Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Сибирский государственный аэрокосмический университет
имени академика М.Ф. Решетнёва»

На правах рукописи

НЕКРАСОВ МИХАИЛ ВИКТОРОВИЧ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА МНОГОПОТОЧНОГО ПРИЁМА, ОБРАБОТКИ И АНАЛИЗА ТЕЛЕМЕТРИЧЕСКОЙ ИНФОРМАЦИИ

Специальность 05.13.06 – Автоматизация и управление технологическими процессами и производствами (промышленность)

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
д.т.н., профессор А.Н. Антамошкин

Красноярск 2014

Оглавление

Введение

Современная автоматизированная система управления космическими аппаратами (АСУ КА) предназначена для обеспечения функционирования бортовых систем КА в течение всего времени его активного существования. АСУ КА представляет собой совокупность бортовых и наземных средств управления технологическими процессами с необходимым программным обеспечением и включает:

- бортовой комплекс управления (БКУ), включая аппаратуру управления по каналам бортового радиокомплекса;
- наземный комплекс управления (НКУ).

В такой системе существует критическая необходимость в скорейшем обнаружении нарушения функционирования технологических процессов, от простого перегорания предохранителя до выявления предотказных состояний бортовой аппаратуры, посредством анализа телеметрической информации. Однако зачастую человек не способен эффективно обрабатывать большой объём поступающей информации, поэтому перспективным является создание комплексов автоматизации технологических процессов приёма, обработки и анализа телеметрической информации. Высокий уровень автоматизации и интеллектуализации системы позволит уменьшить время сбора необходимой информации и повысить эффективность действий операторов анализа для поддержания стабильного функционирования космического аппарата.

В настоящее время в центре управления полётами системы Глонасс используется однопоточная система автоматизированного приёма, обработки и анализа телеметрической информации, способная в единичный момент времени принимать, обрабатывать и анализировать телеметрическую информацию не более чем с одного космического аппарата. Однако для обеспечения целевой задачи – создания глобального навигационного поля – в орбитальной группировке (ОГ) должно находиться 24 космических аппарата. Общее количество сеансов связи за сутки достигает 40, при этом одновременные сеансы связи проводятся с 3-5 космическими аппаратами. Поэтому актуальным на данный момент становится разработка новых методов обеспечения автоматизированной многопоточного приёма, обработки и анализа больших объёмов телеметрической информации с нескольких космических аппаратов одновременно в составе АСУ ОГ КА.

Система приёма, обработки и анализа телеметрической информации, применяемая на сегодняшний день в центре управления полётами, имеет ряд недостатков, связанных, в первую

очередь, с несовершенством архитектуры самой системы, отсутствием централизованного хранения архивов телеметрии и процедурными принципами построения специального программного обеспечения, которое ориентировано на применение ручных технологий и не обеспечивает необходимой глубины контроля. Современная система приёма, обработки и анализа телеметрической информации морально устарела и требует модернизации.

Исходя из экономической целесообразности и обеспечения национальной безопасности для заказов МО РФ размещение элементов наземного комплекса управления отечественными ОГ КА производится только на территории РФ. Элементы наземного комплекса управления обмениваются ТМИ по специализированным протоколам информационно-логического взаимодействия и осуществляют поддержку друг друга по выделенным каналам с целью исключения несанкционированного доступа к циркулирующей информации. С учётом изложенного, применение международных стандартов и разработок при создании многопоточной системы приёма, обработки и анализа телеметрической информации затруднительно. Тем не менее, рекомендации зарубежных разработок по построению телеметрических систем в виде иерархической структуры должны учитываться при модернизации существующей телеметрической системы.

Проблемы обработки телеметрической информации с ОГ КА неоднократно рассматривались в работах Охтилева М.Ю., Чуприкова А.Ю., Ничипоровича О.П, Соколова Б.В (СПИИ РАН, г.Санкт-Петербург), Виноградова А.Н., Заднепровского В.Ф., Куршева Е.П., Хачумова В.М. (ИПС РАН, г.Переславль-Залесский, Ярославская область), Кузина В. А., Атаманчука Ю. И., Кравчука Н. В., Шибанов А. П. (ОКБ «Спектр», г.Рязань), Смирнова С.В., Ватутина В.М., Генералова П.В., Круглова А.В., Тимошиной Н.Е (ОАО «РКС», г. Москва), Валова Н.Н., Скорнякова В.А. (ЦНИИМаш, г.Королёв, Московская область).

В рамках современного уровня развития системного анализа сложных инженерных задач и средств программирования в данном исследовании разработана архитектура многопоточной системы обработки телеметрической информации в составе АСУ ОГ КА.

В качестве **объекта исследования** в работе выступают потоки телеметрической информации в контуре АСУ ОГ КА.

Предметом исследования являются методы автоматизации технологических процессов приёма, обработки и анализа телеметрии в системе обработки телеметрической информации.

Целью диссертации является автоматизация комплекса технологических процессов приёма, обработки и анализа телеметрической информации для повышения эффективности функционирования АСУ ОГ КА. В соответствии с целью исследования основными задачами являются следующие.

1 Исследовать технологические процессы существующей автоматизированной системы приёма, обработки и анализа телеметрической информации в составе АСУ ОГ КА.
2 Выявить ключевые функции для анализа эффективности системы автоматизации технологических процессов приёма, обработки и анализа телеметрической информации.
3 Определить архитектуру новой автоматизированной системы многопоточного приёма, обработки и анализа телеметрической информации в контуре АСУ ОГ КА с учётом ключевых функций системы. Определить принципы взаимодействия подсистем системы приёма, обработки и анализа телеметрической информации.
4 Предложить принципы унификации средств описания исходных данных на обработку телеметрической информации, алгоритмов и методов обработки и анализа телеметрической информации.
5 Предложить принципы построения качественной программной модели автоматизированной системы многопоточного приёма, обработки и анализа телеметрической информации.
6 Разработать программное обеспечение сервера обработки телеметрической информации, позволяющее автоматизировать технологические процессы многопоточного сбора, обработки и анализа телеметрической информации и осуществлять централизованное хранение данных с поддержкой множественного санкционированного доступа клиентов обработки и анализа телеметрической информации.
7 Провести сравнительный анализ ключевых функций старой и новой системы приёма, обработки и анализа телеметрической информации в контуре АСУ ОГ КА.

Решение поставленных задач позволит повысить надёжность и степень автоматизации технологических процессов системы приёма, обработки и анализа телеметрической информации и удовлетворить современные требования к АСУ ОГ КА.

Методы исследования. Исследования проводились с использованием теории системного анализа, методов абстрагирования и конкретизации, методов синтеза специального программного обеспечения, объектно-ориентированного проектирования и программирования.

Научная новизна работы результатов диссертационной работы состоит в следующем:

1 Предложена и обоснована и архитектура новой телеметрической системы в составе АСУ ОГ КА, включающая подсистему многопоточного приёма информации и

позволяющая автоматизировать технологические процессы приёма, обработки и анализа телеметрической информации.

2 Спроектирована библиотека объектно-ориентированных модулей, включающая унифицированные средства описания исходных данных, алгоритмы и методы обработки и анализа телеметрической информации и позволяющая более эффективно организовывать программное обеспечение новой телеметрической системы.

3 Предложена и обоснована архитектура обслуживающей подсистемы, включающая поддержку многопоточного приёма, обработки и анализа телеметрической информации и позволяющая обеспечить множественный санкционированный доступ клиентов обработки и анализа телеметрии и повысить степень доступности телеметрической информации.

4 Предложен метод адаптивной передачи телеметрии потребителям, включающий функцию автоматического выбора оптимального потока телеметрии и позволяющий повысить стабильность и качество приема телеметрической информации на начальных этапах ориентации КА.

Теоретическая значимость результатов диссертационного исследования состоит в создании архитектуры новой автоматизированной многопоточной системы приёма, обработки и анализа телеметрической информации, построении унифицированной библиотеки описания исходных данных, алгоритмов и методов обработки и анализа телеметрии. Результаты, полученные при выполнении диссертационной работы, создают теоретическую основу для разработки методов и алгоритмов, направленных на повышение эффективности технологических процессов приёма, обработки и анализа телеметрической информации.

Практическая ценность работы заключается в применении результатов исследования для построения в контуре АСУ ОГ КА автоматизированной многопоточной системы приёма, обработки и анализа телеметрической информации, организации взаимодействия построенной системы с удалёнными потребителями телеметрической информации, а также в применении результатов исследования для проектирования, разработки и внедрения новой автоматизированной многопоточной системы приёма, обработки и анализа телеметрической информации в состав АСУ ОГ системы Глонасс, Гео-ИК-2, Экспресс-АМ, Экспресс-АТ, Луч-5В. В работе достигнуты следующие практические результаты:

- предложен унифицированный сетевой интерфейс для доступа к источникам телеметрии, включающий взаимодействие с САО-Ц, СОТИ, ЕЦУП РБ и позволяющий расширять функции телеметрической системы при введении новых источников телеметрии;

- разработан набор протоколов взаимодействия между сервером обработки телеметрии и внутренними / внешними для АСУ ОГ КА клиентами, описывающий значения телеметрических параметров и позволяющий передавать состояния бортовых систем КА в унифицированном виде;
- разработана библиотека унифицированного описания исходных данных, алгоритмов и методов обработки и анализа телеметрической информации;
- разработано программное обеспечение, реализующее взаимодействие подсистем внутри системы, а также взаимодействие с внешними абонентами по предложенным принципам;
- разработано кроссплатформенное программное обеспечение сервера обработки телеметрической информации, функционирующее под управлением операционных систем Windows, Linux и включающее метод адаптивной передачи телеметрии потребителям и подсистему защиты информации;
- разработана реляционная модель базы данных для централизованного хранения архивов телеметрической информации;
- использование результатов исследования для построения системы многопоточного приёма телеметрической информации позволит повысить степень доступности телеметрической информации, обеспечить гибкую модульность программного обеспечения, упростить процесс формирования кроссплатформенного информационно-телеметрического обеспечения, обеспечить полноценное взаимодействие внутри системы и с внешними абонентами, а также повысить качество системы приёма телеметрии и эффективность функционирования АСУ КА в целом;
- результаты исследования и созданное на его основе специальное программное обеспечение сервера обработки телеметрии внедрено, что подтверждается актами внедрения, и используются следующими организациями:
  - центры управления полётами системами Глонасс, Гео-ИК-2 (г. Краснознаменск, Московская область);
  - центры управления полётами системами Экспресс-АМ, Экспресс-АТ (г. Москва, г. Железногорск, Красноярского края);
  - центр управления полётом системы Луч-5В (г. Королёв, Московская область)
  - информационно-вычислительный комплекс генерального конструктора Открытого акционерного общества «Информационные спутниковые системы» имени академика М.Ф. Решетнёва (г. Железногорск, Красноярского края).

Публикации. По теме диссертационной работы опубликовано 20 работ, из них 4 в изданиях Перечня ВАК и 2 зарегистрированные программные системы.

