О ВЕРИФИКАЦИИ ЗАДАНИЙ НА РАЗРАБОТКУ РАБОЧИХ БАЗ ДАННЫХ АСУТП АЭС

Будынкова Е.Р.

АКЦИОНЕРНОЕ ОБЩЕСТВО «АЛЬФА-БАНК»,
Россия, г. Москва, ул. Каланчевская, д. 27

azeevaliza@mail.ru

Байбулатов А.А.

Институт проблем управления им. В.А. Трапезникова РАН,
Россия, г. Москва, ул. Профсоюзная, д.65

bajbulatov@mail.ru

Рассмотрена задача верификации заданий на разработку рабочих баз данных верхнего уровня АСУТП АЭС. Выделено место верификации в жизненном цикле разработки. Предложен алгоритм автоматизации решения задачи в части проверки структуры и содержимого заданий. Подробно изложен один из вариантов реализации алгоритма.

Ключевые слова: верификация заданий, рабочие базы данных, АСУТП.

Введение

Разработка рабочих баз данных (РБД) автоматизированных систем управления технологическими процессами (АСУТП) атомных электростанций (АЭС) носит циклический характер. Каждая новая версия заданий задает новый цикл разработки РБД.

Верификация заданий является важной частью в цикле разработки РБД и обеспечивает выполнение следующих функций [1]:

- выявление ошибок;
- контроль и оценку качества;
- предоставление всем заинтересованным лицам информации о текущем состоянии проекта и характеристиках его результатов.

Основной задачей верификации является проверка соответствия реализации требованиям.

Можно рассмотреть два различных пути выполнения верификации: с помощью эксперта (техническая экспертиза) и автоматизированными методами (статические, формальные). Техническая экспертиза показывает хорошие результаты поиска дефектов на ранних стадиях работы над проектом, но требует человеческих ресурсов и зависит от опыта эксперта.

Формальные методы применимы только к тем свойствам, которые выражены формально в рамках некоторой математической модели, на построение которой также нужен человеческий ресурс, после чего задача верификации может быть автоматизирована.

1 Обзор методов верификации

Можно выделить следующие основные методы верификации.

1.1 Экспертиза

Экспертиза объединяет все методы, где верификация производится людьми. Отличается от других методов верификации использованием реализации, а не модели. Среди плюсов данной группы методов поиск дефектов на ранних стадиях разработки системы. В рамках данного метода обычно проводится проверка наличия свойств у системы согласно составленному экспертом списку.

1.2 Статический анализ

Данная группа методов используется для проверки формализованных правил корректного построения артефактов и поиска часто встречающихся дефектов по некоторым шаблонам.

К минусам данного метода относится способность обнаруживать ограниченное количество типов дефектов.

1.3 Формальные методы

Для формальной верификации требуются:

- модель системы, обычно содержащая множество состояний, которые хранят информацию о значениях переменных, программных счетчиках и т. п., и отношение переходов, которое описывает, как система переходит из одного состояния в другое;
- метод спецификации для выражения требований в формальном виде;

- множество правил доказательства, позволяющих определить, удовлетворяет ли модель сформулированным требованиям [2].

1.3.1 Проверка эквивалентности

Для данного типа формальной верификации требуется, чтобы эталонная модель и эталонная реализация были однотипны, чтобы можно было использовать отношение эквивалентности.

1.3.2 Проверка моделей

Для данного типа формальной верификации требования представляются в виде логических формул, а программы – структурами. Программа удовлетворяет требованиям только тогда, когда соответствующая структура являются моделью соответствующей формулы [3].

1.3.3 Дедуктивный анализ

Для данного типа формальной верификации моделью программы является код на языке с формализованной семантикой (сама программа, если язык программирования формализован, псевдокод или блок-схема), а моделью требований — программный контракт, т.е. пара логических формул: пред- и постусловие; отношение соответствия — полная или частичная корректность программы относительно контракта.

