Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем вычислительных комплексов

Маслов Никита Сергеевич

Разработка инструмента анализа и автоматической проверки требований для информационного обмена в бортовых сетях передачи данных

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
А.В.Герасёв

Москва, 2017

Содержание

Аннотация

В настоящее время в авиационных и корабельных бортовых вычислительных комплексах широко используется мультиплексный канал информационного обмена (МКИО), при этом для корректной работы комплекса передаваемые по каналу данные должны соответствовать протоколам, согласованным и утверждённым разработчиками.

Данная работа посвящена формализации описания требований, предъявляемым к обменам и передаваемым данным, а также задаче автоматизации анализа соответствия передаваемых данных протоколам, записанным в предложенном формальном виде.

В работе приведён перечень проверяемых требований, предложена архитектура решения и создано программное средство для проверки требований на основе инструмента Opermon, разрабатываемого в ЛВК.

1 Введение

1.1 Актуальность задачи

В РВС компоненты взаимодействуют друг с другом, используя согласованные и утвержденные разработчиками системы протоколы. В этих протоколах содержится информация об абонентах (компонентах системы), адресации на каналах, наборе передаваемых сообщений, частотах этих сообщений, их форматах, ограничениях на последовательность, описание содержимого этих сообщений.

Одна из задач интеграционного тестирования РВС - проверка соблюдения этих протоколов компонентами системы. Для этого в среде тестирования описываются тесты, которые выполняют соотвествующие проверки: для принятых от тестируемого компонента сообщений проверяются различные характеристики из упомянутых выше. Некоторые детали соблюдения протоколов можно проверить только так: передать компоненту сообщение, содержащее параметр N, получить от него в ответ сообщение с зависимым параметром M и проверить, что преобразование параметра компонентом выполнено правильно. С другой стороны, некоторые ограничения (например, частотные характеристики) возможно проверить только при регистрации обменов на работающей системе и сравниваи характеристик обменов с ограничениями, описанными в протоколах.

Таким образом, целесообразно иметь инструмент, который по регистрируемым обменам в реальном времени или по записанной трассе сможет проверить соответствие этих обменов требованиям протоколов.

2 Постановка задачи

2.1 Формальная постановка задачи

2.2 Требования к анализатору

К решению задачи предъявляется следующий набор требований, связанных с возможными способами прикладного применения анализатора.

- Простота: описание конфигурации анализатора должно быть простым для понимания, создания и редактирования как вручную пользователем анализатора, так и с применением программного обеспечения для автоматического составления описания протоколов.

- Совместимость: формат описания конфигурации анализатора должен быть совместимым с используемым на текущий момент форматом описания протокола для Анализатора МКИО для возможности переиспользования существующего программного обеспечения для автоматического составления описания протоколов.

- Быстродействие: анализатор должен иметь возможность выполняться достаточно быстро для работы в режиме регистрации обменов в реальном времени при условии выполнении анализа на рабочей станции пользователя Анализатора МКИО и проверки достаточно большого количества требований.

- Расширяемость: анализатор должен быть готовым к возможным расширениям сохранением совместимости с предыдущей версией как минимум на уровне формата описания конфигурации. Это значит, что при добавлении новых типов ограничений пользователь должен иметь возможность использовать описания требований, использованных с предыдущей версией анализатора (возможно, с незначительными строго описанными изменениями).

В качестве дополнительного требования, для удобства работы описание конфигурации анализатора требований должно иметь возможность сохранения во внутреннем пользовательском хранилище Анализатора МКИО для последующего переиспользования без необходимости повторного импорта описания из внешнего файла.

3 Формат описания входных данных

3.1 Общая структура файла описания

Формат входных данных обратно совместим с форматом, использованным в некоторых версиях Opermon для описания сообщений и битовых полей на основе спецификации ПИВ.

Входные данные представляют собой XML-документ следующего содержания:

```
<?xml version="1.0"?>

<piv version="1.1">
    <signals>
        <!-- ...signals -->
        <signal identifier="" type="" signed="" twosComplement="" />
        <!-- ... -->
    </signals>

    <abonents>
        <!-- ...abonents -->
        <abonent identifier="" mil1553_addr="">
            <!-- ...type_messages -->
            <mil1553_messages>
                <!-- controller messages -->
                <mil1553_contrMessage identifier="" direction="" addr
                    ="" subaddr="" numWords="">
                    <!-- ...bitfields -->
                    <bitfield identifier="" firstWord="" firstBit=""
                        numBits="" lowerBitCost="">
                        <restrict type="" value="" level="" />
                        <!-- ... -->
                    </bitfield>
                    <!-- ... -->
                </mil1553_contrMessage>

                <!-- terminal messages -->
                <mil1553_termMessage identifier="" direction=""
                    subaddr="" numWords="">
                    <!-- ...bitfields -->

                </mil1553_termMessage>
            </mil1553_messages>
            <!-- ... -->
```

```
        </abonent>
        <!-- ... -->
    </abonents>

    <restricts>
        <!-- restrics -->
        <restrict messageId="" type="" level="" value="">
            <!-- special restriction params -->
            <param name="" value="" />
            <!-- ... -->
        </restrict>
        <!-- ... -->
    </restricts>
</piv>
```

Листинг 1: Структура файла описания входных данных

Атрибуты описания абонента (тег abonent):

- identifier - индентификатор абонента (строка - идентификатор Си);
- mil1553_addr - адрес абонента на шине MIL STD-1553B.

Атрибуты описания сигнала (тег signal):

- identifier - идентификатор сигнала (строка - идентификатор Си);
- type - тип данных сигнала (например, int, unsigned int, double); приведён для справки;
- signed - является ли сигнал знаковым (true|false, по умолчанию true);
- twosComplement - записывается ли отрицательное значение в дополнительном коде (true|false, по умолчанию false для совместимости со старым форматом ПИВ).

