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

Аннотация

В настоящее время в авиационных и корабельных бортовых вычислительных комплексах широко используется мультиплексный канал информационного обмена (МКИО, известный также как MIL STD-1553B), 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 Анализатор MIL STD-1553B [1], разрабатываемого в Лаборатории вычислительных комплексов факультета ВМК МГУ.

Содержание

1 Введение

1.1 Задача анализа корректности работы РВС

Предметом исследования данной работы являются распределённые вычислительные системы (РВС). *Распределённая вычислительная система* - это набор независимых вычислительных устройств, представляющихся их пользователям единой объединённой системой [2]. Частным случаем РВС, рассматриваемым в данной работе, являются распределённые информационно-управляющие системы реального времени [3].

При разработке и отладке РВС возникает задача проверки корректности функционирования системы. Для решения этой задачи используются следующие подходы [3]:

- *тестирование* - исследование системы с вмешательством в работу. Обычно для этого в каналы связи отправляются управляющие сообщения, после чего анализируется отклик системы. Этот подход часто используется для проверки отдельных узлов, при этом управляющие сообщения моделируют процесс функционирования целой системы;

- *мониторинг* - исследование без вмешательства в работу системы. Проводится запись *трассы* обменов - последовательности сообщений и пауз, наблюдаемых в каналах связи функционирующей системы, после чего проводится анализ трассы на предмет некорректной работы узлов.

После сбора трасс обменов требуется провести их анализ. Это можно сделать следующими способами:

- сравнением записанной трассы с эталоном. Решение этой задачи предложено, к примеру, в работе [4]. Но при анализе корректности функционирования достаточно сложной системы (состоящей из многих устройств), задача подготовки эталонной трассы обменов может оказаться неоправданно сложной;

- проверкой соблюдения трассой набора требований. В этом случае проверяется не вся трасса целиком, а некоторые её характеристики (свойства обменов, значения передаваемых параметров и т.п.). При таком подходе упрощается процесс подготовки исходных данных для проведения проверки (описания отдельных требований имеют достаточно простые формулировки), а также появляется возможность анализировать работу отдельных узлов системы вне зависимости от наличия на линии абонентов, не участвующих в проверке.

Как правило, логика функционирования РВС и описание обмена данными между компонентами оформлены в виде набора утверждённых разработчиками РВС *протоколов*, часто представляющих собой неформальное текстовое описание, непригодное для проведения автоматической проверки. Таким образом, возникает необходимость в *формализации* описания требований.

Поскольку в состав РВС могут входить десятки различных устройств, количество проверяемых требований может оказаться таким, что анализ системы на соответствие им без автоматического средства окажется очень сложным или невозможным, что подтверждает актуальность задачи *автоматического* анализа системы.

Таким образом, при разработке и отладке РВС полезно иметь инструмент для проведения автоматического анализа системы на соответствие требованиям, предъявляемым в протоколах.

Проверку требований можно проводить в рамках функционального тестирования бортового комплекса. Однако, это требует ручного написания тестовых процедур для каждого проверяемого требования с использованием императивной семантики, что достаточно трудоёмко. Более того, анализ требований в рамках функционального тестирования будет достаточно ресурсоёмким процессом в силу особенностей инструментария [7].

1.2 Анализатор информационного обмена

В Лаборатории вычислительных комплексов факультета ВМК МГУ разрабатывается семейство инструментов для мониторинга и анализа информационного обмена на каналах связи РВС. Эти средства активно используются при разработке и отладке бортовых РВС для самолётов и кораблей на различных этапах жизненного цикла систем, в том числе при стендовом моделировании [7]. На сегодняшний день инструментарий нашёл применение в разработках таких организаций, как “ОКБ Сухого” и “ОАК - Центр Комплексирования”.

Среди возможностей средств следует отметить следующие:

- регистрация обменов в реальном времени;
- запись трасс обменов;
- получение значений отдельных параметров, передаваемых в сообщениях. Для бортовых РВС примерами параметров являются скорость, высота над уровнем моря, координаты GPS и т.п.

Данное ПО реализовано на языке C++ с использованием инструментария Qt 4 [11]. Средства имеют графический интерфейс пользователя, поддерживающий отображение обменов и параметров в табличном виде с возможностью поиска и фильтрации, а также построение графиков значений параметров.

В то же время, разработанный инструментарий до сих пор не поддерживает функциональность автоматической поверки информационного обмена на соответствие требованиям.

Стоит отметить, что для различных каналов связи существуют особенности, касающиеся структуры обменов. К примеру, на разных каналах могут использоваться специфические флаги ошибок, не имеющие прямых аналогов в других стандартах. Также может различаться природа передаваемых данных (например, в бортовых РВС самолётов используются отдельные каналы для обмена показаниями датчиков и для передачи видеопотоков). Это значит, что к обменам на разных каналах могут предъявляться несовместимые наборы требований.

Одним из типов каналов, поддерживаемых в инструментарии, является MIL STD-1553B [5][6] - канал с централизованным управлением, активно используемый в бортовых РВС для организации обмена данными произвольных типов.

В рамках данной работы предлагается расширение функциональности разработанного инструментария с целью поддержки автоматической проверки требований на примере средства “Анализатор MIL STD-1553B” [1]. Поддержка проверки требований для обменов на других каналах связи является перспективой развития данной работы.

2 Постановка задачи

Целью работы является проектирование и разработка программных средств для анализа трасс информационного обмена по каналу MIL STD-1553B на предмет соответствия зарегистрированных обменов и передаваемых в обменах данных требованиям, описанным в протоколах. Для этого требуется:

1. Провести анализ существующих протоколов информационного взаимодействия и предложить набор ограничений/характеристик, которые можно проверять в ходе работы анализатора.

2. Предложить формальное описание требований к обмену и параметру. Отдельно учесть возможность автоматической генерации части требований по базе данных протокола информационного взаимодействия (БД ПИВ).

3. Спроектировать средство проведения анализа в рамках инструмента Анализатор MIL STD-1553B.

4. Реализовать инструмент анализа, провести апробацию.

К инструменту анализа предъявляется следующий набор требований, связанных с возможными способами прикладного применения инструмента:

- 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;

- формат описания конфигурации анализатора должен быть совместимым с используемым на текущий момент форматом описания протокола для Анализатора MIL STD-1553B для возможности переиспользования существующего программного обеспечения автоматического составления описания протоколов;

- анализатор должен иметь возможность функционирования в режиме регистрации обмена в реальном времени при условии выполнении анализа на рабочей станции пользователя Анализатора MIL STD-1553B и проверки достаточно большого количества требований. (Это значит, что на современном персональном компьютере при регистрации обмена в реальном времени анализ должен выполняться без заметных задержек с точки зрения пользователя);

- анализатор должен быть готовым к возможным расширениям функционала (определению новых типов требований к обменам и параметрам). При добавлении новых типов ограничений пользователь должен иметь возможность использовать описания требований, подготовленные для предыдущей версии анализатора (возможно, с незначительными изменениями).

