МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра компьютерных технологий
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Гнездиловой Анны Владимировны

Тема работы:

РАЗРАБОТКА АЛГОРИТМА ПРЕОБРАЗОВАНИЯ EDTL-СПЕЦИФИКАЦИЙ В ТРЕБОВАНИЯ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

«К защите допущена»
Заведующий кафедрой,
д.т.н., доцент
Зюбин В.Е. /……..
(ФИО) / (подпись)
«31» мая 2022 г.

Руководитель ВКР
к.ф.-м.н, доцент,
КафКТ ФИТ НГУ
Гаранина Н.О. /……...
(ФИО) / (подпись)
«31» мая 2022 г.

Соруководитель ВКР
к.т.н, доцент
КафКТ ФИТ НГУ
Лях Т.В. /……..
(ФИО) / (подпись)
«31» мая 2022 г.

Новосибирск, 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра компьютерных технологий
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

УТВЕРЖДАЮ
Зав. кафедрой Зюбин В.Е.
(фамилия, И., О.)
…….
(подпись)
«29» октября 2021 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студентке Гнездиловой Анны Владимировны, группы 18201
(фамилия, имя, отчество, номер группы)

Тема: «Разработка алгоритма преобразования EDTL-спецификаций в требования на естественном языке»
(полное название темы выпускной квалификационной работы)

утверждена распоряжением проректора по учебной работе от 29.10.2021 № 0297
Срок сдачи студентом готовой работы «31» мая 2022 г.
Исходные данные (или цель работы):
Разработка алгоритма преобразования табличного и формульного представления EDTL-спецификаций свойств систем управления в соответствующее их представление на естественном языке.
Структурные части работы:
Определение предметной области, обзор существующих решений, определение общей схемы преобразования, разработка NL-шаблонов, разработка таблиц трансляции, формулирование списка требований к программе и описание реализации алгоритма.

Руководитель ВКР
к.ф.-м.н, доцент
КафКТ ФИТ НГУ
Гаранина Н.О./……..
(ФИО) / (подпись)
«29» октября 2021 г.

Задание приняла к исполнению
Гнездилова А.В./……..
(ФИО студента) / (подпись)
«29» октября 2021 г.

Соруководитель ВКР
к.т.н., доцент
КафКТ ФИТ НГУ
Лях Т.В./……..
(ФИО) / (подпись)
«29» октября 2021 г.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

CPS (англ. Cyber-Physical System) – киберфизическая система.

EDTL (англ. Event-Driven Temporal Logic) – управляемая событиями временная логика.

LTL (англ. Linear Temporal logic) – линейная временная логика.

NL (англ. Natural Language) - естественный язык.

CNLs (англ. Controlled Natural Languages) – управляемые естественные языки.

FOL (англ. First-Order Logic) – логика первого порядка.

DSL (англ. Domain-Specific Language) – язык программирования, специализированный для конкретной области применения.

ИАиЭ СО РАН – Институт автоматики и электрометрии Сибирского отделения Российской академии наук.

ВВЕДЕНИЕ

При разработке программного обеспечения одной из важнейших задач является формулирование требований корректности. Для многих промышленных систем требования преимущественно выражаются на естественном языке. Естественный язык (Natural Language, NL) выгоден тем, что его можно использовать во всех областях приложений, а также можно понять всем участникам проекта. Несмотря на его повсеместное использование, недисциплинированное применение NL способно вызвать множество проблем с качеством ПО. Общие проблемы с требованиями, выраженными на естественном языке, включают в себя: плохую тестируемость, неправильную реализацию, многословность, недостаточную спецификацию, неполноту, дублирование, пропуски, сложность, расплывчатость и двусмысленность.

Кроме того, требования часто меняются на протяжении всего срока реализации проекта, пока не будет достигнут консенсус между заинтересованными сторонами. Такие изменения приводят к значительным дополнительным расходам, которые варьируются в зависимости от фазы проекта. При этом нельзя забывать о том, что стоимость исправления проблем, связанных с требованиями, быстро возрастает по мере прохождения этапов разработки программного обеспечения.

Наиболее четкие требования к программе удобно представлять на формальном математическом языке, однако такое написание зачастую понятно лишь разработчику. В результате возникает противоречие между стремлением использовать NL на практике и необходимостью быть более точным и обращаться к формальным языкам.

Для смягчения этого противоречия на базе Института автоматики и электрометрии СО РАН был разработан новый способ формулировки требования, с помощью событийно-управляемой темпоральной логики EDTL

(англ. Event-Driven Temporal Logic). Этот формализм позволяет пользователям описать поведение систем управления в терминах событий и логических операций ввода и вывода, а также позволяет смотреть на систему управления как на “черный ящик” [1]. Он является более простым, чем формальные математические языки представления требований, и допускает более естественное выражение в NL, поскольку основано на событийной природе вычислений.

Как следствие, цель работы – разработать алгоритм преобразования представления EDTL-спецификаций свойств систем управления в соответствующее их представление на естественном языке.

Для достижения этой цели были поставлены следующие задачи:

1. Проанализировать особенности спецификации свойств систем управления на языке EDTL.
2. Сформулировать список требований к программе преобразования.
3. Определить общую схему преобразования.
4. Разработать NL-шаблоны для классов EDTL- требований.
5. Сопоставить EDTL-атрибутам конструкции естественного языка.
6. Реализовать алгоритм преобразования.
7. Эмпирически исследовать качество программы преобразования на корпусе EDTL-требований.

Перевод EDTL-требований на естественный язык послужит вспомогательным средством для повышения качества общения между различными участниками процесса разработки индустриальных систем: заказчиками, инженерами по требованиям, инженерами-разработчиками и другими.