2 Роль верификации заданий в жизненном цикле разработки РБД верхнего уровня АСУТП АЭС

2.1 Жизненный цикл разработки РБД

Жизненный цикл разработки РБД относится к итеративной модели и содержит все время от написания технического задания (ТЗ) до момента прекращения использования РБД.

В жизненном цикле РБД можно выделить следующие этапы:

1. разработка требований к исходным данным для разработки РБД;
2. выдача заданий на разработку РБД;
3. верификация заданий на разработку РБД;
4. разработка РБД в части интеграции со смежными программно-техническими комплексами (ПТК) / системами;
5. ввод РБД в эксплуатацию;
6. поддержка РБД в актуальном состоянии.

Поддержка РБД в актуальном состоянии в свою очередь включает в себя удаление/добавление новых параметров, что приводит к повторению выполнения пунктов 2-6.

2.2 Верификация заданий для разработки РБД

Верификация проверяет соответствие одних создаваемых в ходе разработки и сопровождения программного обеспечения (ПО) артефактов другим, ранее созданным или используемым в качестве исходных данных, а также соответствие этих артефактов и процессов их разработки правилам и стандартам.

Согласно стандарту ISO 9126 можно рассматривать представление о качестве ПО с разных точек зрения: разработчика, руководителя и пользователя. Если рассматривать качество с точки зрения руководителя, так называемое внешнее качество, – соответствие требованиям, в данном случае соответствие исходных данных и их ТЗ.

Для этого нужно построить формальную модель для дальнейшего исследования структурной и семантической согласованности отдельных частей формальной модели между собой.

3 Алгоритм верификации заданий для РБД

3.1 Верификация заданий для РБД

Верификация заданий для РБД включает следующие проверки:

- соответствие наименование исходной БД;
- наличие всех необходимых таблиц, с заданными наименованиями (TBL_A, TBL_B, REF_AUX, REF_YP);
- сверку имени, формата, размера и обязательности полей, согласно требованиям к исходным данным;
- соответствие полей KKS и SIGNAL требованиям документа [4];
- в остальных текстовых полях таблицы должны быть использованы символы из ограниченного списка допустимых и в определенной кодировке.

3.2 Алгоритм верификации KKS-кода

Проверка содержимого полей, в частности KKS-кода, происходит в следующие этапы:

1) Правила проверки обработки: в случае ошибки выводить пользователю информационное сообщение о невозможности продолжения проверки.

2) Обработка KKS-кода: чтобы изменить значение по умолчанию, настройте шаблон следующим образом.

а) Считывание: считывание KKS-кода,

б) разбиение: разбиение KKS-кода на группы для последующего анализа на корректность,

в) проверка: рекурсивная проверка кодовых групп по правилам, начиная с кода системной классификации, при отсутствии необходимого правила формирование выходной строки с описанием ошибки,

3) Записать в файл: записать все неправильные случаи в файл и предоставить анализ результатов.

Обработка KKS-кода осуществляется следующим образом: сначала проверяются группы кодов F1F2F3, если значение этой группы правильное, затем по правилам проверяются те группы, которые зависят от этого конкретного значения F1F2F3, FN всегда зависит от F1F2F3, поэтому эта группа проверяется следующей. Если дополнительных зависимых групп нет, то группы проверяются в следующем порядке: G, F0, A1A2, AN, A3 по стандартным правилам. Стандартные правила – это общие правила для всех KKS-кодов [5].

3.3 Модель представления данных

Для формальной верификации необходимо составить модель системы.

В качестве модели системы можно рассмотреть граф формирования KKS-кода, который можно представить множеством деревьев, часть которых представлена на рис.1.