3 Ограничения, предъявляемые к информационному обмену

В рамках данной работы требуется опередедить перечень требований, которые предъявляются к информационному обмену в РВС.

Для формулировки перечня требований были проанализированы фрагменты протоколов, описания структуры баз данных протоколов информационного обмена (БД ПИВ), а также были опрошены эксперты “ОКБ Сухого” и “ОАК - Центр Комплексирования”, занимающиеся разработкой и тестированием РВС с использованием каналов связи MIL STD-1553B.

Требования в протоколах описаны в текстовом виде на естественном языке и используются разработчиками для согласования работы отдельных узлов системы, в том числе при проектировании системы несколькими рабочими группами, а также при отладке.

Набор требований к РВС используется также для того, чтобы уточнить понятие корректного состояния системы. Нарушение требований может привести к некорретному функционированию системы. Таким образом, проверка соблюдения требований позволяет определить возможные проблемы и ошибки системы на разных этапах жизненного цикла РВС.

Требования, выделяемые из протоколов, сводятся к набору простых ограничений, накладываемых на информационный обмен.

Можно выделить два основных класса ограничений:

- ограничения, накладываемые на свойства сообщений;
- ограничения, накладываемые на значения параметров.

Среди свойств сообщений важно отметить следующие:

- состояние флагов ошибок, специфических для различных каналов связи;
- частота появления сообщения в канале связи;
- последовательность появления сообщений различных типов в канале связи.

Отдельно рассматриваются ограничения, накладываемые на значения передаваемых параметров. Часто значения параметров - это числовые представления значений каких-либо физических показателей управляемой системы (скорость, координаты и т.п.). Таким образом, на эти

значения можно накладывать естественные ограничения: пороговые значения, ограничение на производную по времени и т.п.

В ходе обсуждения с разработчиками РВС было предложено также проверять корректность значений у группы логически связанных параметров. Например, бортовая РВС самолёта может получить значение высоты над уровнем моря от нескольких источников: от барометрического датчика и от блока вычисления координат с использованием GPS/ГЛОНАСС. Значительное расхождение значений этих параметров может означать неполадки в работе какого-либо из узлов системы, либо ошибку в программном обеспечении. В любом случае, подобная проверка окажется полезной.

Некоторые параметры используются также в сервисных целях. К примеру, достаточно часто устройства бортовых РВС передают специальный параметр-счётчик, значение которого увеличивается на 1 с заданной частотой, что позволяет другим абонентам определить зависание управляющей программы устройства (при этом значение перестанет увеличиваться со временем). Также сервисные параметры могут использоваться для индикации режима работы устройства (например, с помощью кода ошибки или набора флагов). В этом случае можно определить подмножества допустимых значений, определяющие нормальный и ошибочный режимы работы узла.

Таким образом, в результате анализа был составлен следующий перечень возможных ограничений, предъявляемых к обменам:

- для сообщений:
  - проверка частоты появления сообщения данного типа (равенство константе с заданной погрешностью);
  - проверка флагов ошибок, специфических для конкретного канала связи;
  - проверка последовательности появления сообщений различных типов в канале;
  - проверка значения контрольной суммы для набора передаваеых данных;
- для параметров:
  - проверка частоты обновления значения параметра (минимальное значение частоты с заданной погрешностью);
  - проверка выхода значения параметра за пороговые значения (ограничение сверху и снизу);

– проверка на равенство константе с заданной погрешностью;
– проверка на гладкость (ограничение первой производной значения параметра по времени по модулю сверху с заданной погрешностью);
– проверка равенства значений группы связанных параметров с заданными погрешностями;
– проверка на ошибочность значения (для целочисленных параметров);
– проверка автоматического увеличения значения параметра с определённой частотой (с заданной погрешностью частоты, для целочисленных параметров).

Приведённый выше набор ограничений должен покрыть достаточно большой круг задач автоматической проверки системы на корректное функционирование. Тем не менее, в качестве отдельного требования к разрабатываемому решению предлагается обеспечить возможность реализации проверки других типов ограничений, накладываемых на сообщения или параметры.

4 Модель и формальная постановка задачи

4.1 Определения

В данном разделе приводятся определения, используемые в дальнейшем в формальной постановке задачи и описании решения.

Термины, допускающие использование особенностей конкретного типа каналов связи, сформулируем для типа канала MIL STD-1553B. Формулировка соответствующих определений и свойств для сообщений каналов других типов является перспективой развития данной работы.

Обмен - объект $E = (h_E, p_E, t^s_E, l_E)$, являющийся моделью для завершённого атомарного обмена данными между абонентами канала MIL STD-1553B. Понятие обмена соответствует термину *сообщение* ГОСТ Р 52070-2003 [5]. Здесь

- h_E - **заголовок обмена** - набор сервисных данных об обмене. Заголовок обмена содержит формат сообщения, адреса и подадреса абонентов (получателя и отправителя) и размер полезной нагрузки. В терминах ГОСТ Р 52070-2003 заголовок обмена определяется командным словом (КС) (или командными словами для некоторых форматов [5]);

- p_E - **полезная нагрузка обмена** - последовательность слов данных (СД) сообщения;

- t^s_E - время, прошедшее с момента начала регистрации трассы до момента получения первого бита сообщения (*время начала обмена*);

- l_E - время, прошедшее с момента получения первого бита сообщения до момента получения последнего бита сообщения (*длительность обмена*).

Тип обмена - множество обменов $M_h = \{E | h_E = h\}$ с одинаковым заголовком h. Будем говорить, что обмен E является обменом типа M_h, если $E \in M_h$.

Параметр - объект $S = (h, b^{start}_S, sz_S, sc_S)$, где

- h - заголовок обмена. Будем считать, что h содержит размер полезной нагрузки $size(h)$;

- b^{start}_S - номер первого бита в полезной нагрузке обмена из $\bar{E}_h$, содержащего значение параметра;

- sz_S - длина битового поля параметра;

- sc_S - цена младшего бита битового поля параметра.

При этом выполняется условие $b_S^{start}; 0 < b_S^{start} + sz_S < size(h)$.