Тег restrict также может содержать элементы внутри, если это требуется для определённого типа ограничений.

Атрибуты сообщения MIL STD-1553B для контроллера (тег mil1553_contrMessage):

- identifier - идентификатор сигнала (строка - индентификатор Си);
- direction - направление (input | output - к/от контроллера);

- addr - адрес ОУ (число от 1 до 31);
- subaddr - подадрес ОУ (целое число от 1 до 30);
- numWords - число слов в сообщении (от 1 до 32).

Атрибуты сообщения MIL STD-1553B для оконечного устройства (тег mil1553_termMessage):

- identifier - идентификатор сигнала (строка - индентификатор Си);
- direction - направление (input | output - к/от контроллера);
- subaddr - подадрес ОУ (целое число от 1 до 30);
- numWords - число слов в сообщении (от 1 до 32).

Стоит заметить, что в описании сообщений MIL STD-1553B для оконечных устройств не указан адрес ОУ-получателя сообщения. Это связано с особенностями внутреннего устройства используемых БД ПИВ. Для формирования полного заголовка сообщения требуется найти два "полусообщения" - сообщения mil1553_termMessage у двух абонентов, где атрибуты identifier для сообщений совпадают.

Атрибуты описания ограничений для сигнала (тег restrict внутри тега bitfield):

- type - тип ограничения (см. Ограничения для сигналов);
- value - значение для ограничения (необязательный параметр), зависит от типа ограничения;
- level - уровень критичности ограничения (info, notice, warning, error).

3.2 Ограничения для сообщений

3.2.1 Частота появления сообщения

Проверяется частота появления сообщения требуемого типа на шине.

Подсчёт частоты происходит вычислением временного интервала между получением текущего и предыдущего сообщений (точное время получения сообщений записано в структуре Exchange).

Параметры ограничения:

- *value* - требуемое значение частоты в герцах (Гц);
- maxDeviation - максимальное отклонение значения частоты (по модулю). Может быть записано в абсолютной величине (без суффикса; например, 1.0), так и в процентах (с суффиксом '%'). По умолчанию - 5%.

Описание ограничения в файле протокола:

```
<restrict messageId="message_id" type="frequency" level="warning"
    value="10.0">
    <param name="maxDeviation" value="1.0" />
</restrict>
```

Листинг 2: Частота появления сообщения

3.2.2 Ошибочные состояния

Проверяются ошибочные состояния сообщения (флаги MIL STD-1553B).

Флаги описаны в структуре Exchange.

Параметров у ограничения нет.

Описание ограничения в файле протокола:

```
<restrict messageId="message_id" type="errors" level="warning" />
```

Листинг 3: Ошибочные состояния сообщения

3.2.3 Последовательность сообщений

Проверяется последовательность сообщений.

Последовательность сообщений задаётся с помощью идентификатора последовательности. Каждое сообщение, входящее в последовательность, имеет в ней порядковый номер. Анализатор проверяет, соблюдается ли порядок появления сообщений в канале согласно порядку номеров.

Стоит заметить, что сообщения не обязательно должны следовать друг за другом; между сообщениями одной последовательности могут появляться другие сообщения. Проверяется порядок именно тех сообщений, которые включены в последовательность.

В параметрах ограничения описывается строковой идентификатор последовательности (идентификатор Си) и номер сообщения в последовательности.

Если после анализа файла протокола выяснится, что номера каких-либо сообщений в последовательности совпадают, пользователь получит сообщение об ошибке и ограничение проверяться не будет. Последовательность пар *(номер, тип_сообщения)* будет упорядочена по номеру.

При получении первого сообщения за время работы анализатор выставит внутренний индекс на номер полученного сообщения. При получении следующего, анализатор сравнит номер следующего полученного сообщения со следующим сообщением в последовательности. При несовпадении будет выведено сообщение об ошибке, при этом внутренний индекс вновь будет сброшен.

Параметры ограничения:

- sequenceId - идентификатор последовательности (строка - идентификатор Си);
- order - номер сообщения в последовательности.

Описание ограничения в файле протокола:

```
<restrict messageId="message1" type="sequence" level="warning">
    <param name="sequenceId" value="sequence1" />
    <param name="order" value="1" />
</restrict>
<restrict messageId="message2" type="sequence" level="warning">
    <param name="sequenceId" value="sequence1" />
    <param name="order" value="2" />
</restrict>

<!-- ... -->

<restrict messageId="messageN" type="sequence" level="warning">
    <param name="sequenceId" value="sequence1" />
    <param name="order" value="N" />
</restrict>
```

Листинг 4: Последовательность сообщений

В случае несовпадения параметра level у описаний одной и той же последовательности, пользователь получит предупреждение, при этом будет выбрано самое сильное значение параметра.

3.2.4 Значение контрольной суммы в полезной нагрузке

Проверяется значение контрольной суммы для некоторого диапазона байт в полезной нагрузке сообщения.

Подразумевается, что в полезной нагрузке сообщения можно специально выделить две последовательности байт, где одна из них - блок данных, а вторая - значение контрольной суммы для этого блока данных.

Параметры ограничения:

- function - функция подсчёта контрольной суммы (crc16 на текущий момент);
- dataStart - номер первого слова последовательности блока данных;
- dataSize - длина последовательности блока данных (количество слов);
- checksumWord - номер слова поля контрольной суммы. Подразумевается, что длина поля контрольной суммы известна по функции подсчёта.