Работа состоит из введения, трех глав и заключения. В первой главе определяется предметная область и рассматриваются существующие решения. Во второй главе анализируется EDTL-специфика и LTL-классификация

требований. Третья глава посвящена описанию работы алгоритма и его реализации.

ОСНОВНАЯ ЧАСТЬ

1. Определение предметной области и обзор существующих решений

1.1. Сфера применения

Фундаментальным элементом реализации концепции Индустрии 4.0 является идея киберфизических систем (CPS) – таких интеллектуальных систем, которые включают в себя инженерные взаимодействующие сети физических и вычислительных компонентов [6].

CPS предоставляют новые возможности для повышения качества жизни и обеспечения технологических достижений в критически важных областях, таких как персонализированное здравоохранение, реагирование на чрезвычайные ситуации, управление транспортными потоками, интеллектуальное производство, оборонная и национальная безопасность, а также энергоснабжение и использование. Как следствие, киберфизические системы имеют большой потенциал для внедрения инновационных приложений и влияния на различные сектора экономики в мире.

Растущее значение систем с интенсивным использованием программного обеспечения в промышленных проектах, необходимость выводить на рынок более инновационные, более индивидуальные и более комплексные системы, а также необходимость делать это быстрее, лучше и с высоким уровнем качества, требуют эффективной инженерии требований. Полные требования, свободные от дефектов, являются основой для успешной разработки системы.

Потенциальные риски должны быть идентифицированы во время разработки требований и должны быть уменьшены как можно раньше, чтобы обеспечить успешное продвижение проекта. Ошибки и пробелы в документах с требованиями должны быть обнаружены на ранней стадии, чтобы избежать утомительных процессов внесения изменений.

Более того, велика стоимость ошибок при разработке требований: чем позже в проекте разработки будет исправлен недочет в требованиях, тем выше будут затраты, связанные с его исправлением. Например, усилия по исправлению дефекта требований до 20 раз выше, если исправление выполняется во время программирования, в отличие от исправления того же дефекта во время разработки требований. Если дефект устраняется во время приемочных испытаний, затрачиваемые усилия могут быть в 100 раз выше [5].

При этом, сложность нахождения оптимального решения увеличивается засчет того, что наиболее четкие требования к программе удобно представлять на формальном математическом языке, несмотря на то, что такое написание в реальности зачастую понятно лишь разработчику. Со стороны заказчика же привычнее использовать свой родной – естественный, язык. Поэтому предполагается, что алгоритм преобразования EDTL-требований в NL решит эту проблему, или, как минимум, увеличит для заказчиков и программистов уровень понимания друг друга, и как следствие, сократит ошибки при разработке киберфизических систем.

В лаборатории был сформирован корпус требований к ряду киберфизических систем под названием CPS 1.0. Он состоит из требований к различным системам управления: трехэтажный лифт, сушилка для рук, турникет и шлюз. Требования, представленные в корпусе, сформулированы инженерами на естественном языке, а также записаны в виде EDTL-таблиц с атрибутами, состоящими из формул или переменных.

1. 2. Существующие решения

Несмотря на увеличивающуюся актуальность вопроса перевода формальных выражений на естественный язык, работа над разработкой алгоритмов, имеющих подобную цель, началась давно. На основании этого перед началом реализации метода преобразования EDTL-требований в NL

были изучены уже существующие работы в этой области исследования с целью познания и применения опыта предыдущих разработок, и в то же время избежания повторения ошибок.

Во время анализа предметной области был изучен разработанный ранее метод, реализующий перевод формул логики первого порядка в предложения естественного языка, название которому дали FOLtoNL [3]. Данный алгоритм был полностью реализован в Jess, оболочке экспертной системы на основе Java, и состоит из системы, основанной на конкретных правилах, и словаря.

При таком подходе логика рассматривается как язык представления знаний, а также используется как средство обратной связи с пользователями. Стоит заметить, что в этом примере преобразование логических формул первого порядка на естественный язык достигается с помощью метода, основанного на конкретных ограниченных правилах, при котором используются возможности сопоставления этих правил с выделенными образцами. Программный модуль на основе правил реализует процесс преобразования, который основан на лингвистическом анализе предложения FOL, а электронный лексический словарь предоставляет лексическую и грамматическую информацию, которая помогает в создании предложений NL.

Одной из проблем такого решения является интерпретация предложений, которые полностью относятся к отрицанию. Более того, еще одно ограничение налагается на использование импликации, которая может встречаться только один раз во входном предложении. Также есть ограничение, заключающееся в том, что в настоящее время не принимается во внимание порядок квантификаторов при вводе пользователем. Наконец, используемая лексика содержит ограниченное количество слов: процесс преобразования охватывает небольшой набор формул FOL.

Стоит отметить, что вышеизложенный пример преобразования формального языка в естественный рассматривал перевод не ограниченных по

смысловому содержанию высказываний логики первого порядка. Поставленная же нами задача учитывает проблему качества естественного языка, а именно его расплывчатость, двусмысленность и неполноту, поэтому результат преобразования формальных выражений теперь ограничен фрагментом естественного языка, в котором содержатся только требования к управляющему программному обеспечению.

С этой же мотивацией были задуманы управляемые естественные языки (CNL), цель их создания - предотвратить проблемы с качеством в документах требований, сохраняя при этом гибкость для написания и передачи требований интуитивным и универсально понятным способом. CNL – это набор предопределенных структур предложений, которые ограничивают синтаксис NL и точно определяют семантику операторов, написанных с использованием этих предопределенных структур.