Значение параметра - функция $value(S, E)$, где $h_S = h_E$, определяющая значение параметра S в обмене E: $value(S, E) = bits(p_E, b_S^{start}, sz_S) * sc_S$ ($bits(b, s, l)$ - функция, выделяющая подпоследовательность из l бит в b, начиная с бита s).

Трасса обменов - последовательность обменов $T = \{E_1, ..., E_N\}$, где выполняется свойство $t_{i+1}^s \geq t_i^s + l_i, i = \overline{1, (N-1)}$ - обмены последовательны во времени и не накладываются друг на друга.

Описание ограничения - неформальное (текстовое) описание ограничения $\bar{R}$, накладываемое на параметр/параметры или тип обмена/типы обменов.

Описание протокола - $P = (M, S, \bar{R}_M, \bar{R}_S)$, где:

- $M = \{M_1, ..., M_n\}$ - множество типов обменов в протоколе;
- $S = \{S_1, ..., S_m\}$ - множество параметров, описанных в протоколе;
- $\bar{R}_M : M_i \rightarrow \bar{R}_{M_i}^j, i = \overline{1, n}, j = \overline{1, p_i}$ - описания ограничений для каждого типа обмена (возможно, пустые);
- $\bar{R}_S : S_k \rightarrow \bar{R}_{S_k}^l, k = \overline{1, m}, l = \overline{1, q_k}$ - описания ограничений для каждого параметра (возможно, пустые).

Ограничение - функция $R : T \rightarrow \{0, 1\}$, отображающая множество трасс обменов на булево множество. Функция R соответствует описанию ограничения $\bar{R}$ и равна 1, если для трассы T данное ограничение выполнено. (Будем считать, что задача определения соответствия трассы ограничению алгоритмически разрешима).

4.2 Формальная постановка задачи

Исходные данные:

- зарегистрированная трасса обменов на канале MIL STD-1553B $T = \{E_1, ..., E_N\}$;
- описание протокола $P = (M, S, \bar{R}_M, \bar{R}_S)$.

Требуется:

- предложить способ формального описания ограничений:

- ограничить множество допустимых описаний ограничений для обменов и параметров (определить классы ограничений);
- предложить способ построения наборов функций проверки ограничений R_M и R_S из описаний ограничений $\bar{R}_M$ и $\bar{R}_S$;

- разработать алгоритмы и реализовать программное средство - **анализатор ограничений**, обеспечивающий решение задачи проверки соответствия трассы T ограничениям, накладываемым описанием протокола P, то есть проверки условия

$$\bigwedge_{M_i \in M} \bigwedge_{R^j_{M_i} \in R_{M_i}} R^j_{M_i}(T) \wedge \bigwedge_{S_i \in S} \bigwedge_{R^j_{S_i} \in R_{S_i}} R^j_{S_i}(T) = 1$$

  При невыполнении условия анализатор должен передать пользователю сообщения об ошибках, соответствующие всем несоблюдённым ограничениям (возможно, несколько сообщений для одного описания ограничения).

5 Структура разработанного решения

Решение задачи было разработано с учётом архитектуры актуальной версии Анализатора MIL STD-1553B и программных компонент, реализующих его функциональность.

Для удобства реализации решение разрабатывалось с учётом возможностей инструментария Qt версии 4 и его библиотек для языка C++ [8].

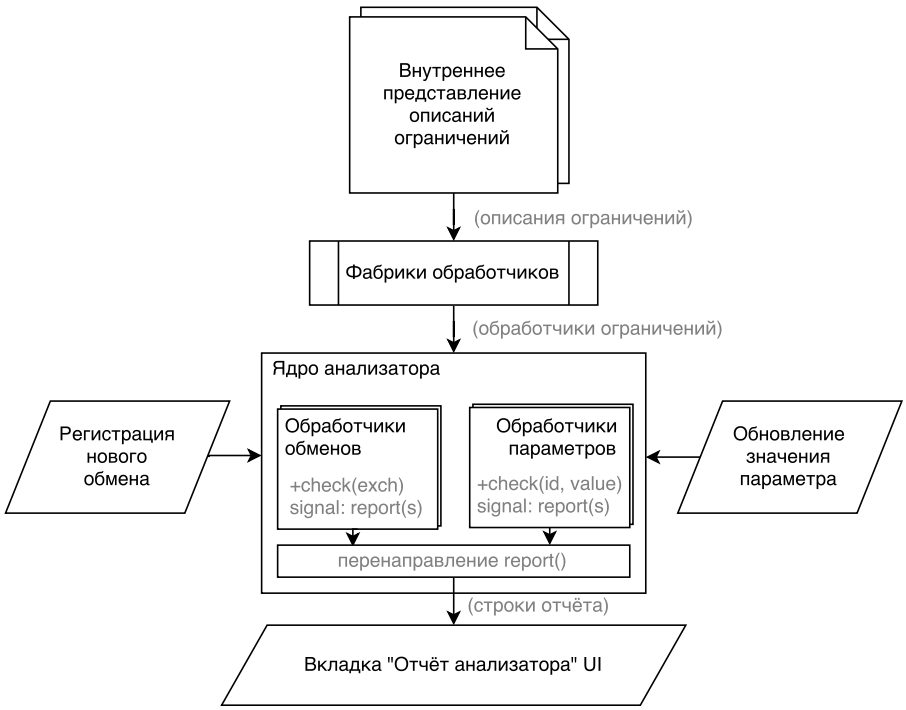


Рис. 1: Общая схема архитектуры решения

5.1 Формат описания входных данных

5.1.1 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, имеющий структуру, описанную в листинге 1 в приложении A

Атрибуты описания абонента (тег abonent):

- identifier - индентификатор абонента (строка - идентификатор Си);

- mil1553_addr - адрес абонента на шине MIL STD-1553B.

Атрибуты описания параметра (тег signal):

- identifier - идентификатор параметра (строка - идентификатор Си);
- type - тип данных параметра (например, int, unsigned int, double); приведён для справки;
- signed - является ли параметр 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 параметра (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 параметра (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 параметра (тег restrict внутри тега bitfield):

- type - тип ограничения (см. раздел 5.7);
- value - значение для ограничения (необязательный параметр), зависит от типа ограничения;
- level - уровень критичности ограничения (info, notice, warning, error).

5.2 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.

5.3 Получение требуемых данных

В данном разделе рассмытриваются способы получения новых зарегистрированных обменов и значений параметров в рамках архитектуры актуальной версии Анализатора MIL STD-1553B, а конкретно - в рамках архитектуры компонента tabexchange, реализующего функционал, необходимый для обработки и отображения обменов и параметров.