Апробация работы. Диссертационная работа и её отдельные разделы докладывались и обсуждались на научно-технической конференций молодых специалистов ОАО ИСС, г. Железногорск, 2008г; XLVII Международной научной студенческой конференции «Студент и научно-технический прогресс», г. Новосибирск, 2009г; Всероссийской научно-практической конференции студентов, аспирантов и молодых специалистов «Актуальные проблемы авиации и космонавтики», г. Красноярск, 2009г; Всероссийской научно-технической конференции «Актуальные проблемы ракетно-космического приборостроения и информационных технологий», г. Москва, 2009г; XIII Международной научной конференции «Решетневские чтения», г. Красноярск, 2009г; XXXIV Академических чтениях по космонавтике «Королёвские чтения», г. Москва, 2009г; III Всероссийской научно-технической конференции «Актуальные проблемы ракетно-космического приборостроения и информационных технологий», г. Москва, 2010г; XLVIII Международной научной студенческой конференции «Студент и научно-технический прогресс», г. Новосибирск, 2010г; Международном конгрессе по интеллектуальным системам и информационным технологиям IS&IT'10, г. Геленджик-Дивноморское, 2010г; XIV Международной научной конференции «Решетнёвские чтения», г. Красноярск, 2010г.

Автор выражает благодарность за ценные рекомендации при обсуждении и оформлении работы Жаворонкову В.Г., Шмику К.Б., Пакману Д.Н.

Глава 1. Общие сведения об архитектуре современной АСУ ОГ КА

Современные автоматизированные системы управления космическим аппаратом имеют сложную многокомпонентную архитектуру. Управление орбитальной группировкой космических аппаратов осуществляется из центра управления полётами, куда стекаются различные виды служебной и специальной информации. Среди множества видов информации, циркулирующих в контуре автоматизированной системы управления, особое место занимают потоки телеметрической информации, позволяющие осуществлять мониторинг состояния космического аппарата. Обработка и анализ телеметрической информации производится средствами специального программного обеспечения обработки телеметрии. Однако с вводом новых типов КА в орбитальную группировку, возрастанием вычислительной нагрузки и расширением функциональных задач становятся очевидными, что существующая система обработки телеметрии более не может обеспечивать полноты и качества выполнения возложенных на неё задач. При проектировании современных средств обработки и анализа телеметрической информации необходимо следовать принципам построения унифицированных систем.

1.1 Структура автоматизированной системы управления космическим аппаратом

Автоматизированная система управления космическим аппаратом, рассматриваемая в [1], [2], [3] предназначена для обеспечения работы бортовых систем космического аппарата в течение всего времени его активного существования, представляющая собой совокупность бортовых и наземных средств управления с необходимым математическим обеспечением, и включает бортовой комплекса управления и наземный комплекс управления. Приведенные составляющие АСУ КА представляют собой самостоятельные функциональные звенья с решением определенных задач управления в контуре АСУ.

В задачи *бортового комплекса управления*, описываемого в [1], [4] входит:

а) обеспечение приведения элементов конструкции и бортовых систем КА в рабочее состояние;

б) управление работой бортовой аппаратуры КА в автономном и оперативном режимах;

в) контроль и диагностика технического состояния бортовых систем;

г) организация работы бортового программного обеспечения.

д) ориентация и управление движением КА;

е) обмен информацией с центром управления полётами (ЦУП) через средства НКУ;

ж) передача на Землю телеметрической информации о состоянии бортовой аппаратуры;

и) обеспечение траекторных измерений средствами НКУ;

к) прием, дешифровку разовых команд (РК), программных команд (ПК), массивов специальной информации, управляющей информации и временной программы управления, принимаемых с наземных измерительных пунктов (НИП);

л) формирование и выдачу на НИП квитанций о прохождении РК, ПК, массивов специальной информации и выполнении программ управления;

м) коррекция шкалы бортового времени и участие в сверке шкал времени.

Необходимо отметить, что одной из важнейших задач БКУ является сбор, обработка, сжатие, хранение и выдача по требованию наземного комплекса управления телеметрической информации (ТМИ) о текущем состоянии бортовых систем.

В свою очередь *наземный комплекс управления* рассматривается в [5], [6], [7] и предназначен обеспечивать решение следующих задач:

- управление КА, размещаемыми на заданных орбитах в период лётно-конструкторских испытаний, опытной и штатной эксплуатации, как в штатных, так и в аварийных ситуациях;
- подготовку и обмен с центром управления плановой и оперативной информацией, обеспечивающей согласованное функционирование бортового радиокомплекса (БРК) КА и наземных средств связи в штатных и аварийных ситуациях;
- долговременное и оперативное планирование работы систем КА и средств НКУ;
- автоматизированную подготовку исходных данных для принятия решений по управлению КА и средствами НКУ;
- автоматизированный анализ и отображение состояния бортовой аппаратуры;
- непрерывный приём со спутника, обработку в темпе приёма, хранение, регистрацию, документирование и воспроизведение телеметрической и траекторной информации;
- круглосуточное автоматизированное формирование команд и программ управления всеми бортовыми системами и их выдачу на спутник;
- измерение и прогноз параметров движения спутника с точностью, необходимой для удержания спутника в заданной орбитальной позиции с необходимой точностью;
- оперативный обмен необходимой управляющей, измерительной и служебной информацией между элементами НКУ;
- управление и непрерывный контроль состояния средств НКУ.

- поддержания заданных технических и баллистических характеристик КА и орбитальной группировки в целом.

С целью обеспечения решения наземным-комплексом управления описанных задач, как правило, в состав НКУ включены следующие элементы:

- основной центр управления полетом;
- резервный центр(ы) управления полетом (РЦУП);
- наземные командно-измерительные станции (КИС);
- станции приема телеметрической информации;
- корелляционно-фазовые пеленгаторы;
- система связи и передачи данных (ССПД);
- система единого времени;
- система диспетчерской связи.

Таким образом, ввиду особенностей структуры АСУ КА выделяют два основных направления передачи информации: прямой канал – передача по направлению «земля-борт» (разовые команды, массивы специальной информации и баллистические параметры) и обратный канал – «борт – земля», по которому транслируется телеметрическая информация.

1.2 Технологические процессы приёма, обработки и анализа телеметрической информация в контуре автоматизированной системы управления космическим аппаратом

В АСУ КА объектом управления является орбитальная группировка космических аппаратов (ОГ КА) (1.2), а задачей управления – поддержание оптимального функционирования бортовых систем КА в течение всего срока активного существования. Решение задачи управления КА осуществляется аппаратно-программными средствами НКУ КА, являющимся составной частью НКУ единой системы спутниковой связи. В [5], [6] рассматриваются различные виды информации, циркулирующие в структуре НКУ, основными из которых являются: разовые команды, программные команды, командно-программная информация, информация функционального контроля, баллистическая информация и телеметрическая информация.

Средствами ЦУП производится выдача управляющих воздействий (УВ) и закладка массивов командно-программной информации (КПИ) через земные станции (ЗС) на борт КА. Для уточнения параметров движения КА производится периодическое измерение текущих навигационных параметров (ИТНП) КА. Особую роль при управлении КА играет телеметрическая информация (ТМИ), позволяющая оценить реакцию управляемого КА на

управляющие воздействия. Телеметрическая информация, поступающая с передающих устройств КА по радиолинии, содержит сведения о работе и состоянии бортовой аппаратуры КА, выполнении программ работы и управляющих воздействий [8], [4].

Технологический процесс сбора телеметрической информации о состоянии КА, передачи и представления ее конечному пользователю (оператору управления, системному специалисту) представляет собой многоступенчатую процедуру, включающую в себя следующие этапы.

Этап *сбора и передачи телеметрической информации*, рассматриваемый в [4], выполняется бортовым информационно-телеметрическим комплексом, представленным на рисунке 1.1. Из рисунка видно, что бортовой информационно-телеметрический комплекс состоит из совокупности датчиков $Д_1$, …, $Д_N$, информационно-телеметрической системы, включающей в себя бортовую и приемно-регистрирующую аппаратуру и аппаратуру обработки телеметрической информации [4].

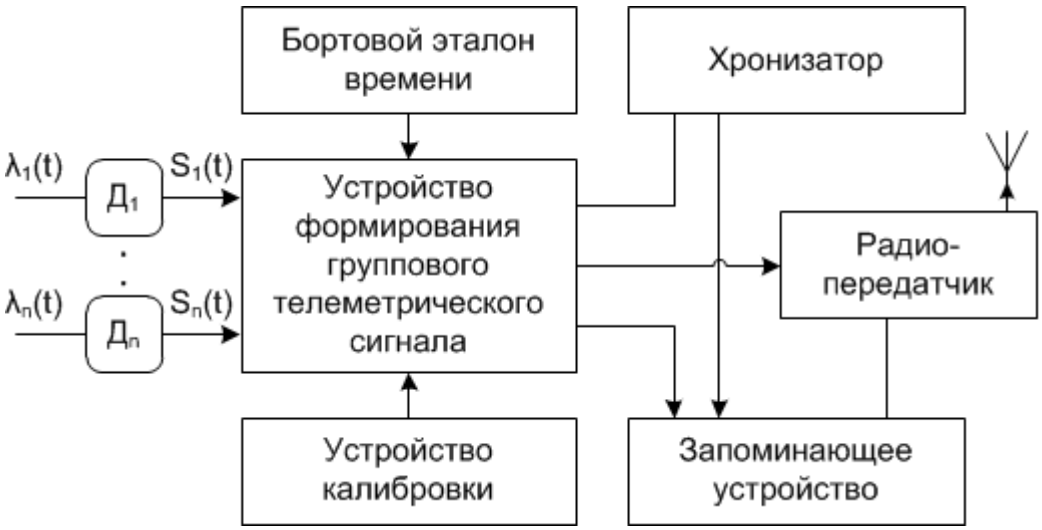

Рисунок 1.1 – Структурная схема бортового информационно-телеметрического комплекса

Телеметрируемые параметры $\lambda_1(t)$, …, $\lambda_N(t)$ с помощью измерительных преобразователей (датчиков) Д преобразуются в первичные электрические сигналы. Между датчиками и входами бортовой информационно-телеметрической системы при необходимости включаются согласующие устройства, обеспечивающие нормализацию сигналов и согласование выходных сопротивлений датчиков с входными сопротивлениями каналов. Под нормализацией подразумевается преобразование различных электрических сигналов в напряжение постоянного тока, пределы изменения которого составляют 0-6,2 В.

Далее, первичные сигналы от датчиков объединяются в групповой телеметрический сигнал, на основе принципов частотного, временного или кодового разделения канальных

сигналов, который размещается в оперативной памяти бортовой аппаратуры телесигнализации (БАТС). Вместе с информационными сообщениями в групповом сигнале передаётся также служебная информация: сигналы синхронизации, «командное слово» и сведения, необходимые для декодирования группового-телеметрического сигнала при приёме. С помощью маркерных сигналов групповой-телеметрический сигнал разделяется на кадры (псевдокадры), которые, в свою очередь, через бортовую командно-измерительную систему (бортовую КИС) транслируются в наземную измерительную станцию.