В частности, в работе [4] был предложен Rimay – управляемый естественный язык, предназначенный для помощи аналитикам в написании функциональных требований. В течении работы над ним авторы ставили следующие цели:

- разработка качественной методологии для систематического определения CNL для функциональных требований - эта методология предназначена для использования во всех областях информационных систем;
- разработка грамматики CNL для представления функциональных требований - эта грамматика основана на опыте в финансовой области, но может быть применима, возможно, с адаптациями, к другим областям информационных систем;
- получение эмпирической оценки CNL (Rimay) на примере промышленного исследования.

Было выяснено, что управляемые естественные языки обеспечивают баланс между удобством использования NL, с одной стороны, и строгостью формальных методов, с другой.

Грамматика Rimay получена в результате качественного исследования, основанного на анализе корпуса требований из 11 различных проектов. В этом исследовании было определено информационное содержание, которое финансовые аналитики должны учитывать в требованиях финансовых приложений. Также была проведена эмпирическая оценка Rimay в реальных условиях. Эта оценка измеряет процент требований, которые могут быть представлены с помощью Rimay. Отмечено, что в среднем 88% требований, которые были оценены в рассматриваемом тематическом исследовании, можно было выразить с помощью Rimay.

Таким образом, описанные выше и некоторые другие теоретические материалы и практические наработки были использованы в работе в качестве отправной точки, так как результаты, полученные в процессе разработки представленных методов подтвердили свою значимость и эффективность.

2. Преобразование EDTL-требований в естественный язык

В лаборатории киберфизических систем Института Автоматики и Электрометрии СО РАН ведутся работы над набором инструментов для решения проблем, возникающих при реализации концепции Индустрии 4.0: созданием теоретических моделей, языковых средств, инструментов для верификации программ и статического анализа. В частности, исследования лаборатории показывают, что основные проблемы корректности, которые возникают в управляющем программном обеспечении, на самом деле, часто связаны не с проблемами несоответствия программы требованиям, а с формулировкой самих требований.

2.1. Синтаксис и семантика EDTL

Разработанный в лаборатории способ представления требований EDTL предназначен для описания поведения систем управления в терминах событий и логических операций ввода и вывода, а также позволяет смотреть на систему управления как на “черный ящик” [2].

EDTL-требование R - это кортеж состоящий из шести атрибутов EDTL: R = (trigger, invariant, final, delay, reaction, release). Атрибуты EDTL-требования – это события системы, ограниченные заданными временными взаимосвязями. В таблице 1 приведена неформальная семантика этих атрибутов.

Таблица 1 – Определения EDTL-атрибутов

Название	Определение EDTL-атрибута
Trigger	Событие, после которого инвариант должен быть истинным, пока не произойдет событие release или reaction; оно также является начальной точкой для тайм-аутов при создании событий final/release (если они есть).

Invariant	Утверждение, которое должно быть истинным после возникновения триггерного события и до событий release или reaction.
Final	Событие, после которого должна произойти реакция с допустимой задержкой; Оно всегда следует за триггером.
Delay	Ограничение времени после финального события, в течение которого должна появиться реакция; реакция на это утверждение должна случиться в пределах допустимой задержки после финального события.
Reaction	Событие, являющееся индикатором выполнения требования.
Release	Событие, которое отменяет требование.

Значения EDTL-атрибутов являются EDTL-формулами, которые, в свою очередь, делятся на два класса: формулы состояний и формулы событий. Неформально формулы состояния задают утверждения о значениях переменных системы в данный момент времени, а формулы событий задают утверждения о событиях, которые происходят или не происходят, то есть об изменении или сохранении значений переменных с предыдущего момента времени.

Определим EDTL-формулы. Пусть p – утверждение, ϕ и ψ EDTL-формулы, тогда формулы состояний:

- *true, false и p* – атомарные EDTL-формулы;
- $\phi \wedge \psi$ – конъюнкция ϕ и ψ;
- $\phi \vee \psi$ – дизъюнкция ϕ и ψ;
- $\neg\phi$ – отрицание ϕ,

а формулы событий с утверждением p:

- $\backslash p$ – это падающий фронт: значение p меняется с истинного на ложное;

- */p* – нарастающий фронт: значение *p* изменяется с ложного на истинное;
- *_p* – низкое установившееся состояние: значение *p* остается равным *false*;
- *~p* – высокое установившееся состоянием: значение *p* остается равным *true*.

EDTL-требование имеет три типа формальных семантик: в виде формулы логики линейного времени LTL, формулы логики первого порядка и в виде конечного автомата Бюхи. В дипломной работе используется LTL-семантика EDTL-требований.

2.2. Синтаксис и семантика LTL

LTL (англ. Linear Temporal Logic) – это модальная временная логика, где модальности относятся ко времени. Она позволяет формулировать свойства исполнимых вычислительных последовательностей системы. LTL состоит из обычной пропозициональной логики, с добавлением темпоральных операторов: унарный оператор *neXt* и бинарный *Until* [7].

Множество базовых формул, которые могут быть выражены в пропозициональной линейной темпоральной логике: Пусть AP – множество атомарных предложений. Тогда:

1. p является формулой для всех $p \in \mathrm{AP}$.
2. Если φ – формула, то φ – формула.
3. Если φ и ψ – формулы, то $\varphi \vee \psi$ – формула.
4. Если φ – формула, то $X\varphi$ – формула.
5. Если φ и ψ – формулы, то $\varphi U \psi$ – формула.