Описание ограничения в файле протокола:

```
<restrict messageId="message_id" type="checksum" level="warning">
    <param name="function" value="crc16" />
    <param name="dataStart" value="0" />
    <param name="dataSize" value="8" />
    <param name="checksumWord" value="8" />
</restrict>
```

Листинг 5: Проверка контрольной суммы

3.3 Ограничения для параметров

3.3.1 Частота обновления параметра

Проверяется минимальная частота обновления значения параметра.

Подсчёт частоты происходит вычислением временного интервала между получением текущего и предыдущего сообщений. Точное время получения значения параметра передаётся вместе со значением в структуре ParameterContainer::ParamValue.

Параметры ограничения:

- *value* - требуемое минимальное значение частоты в герцах (Гц);

Описание ограничения в файле протокола:

```
<signal identifier="signal1">
   <restrict type="minFrequency" level="error" value="1.0" />
</signal>
```

Листинг 6: Частота появления сообщения

3.3.2 Пороговые значения

Проверяется выход значения параметра за некоторое пороговое значение (вверх или вниз).

Для одного параметра можно описать несколько различных пороговых значений (в том числе одного типа) при том, что у ограничений будут различаться уровни критичности (параметры level). В случае, если значение параметра вышло за несколько пороговых значений одного типа (min или max), сообщение будет выведено для порогового значения с самым сильным значением уровня критичности.

Типы ограничений:

- min - минимальное значение параметра;
- max - максимальное значение параметра.

Параметры ограничения:

- *value* - пороговое значение.

Описание ограничения в файле протокола:

```
<signal identifier="signal1">
   <restrict type="max" level="error" value="10.0" />
   <restrict type="max" level="warning" value="9.0" />

   <restrict type="min" level="warning" value="2.0" />
   <restrict type="min" level="error" value="1.0" />
</signal>
```

Листинг 7: Пороговые значения параметра

3.3.3 Равенство константе

Проверяется равенство значения параметра константе (или равенство с допустимой погрешностью).

Параметры ограничения:

- *value* - константа,
- maxDeviation - максимальное отклонение значения от константы в абсолютной величине (по модулю). По умолчанию - 0.

Описание ограничения в файле протокола:

```
<signal identifier="signal1">
   <restrict type="const" level="error" value="10.0" />
</signal>

<signal identifier="signal2">
   <restrict type="const" level="error" value="120.0">
      <param type="maxDeviation" value="2.0" />
   </restrict>
</signal>
```

Листинг 8: Равенство значения параметра константе

3.3.4 Ошибочное значение

Проверяется равенство значения параметра константе, означающей ошибочное состояние параметра. Значение должно быть целочисленным.

Параметры ограничения:

- *value* - ошибочное значение.

Описание ограничения в файле протокола:

```
<signal identifier="signal1">
   <restrict type="error_value" level="error" value="0xDEAD" />
</signal>
```

Листинг 9: Ошибочное значение параметра

3.3.5 Гладкость

Проверяется гладкость параметра - ограничение на максимальную (по модулю) скорость изменения значения параметра.

Скорость изменения параметра измеряется в единицах измерения значения параметра в секунду и считается между двумя соседними событиями обновления значения параметра по формуле:

$v = \frac{val_2 - val_1}{t_2 - t_1}$,

где v - скорость изменения значения параметра, val_2, val_1 - соответственно текущее и предыдущее значения наблюдаемого параметра, t_2, t_1 - время получения текущего и предыдущего значения параметра соответственно (в секундах с момента начала записи трассы).

Параметры ограничения:

- *value* - максимальное значение скорости изменения параметра (по модулю).

Описание ограничения в файле протокола:

```
<signal identifier="signal1">
    <restrict type="smooth" level="error" value="0.25" />
</signal>
```

Листинг 10: Гладкость значения параметра

3.3.6 Связанные параметры

Проверяется соответствие значений нескольких различных параметров, имеющих общую природу.

Например, высота над уровнем моря на борту самолёта может быть получена от модуля позиционирования, использующего GPS, и от модуля, использующего барометрический датчик. Каждый модуль предлагает свой собственный параметр, требуется сравнить эти параметры с заранее заданной погрешностью.

Параметры связываются в группы, определённые с помощью строковых идентификаторов групп. Для каждого отдельного параметра устанавливается максимальная погрешность измерений (в абсолютной или относительной величине), а также “время жизни” - интервал времени, в течение которого значение параметра считается валидным.

При получении нового значения параметра, включённого в группу, происходят следующие действия:

1. Значение параметра и время получения этого значения вносятся в таблицу значений группы;

2. Определяются все “живые” значения параметров из группы (те значения, для времени получения которых верно: $t_{now} \leq t_{recv} + timeout$);

3. Вычисляются абсолютные значения погрешностей для каждого параметра группы;

4. Строится множество отрезков, заданных центральной точкой (значением параметра) и радиусом (величиной погрешности);

5. Строится пересечение полученных отрезков. Если пересечение отрезков пусто - ограничение нарушено.

Параметры ограничения:

- group - идентификатор группы связанных параметров (строка - идентификатор Си);
- measureError - допустимая ошибка измерения параметра. Может быть записана в абсолютной величине (без суффикса) или относительной величине (с суффиксом '%');
- timeout - время жизни последнего полученного значения.

Описание ограничения в файле протокола:

```
<signal identifier="signal1">
    <restrict type="bind" level="error">
        <param name="groupId" value="group1" />
        <param name="measureError" value="5%" />
        <param name="timeout" value="10s" />
    </restrict>
</signal>

<signal identifier="signal2">
    <restrict type="bind" level="error">
        <param name="groupId" value="group1" />
        <param name="measureError" value="0.3" />
        <param name="timeout" value="1s" />
    </restrict>
</signal>
```

Листинг 11: Связанные параметры

В случае несовпадения параметра level у описаний одной и той же группы связанных параметров, пользователь получит предупреждение, при этом будет выбрано самое сильное значение параметра.