Этап *предварительной обработки телеметрической информации*, рассматриваемый в [4], осуществляется средствами наземной КИС в составе НКУ и обеспечивает решение следующих задач:

- прием телеметрической информации наземной КИС и осуществление идентификации, нормализации, усиления телеметрического (ТМ) сигнала;
- аналогово-цифровое преобразования информации средствами КИС;
- выделение синхропосылки, анализ служебной части кадра, нарезка потока ТМИ на псевдокадры;
- привязка псевдокадров к московскому декретному времени;
- формирование блоков информации для передачи в каналы связи;
- передача сформированных блоков телеметрической информации в комплекс обработки информации (КОИ) «Рымник» или систему автоматизированной обработки пункта (САО-П), в зависимости от этапа проведения работ.

Этап *первичной обработки телеметрической информации*, рассматриваемый в [9], [10], осуществляется распределённой системой обработки «Рымник» или системой САО-П и обеспечивает решение следующих задач:

- в случае проведения работ через КОИ «Рымник»:
  - обработка телеметрии средствами КОИ, привязка телеметрической информации ко времени, проверка на достоверность, выделение существенных значений;
  - сбор ТМ-информации с различных наземных измерительных пунктов сектором сбора данных (ССД);
  - передача ТМИ из ССД в СОТИ (сектор обработки ТМИ), расположенный в ЦУП;
- в случае проведения работ через САО-П производится передача телеметрической информации по соответствующему протоколу информационного взаимодействия в систему автоматизированной обработки центра (САО-Ц).

Окончательный этап – этап *вторичной обработки, анализа и представления информации*, рассматриваемый в [11], [12], проводится по результатам первичной обработки и

позволяет путём заданных расчётов определить необходимые характеристики работы бортовых систем. Этап осуществляется аппаратно-программными средствами ЦУП, входящими в состав НКУ и обеспечивает решение следующих задач:

- прием информационных потоков от СОТИ или САО-Ц;
- распределение потоков телеметрии по рабочим станциям ЦУП;
- расчет первичных, вторичных параметров и параметров алгоритма обобщенного контроля;
- анализ результатов обработки телеметрической информации;
- представление результатов операторам управления и системным специалистам.

Общая архитектура современной АСУ ОГ КА, а также циркуляция в ней потоков ТМИ представлены на рисунке 1.2.

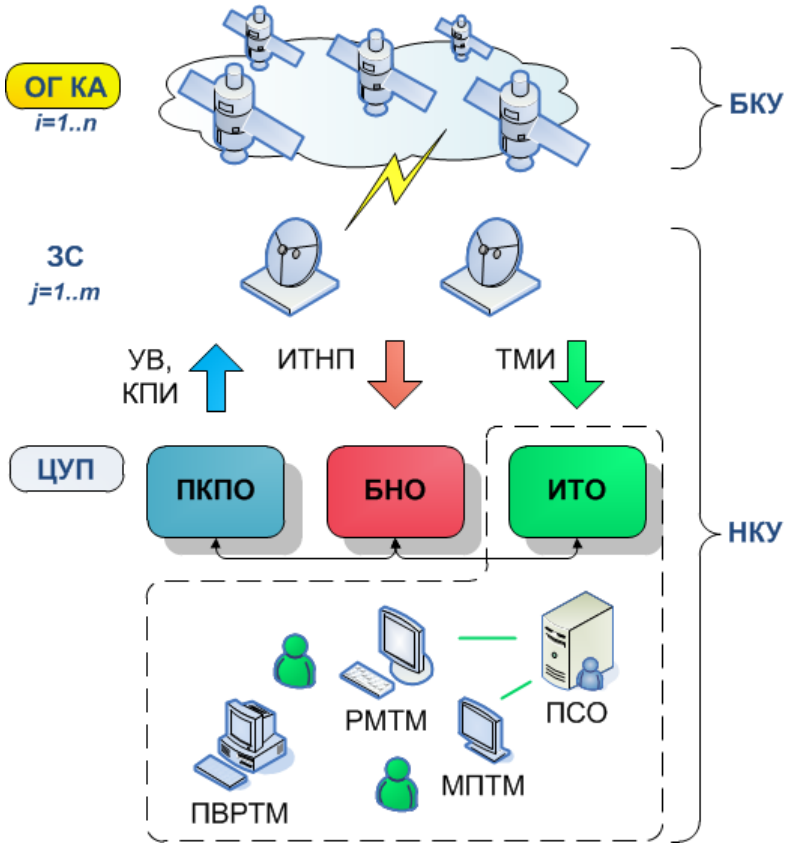

Рисунок 1.2 – Общая архитектура современной АСУ ОГ КА

1.3 Структурно-функциональная схема центра управления полётом космического аппарата и циркуляция технологических процессов приёма, обработки и анализа телеметрической информации в контуре ЦУП КА

В настоящее время ЦУПы КА организуются на базе аппаратно-программного комплекса [3;4], включающего следующие функциональные сектора, изображённые на рисунке 1.3:

Сектор управления, основными задачами которого являются:

- проведение оперативного сеанса управления с КА, включая отработку как заранее подготовленных программ управления, так и программ управления или команд, вводимых оператором в реальном масштабе времени;
- регистрация и обработка квитанций, поступающих от средств НКУ и бортовой аппаратуры КА, автоматизированный контроль выполнения программы управления по поступающей ТМИ и квитанциям;
- прием, обработка, отображение и анализ информации о результатах работы и состоянии наземных средств НКУ;
- выдача рекомендаций по управлению КА, в том числе в нештатных (аварийных) режимах с выдачей алгоритмов по их устранению;
- дистанционное управление и контроль станциями приёма КИС, корреляционно-фазового пеленгатора;
- сбор и накопление статистической информации по проведенным сеансам управления, по выданным управляющим воздействиям (с учетом их классификации по видам), о состоянии орбитальной группировки по результатам обработки ТМИ и отчетов бортового цифрового вычислительного комплекса (БЦВК).

Сектор планирования, основными задачами которого являются:

- автоматизированные работы по реализации планов развертывания, восполнения и поддержания орбитальной группировки КА;
- формирование, отображение и корректировка долгосрочных и оперативных планов работы КА и НКУ для обеспечения реализации штатного технологического цикла управления на разных этапах функционирования всех КА ОГ, в том числе и при возникновении нештатных ситуаций;
- формирование технологических данных для проведения сеансов связи с КА и перекачка их на средства управления;
- расчет программ управления и массивов командно-программной информации для КА.

Баллистический сектор, основными задачами которого являются:

- прием и обработка измерений текущих навигационных параметров с КИС и КФП;
- расчет баллистических характеристик КА из состава ОГ, определение орбитальных параметров КА, разработка планов коррекции орбит КА, расчёт интерференции (влияния солнечного излучения на работу антенных систем наземных станций) и обеспечение операторов управления необходимой баллистической информацией

(тени Луны и Земли, зоны радиовидимости, целеуказания для наведения антенн наземных станций, зоны обслуживания);

- подготовка массивов командно-программной информации для их закладки на борт КА в сеансах управления (в части обеспечения работы бортового баллистического программного обеспечения);
- подготовка исходных технологических данных на запуск ракетоносителя с КА и обеспечение ими подразделений и служб НКУ.

Сектор обработки телеметрической информации, основными задачами которого являются:

- подготовку исходных данных, необходимых для проведения обработки ТМИ, контроля и анализа состояния КА;
- получение и обработка телеметрической информации с сохранением в долговременный архив;
- прием и обработка отчетов БЦВК;
- автоматизированный обобщенный контроль и диагностика состояния КА по ТМИ в течение всего срока активного существования;
- контроль и анализ состояния КА с выдачей заключений и рекомендаций;
- дистанционное управление станциями приёма ТМИ;
- обмен различными видами информации с элементами НКУ, внешними организациями и комплексами;
- представление результатов обработки телеметрической информации системным специалистам и операторам управления.

Для решения общей целевой задачи управления КА и поддержания его активного существования между секторами ЦУП осуществляется информационно-логическое взаимодействие. Частью этого взаимодействия является и обмен телеметрической информацией, представленный на рисунке 1.3. Информационное взаимодействие между секторами осуществляется на программном уровне с использованием локально-вычислительной сети ЦУП.

Основными потребителями ТМИ в ЦУП КА являются операторы управления (анализаторы) и системные специалисты. В их задачи в частности входит анализ состояния всех КА орбитальной группировки и выбор на его основе стратегии по управлению космическими аппаратами.

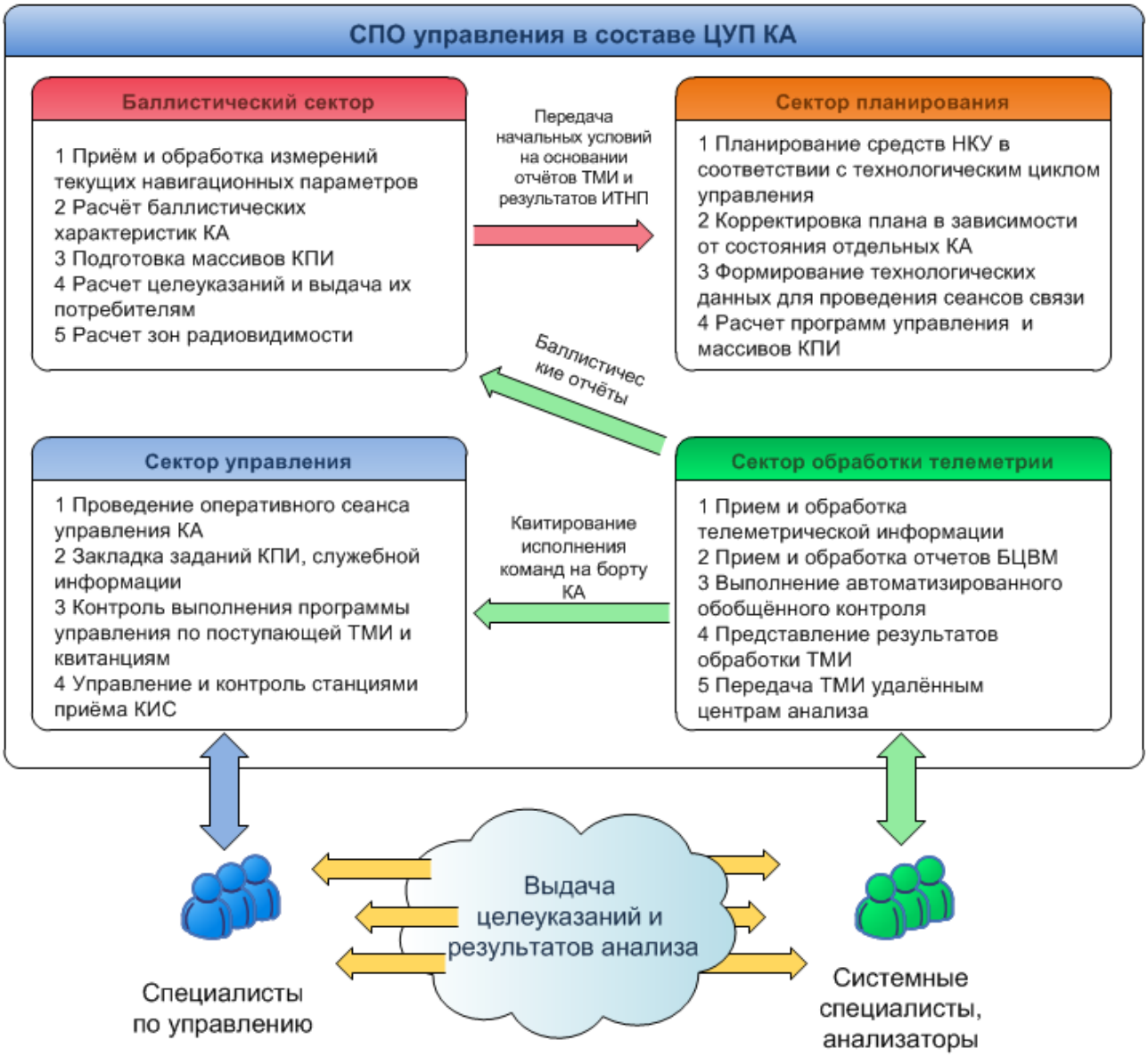

Рисунок 1.3 – Циркуляция ТМИ внутри СПО ЦУП КА

1.4 Анализ основных направлений развития зарубежных систем автоматизированного управления орбитальной группировкой космических аппаратов