Множество формул, построенных в соответствии с этими правилами, называется формулами LTL. Дизъюнктивный базис логики высказываний создают булевы операторы отрицание и дизъюнкция, через них могут быть

выражены любые другие булевы операторы. Темпоральный базис LTL образуют пара темпоральных операторов *U*, *X*, через которые можно выразить выводимые операторы *F* (Future, «когда-нибудь») и *G* (Globally, «всегда»). *Fp* можно считать сокращением для (*true U p*). *Gp* можно считать сокращением для ¬*F*(¬p). Так как true верно во всех состояниях, формула *Fp* в действительности означает, что *p* выполняется в какой-то момент в будущем. Предположим, что нет момента в будущем, когда выполняется *p*. Тогда *p* выполняется все время. Это объясняет определение *Gp*.

Семантика формул LTL определяется на вычислениях, то есть формула принимает истинностное или ложное значение на бесконечных цепочках состояний, в каждом из которых предикаты имеют конкретное истинностное значение – *true* или *false*. Любая формула линейной темпоральный логики на конкретном вычислении будет либо истинна, либо ложна. Формула считается истинной на вычислении, если она истинна в начальном его состоянии [7]. Громоздкие формулы LTL можно свести к более коротким эквивалентными синтаксическими преобразованиями, сохраняющими семантику формулы.

Определим LTL-семантику для EDTL-требований. Требование EDTL req, состоящее из EDTL-атрибутов: trigger, invariant, final, delay, reaction, release, которые являются формулами состояний, выполняется в системе управления *C*, если и только если следующая формула LTL *Φreq* выполняется в структуре Крипке MC для каждого начального пути:

$$\Phi req = G(trig \wedge \neg rel \rightarrow inv \wedge (G(inv \wedge \neg f\,in) \vee (inv \wedge \neg f\,in)U(rel \vee (f\,in \wedge (inv \wedge \neg del)U(rel \vee rea \wedge inv))))).$$

2.3. LTL-классификация и естественный язык

Атрибуты EDTL-требования могут иметь константные значения *true/false* или непостоянное значение. При подстановке значений *true/false* в семантическую формулу, приведенную выше, формульное выражение может

быть упрощено с помощью эквивалентных синтаксических преобразований. При этом оказывается, что различные EDTL-требования могут иметь одинаковую упрощенную LTL-семантику. Это позволяет формировать классы эквивалентности EDTL-требований. На основе этих выводов на базе института была разработана LTL-классификация [9]. Поскольку требования из одного класса имеют одинаковую семантику, было решено для каждого класса разработать единый шаблон на естественном языке.

Классификация содержит 80 одноэлементных классов и 15 многоэлементных классов, а также классы с тривиальной семантикой true и false. В дипломной работе были рассмотрены только те классы, примеры которых содержатся в используемом корпусе требований CPS 1.0. В таблице 2 представлены LTL-формулы общего вида для классов с встречающимися примерами в корпусе. В третьем столбце этой таблицы содержится информация о количестве примеров требований для каждого класса.

Таблица 2 – LTL-классификация требований из корпуса CPS 1.0

LTL-класс	Общий вид LTL-формулы	Частота встречаемости в корпусе CPS 1.0.
#1A	$G(trig \rightarrow rea)$	9
#2	$G(inv)$	4
#27A	$G(trig \rightarrow (G(\neg fin) \vee (\neg fin\ U\ (rel \vee (fin \wedge rea)))))$	4
#8	$G(trig \rightarrow inv)$	3
#4	$G(\neg trig)$	1
#6A	$G(trig \rightarrow (rel \vee inv))$	1
#13A	$G(trig \rightarrow (inv \wedge rea))$	1
#16A	$G(trig \rightarrow (G(inv \wedge \neg fin) \vee$	1

	((inv ∧ ¬fin) U (rel ∨ (fin ∧ (inv ∧ rea))))))	
#25A	G(trig → (G(inv ∧ ¬fin) ∨ ((inv ∧ ¬fin) U (fin ∧ ((inv ∧ ¬del) U (inv ∧ rea))))))	1
#36A	G(trig → (¬del U rea))	1
#37A	G(trig → (¬del U (rel ∨ rea)))	1

2.4. Требования к программе преобразования EDTL в NL

На основе анализа были выдвинуты следующие требования к программе преобразования:

1. Программа должна быть реализована на языке XTend.
2. Преобразование должно основываться на LTL семантике EDTL требований.
3. При разработке алгоритма преобразования необходимо обеспечить:
   - представление EDTL-требования в виде AST-дерева;
   - описание NL-шаблонов в формате csv;
   - описание таблицы соответствия значений EDTL-атрибутов классам в формате csv;
   - фиксацию результата в формате txt.

В ИАиЭ СО РАН разрабатывается web-IDE EDTL для автоматизированной работы с EDTL-требованиями. Средой разработки web-IDE EDTL была выбрана Eclipse IDE с плагином Xtext, который предоставляет возможность реализации собственного DSL [10].

Также в институте было реализовано ядро web-IDE EDTL с помощью фреймворка Xtext. Ядро включает в себя: синтаксически-ориентированный

редактор EDTL-требований, парсер EDTL-требований в абстрактное синтаксическое дерево, синтаксический анализ, семантический контроль, реализацию макросов и аббревиатур [8].

Для реализации алгоритма преобразования был выбран язык программирования Xtend, который является расширением языка Java и предоставляет методы необходимые для работы с абстрактным синтаксическим деревом [11].

3. Алгоритм преобразования

Таким образом, на основе анализа предметной области и требований к программе преобразования был определен общий вид алгоритма.