5.3.1 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.

5.3.2 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 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 [8].

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

5.4 Обработка данных

5.4.1 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.

5.4.2 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 возможно 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 поиск 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().

5.4.3 Фабрика обработчиков

Для создания обработчиков используется специальный объект-фабрика [10]. 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.

5.5 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.

5.6 Ограничения для сообщений

5.6.1 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%.

Пример описания ограничения в файле протокола приведён в листинге 2 в приложении А.

5.6.2 Ошибочные состояния

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

Флаги описаны в структуре Exchange.

Параметров у ограничения нет.

Пример описания ограничения в файле протокола приведён в листинге 3 в приложении А.

5.6.3 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.

Пример описания ограничения в файле протокола приведён в листинге 4 в приложении А.

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

5.6.4 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-ccitt в данной реализации);
- 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.

Пример описания ограничения в файле протокола приведён в листинге 5 в приложении А.

5.7 Ограничения для параметров

5.7.1 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);

Пример описания ограничения в файле протокола приведён в листинге 6 в приложении А.

5.7.2 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.

Пример описания ограничения в файле протокола приведён в листинге 7 в приложении А.

5.7.3 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.

Пример описания ограничения в файле протокола приведён в листинге 8 в приложении А.

5.7.4 Ошибочное значение

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

Параметры ограничения:

- *value* - ошибочное значение.

Пример описания ограничения в файле протокола приведён в листинге 9 в приложении А.

5.7.5 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).

Пример описания ограничения в файле протокола приведён в листинге 10 в приложении А.

5.7.6 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.

Пример описания ограничения в файле протокола приведён в листинге 11 в приложении А.

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.

5.7.7 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. Для этого сначала определяется актуальность значения (изменилось ли значение за заданный интервал времени), после чего проверяется, увеличилось ли оно на 1 (с учётом маски значения, т.е. 0xFFFF + 1 = 0).

Параметры ограничения:

- timeout - длина максимального временного интервала, в течение которого значение параметра может оставаться неизменным;
- mask - маска значения (по сути, максимальное возможное значение счётчика).

Пример описания ограничения в файле протокола приведён в листинге 12 в приложении А.

6 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.

6.1 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 5.2) 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].

6.2 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 MIL STD-1553B 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.

6.3 Импорт описания ограничений

Импорт файла описания протокола в tabexchange выполняется при помощи следующих классов:

- 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.

6.4 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.

6.5 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.

6.6 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.

6.7 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)*.

6.8 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 посредника 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).

6.9 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 MIL STD-1553B.

6.10 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 6.9). 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 MIL STD-1553B).

Снимок экрана пользовательского интерфейса анализатора ограничений предложен на рисунке 2.

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;

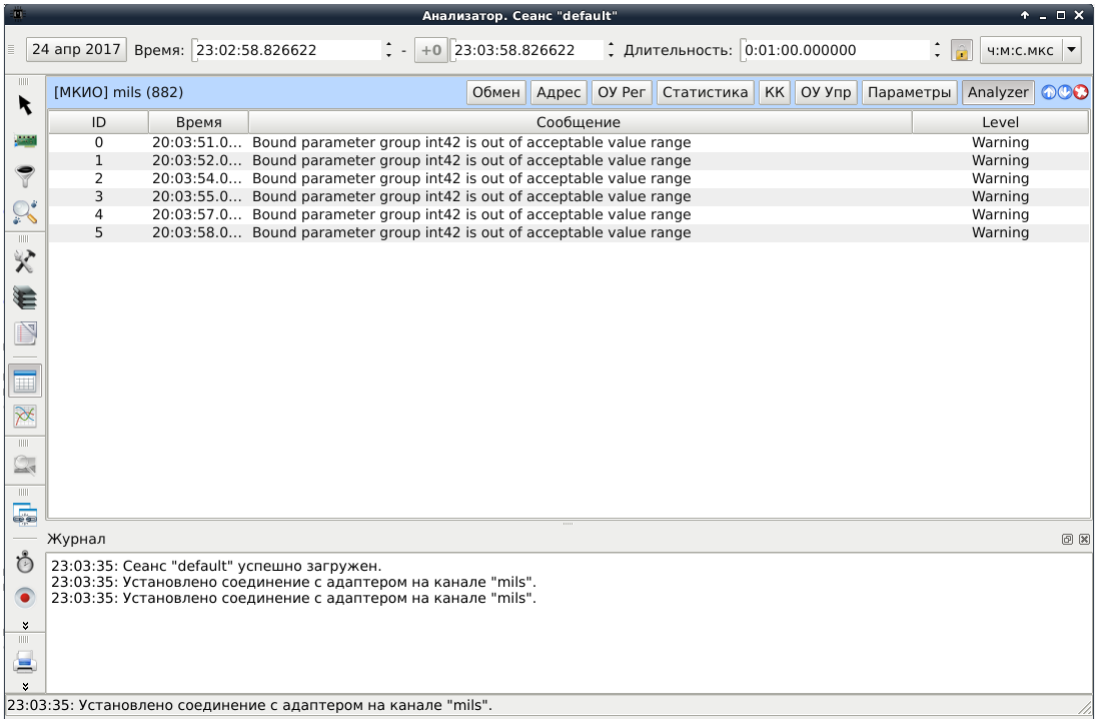


Рис. 2: Пользовательский интерфейс анализатора ограничений

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

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

7 Результаты апробации решения

7.1 Порядок проведения апробации

В силу того, что реальные бортовые РВС являются закрытыми (иногда секретными) разработками, проверка полученного решения на реальных системах не представляется возможным. Поэтому для проведения апробации были использованы тестовые данные, полученные с помощью генератора трасс обменов.

Генератор трасс обменов эмулирует работу нескольких абонентов и контроллера канала на специальном виртуальном адаптере-петле. В ходе работы генератора устройства "обмениваются" значениями некоторых параметров (частичный перечень которых приведён в приложении В).

Изначально генератор использовался для демонстрации работы анализатора при физическом отсутствии подключения к шине. В данной работе генератор был модифицирован для того, чтобы провести проверку корректности обработки всех требований, сформулированных в работе.

В ходе апробации полученного решения проводились следующие проверки:

- проверка корректности работы анализатора - для заданной трассы и описания требований проверялось обнаружение несоответствий и отсутствие ложных срабатываний анализатора;
- проверка быстродействия анализатора для определения возможности работы в режиме регистрации обмена в реальном времени.

Подробное описание характеристик генератора приведено в приложении В.