Космическая навигационная система GPS (США) в настоящее время является единственной в мире в полной мере функционирующей системой и несет на себе основную нагрузку по обеспечению потребителей всего мира навигационной информацией [13], [14]. Управление этой глобальной услугой принадлежит Министерству Обороны и Министерству транспорта Соединенных Штатах, с ежедневным контролем над системой, осуществляемым Командованием Военно-воздушных и космических сил США. Орбитальная группировка минимального состава включает 24 КА, расположенных в шести орбитальных плоскостях по 4

КА в плоскости, высота орбиты 20180 км, наклонение 56°. Возможно увеличение количества КА в каждой плоскости до 6 КА [13], [14], [15]. В настоящее время в составе ОГ GPS находится 30 КА в штатном использовании.

Наземный контур управления GPS состоит из глобальной сети наземных средств, обеспечивающих измерение орбит, мониторинг и анализ состояния КА, а также выдачу команд управления для ОГ КА. Функционирующий в настоящее время наземный контур управления состоит из главного ЦУП, одного резервного ЦУП, 12 антенн КИС и 16 станций мониторинга [16]. Расположение средств приведено на рисунке 1.4.

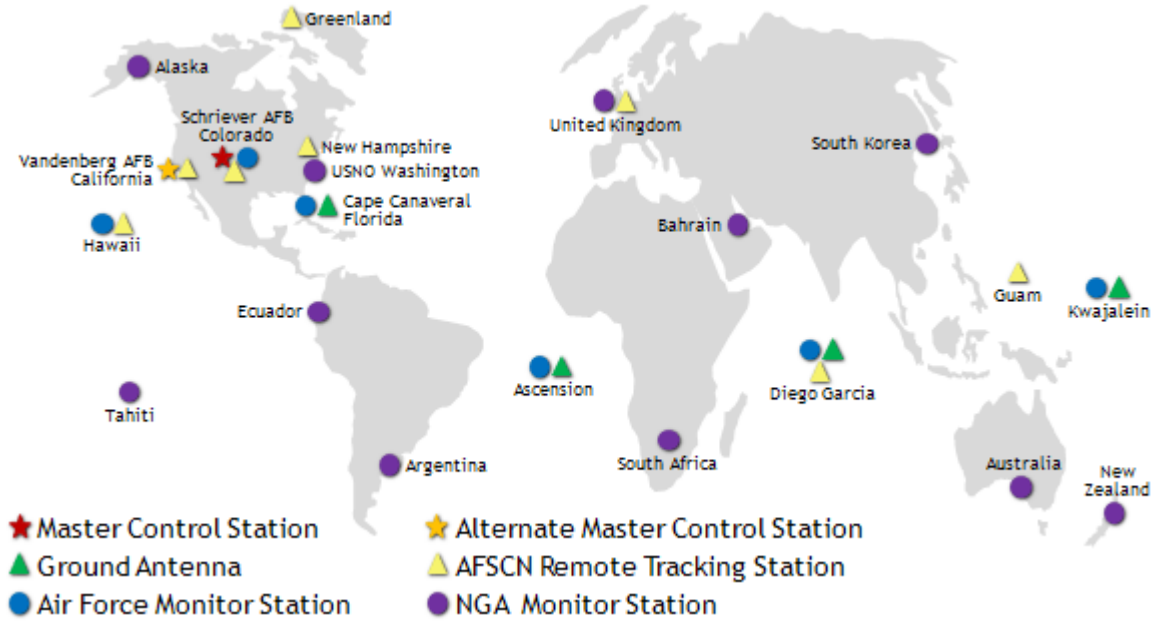

Рисунок 1.4 – Наземный контур управления системы GPS

Другая навигационная система, представляющая интерес в смысле использования группировки космических аппаратов – глобальная система определения местоположения Galileo, создаваемая Европейский Союзом с 2003 года. Навигационная система Galileo будет включать в себя 30 спутников (27 действующих и 3 резервных), размещенных на трех круговых околоземных орбитах высотой 23616 км и наклонением 56° [17], [18]. Наклонение орбиты находится между параметрами американской системы GPS - 55° и российской ГЛОНАСС - 64,8°. Навигационные сигналы этой системы будут обеспечивать надежное покрытие поверхности Земли вплоть до широты 75° и даже в более высоких широтах. Большое число космических аппаратов и оптимизация их размещения на орбитах гарантируют высокую надежность системы. Европейская спутниковая навигационная система Galileo планируется под гражданское управление.

Инфраструктура наземного контура управления системы Galileo представлена на

рисунке 1.5 состоит из датчиковых станций по всему миру, двух центров управления полётами, станций выдачи управляющих воздействий, а также командно-измерительных станций [19] .

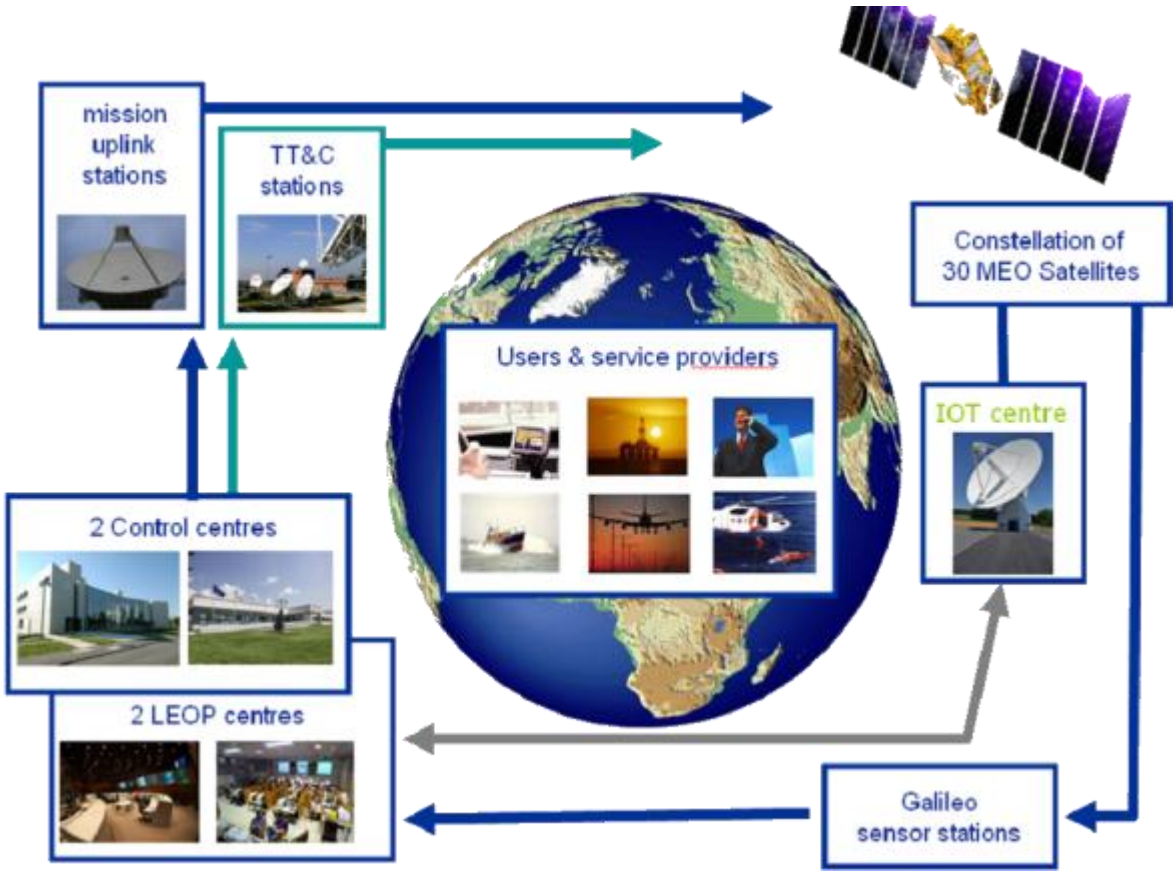

Рисунок 1.5 – Инфраструктура наземного контура управления системы Galileo

Создание системы производится в несколько этапов [20].

Первый этап — планирование и определения задач стоимостью

Второй этап состоит в запуске двух опытных спутников и развития инфраструктуры (наземных станций для них) стоимостью 1,5 млрд евро.

Третий этап состоит в выводе на орбиты четырёх спутников Galileo IOV (in-orbit validation), которые, будучи запущенными парами (два 20 октября 2011 года и ещё два в октябре 2012 года), создали первое мини-созвездие Galileo. Запуски состоялись с помощью ракеты «Союз-СТБ» с космодрома в Куру. Первые четыре спутника строятся партнерством EADS Astrium-Thales Alenia Space. Спутники будут расположены на круговых орбитах на высоте 23 222 км [20].

Четвёртый этап проекта будет запущен предположительно с 2014 года. К 2015 году на орбиту будут выведены ещё 14 спутников, остальные — к 2020 году [20].

После завершения развертывания группировки, спутники обеспечат в любой точке планеты, включая Северный и Южный полюса, 90%-ю вероятность одновременного приема

сигнала от четырёх спутников. Благодаря доступу к точному сигналу в двух частотных диапазонах, клиенты Galileo получат информацию о своем местоположении с точностью 4 м в горизонтальной плоскости и 8 м в вертикальной при доверительном интервале 0,95 [20].

