольку не из каждого описание сети можно сразу построить графовую модель устройств.

6) Предложен способ получения графовых моделей сети Smart Grid на основе её описания. Возможно получение 2 вариантов графа: граф соединительных узлов и граф устройств

7) На основе анализа литературы определено, что в настоящее время существует несколько исследовательских работ, включающих как МЭК 61850, так и МЭК 61499 с предлагаемой архитектурой и средой совместного выполнения. Однако ряд вопросов, в частности, вопросов, связанных с автоматической генерацией HMI и схем защиты, остается нерешенным.

В ходе работы над третьей главой в Роспатенте были зарегистрированы следующие программы для ЭВМ:

Транслятор SCL-описаний в функционально-блочное представление человеко-машинного интерфейса электроподстанций IEC 61850 № 2018612517 от 27.12.2017

Заключение

В ходе выполнения данной работы были получены следующие научные и практические результаты:

1) Предложена методика генерации ФБ, реализующих HMI для автоматизации ЦПС МЭК 61850, отличающаяся от существующих возможностью внесения изменений в структуру начальной сети для генерации оптимизированной системы управления, что позволяет значительно упрощать (или усложнять) системы управления, но только если это необходимо.

2) Разработан модифицированный алгоритм Ли, отличающийся от известных возможностью соединения не только узла с узлом, но и узла с уже проложенным соединением, причем в качестве исходной вершины для распространения волны всегда выбирается неподключенная вершина или вершина с наименьшей степенью связности, что позволяет ускорить процесс трассировки пути и уменьшить общее время работы алгоритма, а также повысить читабельность экранных представлений электрических схем в HMI;

3) Предложена методика автоматической генерации функционально-блочной модели системы защиты от перегрузок для цифровых подстанций МЭК 61850, которая отличается выделением главного элемента защиты, отключение которого выводит из строя всю систему, что позволяет использовать алгоритмы достижения консенсуса вследствие чего система будет крайне устойчивой.

4) Разработана библиотека ФБ, моделирующих и визуализирующих типы устройств на ЦПС, а также управляющих ими;

5) Разработан автоматический транслятор, преобразующий исходное SCD-описание ЦПС в систему ФБ, реализующую HMI, который в дальнейшем может интегрироваться в систему автоматизации ЦПС в целом. В рамках разработки данного транслятора:

6) Разработан способ представления описания ЦПС на языке Prolog. Данный метод является универсальным и достаточно легко адаптируемым в случае внесения изменений в стандарт МЭК 61850.

7) Разработан способ представления ФБ на языке Prolog

8) Предложен способ трансляции SCL-Prolog в FB-Prolog на основании правил преобразования.

9) Разработан автоматический транслятор для получения Prolog-представления описания сети Smart Grid на основе SCD - описания.

10) Разработан автоматический транслятор FB-Prolog описания ЦПС в систему ФБ, реализующую HMI. В рамках разработки данного транслятора:

11) Выведена формула для преобразования матрицы узлов в матрицу устройств. Данная формула позволяет размещать на плоскости графовую модель соединительных узлов, а затем превращать её в модель устройств, поскольку не из каждого описание сети можно сразу построить графовую модель устройств.

12) Предложен способ получения графовых моделей сети Smart Grid на основе её описания. Возможно получение 2 вариантов графа: граф соединительных узлов и граф устройств

13) Выведена формула для преобразования матрицы узлов в матрицу устройств. Данная формула позволяет размещать на плоскости графовую модель соединительных узлов, а затем превращать её в модель устройств, поскольку не из каждого описание сети можно сразу построить графовую модель устройств.

В ходе работы над магистерской в Роспатенте были зарегистрированы следующие программы для ЭВМ:

Транслятор SCL-описаний цифровых подстанций МЭК 61850 в набор фактов языка Пролог. №2018613272 от 19.01.2018

Транслятор SCL-описаний в функционально-блочное представление человеко-машинного интерфейса электроподстанций IEC 61850 № 2018612517 от 27.12.2017

Функционально-блочная модель защиты от перегрузок цифровых подстанций МЭК 61850 (в процессе регистрации).