7.2 Проверка корректности работы анализатора

Апробация возможностей анализатора проводилась на наборах требований, составленных для трасс обменов от генератора. Подробные описания примеров тестовых протоколов приведены в приложении В.

7.2.1 Пример 1

В примере 1 проводится проверка следующих классов требований:

- частота появления сообщения в канале;

- наличие в сообщении флагов ошибок, специфических для MIL STD-1553B;
- значение контрольной суммы в полезной нагрузке сообщения для MIL STD-1553B;
- последовательность появления сообщений из группы;
- минимальная частота обновления значения параметра;
- равенство значения параметра константе;
- проверка равенства значений группы связанных параметров.

7.2.2 Пример 2

В примере 2 проверяются следующие классы требований:

- ошибочное значение параметра;
- автоматическое увеличение значения параметра (автоинкремент).

7.2.3 Пример 3

В примере 3 проверяются следующие классы требований:

- гладкость значения параметра во времени;
- проверка выхода значения параметра за пороговые значения (в протоколе приведено несколько пороговых значений с разными направлениями и уровнями).

7.2.4 Результат апробации

В результате проверки были успешно обнаружены нарушения всех классов требований, описанных в данной работе, при этом ложные срабатывания анализатора не наблюдались.

7.3 Проверка быстродействия анализатора

Для проведения проверки быстродействия анализатора был модифицирован генератор обменов, использованный для предыдущих проверок. Целью модификации было получение большего числа типов сообщений.

Для проведения проверки записана трасса обменов с помощью модифицированного генератора длительностью 20 минут, содержащая $N_{exch} = 108873$ обмена.

Файл протокола содержит описания сообщений, получаемых от модифицированного генератора (всего до 250 типов сообщений). Максимальное количество типов сообщений выбрано в соответствии с таковым в изученных трассах обменов, полученных с реальных РВС. Для каждого из типов сообщений описываются ограничения; количество ограничений для каждого из проведённых тестов приводится в таблице 1.

Для типов сообщений, подразумевающих передачу полезной нагрузки, описаны также параметры и ограничения для них. Количество параметров в протоколе и количество ограничений для каждого из параметров приведены в таблице 1.

Проверка проводилась для разного количества описаний сообщений и параметров, а также разного количества ограничений на сообщения и параметры.

Проект собирался с выставленными флагами оптимизации (перед сборкой была создана глобальная переменная окружения CL_OPTIMIZE=1, используемая системой сборки cvslvk [13]).

Конфигурация тестовой ЭВМ:

- Процессор: Intel Core i5-3320M, 2.6 ГГц;
- ОЗУ: 1024 МБ;
- ОС: Debian Linux 7 (wheezy), 32-разрядная;
- SSD.

Тестирование решения проводилось на виртуальной ЭВМ в среде Oracle VM VirtualBox [12].

Для каждого из описаний требований проводилась операция загрузки трассы. Операция проводилась 5 раз для каждого описания требований, после чего подсчитывалось среднее значение времени загрузки трассы и среднее значение времени, потраченного в контексте анализатора. *Время в контексте анализатора* - время, потраченное в методах-обработчиках:

- AnalyzerCore::onParamValueUpdated(),

- AnalyzerCore::onAddExchange().

Время загрузки трассы оценивается временем выполнения метода

- ExchangeContainer::addSubscriber().

Для 5 запусков решения среднеквадратичное отклонение результатов измерения оценки времени выполнения для обоих случаев не превысило 2.5%.

Результаты проверки приведены в таблице 1.

Легенда таблицы:

- N_{msgs} - общее число сообщений, описанных в файле протокола;
- N_{params} - общее число параметров, описанных в файле протокола;
- M_{msg} - количество ограничений на каждое из описанных сообщений;
- M_{param} - количество ограничений на каждый из описанных параметров;
- t_{total} - среднее значение оценки времени загрузки трассы с данным описанием протокола (в секундах);
- t_a - среднее значение оценки времени, потраченного в контексте анализатора при загрузке трассы с данным описанием протокола (в секундах);
- % - доля времени, потраченного в контексте анализатора, по отношению к общему времени загрузки трассы (в процентах, для средних значений оценки времени).

№ теста	N_{msgs}	N_{params}	M_{msg}	M_{param}	t_{total}, с	t_a, с	%
1	128	128	5	5	3.610892	0.572862	15.86
2	128	240	5	5	5.470608	0.888364	16.23
3	240	128	5	5	3.517034	0.553243	15.73
4	240	240	5	5	5.459458	0.889838	16.29
5	128	128	10	10	4.250614	1.249704	29.4

Таблица 1: Результаты проверки быстродействия решения

В самом худшем случае на анализ одного обмена (с учётом накладных расходов) в среднем приходится $\hat{t_a} = t_{total}/N_{exch} \approx 3.9 * 10^{-5}$ с.

В ходе исследования были изучены трассы обменов на реальных распределённых системах. Используем данные, полученные при изучении одной из этих трасс, для оценки возможности использования анализатора при регистрации обменов в реальном времени.

Средняя длительность обмена в трассе обмена, записанной на реальной системе: $\hat{t_{ex}} \approx 3.6 * 10^{-4}$ с. При длительности записанной трассы $t_T = 50$ с и количестве зарегистрированных обменов $\hat{N_{exch}} = 28452$ получаем среднюю продолжительность паузы между обменами:

$$\hat{p} = \frac{t_T - \hat{t_{ex}} * \hat{N_{exch}}}{\hat{N_{exch}} - 1} \approx 1.39 * 10^{-3} c,$$

что значительно больше средней оценки времени анализа одного обмена, полученной в результате проверки быстродействия в худшем случае:

$$\hat{p} = 1.39 * 10^{-3} > 3.9 * 10^{-5} = \hat{t_a}.$$

Также можно оценить максимальное количество обменов, обрабатываемых анализатором за секунду (с учётом накладных расходов) на предложенной конфигурации тестовой ЭВМ:

$$v = 1/\hat{t_a} \approx 2.56 * 10^3$$

7.3.1 Результаты

Проверка быстродействия показала, что анализатор способен быстро обрабатывать достаточно большое количество ограничений и готов к работе как в режиме анализа записанной трассы, так и при регистрации обменов в реальном времени.

8 Заключение

В рамках настоящей работы было сделано следующее:

- проведён анализ существующих протоколов информационного взаимодействия и предложен набор требований, предъявляемых к обменам и параметрам;
- предложено формальное описание требований к обменам и параметрам;
- разработан метод проверки требований;
- выполнена программная реализация анализатора требований и проведена апробация разработанного инструмента на сгенерированных трассах обменов, а также показана возможность использования инструмента при анализе систем в режиме реального времени.