3.3.7 Автоинкремент значения параметра

Автоинкремент - свойство целочисленного значения параметра увеличиваться на 1 с заданной частотой. Такие параметры могут использоваться для проверки работоспособности модуля абонента.

Проверяется соблюдение свойства автоинкремента значения параметра.

Параметры ограничения:

- timeout - длина максимального временного интервала, в течение которого значение параметра может оставаться неизменным.

Описание ограничения в файле протокола:

```
<signal identifier="signal1">
    <restrict type="autoincrement" level="error">
        <param name="timeout" value="1s" />
    </restrict>
</signal>
```

Листинг 12: Автоинкремент

4 Структура разработанного решения

Решение задачи было разработано с учётом архитектуры актуальной версии инструмента Opermon и программных компонент, реализующих его функциональность.

Для удобства реализации решение разрабатывалось с учётом возможностей инструментария Qt версии 4 и его библиотек для языка C++ [2].

Рис. 1: Общая схема архитектуры решения

4.1 Внутреннее представление конфигурации анализатора

Пользователь передаёт анализатору данные о накладываемых ограничениях в составе файла описания протокола. При загрузке этого описания происходит преобразование данных во *внутреннее представление*, более пригодное для хранения в ОЗУ и с возможностью быстрого удобного доступа к отдельным элементам описания. При этом, внутреннее представление не содержит никакой информации, которую нельзя получить только из файла описания (взаимно однозначное соответствие описания и его внутреннего представления).

Внутреннее представление описания конфигурации анализатора - это пара массивов A_{exch}, A_{param}, содержащих пары *(идентификатор_ объекта, список_ ограничений)*. Здесь идентификатор объекта - набор признаков конкретного типа обмена или параметра, по которым объект однозначно определяется в системе. Массив A_{exch} содержит описания ограничений для обменов, A_{param} - для параметров.

Список ограничений - это массив, содержащий внутренние представления описаний ограничений, накладываемых на конкретный обмен или параметр.

Внутреннее представление отдельного ограничения - это структрура данных, которая имеет следующий набор полей данных:

- тип ограничения - один из элементов множества допустимых ограничений для заданного типа объекта;
- уровень критичности нарушения - элемент из множества уровней критичности нарушения ограничения *(Info, Notice, Warning, Error)*;
- значение ограничения - поле, хранящее значение атрибута value для данного ограничения;
- список дополнительных параметров - набор пар *(имя_ параметра, значение_ параметра)*. Набор допустимых имён и значений конкретных параметров задан отдельно для каждого типа ограничения.

Необходимость использования общего формата описания ограничений диктуется требованиями простоты и расширяемости: у программиста не должно возникнуть необходимости в редактировании средства загрузки параметров при добавлении новых типов ограничений.

При преобразовании данных из текстового представления во внутреннее представление, значения неперечисляемых типов должны сохраняться в поле специального типа, допускающего последующее преобразование к различным машинным типам данных (как минимум, к целочисленным, строковым и значениям с плавающей точкой). Это диктуется требованием к расширяемости, так как обработчики конкретных ограничений должны иметь возможность использовать данные из описания конфигурации любым необходимым образом. В библиотеке инструментария Qt для хранения таких данных предложен специальный тип QVariant.

4.2 Получение требуемых данных

В данном разделе речь пойдёт о способах получения новых зарегистрированных обменов и значений параметров в рамках архитектуры актуальной версии Анализатора МКИО, а конкретно - в рамках архитектуры компонента tabexchange, реализующего функционал, необходимый для обработки и отображения обменов и параметров.

4.2.1 Получение новых обменов

В tabexchange есть множество объектов, получающих уведомления при регистрации новых обменов. Соответственно, для получения анализатором информации о новых обменах требуется такой объект, который будет передавать информацию о новых обменах непосредственно анализатору.

Уведомления рассылаются с помощью вызова виртуального метода addExchange() у каждого из таких объектов при регистрации нового обмена.

4.2.2 Получение новых значений параметров

В актуальной версии tabexchange не было реализовано функционала для рассылки уведомлений о получении новых значений параметров. Значения параметров запрашивались порциями по сигналу от внешних таймеров. Этого функционала вполне достаточно для реализации отображения значения параметра в соответствующей вкладке, а также для построения графиков зависимости значения параметра от времени.

Тем не менее, для анализатора требований требуется получение информации о новых значениях параметра сразу после получения этого значения из отдельного обмена. Поэтому в рамках данной работы был немного изменён и дополнен класс, реализующий хранение и подсчёт значений параметров при регистрации нового обмена (с полным сохранением уже существовавшего функционала).

В дополнение к существующим полям, в класс был добавлен *фильтр оперативно наблюдаемых параметров* - массив, содержащий идентификаторы параметров, при изменении значений которых требуется рассылка уведомлений.

При получении нового значения параметра из этого списка, происходит рассылка уведомления всем объектам-наблюдателям. Здесь уведомления рассылаются с использованием механизма “сигнал-слот”, реализованного в инструментарии Qt [2].

Обновление фильтра оперативно наблюдаемых параметров происходит при загрузке новых описаний ограничений.

4.3 Обработка данных

4.3.1 Обработчик ограничения

Обработчик ограничения - объект специального типа, задача которого - проверять очередной поступивший обмен (очередное значение параметра) на корректность.