В алгоритме были выделены следующие этапы:

1. обработка входного EDTL-требования:
    a. извлекаются значения константных атрибутов;
    b. извлекаются комментарии к неконстантным атрибутам.
2. определение подходящего NL-шаблона:
    a. по значениям константных атрибутов определяется класс EDTL-требования;
    b. по классу EDTL-требования определяется шаблон на естественном языке.
3. подстановка в шаблон подходящих конструкций естественного языка:
    a. используются комментарии, соответствующие не константным атрибутам.

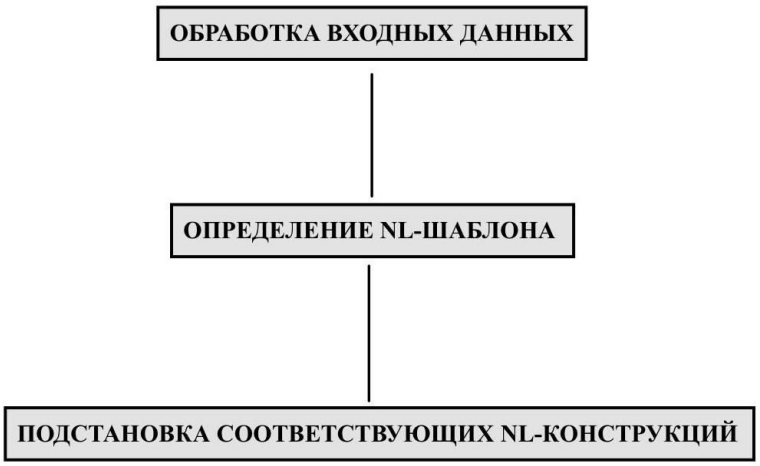

Рисунок 1 – Общая схема алгоритма преобразования

3.1. Разработка NL-шаблонов

Как следствие, для работы алгоритма необходимо было разработать NL-шаблоны. Формирование шаблонов естественного языка осуществлялось на основе анализа корпуса требований следующих систем управления: сушилка для рук, турникет, лифт и шлюз. Данный корпус представлен в приложении.

Для составления NL-шаблонов была использована существующая LTL-классификация, которая определяет в один класс некоторые наборы EDTL-требований, соответсвующих одинаковой LTL-формуле. Так как требования из одного набора имеют одинаковую семантику, были разработаны единые шаблоны для классов на естественном языке.

В таблице 3 в первой колонке перечислены рассмотренные классы требований, во второй колонке значения константных атрибутов этих классов, которые определяют принадлежность требования к этому классу. В третьей колонке представлены шаблоны для соответствующих классов на естественном языке. Данная таблица используется на втором шаге алгоритма преобразования. На третьем шаге алгоритма вместо символьных значений атрибутов в NL-шаблонах подставляются конкретные комментарии, соответствующие этим атрибутам во входном EDTL-требовании.

Таблица 3 – Перечень, разработанных NL-шаблонов

LTL-класс	EDTL-атрибуты	NL-шаблон
#1A	<trigger> = other <release> = false <final> = true <delay> = true <invariant> = true <reaction> = other	После <trigger> происходит <reaction>.
#2	<trigger> = true <release> = false	Система работает, если <invariant>.

	<final> = true <delay> = true <invariant> = other <reaction> = true	
#27A	<trigger> = other <release> = other <final> = other <delay> = true <invariant> = true <reaction> = other	Когда случилось событие <trigger>, после <final> произойдет <reaction>, если не будет <release>.
#8	<trigger> = other <release> = false <final> = true <delay> = true <invariant> = other <reaction> = true	Происходит <invariant>, если <trigger>.
#4	<trigger> = other <release> = false <final> = true <delay> = true <invariant> = true <reaction> = false	Событие <trigger> невозможно.
#6A	<trigger> = other <release> = other <final> = true <delay> = true <invariant> = other <reaction> = true	После <trigger> будет <invariant>, пока не будет <release>.
#13A	<trigger> = other <release> = false <final> = true <delay> = true <invariant> = other <reaction> = other	В то время как <invariant> и <trigger>, то <reaction>.

#16A	<trigger> = other <release> = other <final> = other <delay> = true <invariant> = other <reaction> = other	Если <trigger>, то после <final> будет <reaction>, в случае если не <release>.
#25A	<trigger> = other <release> = false <final> = other <delay> = other <invariant> = other <reaction> = other	Если <trigger> , то до события <final> не будет <reaction>, но затем после события <delay> точно будет <reaction>.
#36A	<trigger> = other <release> = false <final> = true <delay> = other <invariant> = true <reaction> = other	Если <trigger>, то после <delay> точно происходит <reaction>.
#37A	<trigger> = other <release> = other <final> = true <delay> = other <invariant> = true <reaction> = other	После того как <trigger> происходит <reaction> или происходит <delay>, а затем происходит <release>.

3.2. Электронный словарь для разработки комментариев к атрибутам требований

Для корпуса требований были разработаны следующие электронные словари для использования инженерами по требованиям при написании комментариев к неконстантным атрибутам EDTL-требований. Эти комментарии в дальнейшем используются на третьем шаге алгоритма. Словарь для

требований к турникета приведен в таблице 4, а для требований к шлюзу в таблице 5. Полный список разработанных словарей приведен в приложении.