В качестве перспектив развития данной работы можно предложить следующие:

1. Усовершенствование интеграции с инструментами Анализатора MIL STD-1553B:
    - 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.

Список литературы

1. Анализатор MIL STD-1553B. Свидетельство о государственной регистрации программы для ЭВМ №2013615105 от 28 мая 2013 г.

2. Таненбаум Э. Распределённые системы. Принципы и парадигмы. - СПб.: Питер, 2003. - С. 16

3. Балашов В. В. Курс лекций по системам реального времени: цели, структура, направления развития // Научная конференция "Ломоносовские чтения"2017. — МАКС пресс Москва, 2017. — С. 77.

4. Гордеев А.И. Разработка средств анализа и сравнения результатов регистрации обменов по МКИО. Дипломная работа, н.р. м.н.с. Чистолинов М.В. Москва, 2008.

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

6. MIL-STD-1553B: Digital Time Division Command/Response Multiplex Data Bus. United States Department of Defense, 1978. [PDF] (https://snebulos.mit.edu/projects/reference/MIL-STD/MIL-STD-1553B.pdf)

7. Balashov V. V., Chistolinov M. V., Smeliansky R. L. A family of testbenches to support testing of real-time avionics systems // Proc. of the 5th EUCASS conference for aerospace science (EUCASS 2013). — Munich. Germany, 2013.

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

9. Страуструп Б. Язык программирования C++. М.:Бином, 2012. - С.396

10. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. - СПб. Питер, 2016. 366 с.

11. Qt product page [HTML] (https://www.qt.io/)

12. Oracle VM VirtualBox product page [HTML] (https://www.virtualbox.org/)

13. Описание системы сборки cvslvk [LyX] (https://github.com/redlab-i/cvslvk/blob/master/cvslvk.lyx)

A Примеры описаний ограничений

```
<?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

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

Листинг 2: Частота появления сообщения

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

Листинг 3: Ошибочные состояния сообщения

```
<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

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

Листинг 5: Проверка контрольной суммы

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

Листинг 6: Частота появления сообщения

```
<!-- bitfield 1 -->
    <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" />
<!-- /bitfield 1 -->
```

Листинг 7: Пороговые значения параметра

```
<!-- bitfield 1 -->
    <restrict type="const" level="error" value="10.0" />
<!-- /bitfield 1 -->

<!-- bitfield 2 -->
    <restrict type="const" level="error" value="120.0">
        <param type="maxDeviation" value="2.0" />
    </restrict>
<!-- /bitfield 2 -->
```

Листинг 8: Равенство значения параметра константе

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

Листинг 9: Ошибочное значение параметра

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

Листинг 10: Гладкость значения параметра

```
<!-- bitfield 1 -->
    <restrict type="bind" level="error">
        <param name="groupId" value="group1" />
        <param name="measureError" value="5%" />
        <param name="timeout" value="10s" />
    </restrict>
<!-- /bitfield 1 -->
```

```
<!-- bitfield 2 -->
    <restrict type="bind" level="error">
        <param name="groupId" value="group1" />
        <param name="measureError" value="0.3" />
        <param name="timeout" value="1s" />
    </restrict>
<!-- /bitfield 2 -->
```

Листинг 11: Связанные параметры

```
<restrict type="autoincrement" level="error">
    <param name="timeout" value="1s" />
    <param name="mask" value="0xFF" />
</restrict>
```

Листинг 12: Автоинкремент

В Исходные данные и результаты экспериментов

В.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 MIL STD-1553B 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, эмулируя работу нескольких устройств на канале.

Перечень генерируемых параметров приведён в таблице 2. 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 каждый промежуток времени, заданный в исходном коде генератора (по умолчанию - 300 мс, что даёт частоту обновления значения параметра около 3.3 Гц).

Значение $crc(1:6)$ - значение контрольной суммы сообщения, получаемого абонентом с адресом 1 и подадресом 6 от контроллера канала. Контрольная сумма считается от слов 2-4 и располагается в первом слове сообщения.

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	$10^6 * sin(i/10)$	КК	КК	2	1	4	0	0	32	1.0
2	i; 0x42	КК	КК	2	1	4	2	0	16	1.0
3	$10^6 * sin(i/10)$	КК	КК	2	3	4	0	0	32	1.0
4	i; 0x42	КК	КК	2	3	4	2	0	16	1.0
5	$10^6 * sin(i/10)$	КК	КК	2	4	4	0	0	32	1.0
6	i; 0x42	КК	КК	2	4	4	2	0	16	1.0
7	$crc(0x01 : 0x06)$	1	6	КК	КК	4	0	0	16	1.0

Таблица 2: Перечень параметров, предоставляемых генератором

В значения некоторых параметров генератором вносятся ошибки. Ошибки для таких параметров описаны в таблице 3. 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
7	$crc(1 : 6) + 1$	$rnd()$

Таблица 3: Ошибочные значения параметров

Ниже приводятся примеры описаний протоколов, графики значений параметров, описанных в этих протоколах и результаты работы анализатора для этих описаний протоколов. Проверка работы анализатора проводилась в режиме регистрации обменов в реальном времени.

B.2 Исходные данные и результаты тестирования

B.2.1 Пример 1

В примере 1 наблюдается изменение младших 8 бит значений параметров 4 и 6 из таблицы 2 (назовём полученные параметры 4a и 6a соответственно). Значение параметра 4a постоянно и равно 42. Если для параметра 6a не выполнено условие генерации ошибки, его значение равно значению параметра 4a, иначе - константе 173 (0xAD).

Параметры связываются в группу с идентификатором *int42* со значениями погрешности по умолчанию (5%). Поскольку параметр 6a принимает ошибочные значения со значительным отклонением (больше 5%), в отчёте анализатора должны появиться сообщения о расхождении значений параметров в группе *int42*.

Также в примере 1 проверяются свойства сообщений. Для проверки частоты появления обмена используется сообщение, содержащее параметр 4. Для проверки флагов ошибок выбрано сообщение, для которого генератор иногда не публикует ответного слова. Для проверки значения контрольной суммы используется сообщение, содержащее параметр 7. Последовательность обменов проверяется для сообщений, содержащих параметры 4 и 6. Для демонстрации нарушения порядка следования этих сообщений в регистрации сделана пауза, при этом одно из сообщений повторяется с пропуском второго.