Таким образом, особенности реализаций крупных международных космических проектов характеризуются использованием высокотехнологического оборудования для решения задач автоматизации в космосе и из космоса. Необходимость обеспечения функционирования данного оборудования сказалась на требованиях, предъявляемых к наземным станциями слежения, связи и системам передачи данных из космоса. Целью функционирования телеметрической системы всегда является надёжная и достоверная доставка пользователям измерительной информации от удалённых бортовых источников [4].

Для зарубежных телеметрических систем, используемых в международных проектах, можно выделить ряд особенностей их построения и способов функционирования.

1 Телеметрические системы, системы контроля и диагностики КА построены как централизованные иерархические системы, основанные на встроенных в системы средствах и реализации алгоритмов контроля и диагностики в бортовом вычислительном комплексе [4].
2 Телеметрические системы (являющиеся, по сути, системами сбора и передачи данных) передают информацию о результатах диагностики оборудования, а также информацию о научных экспериментах по одним и тем же выделенным каналам связи [4].
3 Принадлежность различных элементов (сегментов) проекта различным странам, космическим агентствам и организациям, кроме того необходимость обмена телеметрической информацией и аппаратной поддержки друг друга являются причиной высокой стандартизации как услуг передачи информации, так и используемого оборудования [4].

1.5 Предложения по усовершенствованию специального программного обеспечения автоматизации технологических процессов приёма, обработки и анализа телеметрической информации

В настоящее время программные средства обработки, анализа и представления телеметрической информации в центрах управления космическими аппаратами морально устарели и не соответствуют современным требованиям и возможностям. В то же время с развитием самой платформы космического аппарата растет потребность в обработке все

большего потока информации с максимально удобным представлением ее результатов оператору управления или системному специалисту.

Для обеспечения решения целевой задачи – создания глобального навигационного поля – в орбитальной группировке должно находиться 24 космических аппарата. Общее количество сеансов связи в сутки достигает 100, при этом одновременные сеансы связи проводятся с 3-5 космическими аппаратами. Поэтому актуальным на данный момент становится разработка новых методов обеспечения многопоточной обработки больших объёмов телеметрической информации с нескольких космических аппаратов одновременно.

В существующей схеме обработки ТМИ сервер проведения сеанса обработки (КП ПСО) обеспечивает однопоточный режим работы. То есть, проведение сеанса возможно только с одним аппаратом. В режиме проведения параллельных сеансов связи с КА для обеспечения приёма нескольких потоков телеметрии необходима организация соответствующего количества телеметрических серверов на различных рабочих местах. Данная ситуация ухудшает оперативность работы телеметрического сектора, поскольку увеличивается время настройки клиентов на телеметрический сеанс. Такая реализация имеет проблему децентрализованного размещения телеметрических архивов, создаваемых серверами для послесеансного анализа. Как известно в случае децентрализованного хранения информации остаётся актуальной проблема синхронизации сведений, которая в настоящей реализации системы обработки ТМИ не решена, а также проблема обеспечения санкционированного доступа к архивной информации.

Сеансовый режим работы системы телеметрии требует от оператора сектора анализа постоянного участия при организации сеансов, в то время как при возрастающем количестве проводимых сеансов данная процедура должна быть автоматизирована.

При проведении сеансов приёма и анализа телеметрии каждый клиент телеметрии производит ручной ввод параметров предстоящего сеанса, что с одной стороны приводит к возможным ошибкам ввода оператора, с другой – к нежелательным задержкам при организации подключения к серверу.

Во время сеанса приёма телеметрии с НИП сигнал/качество телеметрической информации может ухудшаться (возрастание процента сбойной информации) или полностью пропадать в виду атмосферных или иных возмущений, вносимых в радио-канал между КА и НИП. В этом случае для выбора другого НИП, возможно принимающего более качественный сигнал, оператору сектора анализа необходимо: во-первых организовать новый сеанс с подключением к требуемому НИП на КП ПСО, во-вторых повторно выполнить вручную процедуру организации сеанса для клиента телеметрии, со всеми вытекающими последствиями.

Использование различных аппаратно-программных средств для приёма ТМ информации от различных источников (САО-Ц, СОТИ) затрудняется их сопровождение, поддержку и эксплуатацию.

Кроме того, существующая система обработки телеметрии не предоставляет никаких инструментов в части передачи результатов обработки удалённым потребителям и внешним системам, таким, как, например, РЦУП или центр управления БРК.

Важной задачей АСУ КА является задача оценки состояния космического аппарата на участке выведения ещё до наступления контакта отделения посредством приёма телеметрии с разгонного блока. Решение этой задачи требует организации советующего информационно-логического взаимодействия между НИП и ЦУП с разработкой организационно-технических мероприятий и протоколов обмена специализированной телеметрической информацией.

Одной из ключевых задач в условиях расширения количества КА в орбитальной группировке становится задача автоматизированной подготовки различного рода отчётов (за неделю, за месяц и т.д.) на основании результатов обобщённой обработки телеметрической информации.

В имеющейся системе обработки телеметрии отсутствуют средства защиты от несанкционированного доступа, такие как авторизации клиентов телеметрии и защита передаваемой информации, что не соответствует требованиям безопасности центров управления специализированного назначения.

Решение описываемых проблем имеющейся системы обработки телеметрии весьма затруднительно ввиду использовавшихся процедурных методов при разработке ПО. Производительность, модульность и расширяемость существующей системы могут быть обеспечены только за счёт проектирования новой архитектуры системы многопоточной обработки телеметрии, с применением методов системного анализа и разработки программных компонент на основе объектно-ориентированных принципов, соответствующих стандартам качества программного обеспечения.

1.6 Принципы построения унифицированной системы приёма, обработки и анализа телеметрической информации

До настоящего времени подход к разработке специального программного обеспечения обработки телеметрической информации (СПО ОТИ) и его функциональному структурированию определялся требованиями, предъявляемыми специалистами по управлению конкретным КА и заложенными аппаратными характеристиками ЦУП, что привело к наличию большого числа различных версий программного обеспечения для каждого из аппаратов или

орбитальной их группировки. Сложившаяся ситуация препятствовала эффективному техническому сопровождению программного обеспечения, включая его наращивание и модернизацию. Увеличение числа и типов разрабатываемых космических аппаратов ещё больше обострило данную проблему.

Таким образом, актуальной стала задача унификации СПО ОТИ, т. е. создание единой системы обработки телеметрической информации. Решение обозначенной проблемы позволит обеспечить гибкость функциональной архитектуры программного обеспечения, а также его независимость от обслуживаемого космического аппарата. В работе [21] предлагается идеология построения унифицированного программного комплекса СПО ОТИ. Данная идеология предполагает реализацию следующих принципов:

1 системность, предопределяемую:
   - наличием функциональных и информационных связей в системе;
   - разделением функциональных задач системы между отдельными модулями на основе методов системного анализа;
2 гибкость программного обеспечения и совместимость его функциональных характеристик, обусловленные:
   - модульностью программного обеспечения, возможностью его расширения и изменения в соответствии с решаемыми задачами;
   - независимостью от операционной системы;
   - определение ключевых задач протоколов информационного взаимодействия с сопрягаемыми элементами системы;
3 автоматизации работы модулей системы, для чего требуется:
   - определение минимально необходимой входной информации;
   - минимизация степени участия человека в управлении системой;
   - обеспечение заданной степени обобщения выходной информации для передачи в смежные модули управления;
4 заданный уровень функциональности системы, обеспечивающий:
   - получение, обработку и хранение телеметрической информации различных типов;
   - автоматизированный контроль и диагностику состояния КА по ТМИ;
   - эффективное представление результатов обработки ТМИ.

1.7 Выводы по главе 1

1 Проведён анализ структуры автоматизированной системы управления космическим аппаратом.

2 Рассмотрена структурно-функциональная схема центра управления полётами. Обозначены задачи, решаемые секторами управления в составе центра управлениям полётами. Особо выделены задачи системы обработки и анализа телеметрической информации

3 Отмечена необходимость усовершенствования специального программного обеспечения обработки телеметрической информации. Показано, что с вводом новых типов КА в орбитальную группировку, возрастанием вычислительной нагрузки и расширением функциональных задач становятся очевидными, существующая система обработки телеметрии более не может обеспечивать полноты и качества выполнения возложенных на неё задач.

4 Описаны общие принципы построения унифицированных систем обработки телеметрической информации.

Глава 2. Проектирование автоматизированной системы многопоточного приёма, обработки и анализа телеметрической информации

До начала проектирования системы необходимо выделить ключевые функции (требования) будущей многопоточной системы приёма телеметрической информации. Последующий анализ сформулированных функций позволит произвести более тщательную разработку общей структуры системы. При этом будут определены элементы системы, задачи, решаемые каждой из подсистем, а также виды внутреннего и внешнего информационного взаимодействия. Центральным элементом новой системы является обслуживающая подсистема, поэтому построение её архитектуры должно быть произведено с особой тщательностью. Кроме того будут рассмотрены задачи и функции клиентов обработки телеметрии с целью формирования интерфейсов информационно-логического взаимодействия с сервером обработки телеметрической информации.

2.1 Определение ключевых функций системы приёма телеметрической информации

В соответствии с задачами исследования необходимо определить ключевые функции системы обработки телеметрии, которые будут заложены в основу проектирования новой системы. Под ключевыми функциями будем понимать набор требований и функциональных задач системы обработки телеметрической информации, предъявляемых к ней с точки зрения системы АСУ КА в целом. Перечь таких ключевых функций определяется требованиями технического задания на систему обработки телеметрии и включает следующие функции, представленные в таблице 2.1.