Таблица 4 – Словарь трансляции для требований к турникету

Значение EDTL-атрибута	NL-конструкция
payed.RE	появления с монетоприемника сигнала получения оплаты
open	формируется сигнал открытия турникета
open.RE	появления сигнала открытия турникета
passed(10s)	прошло 10 секунд
¬open	выключение сигнала на открытие
¬open'	выключение сигнала на открытие*
passed(1s)	прошла 1 секунда
passed(9s)	прошло 9 секунд

Таблица 5 – Словарь трансляции для требований к шлюзу

Значение EDTL-атрибута	NL-конструкция
¬atLow	уровни камеры и нижнего бьефа не выровнены
¬lowerGate	нижние ворота закрыты
¬atHigh	уровни камеры и верхнего бьефа не выровнены
¬upperGate	верхние ворота закрыты
¬(upperGate ∧ lowerGate)	исключена одновременная подача сигналов на открытие верхних и нижних ворот шлюза
¬(openHighV ∧ openLowV)	исключена одновременная подача сигналов для верхнего и нижнего клапанов

¬((uGate ∧ lGate) ∨ (uGate ∧ openHighV) ∨ (uGate ∧ openLowV) ∨ ∨ (lGate ∧ openHighV) ∨ (lGate ∧ openLowV) ∨ (openHighV ∧ openLowV))	открыто только что-то одно: либо одни ворота, либо один клапан
lowerGate.FE	закрытие нижних ворот
¬Chmbr2LowLight* ∧ ¬Low2ChmbrLight*	светофор гаснет
upperGate.FE	закрытие верхних ворот
¬LC_LH_LED* ∧ ¬LH_LC_LED*	светофор выключается
lowerGate.RE	открытие нижних ворот
shipInChmbr ∨ shipInHigh	появляется корабль в камере или с противоположной стороны
lowerGate	нижние ворота открыты

3.3. Описание реализации

Программа преобразования написана на языке Xtend и использует разработанное ядро web-IDE EDTL для работы с абстрактным синтаксическим деревом, которое является удобным представлением EDTL-требования.

Входными данными для работы программы являются:

- AST-дерево для исходного EDTL-требования с комментариями на естественном языке, задающими конструкции естественного языка для атрибутов;
- csv-таблица с перечнем разработанных NL-шаблонов;
- csv-таблица с наборами атрибутов значений EDTL-формул (true, false, other), соответствующими конкретным LTL-классам.

Выходными данными для работы программы является файл в формате txt.

Программа, реализующая вышеописанный алгоритм, осуществляет следующие шаги:

1. из входного AST-дерева извлекаются значения константных атрибутов;
2. по значениям константных атрибутов определяется класс EDTL-требования при использовании таблицы LTL-классификации формата csv;
3. по классу EDTL-требования определяется шаблон на естественном языке при использовании таблицы NL-шаблонов формата csv;
4. из AST-дерева извлекаются комментарии, соответствующие не константным атрибутам;
5. комментарии на естественном языке подставляются в шаблон вместо символьных значений атрибутов, и результат подстановки выводится в текстовом файле.

Рассмотрим пример преобразования одного из EDTL-требований для турникета. Выбранное требование из CPS 1.0 представлено в таблице 6. EDTL-атрибуты: trigger, delay и reaction являются неконстантными, следовательно для них во входном файле программы с описанием требования будут добавлены комментарии на естественном языке из электронного словаря трансляции, которые представлены в таблице 7.

Таблица 6 - Таблица со значениями EDTL-атрибутов для одного из требований

Требование на естественном языке	trigger	release	final	delay	invariant	reaction
Сигнал open должен быть в состоянии true не более 10 секунд.	open.RE	false	true	passed (10s)	true	¬open

Таблица 7 - Таблица с комментариями для неконстантных атрибутов

Значение EDTL-атрибута	NL-конструкция на русском языке	NL-конструкция на английском языке
open.RE	появления сигнала открытия турникета	signal ‘open’ turned on
passed(10s)	прошло 10 секунд	10 seconds have passed
¬open	выключение сигнала на открытие	signal ‘open’ turned off

С учетом правил EDTL грамматики объявляем переменные во входном файле с описанием требования, а также задаем значение всем EDTL-атрибутам: TRUE, FALSE или OTHER. На рисунке 2 изображен пример описания требования в текстовом формате языка EDTL.

```
VAR_INPUT
    OPEN : BOOL;
    OTHER2 : INT;
END_VAR

ABBR OTHER1
OPEN
END_ABBR

ABBR OTHER3
!OPEN
END_ABBR

REQ NAME
    TRIGGER := OTHER1; NL: "signal 'open' turned on";
    INVARIANT := TRUE;
    FINAL := TRUE;
    DELAY := OTHER2; NL: "10 seconds have passed";
    REACTION := OTHER3; NL: "signal 'open' turned off";
    RELEASE := FALSE;
END_REQ
```

Рисунок 2 – Пример файла текстового формата языка EDTL с требованием к турникету

Согласно LTL-классификации такое сочетание константных и неконстаных атрибутов соответствует классу #36A. Разработанный NL-шаблон для требования из данного класса представлен в таблице 8.

Таблица 8 - NL-шаблон для класса #36A

LTL-класс	EDTL-атрибуты	NL-шаблон
#36A	<trigger> = other <release> = false <final> = true <delay> = other <invariant> = true <reaction> = other	Если <trigger>, то после <delay> точно происходит <reaction>.

На рисунке 3 представлен результат работы программы, то есть текстовый файл с преобразованным на естественный язык требованием.