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

<piv version="1.1">
       <signals>
          <signal identifier="int42_2" type="int" signed="false" />
          <signal identifier="int42_3" type="int" signed="false" />
       </signals>

       <abonents>
          <abonent identifier="src0" mil1553_addr="1">
             <mil1553_messages>
                <mil1553_contrMessage identifier="crc_msg"
                   direction="output" addr="1" subaddr="6"
                   numWords="4" />
             </mil1553_messages>
          </abonent>
          <abonent identifier="src1" mil1553_addr="2">
             <mil1553_messages>
                <mil1553_contrMessage identifier="src1_msg2"
```

```
            direction="input" addr="2" subaddr="3" numWords
            ="4">
            <bitfield identifier="int" signal="int42_2"
                firstWord="2" firstBit="20" numBits="8"
                lowerBitCost="1">
                <restrict type="bind" level="warning">
                    <param name="groupId" value="int42" />
                    <param name="timeout" value="1s" />
                </restrict>
                <restrict type="minFrequency" level="
                    warning" value="3.3" />
                <restrict type="const" level="error" value
                    ="66" />
            </bitfield>
        </mil1553_contrMessage>

        <mil1553_contrMessage identifier="src1_msg3"
            direction="input" addr="2" subaddr="4" numWords
            ="4">
            <bitfield identifier="int" signal="int42_3"
                firstWord="2" firstBit="20" numBits="8"
                lowerBitCost="1">
                <restrict type="bind" level="warning">
                    <param name="groupId" value="int42" />
                    <param name="timeout" value="1s" />
                </restrict>
                <restrict type="const" level="error" value
                    ="66" />
            </bitfield>
        </mil1553_contrMessage>
    </mil1553_messages>
</abonent>
<abonent identifier="src2" mil1553_addr="5">
    <mil1553_messages>
        <mil1553_termMessage identifier="buggy" direction
            ="output" subaddr="5" numWords="8">
        </mil1553_termMessage>
    </mil1553_messages>
</abonent>
<abonent identifier="src3" mil1553_addr="6">
    <mil1553_messages>
        <mil1553_termMessage identifier="buggy" direction
            ="input" subaddr="5" numWords="8">
```

```
                </mil1553_termMessage>
            </mil1553_messages>
        </abonent>
    </abonents>

    <restricts>
        <restrict messageId="src1_msg2" type="frequency" level="
            warning" value="3.3" />
        <restrict messageId="buggy" type="errors" level="error" />
        <restrict messageId="crc_msg" type="checksum" level="error
            ">
            <param name="function" value="crc16-ccitt" />
            <param name="dataStart" value="1" />
            <param name="dataSize" value="3" />
            <param name="checksumWord" value="0" />
        </restrict>

        <!-- order -->
        <restrict messageId="src1_msg2" type="sequence" level="
            warning">
            <param name="sequenceId" value="seq1" />
            <param name="order" value="1" />
        </restrict>
        <restrict messageId="src1_msg3" type="sequence" level="
            warning">
            <param name="sequenceId" value="seq1" />
            <param name="order" value="2" />
        </restrict>
    </restricts>
</piv>
```

Листинг 13: Пример описания протокола 1

Рис. 3: График изменения значений параметров (красный - 4а, зелёный - 6а)

[МКИО] mils (243) — Обмен | Адрес | ОУ Рег | Статистика | КК | ОУ Упр | Параметры | Analyzer

ID	Время	Сообщение	Level
0	21:48:19.7...	Sequence seq1 broken: unexpected message 2, expected 1	Warning
1	21:48:19.7...	Exchange Exchange(21:48:19.780141) contains errors	Ошибка
2	21:48:19.9...	Param src1_msg2:int is out of frequency: 0.000596948 (3.3 required)	Warning
3	21:48:19.9...	Exchange Exchange(21:48:19.975424) is out of frequency: 0.000596948 (3.3 required)	Warning
4	21:48:20.0...	Exchange Exchange(21:48:20.074852) contains errors	Ошибка
5	21:48:20.3...	Bound parameter group int42 is out of acceptable value range	Warning
6	21:48:20.3...	Parameter src1_msg3:int is out of const value: 173 (66 expected)	Ошибка
7	21:48:20.3...	Exchange Exchange(21:48:20.375283) contains errors	Ошибка
8	21:48:20.5...	Bound parameter group int42 is out of acceptable value range	Warning
9	21:48:20.6...	Exchange Exchange(21:48:20.674546) contains errors	Ошибка
10	21:48:20.9...	Exchange Exchange(21:48:20.973080) contains errors	Ошибка
11	21:48:21.2...	Exchange Exchange(21:48:21.277248) contains errors	Ошибка
12	21:48:21.5...	Exchange Exchange(21:48:21.579584) contains errors	Ошибка
13	21:48:21.8...	Exchange Exchange(21:48:21.874172) contains errors	Ошибка
14	21:48:22.1...	Exchange Exchange(21:48:22.180330) contains errors	Ошибка
15	21:48:22.1...	Data checksum mismatch: 1197 (expected 1196)	Ошибка

Рис. 4: Результат работы анализатора для протокола 1

B.2.2 Пример 2

В примере 2 наблюдается изменение значения параметра 6 из таблицы 2.

Для проверки факта равенства младшего байта параметра ошибочному значению 173 (0xAD) используется ограничение error_value. Старший байт параметра рассматривается как автоматически увеличивающееся значение-счётчик (имеющее выбросы). В регистрации трассы были сделаны паузы для того, чтобы протестировать проверку опоздания момента увеличения значения счётчика.

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

<piv version="1.1">
```

```
        <signals>
            <signal identifier="int42_3" type="int" signed="false" />
            <signal identifier="int42_3" type="autoinc" signed="false"
                 />
        </signals>

        <abonents>
            <abonent identifier="src1" mil1553_addr="2">
                <mil1553_messages>
                    <mil1553_contrMessage identifier="src1_msg3"
                        direction="input" addr="2" subaddr="4" numWords
                        ="4">
                        <bitfield identifier="int" signal="int42_3"
                            firstWord="2" firstBit="20" numBits="8"
                            lowerBitCost="1">
                            <restrict type="errorvalue" value="0xAD"
                                level="error" />
                        </bitfield>
                        <bitfield identifier="autoinc" signal="autoinc"
                             firstWord="2" firstBit="12" numBits="8"
                            lowerBitCost="1">
                            <restrict type="autoincrement" level="error
                                ">
                                <param name="timeout" value="1s" />
                                <param name="mask" value="0xFF" />
                            </restrict>
                        </bitfield>
                    </mil1553_contrMessage>
                </mil1553_messages>
            </abonent>
        </abonents>
</piv>
```