Направлением дальнейших исследований является добавление поддержки различных языков описания подстанций и возможность реализовывать раличные элементы управления, а не только схемы защиты.

Список использованных источников

1) N. Higgins, V. Vyatkin, N. K. C. Nair, and K. Schwarz, \Distributed power system automation with МЭК 61850, МЭК 61499, and intelligent control," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 1, pp. 81-92, Jan 2011.

2) G. Zhabelova and V. Vyatkin, \Multiagent smart grid automation architecture based on МЭК 61850/61499 intelligent logical nodes," IEEE Transactions on Industrial Electronics, vol. 59, no. 5, pp. 2351-2362, May 2012.175

3) C. h. Yang, G. Zhabelova, C.-W. Yang, and V. Vyatkin, \Cosimulation environmentfor event-driven distributed controls of smart grid," IEEE Transactions on Industrial Informatics, vol. 9, no. 3, pp. 1423-1435, Aug 2013.

4) nxtControl, \nxtstudio," 2010. [Online]. Available: https://www.nxtcontrol.com

5) T. Strasser, M. Stifter, F. Andren, D. B. de Castro, and W. Hribernik, \ Applying open standards and open source software for smart grid applications: Simulation of distributed intelligent control of power systems," in 2011 IEEE Power and Energy Society General Meeting, July 2011, pp. 1{8.

6) T. Strasser, M. Rooker, G. Ebenhofer, A. Zoitl, C. Sunder, A. Valentini, and A. Martel, \Framework for distributed industrial automation and control (4diac)," in 2008 6th IEEE International Conference on Industrial Informatics, July 2008, pp. 283-288.

7) J. B_ezivin, \On the uni_cation power of models," Software & Systems Modeling, vol. 4, no. 2, pp. 171{188, May 2005.

8) V. Vyatkin, \Guest editorial: Special section on software engineering in industrial automation," IEEE Transactions on Industrial Informatics, vol. 9, no. 4, pp. 2337-2339, Nov 2013.

9) S. Blair, F. Co_ele, C. Booth, and G. Burt, \An open platform for rapid-prototyping protection and control schemes with МЭК 61850," in 2013 IEEE Power Energy Society General Meeting, July 2013, pp. 1-1.

10) C.-W. Yang, G. Zhabelova, and V. Vyatkin, \Sysgrid: МЭК 61850 and МЭК 61499 standard based engineering tool for smart grid automation design," EAI Endorsed Transactions on Energy Web, vol. 3, pp. 1-8, July 2014.

11) A. J. Wood and B. F. Wollenberg, Power generation, operation, and control. John Wiley & Sons, 2012

12) Y. Cai, T. Huang, E. Bompard, Y. Cao, and Y. Li, \Self-sustainable community of electricity prosumers in the emerging distribution system," IEEE Transactions on Smart Grid, vol. 8, no. 5, pp. 2207-2216, 2017.

13) D. Von Dollen et al., \Report to nist on the smart grid interoperability standards roadmap," Electric Power Research Institute (EPRI) and National Institute of Standards and Technology, 2009.

14) E. A. Lee and S. A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition ed., 2015.

15) V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, et al., "Smart Grid Technologies: Communication Technologies and Standards," Industrial Informatics, IEEE Transactions on, vol. 7(4), pp. 529-539, 2011.

16) МЭК 61850: Communication networks and systems for power utility automation, I. E. Commission, 2009.

17) S. Sendall and W. Kozaczynski, \Model transformation: the heart and soul of model-driven software development," IEEE Software, vol. 20, no. 5, pp. 42-45, Sept 2003.

18) M. Broy, \The 'grand challenge' in informatics: Engineering software-intensive systems," Computer, vol. 39, no. 10, pp. 72-80, 2006.

19) NIST, \Nist framework and roadmap for smart grid interoperability standards, release 3.0," 2014. [Online]. Available: http://www.nist.gov/smartgrid/upload/NIST-SP-1108r3.pdf

20) \МЭК 61850-7-4: Basic communication structure - compatible logical node classes and data object classes," International Standard, First Edition, Geneva, 2009.