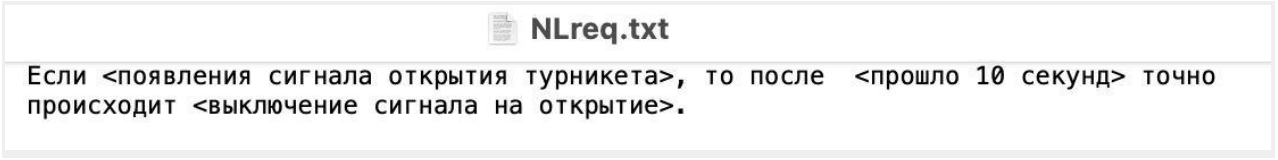

Рисунок 3 – Пример результата работы программы преобразования

ЗАКЛЮЧЕНИЕ

В ходе работы над дипломом проводился анализ предметной области, для EDTL-требований разрабатывались шаблоны на естественном языке, составлялись электронные словари для неконстантных EDTL-атрибутов из корпуса требований CPS 1.0, разрабатывался алгоритм преобразования, на корпусе EDTL-требований эмпирически исследовалось качество его реализации.

Получены следующие результаты:

- разработаны шаблоны на естественном языке для EDTL-требований;
- составлены электронные словари для неконстантных EDTL-атрибутов из корпуса требований;
- разработан и реализован алгоритм преобразования;
- алгоритм преобразования апробирован на корпусе требований.

Планируется разработать шаблоны для других семантических классов EDTL-требований, разработать подход преобразования в естественный язык в случае отсутствия комментариев к неконстантным атрибутам, а также одной из будущих целей является автоматическая разработка электронного словаря на основе предоставленных сведений о предметной области.

Результаты работы были представлены на Международной научной студенческой конференции 2022 в секции «Информационные технологии» в подсекции «Технология искусственного интеллекта» и отмечены дипломом II-степени.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлена с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

Гнездилова Анна Владимировна _____

ФИО студента *Подпись студента*

« ____ »_____20 __г.

(заполняется от руки)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Garanina N., Koznov D. Static Checking Consistency of Temporal Requirements for Control Software //International Conference on Model and Data Engineering. – Springer, Cham, 2021. – С. 189-203.
2. Zyubin V., Anureev I., Garanina N., Staroletov S., Rozov A., Liakh T. Event-Driven Temporal Logic Pattern for Control Software Requirements Specification //International Conference on Fundamentals of Software Engineering. – Springer, Cham, 2021. – С. 92-107.
3. Mpagouli A., Hatzilygeroudis I. A Knowledge-based System for Translating FOL Formulas into NL Sentences //IFIP International Conference on Artificial Intelligence Applications and Innovations. – Springer, Boston, MA, 2009. – С. 157-163.
4. Veizaga A. et al. On systematically building a controlled natural language for functional requirements //Empirical Software Engineering. – 2021. – Т. 26. – №. 4. – С. 1-53.
5. Reiter E., Dale R. Building applied natural language generation systems //Natural Language Engineering. – 1997. – Т. 3. – №. 1. – С. 57-87.
6. Pohl K. Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant. – Rocky Nook, Inc., 2016.
7. Карпов Ю. Г. Model Checking. Верификация параллельных и распределенных программных систем. – БХВ-Петербург, 2010.
8. Козлова А. В. Интегрированная среда разработки для EDTL-требований. МНСК-2022. – В печати.
9. Гетманова А. Н. Разработка транслятора EDTL-требований в формулы логики линейного времени LTL. МНСК-2022. – В печати.

10. Xtext Documentation [Электронный ресурс] // Xtext [сайт]. URL: https://www.eclipse.org/Xtext/documentation/ (дата обращения: 21.04.2022)
11. Xtend Expressions [Электронный ресурс] // Xtend [сайт]. URL: https://www.eclipse.org/xtend/documentation/203_xtend_expressions.html (дата обращения: 21.04.2022)

ПРИЛОЖЕНИЕ А

Корпус требований CPS 1.0.

Требование на естественном языке	trigger	release	final	delay	invariant	reaction
Должна быть исключена одновременная подача сигналов на перемещение лифта вверх и вниз.	Up ∧ Down	false	true	true	true	false
Должна быть исключена одновременная подача сигналов на перемещение лифта вверх и вниз.	true	false	true	true	¬(Up ∧ Down)	true
Двери лифта должны быть закрыты, пока он в движении.	(LED_call1 ∨ LED_button1) ∨ (¬(LED_call2 ∨ LED_button2) ∧ (LED_call0 ∨ LED_button0))	false	true	true	door0..2closed	true
Лифт должен продолжать движение вверх, пока в этом направлении остаются запросы.	onfloor1 ∧ door1closed.RE ∧ (Down.L > Up.L) ∧ (Up.L > 0)	(LED_call1 ∨ LED_b1) ∨ (¬(LED_call2 ∨ LED_b2) ∧ (LED_call0 ∨ LED_b0))	(LED_call2 ∨ LED_b2)	true	true	Up*
Лифт должен продолжать движение вниз, пока в этом направлении остаются запросы.	onfloor1 ∧ door1closed.RE ∧ (Down.L > Up.L) ∧ (Up.L > 0)	(LED_call2 ∨ LED_b2) ∨ (LED_call1 ∨ LED_b1)	(LED_call0 ∨ LED_b0)	true	true	Down*
Если запросов нет, лифт должен остановиться и простаивать.	¬(call0..2_LED ∨ b0..2_LED)	false	true	true	¬Up ∧ ¬Down	¬Up* ∧ ¬Down*
Если стоим на этаже и	¬Up ∧ ¬Down ∧ onfloor0	¬door0closed	call0_LED ∨ b0_LED	true	true	Door0*