Листинг 14: Пример описания протокола 2

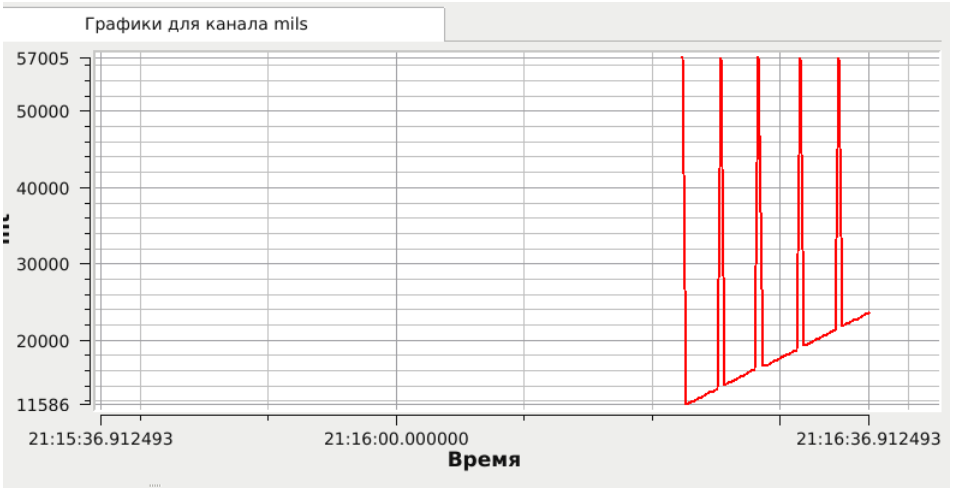


Рис. 5: График изменения значений параметра 6

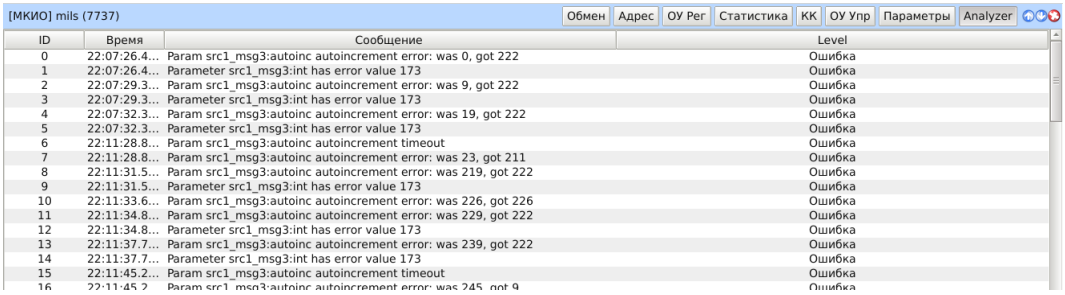


Рис. 6: Результат работы анализатора

B.2.3 Пример 3

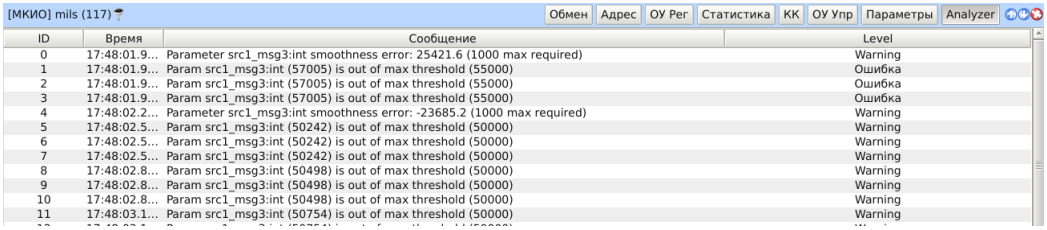
В примере 3 наблюдается изменение значения параметра 6 из таблицы 2. Для обнаружения “выбросов” (значительных (>5%) отклонений значения параметра от исходной функции) используется ограничение smooth с максимальным значением производной по модулю, равным 1000 (значение подобрано эмпирически). Также в этом примере провеяется корректность определения выхода значения параметра за заданные пороги (для этого же параметра).

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

<piv version="1.1">
       <signals>
           <signal identifier="int42_3" type="int" signed="true"
               twosComplement="true" />
```

```
        </signals>

        <abonents>
            <abonent identifier="src1" mil1553_addr="2">
                <mil1553_messages>
                    <mil1553_contrMessage identifier="src1_msg3"
                        direction="input" addr="2" subaddr="4" numWords
                        ="4">
                        <bitfield identifier="int" signal="int42_3"
                            firstWord="2" firstBit="20" numBits="16"
                            lowerBitCost="1">
                            <restrict type="smooth" value="1000" level
                                ="warning">
                                <param name="maxDeviation" value="0" />
                            </restrict>
                            <restrict type="max" level="error" value
                                ="55000" />
                            <restrict type="max" level="warning" value
                                ="50000" />
                            <restrict type="min" level="warning" value
                                ="25000" />
                        </bitfield>
                    </mil1553_contrMessage>
                </mil1553_messages>
            </abonent>
        </abonents>
</piv>
```

Листинг 15: Пример описания протокола 3

Рис. 7: График изменения значений параметра 6



ID	Время	Сообщение	Level
0	17:48:01.9...	Parameter src1_msg3:int smoothness error: 25421.6 (1000 max required)	Warning
1	17:48:01.9...	Param src1_msg3:int (57005) is out of max threshold (55000)	Ошибка
2	17:48:01.9...	Param src1_msg3:int (57005) is out of max threshold (55000)	Ошибка
3	17:48:01.9...	Param src1_msg3:int (57005) is out of max threshold (55000)	Ошибка
4	17:48:02.2...	Parameter src1_msg3:int smoothness error: -23685.2 (1000 max required)	Warning
5	17:48:02.5...	Param src1_msg3:int (50242) is out of max threshold (50000)	Warning
6	17:48:02.5...	Param src1_msg3:int (50242) is out of max threshold (50000)	Warning
7	17:48:02.5...	Param src1_msg3:int (50242) is out of max threshold (50000)	Warning
8	17:48:02.8...	Param src1_msg3:int (50498) is out of max threshold (50000)	Warning
9	17:48:02.8...	Param src1_msg3:int (50498) is out of max threshold (50000)	Warning
10	17:48:02.8...	Param src1_msg3:int (50498) is out of max threshold (50000)	Warning
11	17:48:03.1...	Param src1_msg3:int (50754) is out of max threshold (50000)	Warning

Рис. 8: Результат работы анализатора