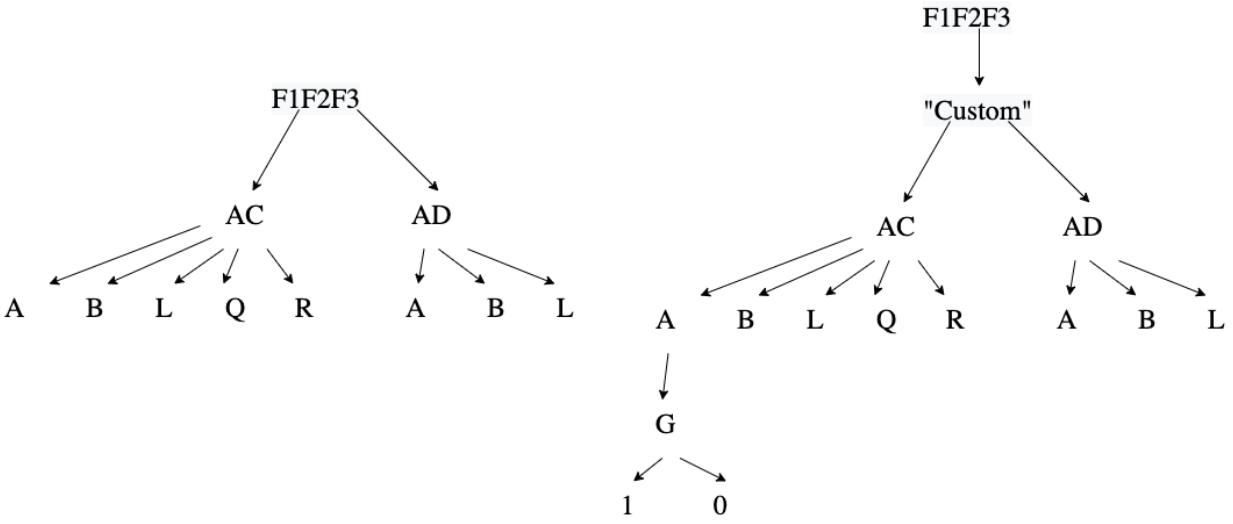

Рис. 1. Часть модели представления KKS-кода

Корневой вершиной отдельно взятого дерева являются значения группы кода F1F2F3, так как именно от этой группы кодов зависят остальные.

4 Пример реализации алгоритма верификации заданий для РБД

В качестве реализации можно предложить автоматизированную проверку заданий.

Алгоритм проверки исходных данных реализован на языке программирования Python 3.7.

Описание программы можно разделить на несколько частей:

- главная область программы;
- класс KKS_code;
- класс SIGNAL_code;
- класс Table_format;
- класс Field_format.

Основная область программы: в этой области осуществляется доступ к базе данных с исходными данными, для которой выполняются все проверки, а также из нее считываются поля SIGNAL и KKS для дальнейшей проверки. Также в этой области читаются все файлы конфигурации, и они

проверяются на правильность. После этого осуществляется доступ к классам KKS_code и SIGNAL code, и поля проверяются напрямую. По окончании верификации подготовленные данные о ее результатах записываются в выходные файлы.

Функции основной области программы представлены в таблице 1.

Таблица 1. Функции основной области программы

Наименование функции	**Входные данные**	**Описание**
check_error_table(table_name)	Имя файла	Проверка конфигурационного файла table_field.json
check_error_KKS_structure (KKS_structure)	Имя файла	Проверка конфигурационного файла KKS_structure.json
check_error_KKS_F1F2F3 _techsys(KKS)	Имя файла	Проверка конфигурационного файла KKS_F1F2F3_techsys.json
check_error_KKS_F1F2F3 _building_constructions(KKS)	Имя файла	Проверка конфигурационного файла KKS_F1F2F3_building_constr uctions.json

Функции класса KKS_code представлены в таблице 2. Цель этого класса - разбить код на группы (G, F0, F1F2F3, FN, A1A2, AN, A3), которые хранятся в одноименных полях, проверить KKS-код и подготовить данные для записи. в выходной файл.