Таблица 2.1 – Базовые функции системы обработки телеметрии

Индекс	**Описание функции**
A001	хранение результатов управления и контроля в течение всего жизненного цикла КА
A002	непрерывный автоматизированный контроль, диагностика и отображение состояния КА при штатном и нештатном его функционировании с использованием графиков, мнемосхем и элементов визуализации, а также средств звуковой и световой сигнализации
A003	одновременный мониторинг не менее чем 4 КА

Продолжение таблицы 2.1

Индекс	**Описание функции**
A004	взаимодействие с САО-Ц, СОТИ для получения потоков ТМИ с наземных станций управления
A005	распознавание, выделение, проверка на достоверность и запись ТМИ в оперативный архив
A006	допусковый контроль параметров
A007	отображение значений заявленных оператором ТМ параметров и состояния систем спутника, в том числе в виде мнемосхем
A008	отображение каналов псевдокадра ТМИ в необработанном виде
A009	построение графиков поведения параметров
A010	отображение блоков информации (формуляров)
A011	отображение отчетной информации БЦВК с полной обработкой параметров во фразах и событиях
A012	сбор, накопление и систематизация информации о состоянии спутника по результатам обработки ТМИ и отчетов БЦВК
A013	непрерывный автоматизированный обобщенный контроль состояния спутника по ТМИ, с использованием звуковой и цветовой сигнализации в случае выхода любого параметра за пределы допуска
A014	предоставление имитационного режима функционирования (для отладочных и тренировочных целей) с использованием динамического программного имитатора
A015	выборку из архива любой группы ТМ параметров на заданном пользователем интервале времени (в пределах срока хранения) и отображение в заданном виде
A016	обслуживание заявок внутренних и внешних, по отношению к ЦУП, потребителей на получение ТМИ по согласованным протоколам
A017	наличие средств контроля входной информации
A018	независимость функциональных задач СПО от места их выполнения в сети рабочих станций ЦУП (РЦУП)
A019	предварительная подготовка и отладка исходных данных для обработки ТМИ, используемых при летной эксплуатации КА
A020	использование средств парольной защиты

Продолжение таблицы 2.1

Индекс	**Описание функции**
А021	функционирование программных средств в среде операционной системы Windows XP и выше, ОС МСВС 3.0
А022	наличие справочной системы
А023	устойчивость к отказам

Также сформулируем набор дополнительных функций системы обработки телеметрии по результатам предварительного исследования недостатков существующей системы (таблица 2.2):

Таблица 2.2 – Дополнительные функции системы обработки телеметрии

Индекс	**Описание функции**
Б001	масштабируемость СПО ОТИ
Б002	минимизация степени участия человека в управлении системой
Б003	обеспечение заданной степени обобщения выходной информации для передачи в смежные модули управления
Б004	параллельный приём и обработка телеметрической информации не менее чем с 8 НИП
Б005	одновременный приём и обработка телеметрической информации форматов САО-Ц, СОТИ, ЕЦУП РБ
Б006	обеспечение автоматического проведения сеанса съёма телеметрии в соответствии с планом
Б007	централизованное хранение архивов телеметрической информации
Б008	сжатие архивов телеметрической информации в реальном масштабе времени
Б009	обеспечение активного (приём информации в ЦУП от САО-Ц, СОТИ) и пассивного (приём информации от в РЦУП) режимов обработки телеметрии
Б010	возможность автономного функционирования без базы данных (БД) в случае нештатного разрыва связи с системой управления базой данных (СУБД)
Б011	адаптивный приём-передача потока телеметрических кадров клиентам телеметрии

Продолжение таблицы 2.2

Б012	автоматизированное формирование отчётов о состоянии отдельных параметров и целых групп с возможностью статистического анализа, представления результатов в виде графиков, подсчёт наработки бортовой аппаратуры. Их долговременное хранение в центральной БД
Б013	приём телеметрической информации с разгонного блока на участке выведения
Б014	система защиты от несанкционированного доступа
Б015	система восстановления после программных сбоев
Б016	наличие конфигурируемой программы установки дистрибутива
Б017	протоколирование событий
Б018	расширенное протоколирование событий по типам: сообщения базы данных, сетевого взаимодействия, служебные события, действия оператора
Б019	наличие методов автоматизированной организации сеанса съёма телеметрии
Б020	использование преимуществ многоядерной аппаратной конфигурации
Б021	наличие системы автоматического формирования и доставки отчётов об ошибках
Б022	использование системы отслеживания ошибок

2.2 Разработка общей структуры автоматизированной системы многопоточного приёма, обработки и анализа телеметрической информации

Для внесения ясности в дальнейшие рассуждения дадим следующие определения.

Однопоточная система обработки телеметрической информации – это аппаратно-программный комплекс, способный в любой момент времени принимать, обрабатывать и анализировать не более одного телеметрического потока, то есть с одного КА через один НИП. Такую систему можно сравнительно легко реорганизовать в псевдо-многопоточную путём создания дополнительных рабочих мест с размещением на них программных серверов обработки телеметрической информации. Однако данное решение является временным, не обеспечивает достаточной степени надёжности и к тому же не эффективно с экономической точки зрения. Функциональная структура такой системы была приведена на рисунке 3.1.

Многопоточная система обработки телеметрической информации – это аппаратно-программный комплекс, способный в любой момент времени принимать, обрабатывать и анализировать всё множество потоков телеметрической информации с n КА в орбитальной группировке через m НИП (требования А003, Б004) и обеспечивающий поддержку протоколов

доставки телеметрии от источников САО-Ц, СОТИ, ЕЦУП РБ (требования А004, Б005). Функционирование такой системы в контуре АСУ КА приведено на рисунке 2.1.

Из рисунка видно, что множество космических аппаратов орбитальной группировки $КА_1, \ldots, КА_n$ порождает эквивалентное множество потоков телеметрической информации, принимаемое наземными измерительными пунктами $НИП_1, \ldots, НИП_m$. Для приёма телеметрической информации на каждом НИП организуются сеансы связи с КА. В зависимости от аппаратно-программной конфигурации НИП, а также от загруженности каналов системы передачи данных НИП транслирует потоки телеметрической информации в ЦУП одним из протоколов доставки САО-Ц, СОТИ или ЕЦУП РБ. Таким образом, в многопоточной системе ОТИ для приёма информации организуются m сеансов, что соответствует общему числу потоков телеметрии через все доступные НИП. Ядром такой системы должна выступать обслуживающая подсистема, коммутирующая потоки телеметрии между различными источниками (НИП) и подсистемами отображения.

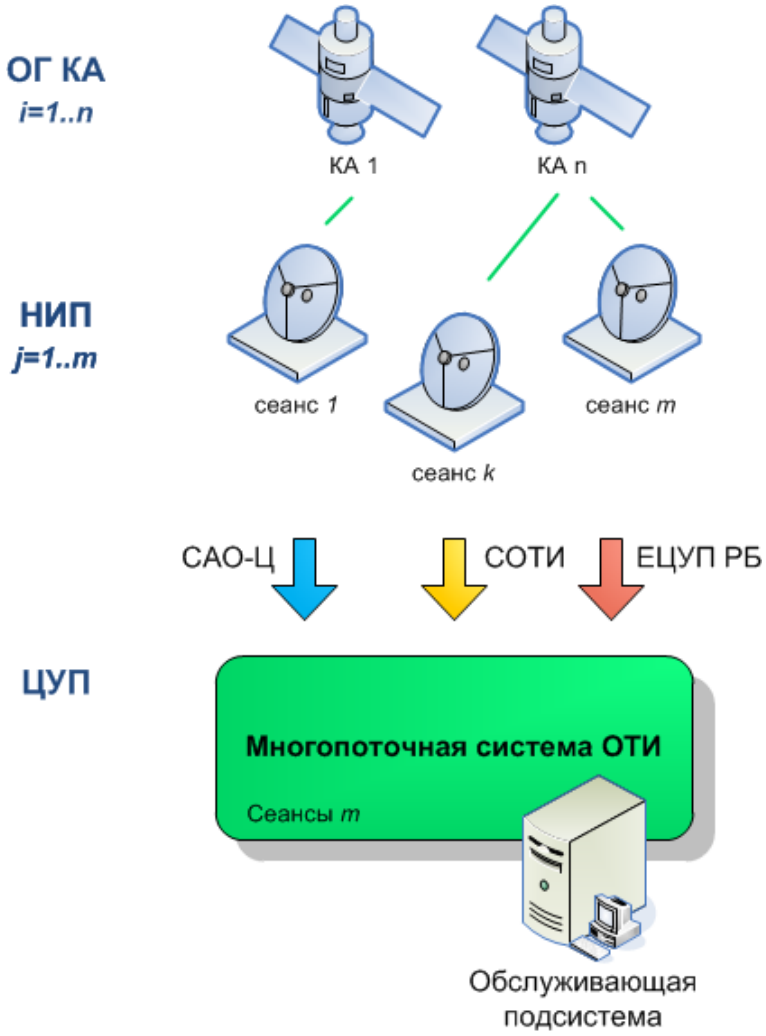

Рисунок 2.1 – Многопоточная система ОТИ в контуре АСУ КА

Для построения внутренней структуры многопоточной системы обработки телеметрической информации произведём классификацию множества сформулированных функций: а) по способу решения задачи обработки и анализа и б) по назначению. По способу решения задач обработки и анализа будем выделять сеансовые (онлайн) и внесеансовые (офлайн) функции. В свою очередь по назначению будем выделять функции обслуживающей подсистемы, функции подсистемы мониторинга и функции вспомогательных подсистем, как, например, подсистема подготовки исходных данных.

Построим матрицу трассируемости функций в функциональные подсистемы обработки телеметрии. Поскольку отдельные подсистемы ОТИ должны обеспечивать решение строго определённых онлайн или офлайн задач, то к группе онлайн функций отнесём обслуживающую подсистему и подсистему мониторинга. А к группе офлайн функций подсистему внесеансного мониторинга и вспомогательные подсистемы. Результирующая матрица представлена в таблице 2.3.

Таблица 2.3 – Матрица трассируемости функций в функциональные подсистемы

	Сеансный анализ		**Внесеансный анализ**	
	обслуживающая подсистема	**подсистема мониторинга**	**подсистема внесеансного мониторинга**	**вспомогательные подсистемы**
Основные функции				
A001	+			
A002	+	+		
A003	+	+		
A004	+			
A005	+			
A006	+	+		
A007		+	+	
A008	+	+	+	
A009		+	+	
A010		+	+	
A011		+	+	
A012	+			
A013		+	+	
A014	+	+		
A015			+	
A016	+			
A017	+	+	+	
A018	+	+	+	+

Продолжение таблицы 2.3

А019				+
А020	+	+	+	+
А021	+	+	+	+
А022	+	+	+	+
А023	+	+	+	+
Дополнительные функции				
Б001	+	+	+	+
Б002	+	+	+	+
Б003	+			+
Б004	+	+		
Б005	+	+		
Б006	+	+		
Б007	+			
Б008	+			
Б009	+	+		
Б010	+	+	+	
Б011	+	+		
Б012	+	+		
Б013	+	+		
Б014	+	+	+	+
Б015	+	+	+	+
Б016	+	+	+	+
Б017	+	+	+	+
Б018	+	+	+	+
Б019	+	+		
Б020	+	+		
Б021	+	+	+	+
Б022	+	+	+	+

Проиллюстрируем на схеме информационное взаимодействие рассматриваемых подсистем (рисунок 2.2).