Обработчик ограничения должен иметь метод *check()*, который вызывается при регистрации очередного обмена (очередного значения параметра) типа, соответствующего данному обработчику согласно описанию ограничений. В случае возникновения ошибки анализа, обработчик посылает уведомление *report()*, передавая в качестве аргумента указатель на объект строки отчёта, содержащей собственно сообщение об ошибке, уровень критичности ошибки и (опционально) ссылку на объект обмена, появление которого спровоцировало ошибку.

Отправка уведомления происходит с использованием механизма "сигнал-слот", реализованного в инструментарии Qt.

4.3.2 Ядро анализатора

Ядро анализатора - объект специального типа, задача которого заключается в распределении событий о регистрации новых обменов / новых значений параметров между отдельными обработчиками, а также перенаправлении уведомлений о возникновении ошибок от обработчиков к другим получателям (это сделано для инкапсуляции, чтобы не усложнять граф внешних связей анализатора).

Ядро хранит массивы пар *(ключ_объекта, список_обработчиков)* отдельно для обменов и параметров. Ключ объекта здесь - значение, которое несложно получить или вычислить из описания зарегистрированного обмена (параметра), при этом позволяющее провести быстрый поиск нужного списка обработчиков среди всех пар (например, с применением двоичного дерева поиска - тогда ключи объектов должны быть сравниваемыми, или с применением хэш-таблицы - тогда ключи объектов должны быть хешируемыми).

Ядро анализатора имеет методы для реакции на события "регистрация нового обмена" (*addExchange()*) и "обновление значения параметра". При возникновении этих событий, ядро анализатора вызывает метод *check()* для всех обработчиков данного обмена (параметра), проведя по-

иск требуемого списка среди всех пар по идентификатору полученного объекта. Таким образом гарантируется, что обработчик будет получать на вход только те обмены или параметры, которые соответствуют описанию ограничения, что упрощает написание кода обработчика и ускоряет работу всего анализатора (исключаются бесполезные вызовы обработчиков).

При добавлении очередного ограничения, ядро анализатора запрашивает у фабрики обработчиков очередной объект и помещает его в соответствующий список обработчиков, а также привязывает сигнал *report()* обработчика с собственным слотом для перенаправления. Важно заметить, что привязка сигнала происходит с флагом UniqueConnection, чтобы исключить привязывание сигнала одного и того же обработчика несколько раз (что может возникнуть при добавлении группового обработчика).

При удалении ограничения, ядро удаляет соответствующий элемент из списка и отвязывает сигнал report().

4.3.3 Фабрика обработчиков

Для создания обработчиков используется специальный объект-фабрика. Фабрика в данном случае позволяет решить сразу две проблемы: она позволяет создавать объекты различных типов по текстовому описанию имени, а также следит за необходимостью создания объектов отдельных типов. Например, при конфигурировании обработчика ограничения на связанность значений параметров, объект обработчика требуется создавать только для новых имён групп; если запрашивается обработчик для группы, определённой ранее, фабрика вернёт указатель на ранее созданный объект.

Таким образом, например, для обработчиков групп связанных параметров, фабрика должна хранить массив пар *(идентификатор_ группы, обработчик)*. Для того, чтобы уже созданный обработчик узнал о существовании ещё одного параметра в группе, фабрика при запросе объекта вызывает у нужного обработчика метод *attach()*, принимающий в качестве аргумента ссылку на внутреннее представление очередного ограничения.

Создание обработчика происходит явным вызовом метода-конструктора, принимающего в качестве аргумента ссылку на внутреннее представление ограничения.

4.4 Передача результатов проверки

По результатам проверки, обработчик отдельного ограничения может оповестить систему о возникновении ошибки (или не оповестить, если ошибки обнаружено не было). Оповещение происходит в рамках выполнения метода *check()* обработчика ограничения.

Для оповещения используется механизм “сигнал-слот”, реализованный в инструментарии Qt.

На оповещения от ядра анализатора подписан объект модели таблицы отчёта анализатора. При получении оповещения, новое сообщение об ошибке добавляется в список строк таблицы, после чего запрашивается перерисовка пользовательского интерфейса. Таким образом, реализуется возможность использования анализатора при регистрации обменов в режиме реального времени.

5 Описание реализации

В данном разделе приводится краткое описание реализации решения на языке C++ с использованием возможностей инструментария Qt и его библиотеки для языка C++, а также с использованием инструментария, реализованного в рамках tabexchange.

5.1 Внутреннее представление ограничения

В качестве внутреннего представления отдельных ограничений используются специальные структуры:

- ExchangeRestrictDescription,
- ParamRestrictDescription,

наследующие тип RescrictDescription. Структуры содержат поля для хранения информации согласно описанию внутреннего представления (см. раздел 4.1) с дополнительным полем для идентификатора объекта - текстового идентификатора обмена (параметра). Это нужно для того, чтобы избавить фабрику обработчиков от необходимости взаимодействовать с полной иерархией объектов хранилища конфигурации анализатора.

Дополнительно в рамках этих структур реализованы методы для сохранения и загрузки полей из контейнера QSettings, а также метод для загрузки значений полей при импорте XML-описания протокола.

Структуры определены в файлах tabexchange/analyzercore.[h|cpp].

5.2 Контейнеры для описания ограничений

Для хранения внутреннего представления описания ограничений были реализованы контейнеры

- ExchangeRestrictionContainer,
- ParamRestrictionContainer.

Контейнеры реализованы на базе класса NamedObjectContainer, используемого в tabexchange для создания контейнеров для именованных объектов с возможностью добавления, использования, редактирования (с оповещением классов-пользователей контейнера) элементов, а также автоматического сохранения содержимого контейнера в пользовательский репозиторий Анализатора МКИО и последующей загрузки.