Таблица 2. Функции класса KKS_code

Наименование функции	**Входные данные**	**Описание**
set_message(self,* params)	Список параметров	Создание строки таблицы выходных данных из входных параметров
message_G(self)	-	Создание списка выходных параметров при ошибке в группе G
message_F0(self)	-	Создание списка выходных параметров при ошибке в группе F0
message_F1F2F3(self)	-	Создание списка выходных параметров при ошибке в группе F1F2F3
message_F1F2F3_1(self)	-	Создание списка выходных параметров при ошибке в группе F1F2
message_F1F2F3_2(self)	-	Создание списка выходных параметров при ошибке в группе F3
message_FN(self)	-	Создание списка выходных параметров при ошибке в группе FN
message_A1A2(self)	-	Формирование списка выходных параметров при ошибке в группе A1A2
message_AN(self)	-	Создание списка выходных параметров при ошибке в группе AN
message_A3(self)	-	Создание списка выходных параметров при ошибке в группе A3
check_G_default(self)	-	Проверка группы G по файлу KKS_structure.json
check_F0_default(self)	-	Проверка группы F0 по файлу KKS_structure.json
check_FN_default(self)	-	Проверка группы FN по файлу KKS_structure.json
check_FN_custom(self)	-	Проверка группы FN согласно KKS_F1F2F3_techsys.json
check_A1A2_defa ult(self)	-	Проверить группу A1A2 по файлу KKS_structure.json
check_AN_default(self)	-	Проверка группы AN по файлу KKS_structure.json
check_A3_default(self)	-	Проверка группы A3 по файлу KKS_structure.json

Функции класса SIGNAL_code представлены в таблице 3. Цель этого класса – разбить код на группы (B1B2, BN), которые хранятся в одноименных полях, проверить поле SIGNAL и подготовить данные для записи в выходной файл.

Таблица 3. Функции класса SIGNAL_code

Наименование функции	Входные данные	Описание
check_B1B2(self)	-	Верификация поля SIGNAL

Функции класса Table_format представлены в таблице 4. Цель этого класса – проверить наименование таблицы и вызвать обработку проверки для каждой колонки таблицы.

Таблица 4. Функции класса Table_format

Наименование функции	Входные данные	Описание
check_name(self)	-	Проверка наименования таблицы
check_field(field_item)	Переменная класса Field_format	Проверка

Функции класса Field_format представлены в таблице 5. Цель этого класса – выполнить сверку имени, формата, размера и обязательности полей, согласно требованиям к исходным данным.

Таблица 5. Функции класса Field_format

Наименование функции	Входные данные	Описание
check_name(self)	-	Проверка наименования поля
check_type(self)	-	Проверка формата поля
check_size(self)	-	Проверка размерности поля
check_is_mandatory(self)	-	Проверка поля на обязательность

В качестве развития текущей реализации может быть предложено расширение путем выноса большей части настроек на графический интерфейс для пользователя.

Заключение

Верификация заданий является важным этапом в жизненном цикле разработки РБД верхнего уровня АСУТП АЭС. Выполнение последующего этапа разработки РБД напрямую зависит от корректной реализации этапа верификации. Использование автоматизированных средств позволяет сократить время на обработку данных.

Литература

1. *Кулямин В.В.* Методы верификации программного обеспечения М: Институт системного программирования. 2007. – 111 с.
2. *Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р.* Верификация автоматных программ. СПб: Наука. 2011. – 244 с.
3. Введение в формальные методы верификации программ: учебное пособие / А. С. Камкин. – Москва: МАКС Пресс. 2018. – 272 с.
4. Требования к представлению исходных данных для проектирования рабочих баз данных в части интеграции с ПТК КЭ СУЗ, СКУД, АСРК, СТД ГЦНА, подсистемой инициирующей части УСБ АЗ-ПЗ – АСУЗ ЕР-РР CSS 14R и формирования протоколов текущих событий R188.KK34.0.0.AP.TT.WD1000-01.5. М.: ИПУ РАН. 2020. – 14 с.
5. *Будынкова Е.Р., Байбулатов А.А.* Пример автоматизации проверки исходных данных для проектирования рабочих баз данных АСУТП АЭС // Труды 13-й Международной конференции «Управление развитием крупномасштабных систем» (MLSD'2020, Москва) , под общей редакцией С.Н.Васильева, А.Д.Цвиркуна, М.: ИПУ РАН, 2020. С. 1353-1359.