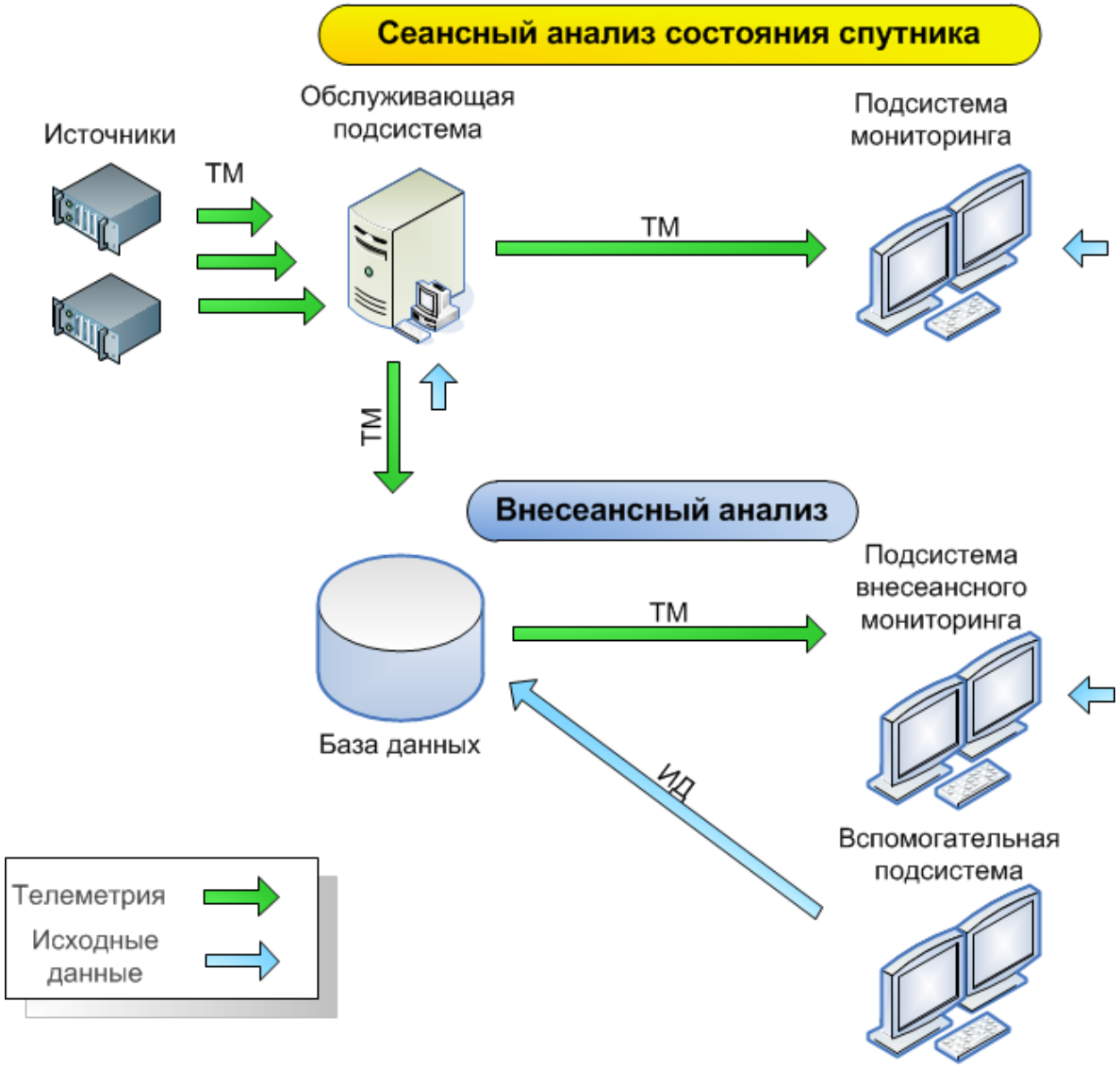

Рисунок 2.2 – Информационное взаимодействие подсистем обработки телеметрической информации

По схеме видно, что в многопоточной системе обработки телеметрической информации центральным элементом является обслуживающая подсистема, которая обеспечивает одновременный приём от источников (САО-Ц, СОТИ, ЕЦУП РБ) множества потоков телеметрической информации, организует долговременный архив в БД ЦУП, предоставляет санкционированный доступ к телеметрической информации для подсистем мониторинга и рассылает принимаемую информацию потребителям по согласованному протоколу.

Подсистема сеансного (онлайн) мониторинга принимает телеметрическую информацию от обслуживающей подсистемы и выполняет задачи сеансного анализа телеметрии, основными из которых являются: первичная и вторичная обработка данных, формирование параметров обобщённого контроля, построение отвечающих им зависимостей, обработка отчётов бортового вычислительного комплекса, построение графиков поведения телеметрических параметров, обработка отчётов БЦВК, построение модели функционирования спутника в виде

мнемонических схем. Следуя принципу автоматизации функционирования модулей системы [21] из общего множества задач подсистемы сеансного мониторинга можно выделить следующие модули: модуль рабочего места обработки телеметрии (РМТМ) для отображения и анализа примитивных объектов (параметры, формуляры, графики) и модуль мнемонического представления телеметрической модели (МПТМ) для анализа комплексных объектов – мнемонических диаграмм.

Накопленная в БД ЦУП телеметрическая информация оценивается подсистемой внесеансного (офлайн) мониторинга (ПВРТМ). Набор функциональных возможностей подсистемы совпадает с возможностями подсистемы онлайн мониторинга и расширяет их методами статистического анализа и прогнозирования временных рядов значений телеметрических параметров.

Все рассмотренные подсистемы во время своего функционирования используют исходные данные (ИД) на обработку информации, расположенные в БД ЦУП и формируемые подсистемой подготовки исходных данных. Исходя из логики обработки телеметрической информации и в соответствии со структурами, формируемыми посредством БЦВК, данная подсистема обеспечивает формирование унифицированных структур для обработки ТМИ, поступающей в ЦУП КА. Помимо логики обработки параметров, генерируемых на борту, названная подсистема позволяет формировать группу параметров обобщенного контроля состояния КА, которые, в свою очередь, могут быть выстроены в многоуровневую иерархию параметров, начиная от первичных параметров (формируемых на основе бортовых датчиков) и заканчивая параметрами, описывающими системы КА и аппарат в целом. Совокупность сформированных унифицированных структур выступает в качестве исходных данных на обработку телеметрии для обслуживающей подсистемы и подсистем мониторинга.

Преимущество предлагаемой структуры заключается в том, что для её внедрения требуется модификация только внутренней архитектуры системы обработки телеметрической информации. Внешнее взаимодействие системы ОТИ с другими системами ЦУП (навигационно-баллистическое обеспечение, планирование и командно-программное обеспечение) сохраняется без изменений и определяется соответствующими информационно-логическими протоколами.

Важно отметить, что более 70% функций системы многопоточной обработки телеметрической информации, представленные в таблице 2.3, являются общими для нескольких подсистем. Однако для исключения дублирования [21] реализации совпадающих функций различными подсистемами телеметрии необходимо разработать унифицированные методы

решения таких задач, а совокупность общих методов организовать в виде разделяемой библиотеки методов обработки и анализа телеметрической информации.

2.3 Подсистема на уровне сервера обработки телеметрии

2.3.1 Архитектура обслуживающей подсистемы

Согласно предложенной выше (рисунок 2.2) архитектуре системы обработки телеметрической информации центральным элементом в такой системе является обслуживающая подсистема или сервер обработки телеметрии. Поэтому необходимо с особой тщательностью подойти к построению архитектуры данной подсистемы.

По своему назначению подсистема обработки телеметрической информации должна обладать следующими функциями: одновременный мониторинг не менее чем 24 КА, параллельный приём и обработка телеметрической информации не менее чем с 8 НИП, одновременный приём и обработку телеметрической информации форматов САО-Ц, СОТИ, ЕЦУП-РБ, централизованное хранение архивов телеметрической информации и др. (полный перечень приведён в таблице 2.3). С учётом сформулированных требований была построена архитектура подсистемы обработки телеметрической информации, представленная на рисунке 2.3.

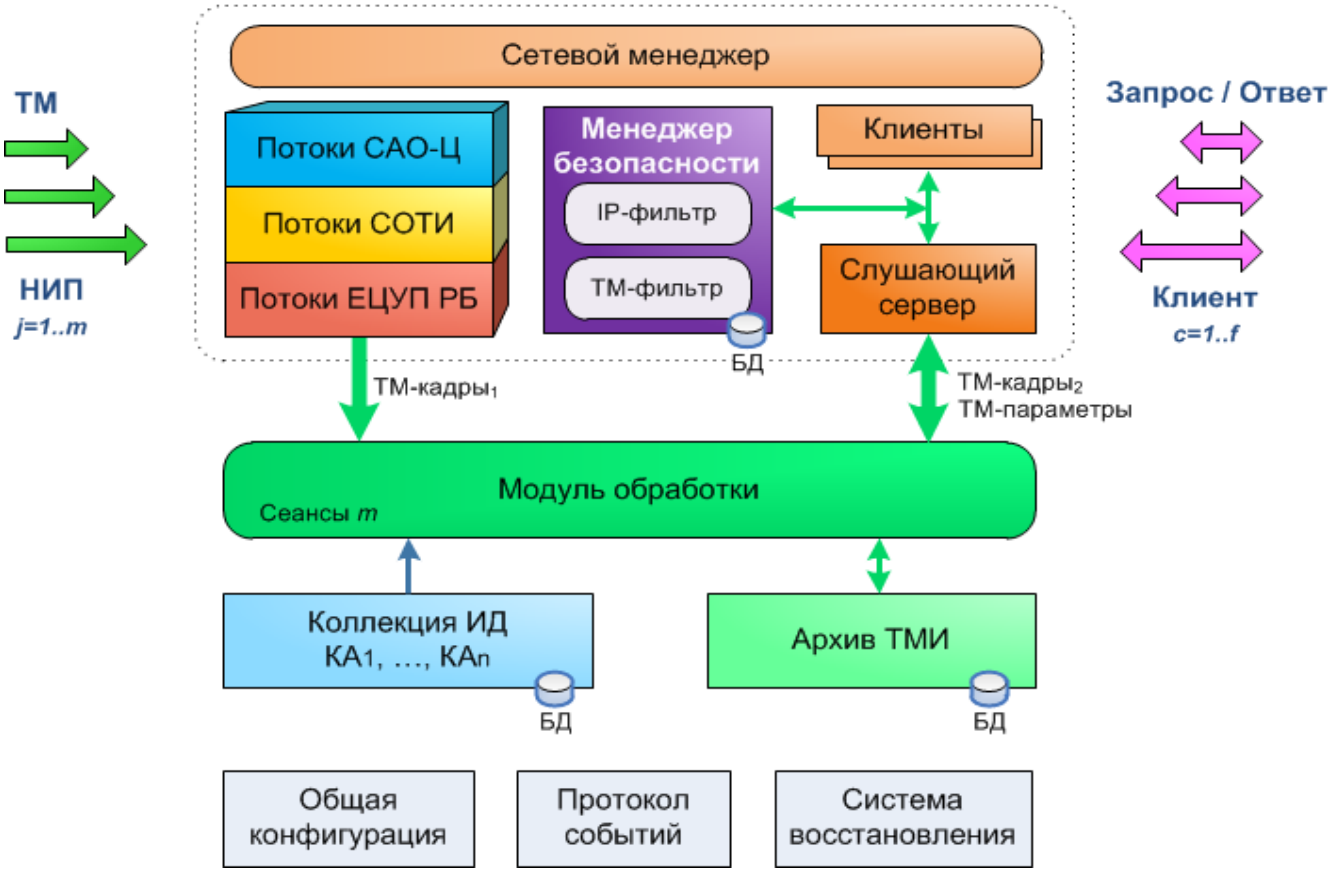

Рисунок 2.3 – Архитектура подсистемы сервера обработки телеметрии

Задачи приёма телеметрических кадров от НИП и рассылки обработанных кадров и ТМ-параметров клиентам решает *сетевой менеджер*. В составе сетевого менеджера имеется блок приёма телеметрических кадров от НИП, состоящий из трёх модулей, предназначенных для обработки потоков телеметрии в форматах САО-Ц, СОТИ или ЕЦУП РБ соответственно. Каждый модуль приёма обеспечивает установление и поддержание множества каналов приёма телеметрической информации. К задачам блока приёма относятся выделение ТМ-кадров из входящего потока сообщений, приведение телеметрических кадров к единой структуре и их дальнейшая передача в м*одуль обработки*.