Именами в контейнерах служат текстовые идентификаторы: для обменов - идентификаторы сообщений, совпадающие с таковыми в контейнере описаний сообщений MessageContainer, для параметров - пара *(имя_сообщения, имя_параметра)*, позволяющая также найти описание параметра в контейнере описания параметров ParameterContainer.

В качестве хранимых объектов используются массивы описаний ограничений, соответствующих каждому из типов обменов (параметров).

В рамках данной работы в класс TabExchangeImpl были добавлены методы для получения данных контейнеров.

Создание объектов контейнеров, а также реакция на запросы загрузки и сохранения данных в репозитории происходит в объекте класса TabExchangeImpl.

5.3 Импорт описания ограничений

За импорт файла описания протокола в tabexchange отвечают следующие классы:

- BDPivImporter (файл tabexchange/mppexportimport.cpp) - класс, реализующий разбор XML-файла описания протокола преобразованием в представление QSettings, используемое в Qt-приложениях для хранения конфигурации в виде пар *(ключ, значение)* с иерархией ключей. Объект класса заворачивается в функцию read(), передаваемую QSettings::registerFormat() для обобщённой обработки конфигурационных файлов различных форматов;.

- MPPExportImportModel, MPPExportImportDialog (файлы mppexportimport.cpp, mppexportimport.h в директории tabexchange) - классы, реализующие диалоговое окно импорта конфигурационного файла с возможностью разрешения конфликтов импорта.

В реализации анализатора ограничений были внесены следующие изменения:

- в класс BDPivImporter добавлена обработка XML-тегов <restrict> и <restricts>;

- в метод MPPExportImportDialog::accept() добавлено заполнение контейнеров ExchangeRestrictionContainer и ParamRestrictionContainer в соответствии с описанием ограничений в загружаемом файле описания протокола.

5.4 Контейнеры для сообщений и параметров

В tabexchange реализованы специальные контейнеры для работы с описаниями сообщений и параметров:

- MessageContainer,
- ParameterContainer.

В MessageContainer хранятся описания сообщений, в том числе шаблонные (одному описанию сообщения может соответствовать несколько реальных типов обменов). После незначительных изменений в коде контейнера, он используется в реализации анализатора ограничений для получения внутренних идентификаторов обменов (для каналов MIL STD-1553B, например, командных слов обмена). Внутренние идентификаторы обменов используются ядром анализатора для быстрого определения типа полученного обмена в обработчике события *addExchange()*.

В ParameterContainer хранятся описания параметров, получаемых из сообщений, а также производится вычисление значений параметров при регистрации очередного обмена. В рамках данной работы в реализацию контейнера была добавлена рассылка уведомления о получении нового значения параметра (с использованием механизма “сигнал-слот”, реализованного в инструментарии Qt). Это уведомление используется ядром анализатора для запуска соответствующих обработчиков ограничений.

5.5 Обработчики ограничений

Для классов обработчиков ограничений описаны соответствующие базовые классы (файлы analyzercore.[cpp|h] в директории tabexchange):

- AnalyzerExchangeCheckerBase,
- AnalyzerParamCheckerBase.

Эти классы содержат описание сигнала *report()*, используемого для уведомления о новой строке отчёта, а также прототип чистой виртуальной функции *check()*, принимающей в качестве аргументов:

- для обменов - константный указатель на объект;
- для параметров - текстовый идентификатор параметра и константный указатель на структуру ParamValue, содержащую последнее полученное значение параметра и время его получения.

Все классы, реализующие обработку ограничений для обменов или параметров, должны наследоваться от одного из этих базовых классов.

Реализация обработчиков конкретных ограничений приводится в файлах analyzerfactory.[cpp|h] в директории tabexchange.

5.6 Фабрики обработчиков ограничений

Для создания и раздачи объектов обработчиков ограничений реализованы два класса фабрик объектов:

- ExchangeCheckerFactory,
- ParamCheckerFactory,

реализующие фабрику обработчиков ограничений для обменов и параметров соответственно.

Реализация фабрик обработчиков содержит всю логику создания новых объектов обработчиков и выдачи уже существующих. В частности:

- ExchangeCheckerFactory хранит множество пар *(имя_ последовательности, обработчик)* для обработчиков ограничений на последовательность обменов. Для хранения множества используется тип данных QMap<QString, ExchangeSequenceChecker *>, реализующий ассоциативный массив на основе хэш-таблицы;
- ParamCheckerFactory хранит множество пар *(имя_ группы, обработчик)* для обработчиков ограничений на группы связанных параметров. Для хранения множества используется тип данных QMap<QString, ParamBindChecker *>, реализующий ассоциативный массив на основе хэш-таблицы;
- ParamCheckerFactory хранит множество пар *(ID_ параметра, обработчик)* для обработчиков ограничений на пороговые значения параметров. Это сделано для реализации механизма “подавления” ненужных сообщений внутри обработчика (если значение параметра пересекло несколько пороговых, сообщение будет выведено для порога с самым высоким уровнем критичности). Таким образом, для каждого параметра с наложенными ограничениями на пороговые значения будет создан только один обработчик, следящий за всеми порогами сразу. Для хранения множества используется тип данных QMap<QString, ParamThresholdChecker *>, реализующий ассоциативный массив на основе хэш-таблицы.

5.7 Ядро анализатора

Ядро анализатора ограничений представлено классом AnalyzerCore в файлах analyzercore.[cpp|h] директории tabexchange.