есть необработанный запрос на открытие на текущем этаже, то двери должны открыться.						
Когда кабина лифта достигает нужного места на запрошенном этаже, лифт должен открыть дверь.	Up.FE ∨ Down.FE ∧ floor0 ∧ (LED_call0 ∨ LED_b0)	false	true	true	true	door_open*
Светодиод кнопки вызова лифта загорается в момент отпускания кнопки.	call0.FE	false	true	true	true	call0_LED*
Светодиоды кнопок вызова и перемещения лифта на нулевой этаж гаснут в момент закрытия двери на нулевом этаже.	door0closed.FE	false	true	true	true	¬call0_LED ∧ ¬b0_LED
После появления с монетоприемника сигнала получения оплаты немедленно должен быть сформирован сигнал открытия турникета.	payed.RE	false	true	true	true	open
Сигнал open должен быть в состоянии true не более 10 секунд.	open.RE	false	true	passed (10s)	true	¬open
Сигнал open должен быть в состоянии true не более 10 секунд. *(снятие open сторонним агентом)	open.RE	¬open	true	passed (10s)	true	¬open'

Сигнал open должен быть в состоянии true не менее 1 секунды, но не более 10 секунд.	open.RE	false	passed(1s)	passed (9s)	open	¬open'
Если сушилка выключена и появляются руки, то сушилка должна немедленно включиться.	H.RE ∧ ¬D	false	true	true	true	D'
Если сушилка включена и руки есть, то сушилка не будет выключаться.	H ∧ D	false	true	true	true	D'
Если нет рук и сушилка выключена, она не включится, пока не появятся руки.	¬H ∧ ¬D	false	true	true	true	¬D'
Если сушилка включена, то она выключится, если в течение 1 секунды не будет рук.	D ∧ H.FE	H	passed(1s)	true	D	¬D'
Нижние ворота должны быть закрыты, если уровни камеры и нижнего бьефа не выровнены.	¬atLow	false	true	true	¬lowerGate	true
Верхние ворота должны быть закрыты, если уровни камеры и верхнего бьефа не выровнены.	¬atHigh	false	true	true	¬upperGate	true
Должна быть исключена одновременная подача сигналов на открытие верхних и нижних ворот шлюза.	true	false	true	true	¬(upperGate ∧ lowerGate)	true

Должна быть исключена одновременная подача сигналов для верхнего и нижнего клапанов.	true	false	true	true	¬(openHighV ∧ openLowV)	true
Должно быть открыто только что-то одно: либо одни ворота, либо один клапан.	true	false	true	true	¬((uGate ∧ lGate) ∨ (uGate ∧ openHighV) ∨ (uGate ∧ openLowV) ∨ ∨ (lGate ∧ openHighV) ∨ (lGate ∧ openLowV) ∨ (openHighV∧ openLowV))	true
Светофор гаснет после закрытия нижних ворот.	lowerGate.FE	false	true	true	true	¬Chmbr2LowLight* ∧ ¬Low2ChmbrLight*
Светофор гаснет после закрытия верхних ворот.	upperGate.FE	false	true	true	true	¬LC_LH_LED* ∧ ¬LH_LC_LED*
После открытия ворота остаются в открытом состоянии, если не появляется корабль в камере или с противоположной стороны.	lowerGate.RE	shipInChmbr ∨ shipInHigh	true	true	lowerGate	true

ПРИЛОЖЕНИЕ Б

Электронный словарь трансляции для требований к лифту.

Значение EDTL-атрибута	NL-конструкция
Up ∧ Down	одновременный вызов сигналов вверх и вниз
¬(Up ∧ Down)	нет вызова сигнала вверх и нет вызова сигнала вниз
(LED_call1 ∨ LED_b1) ∨ (¬(LED_call2 ∨ LED_b2) ∧ (LED_call0 ∨ LED_b0))	лифт в движении
door0..2closed	двери лифта закрыты
onfloor1 ∧ door1closed.RE ∧ (Down.L > Up.L) ∧ (Up.L > 0)	запрос лифта на движение
(LED_call1 ∨ LED_b1) ∨ (¬(LED_call2 ∨ LED_b2) ∧ (LED_call0 ∨ LED_b0))	нет вызова движения лифта
(LED_call2 ∨ LED_b2)	кнопка вызова лифта вверх нажата
Up*	движение лифта вверх
(LED_call2 ∨ LED_button2) ∨ (LED_call1 ∨ LED_button1)	кнопка вызова лифта вниз нажата
Down*	движение лифта вниз
¬(call0..2_LED ∨ b0..2_LED)	нет никаких вызовов лифта на движение
¬Up ∧ ¬Down	лифт стоит
¬Up* ∧ ¬Down*	лифт стоит*
¬Up ∧ ¬Down ∧ onfloor0	лифт стоит на нулевом этаже
¬door0closed	двери лифта на нулевом этаже уже открыты

call0_LED ∨ button0_LED	запрос на открытие дверей лифта на нулевом этаже
Door0*	открытие дверей лифта
Up.FE ∨ Down.FE ∧ floor0 ∧ (LED_call0 ∨ LED_button0)	лифт приехал на нужный этаж
door_open*	лифт открывает двери
call0.FE	кнопка вызова лифта нажата
call0_LED*	светодиод кнопки вызовы лифта загорается
door0closed.FE	закрытие дверей на нулевом этаже
¬call0_LED ∧ ¬button0_LED	светодиоды кнопок вызова и перемещения лифта на нулевой этаж гаснут

Электронный словарь трансляции для требований к сушилке для рук.

Значение EDTL-атрибута	NL-конструкция
H.RE ∧ ¬D	сушилка выключена и появляются руки
D'	сушилка включена
H ∧ D	сушилка включена и руки есть
¬H ∧ ¬D	нет рук и сушилка выключена
¬D'	сушилка выключена
D ∧ H.FE	сушилка включена и убираются руки
H	появление рук
passed(1s)	прошла 1 секунда
D	сушилка включена*