21) \МЭК 61499-1: Function blocks-part 1 architecture," International Standard, First Edition, Geneva, vol. 1, 2012.

22) International Standard МЭК 61499. Function blocks for industrial-process measurement and control systems (edition 2.0). Part 1: Architecture / International Electrotechnical Commission. – Geneva, 2005. – 245 p.

23) Dubinin, V. Semantics-Robust Design Patterns for МЭК 61499 / V. Dubinin, V. Vyatkin // IEEE Transactions on Industrial Informatics. – 2012. – Vol. 8, Issue 2. – P. 279–290.

24) International Standard МЭК 61131-3 (edition 2.0): Programmable Controllers / International Electrotechnical Commission. – Geneva, 2003. – 230 p.

25) Apostolov A., Vandiver B. МЭК 61850 GOOSE Applications to Distribution Protection Schemes // Energize - December 2010. – P. 26-29.

26) Yang C.-W., Vyatkin V. Modeling of МЭК 61850 Message Passing for Automatic Generation of Distributed Control // 41st Annual Conference of the IEEE Industrial Electronics Society (МЭКОN 2015), p. 1-6.

27) Vyatkin V. МЭК 61499 Function Blocks for Embedded and Distributed Control Systems Design, Third Edition. – Instrumentation Society of America (ISA), 2016. – 261 p.

28) Web-site nxtStudio (nxtControl). Available at: http://www.nxtcontrol.com/

29) SWI-Prolog [Электронный ресурс]. – http://www.swi-prolog.org/

30) Yang C.-W., Dubinin V., Vyatkin V. Ontology Driven Approach to Generate Distributed Automation Control from Substation Automation Design // IEEE Transactions on Industrial Informatics, 2017, Volume 13, Issue 2, P. 668-679.

31) *Web-site VisualSCL*. Available at: http://ase-visual-scl.software.informer.com/1.0/

32) Norenkov I.P., Manichev V.B. Osnovy teorii i proektirovaniya SAPR [Fundamentals of CAD theory and design] — M.: Vyssh. shk., 1990.— 335 p.

33) Lee C.Y. An Algorithm for Path Connections and Its Applications // IRE Transactions on Electronic Computers, vol. EC-10, no. 3, 1961, pp. 346-365.

34) Войнов, А. С., Дубинин, В. Н., Вяткин, В. В., Янг, Ч. В. Транслятор SCL XML-описаний SMART GRID по стандарту IEC 61850 в код языка prolog // Новые информационные технологии и системы. – 2016. – С. 20-23.

35) Войнов А.С., Сенокосов И. В., Представление функциональных блоков на языке Пролог // Сб. статей V Еже-годной межвузов-ской науч.-практ. конф. «Информа-ционные техноло-гии в науке и обра-зовании. Проблемы и перспективы» (МНПК – 2018), Пенза, 2018

36) Войнов А.С., Сенокосов И. В., Представление функциональных блоков на языке Prolog // Сб. статей V Еже-годной межвузов-ской науч.-практ. конф. «Информа-ционные техноло-гии в науке и обра-зовании. Проблемы и перспективы» (МНПК – 2018), Пенза, 2018

37) Войнов А.С., Дубинин В.Н., Сенокосов И. В., Климкина Л.П., Вяткин В.В. Функционально-блочное моделирование системы защиты от перегрузок цифровых подстанций МЭК 61850//Международный научно-технический форум «Современные технологии в науке и образовании» (СТНО-2018)

38) Транслятор SCL-описаний цифровых подстанций МЭК 61850 в набор фактов языка Пролог. Войнов А.С., Сенокосов И.В., Дубинин В.Н., Свидетельство о регистрации программы для ЭВМ RUS 2018613272 19.01.2018

39) Войнов, А. С., Сенокосов И.В. Синтаксический анализ файлов описания сети Smart Grid и генерация функциональных блоков в среде разработки NxtStudio // Новые информационные технологии и системы (НИТиС 2016) : тр. XII Междунар. науч.-техн. конф. – Пенза, 2016. – С. 20-23.