Объект класса AnalyzerCore содержит следующие (скрытые) поля данных:

- *mExchangeCheckers* - отображение множества внутренних идентификаторов обменов на списки обработчиков ограничений. Для хранения отображения используется тип данных QMap<ExchangeSignature, AnalyzerExchangeCheckers>, реализующий ассоциативный массив на основе хэш-таблицы;
- *mParamCheckers* - отображение множества текстовых идентификаторов параметров на списки обработчиков ограничений. Для хранения отображения используется тип данных QMap<QString, AnalyzerParamCheckers> , реализующий ассоциативный массив на основе хэш-таблицы.

Класс AnalyzerCore предоставляет следующий набор методов для управления списками обработчиков ограничений:

- *addExchangeAnalyzer(signature, checker)* - добавляет обработчик ограничения для обменов с заданным внутренним идентификатором (signature);
- *clearExchangeAnalyzers(signature)* - удаляет все обработчики ограничений для обменов с заданным внутренним идентификатором (signature);
- *addParamAnalyzer(paramId, checker)* - добавляет обработчик ограничения для параметра с заданным текстовым идентификатором;
- *clearParamAnalyzers(paramId)* - удаляет все обработчики ограничений для параметра с заданным текстовым идентификатором.

Для получения информации о новых обменах и значениях параметров используются слоты Qt:

- *onAddExchange(exch)* - реакция на регистрацию нового обмена *exch* в системе;
- *onParamValueUpdated(paramId, paramValue)* - реакция на обновление значения параметра с текстовым идентификатором *paramId*.

Для перенаправления сообщений об ошибках от обработчиков к внешним потребителям реализован скрытый (private) слот *onReport(item)*, перенаправляющий уведомление в сигнал *report(item)*.

5.8 Конфигуратор

Для того, чтобы собрать процедуры подготовки ядра анализатора и установку всех связей в одном месте, реализован специальный вспомогательный класс AnalyzerConfigurator, определённый в файлах analyzerconfigurator.[cpp|h].

Объект класса выступает в качестве прослойки между ядром анализатора и набором внешних контейнеров:

- MessageContainer,
- ParameterContainer,
- ExchangeRestrictionContainer,
- ParamRestrictionContainer.

Задачи конфигуратора:

- получение событий об обновлениях контейнеров с конфигурацией анализатора и вызов соответствующих методов ядра;
- конфигурация контейнера параметров для рассылки уведомлений об изменениях значений наблюдаемых параметров;
- преобразование текстовых идентификаторов сообщений во внутренние идентификаторы (с использованием инструментария MessageContainer).

5.9 Строка отчёта анализатора

Каждая строка отчёта анализатора представляется объектом класса AnalyzerReportItem, описанного в файле analyzercore.h.

Строка отчёта анализатора содержит следующие поля данных:

- время возникновения события;
- строка сообщения;
- уровень критичности ошибки;
- указатель на обмен, “спровоцировавший” возникновение ошибки.

Объекты этого класса рассылаются обработчиками ограничений при возникновении ошибок. Объект передаётся в ядро анализатора для пересылки, после чего отправляется модели вкладки отчёта анализатора для отображения ошибки в пользовательском интерфейсе Анализатора МКИО.

5.10 Вкладка анализатора

Вкладка анализатора реализована классами

- AnalyzerReportTab,
- AnalyzerReportModel

в файлах analyzerreporttab.[cpp|h] в директории tabexchange.

Вкладка анализатора представляет собой таблицу с колонками, соответствующими полям данных строки отчёта анализатора (см. раздел 5.9). Данные появляются в таблице автоматически по мере работы анализатора, что позволяет использовать анализатор в режиме регистрации обменов в реальном времени. Очистка отчёта происходит по запросу пользователя (при нажатии на кнопку “Очистить” в графическом интерфейсе пользователя Анализатора МКИО).

Снимок экрана пользовательского интерфейса анализатора ограничений предложен на рисунке 2.

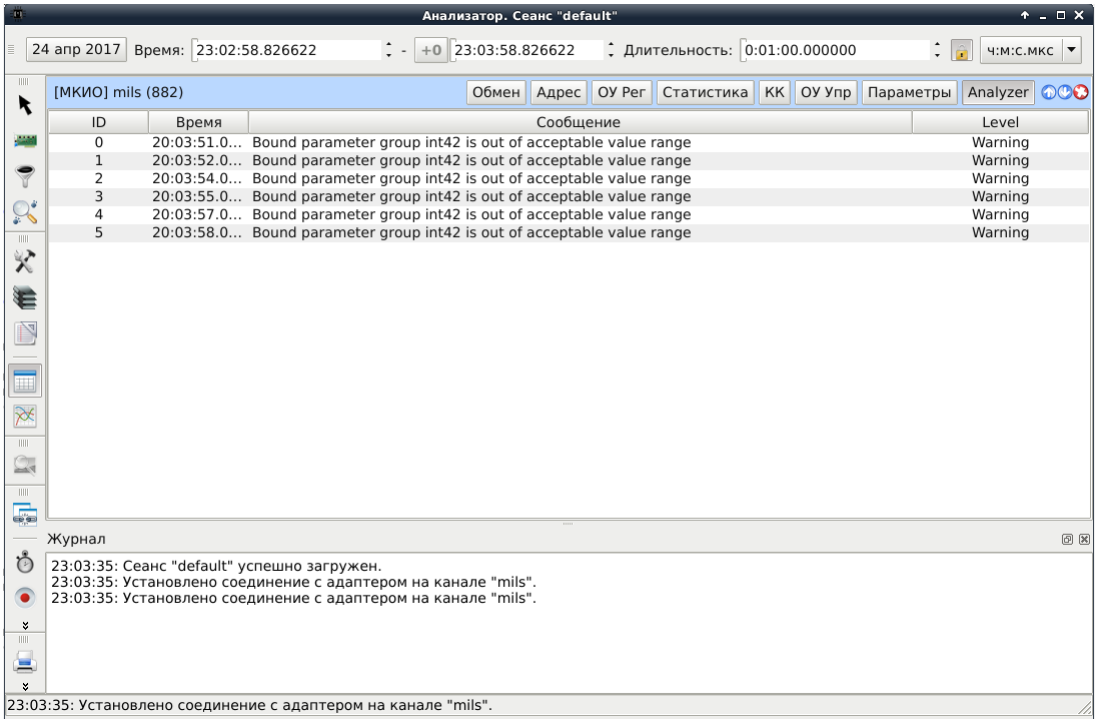

Рис. 2: Пользовательский интерфейс анализатора ограничений

Объект модели вкладки (класса AnalyzerReportModel) является получателем новых зарегистрированных обменов (принимая указатель на

обмен в функции *onAddExchange(exch)*), после чего перенаправляет событие в виде собственного сигнала *addExchange(exch)*.

При создании объектов вкладки создаётся объект ядра анализатора и конфигуратор, а также проводится соединение сигналов и слотов для работы анализатора:

1. сигнал *paramValueUpdated* контейнера параметров привязывается к слоту *onParamValueUpdated* ядра анализатора;

2. сигнал *addExchange* модели вкладки привязывается к слоту *onAddExchange* ядра анализатора;

3. сигнал *report* ядра анализатора привязывается к слоту *getReportItem* модели вкладки.

Создание объектов вкладки происходит в фабрике каналов средства (файл sma/channelfactory_su.cpp).

6 Результаты апробации решения

6.1 Генератор трасс

Для проверки общей работоспособности решения без доступа к реальным или смоделированным ВС полезно провести апробацию средства на данных, полученных с помощью генератора трасс обменов.

Такой генератор был разработан ранее для демонстрационных целей. Для передачи данных агенту Анализатора МКИО используется специальное виртуальное устройство-петля. С помощью генератора можно получить последовательности обменов, подходящие для проверки возможностей анализатора ограничений.

Во время работы генератор непрерывно передаёт набор сообщений определённых типов и содержания с частотой.

Перечень генерируемых параметров приведён в таблице 1. Для каждого параметра приводится функция значения, а также описание обмена, передающего значение параметра (адреса и подадреса отправителя и получателя, формат сообщения, размер сообщения, номер первого слова, номер первого бита и длина битового поля). Параметры по умолчанию целочисленные.

Значение i в определении функции - значение целочисленного счётчика, увеличиваемое на 1 каждые 300 мс.

Легенда таблицы:

- A_{src} - адрес источника сообщения (0-31, либо КК - контроллер канала);
- SA_{src} - подадрес источника сообщения (0-30, либо КК - контроллер канала);
- A_{dst} - адрес получателя сообщения (0-31, либо КК - контроллер канала);
- SA_{dst} - подадрес получателя сообщения (0-30, либо КК - контроллер канала);
- Sz_{msg} - размер сообщения (количество слов);
- St_{word} - номер слова, в котором начинается битовое поле параметра;
- St_{bit} - номер первого бита битового поля параметра в этом слове;
- Sz_{bf} - количество бит в битовом поле параметра;

- Sc - цена младшего бита битового поля.

№	Функция	A_{src}	SA_{src}	A_{dst}	SA_{dst}	Sz_{msg}	St_{word}	St_{bit}	Sz_{bf}	Sc
1	$10^6 * sin(i/10)$	KK	KK	2	1	4	0	0	32	1.0
2	i; 0x42	KK	KK	2	1	4	2	0	16	1.0
3	$10^6 * sin(i/10)$	KK	KK	2	3	4	0	0	32	1.0
4	i; 0x42	KK	KK	2	3	4	2	0	16	1.0
5	$10^6 * sin(i/10)$	KK	KK	2	4	4	0	0	32	1.0
6	i; 0x42	KK	KK	2	4	4	2	0	16	1.0

Таблица 1: Перечень параметров, предоставляемых генератором

В значения некоторых параметров генератором вносятся ошибки. Ошибки для таких параметров описаны в таблице 2. Частота возникновения здесь - количество циклов генератора, определяемых частотой увеличения внутреннего целочисленного счётчика i. $rnd()$ - значение, определяемое с помощью генератора псевдослучайных чисел (функции *random()* стандартной библиотеки языка Си).

№	Ошибочное значение	Частота возникновения
5	побитовое НЕ верного значения	$rnd()$
6	0xDEAD	10

Таблица 2: Ошибочные значения параметров

7 Перспективы развития

Данная работа имеет следующие перспективы для возможного развития:

1. Усовершенствование интеграции с инструментами средства Opermon:
    - установка связей между строками отчёта анализатора и списком обменов для выделения в пользовательском интерфейсе обменов с обнаруженными проблемами;
    - разработка пользовательского интерфейса для редактирования требований к обменам, минуя подготовку файла протокола.
2. Усовершенствование пользовательского интерфейса вкладки с отчётом анализатора:
    - добавление дополнительных колонок данных: “Тип ограничения”, “Канал” и т.п.;
    - возможность сортировки по колонкам в таблице с отчётом анализатора;
    - возможность фильтровать элементы отчёта по содержанию.
3. Расширение сферы применения разработанного средства на другие каналы информационного обмена.

8 Список литературы

1. Государственный стандарт РФ “Интерфейс магистральный последовательный системы электронных модулей” ГОСТ Р 52070-2003

2. Бланшет Ж. Qt 4: программирование GUI на C++, второе издание. КУДИЦ-Пресс, 2008. 736 с.