40) Voinov A., Yang C. W., Vyatkin V. Automatic generation of function block systems implementing HMI for energy distribution automation //Industrial Informatics (INDIN), 2017 IEEE 15th International Conference on. – IEEE, 2017. – С. 706-713.

41) Войнов А. С., Сенокосов И. В., Чен-Вэй Я. Транслятор SCL-описания сетей SMART GRID в системы функциональных блоков //Новые информационные технологии и системы. – 2017. – С. 280-283.

42) Войнов А. С., Чен-Вэй Я. Использование алгоритма ЛИ в трассировке соединений в диаграммах SLD //Новые информационные технологии и системы. – 2017. – С. 283-286.

43) Войнов А.С., Дубинин В.Н., Вяткин В.В., Ч.-В. Янг Методика автоматической генерации систем функциональных блоков, реализующих человеко-машинный интерфейс для автоматизации цифровых подстанций МЭК 61850 //"Известия высших учебных заведений. Поволжский регион. Технические науки" №1, 2018

44) Транслятор SCL-описаний в функционально-блочное представление человеко-машинного интерфейса электроподстанций IEC 61850. Войнов А.С., Дроздов Д.Н., Дубинин В.Н., Сенокосов И.В. Свидетельство о регистрации программы для ЭВМ RUS 2018612517 27.12.2017

ПРИЛОЖЕНИЕ А – СВИДЕТЕЛЬСТВО О РЕГИСТРАЦИИ ПРОГРАММЫ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018613272

Транслятор SCL-описаний цифровых подстанций МЭК 61850 в набор фактов языка Пролог

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Пензенский государственный университет» (RU)***

Авторы: ***Войнов Артем Сергеевич (RU), Сенокосов Илья Владимирович (RU), Дубинин Виктор Николаевич (RU)***

Заявка № **2018610404**
Дата поступления **19 января 2018 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***06 марта 2018 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018612517

Транслятор SCL-описаний в функционально-блочное представление человеко-машинного интерфейса электроподстанций IEC 61850

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Пензенский государственный университет» (RU)***

Авторы: ***Войнов Артем Сергеевич (RU), Дроздов Дмитрий Николаевич (RU), Дубинин Виктор Николаевич (RU), Сенокосов Илья Владимирович (RU)***

Заявка № **2017663675**
Дата поступления **27 декабря 2017 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***19 февраля 2018 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

IEEE 15th International Conference
on Industrial Informatics INDIN´2017
– The Undergoing Industrial Informatics R-Evolution –
24th to 26th July, 2017
Emden, Germany

Certificate of Attendance

This is to certify that

Artem Voinov

has attended the 15th International Conference

on Industrial Informatics 24-26 July 2017, Emden, Germany.

Luis Gomes
General Chair

Kamal Al-Haddad
President
Industrial Electronics Society

Armando Walter Colombo
General Chair

IEEE
Advancing Technology for Humanity

ICS

I2AR

FCT
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЧЕТНАЯ ГРАМОТА

награждаются

А. С. ВОЙНОВ,
И ЧЕН-ВЭЙ ЯШ

за лучший студенческий доклад
на XIV Международной научно-технической конференции
«НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ»
(НИТиС-2017)
секция «Системы автоматизированного проектирования»

Ректор университета

А. Д. Гуляков

2017 год *2017 год*

ФГБОУ ВО «Пензенский государственный университет»
Политехнический институт
Факультет вычислительной техники

Неделя науки 2018

Диплом

НАГРАЖДАЕТСЯ

Войнов Артём Сергеевич
Сенокосов Илья Владимирович

руководитель: Дубинин Виктор Николаевич

победители секции
«Вычислительные системы, сети и комплексы»
V ежегодной межвузовской научно-практической
конференции «Информационные технологии
в науке и образовании.
Проблемы и перспективы»

Ректор ПГУ А. Д. Гуляков

2018 г.