Модуль обработки, принимая очередной ТМ-кадр, на основании коллекции загруженных исходных данных на обработку телеметрической информации производит контроль достоверности с выставлением признаков результата контроля. Дополнительно производит расчёт первичных, вторичных и параметров автоматизированного обобщённого контроля (АОК), выделяет отчёты бортового компьютера и передаёт ТМ-кадры и отчёты бортового компьютера в модуль ведения архивов для записи в оперативный архив. Данные из оперативного архива с определённой периодичностью записываются в долговременный архив базы данных в сжатом виде. Кроме того, модуль обработки транслирует обработанные и унифицированные ТМ-кадры, значения ТМ-параметров в слушающий сервер для последующей рассылки клиентам.

В составе сетевого менеджера имеется блок управления подключениями клиентов, состоящий из *слушающего сервера*, *менеджера безопасности* и *коллекции соединений с клиентами*. Циклограмма взаимодействия сервера и клиента выглядит следующим образом.

1 Внешний клиент отправляет в слушающий сервер запрос на подключение.

2 Слушающий сервер принимает запрос на подключение и, используя менеджер безопасности, проверяет валидность IP-адреса внешнего клиента. В случае успеха происходит переход к шагу 3, иначе отказ в подключении и разрыв соединения.

3 Сервер организует внутренний клиентский объект для каждого внешнего клиента успешно прошедшего проверку IP-адреса и помещает его в коллекцию соединений с внешними клиентами. Дальнейшее взаимодействие внешнего клиента и сервера обработки телеметрии производится через соответствующий внутренний клиентский объект.

4 Внешний клиент отправляет во внутренний клиент параметры телеметрической учётной записи для получения телеметрических кадров и/или телеметрических параметров

5 Внутренний клиент принимает параметры учётной записи и, используя менеджер безопасности, производит аутентификацию. В случае успеха переход к шагу 6, иначе отправка сообщения об ошибке авторизации и дальнейшая передача телеметрии блокируется.

6 Внешний клиент отправляет во внутренний клиент параметры сеанса связи: номер КА, номер НИП, номер витка, номер сеанса.

7 Внутренний клиент регистрирует принятые параметры сеанса связи.

8 При получении очередного ТМ-кадра от модуля обработки слушающий сервер осуществляет его рассылку всем внутренним клиентам, чьи параметрами сеанса соответствуют текущему ТМ-кадру.

9 Внутренний клиент при получении от слушающего сервера очередного ТМ-кадра обращается к менеджеру безопасности для применения ТМ-фильтра к ТМ-кадру. После чего производится непосредственная отправка отфильтрованного ТМ-кадра внешнему клиенту.

Совокупность конфигурационных параметров обслуживающей подсистемы, к которым относятся порт слушающего сервера, учётная запись базы данных, путь к оперативному архиву, настройки журналов протоколирования, множество IP-адресов источников телеметрии – поддерживается *модулем общей конфигурации.* Параметры общей конфигурации используются всеми модулями обслуживающей подсистемы.

Модуль протокола событий является средством сопровождения журналов действий оператора, операций ввода/вывода, сетевых событий, сообщений базы данных и также используется всеми модулями обслуживающей подсистемы.

Отказоустойчивость системы достигается, прежде всего, тщательным проектированием. Однако даже при соблюдении общих правил и рекомендаций к проектированию в процессе штатной эксплуатации могут возникать аппаратные и программные сбои. Сокращение времени восстановления после программных сбоев достигается за счёт применения *системы восстановления*, которая дополнительно обеспечивает формирование и отправку отчётов об ошибках разработчику для дальнейшего изучения, анализа и устранения неисправностей.

2.3.2 Расчёт показателей эффективности обслуживающей подсистемы

Построенная в соответствии с заявленными функциями подсистема может быть проанализирована средствами имитационного моделирования, поскольку по своей сути представляет систему массового обслуживания (СМО), которая производит обслуживание поступающих в неё требований – заявок. Заявками для обслуживающей подсистемы с одной стороны выступают телеметрические кадры о состоянии $КА_i$, принимаемые в различных

форматах от $НИП_j$, и представляющие собой входящий поток информации. С другой стороны заявки-запросы от клиентов обработки телеметрии на получение значений ТМ-параметров или отдельных телеметрических кадров – то есть исходящий поток информации.

Под эффективностью обслуживающей системы понимают характеристику уровня выполнения этой системой тех функций, для которых она предназначена. Выбор показателя эффективности зависит от той задачи, которая поставлена перед исследованием [22]. Приведём наиболее часто используемые показатели эффективности обслуживающих систем [23]:

- среднее число требований, которое может обслужить система за единицу времени (абсолютная пропускная способность);
- отношение среднего числа требований, которое может обслужить система за единицу времени, к среднему числу поступивших за это время требований (относительная пропускная способность);
- среднее число требований, находящихся в очереди;
- среднее число занятых каналов;
- среднее время ожидания в очереди;

Для сервера обработки телеметрической информации в качестве основного показателя эффективности выберем относительную и абсолютную пропускные способности входящего и исходящего потоков.

Поскольку приём телеметрической информации обеспечивается множеством независимых каналов приёма, то входящий поток представляет собой m независимых одноканальных СМО с ожиданием.

Рассмотрим одноканальную СМО с ожиданием [24], [25], [26]. Входящий поток заявок на обслуживание поток имеет интенсивность λ. Интенсивность потока обслуживания равна μ. Длительность обслуживания — случайная величина, подчинённая показательному закону распределения. Заявка, поступившая в момент, когда канал занят, становится в ограниченную очередь и ожидает обслуживания. Независимо от того, сколько требований поступает на вход обслуживающей системы, данная система (очередь + обслуживаемые клиенты) не может вместить более N-требований (заявок), из которых одна обслуживается, а (N-1) ожидают. Клиенты, не попавшие в ожидание, теряются. Источник, порождающий заявки на обслуживание, имеет неограниченную емкость [24], [25].

Обозначим P_n – вероятность того, что в системе находится n заявок. Эта величина вычисляется по формуле:

$$P_n = \begin{cases} \frac{1-\rho}{1-\rho^{N+1}} \cdot \rho^n, \ \ \rho \neq 1, n = 0,1,2,\dots,N; \\ \frac{1}{(N+1)}, \rho = 1; \end{cases} \quad (2.1)$$

Здесь $\rho = {}^{\lambda}/_{\mu}$ – приведенная интенсивность потока, которая показывает степень согласованности входного и выходного потоков заявок канала обслуживания и определяет устойчивость системы массового обслуживания. Тогда вероятность того, что канал обслуживания свободен и в системе нет ни одного клиента, равна: $P_0 = \frac{1-\rho}{1-\rho^{N+1}}$.

С учетом этого можно обозначить

$$P_n = \begin{cases} P_0 \cdot \rho^n, \ \ \rho \neq 1, n = 0,1,2,\dots,N; \\ \frac{1}{(N+1)}, \rho = 1; \end{cases} \quad (2.2)$$

Определим характеристики одноканальной СМО с ожиданием и ограниченной длиной очереди, равной (N-1) [24], [27]:

- вероятность отказа в обслуживании заявки:

$$P_{\text{отк}} = P_n = \begin{cases} \left(\frac{1-\rho}{1-\rho^{N+1}}\right) \cdot \rho^N, \ \ \rho \neq 1; \\ \frac{1}{(N+1)}, \rho = 1; \end{cases} \quad (2.3)$$

- относительная пропускная способность системы:

$$q = 1 - P_{\text{отк}} = \begin{cases} 1 - \left(\frac{1-\rho}{1-\rho^{N+1}}\right) \cdot \rho^N, \ \ \rho \neq 1; \\ 1 - \frac{1}{(N+1)}, \rho = 1; \end{cases} \quad (2.4)$$

- абсолютная пропускная способность:

$$A = q \cdot \lambda \quad (2.5)$$

- среднее число находящихся в системе заявок:

$$L_s = \sum_{n=0}^{N} n \cdot P_n = \begin{cases} \frac{\rho \cdot [1 - (N+1) \cdot \rho^N + N \cdot \rho^{N+1}]}{(1-\rho) \cdot (1-\rho^{N+1})}, \ \ \rho \neq 1; \\ {}^{N}/_{2}, \rho = 1; \end{cases} \quad (2.6)$$

- среднее время пребывания заявки в системе:

$$W_s = \frac{L_s}{\lambda \cdot (1 - P_N)} \quad (2.7)$$

- средняя продолжительность пребывания клиента (заявки) в очереди:

$$W_q = W_s - {}^1/_\mu \tag{2.8}$$

- среднее число заявок (клиентов) в очереди (длина очереди):

$$L_q = \lambda \cdot (1 - P_N) \cdot W_q \tag{2.9}$$

Применительно к системе обработки телеметрической информации имеем следующие входные сведения: интенсивность каждого потока поступления заявок (телеметрических кадров) при информативности передачи телеметрии 8000 бит/с составляет $\lambda = 1.8$ кадров/сек, максимальная длина очереди ожидания $m = 10$ кадров и все потоки событий (получения кадров и обслуживания) имеют характер простейших пуассоновских потоков. Показатель интенсивности обслуживания μ каждого приёмного канала является неизвестным параметром и в аппаратно-программной реализации предлагаемой архитектуры зависит, во-первых, от производительности аппаратной платформы, а во-вторых, от общей загруженности самой обслуживающей системы (количество активных каналов приёма телеметрии, количество клиентов телеметрии, объём исходных данных на обработку телеметрии). Поэтому необходимо определить оптимальное значение интенсивности обслуживания μ, при котором поток телеметрических кадров каждого канала приёма будет обрабатываться с минимальными потерями. Условием оптимальности будем считать удовлетворение в среднем из каждых 100 заявок не менее 99 заявок на обработку (то есть отказ должны получать не более 1% заявок).

Будем постепенно увеличивать интенсивность обслуживания μ и определим по формулам (2.3), (2.4), (2.5) для получаемой СМО характеристики обслуживания. Например, при $\mu = \lambda = 1.8$ имеем:

$P_0 = 0.28554$, $q = 0.91667$, $A = 1.65$ и т.д

Значение характеристик СМО сведём в таблицу 2.4.

Таблица 2.4 – Сводная таблица характеристик одноканальной СМО при $\mu = 1.8 \ldots 2.6$

Характеристика обслуживания	**Значение интенсивности обслуживания μ**					
	1.8	2	2.2	2.4	2.5*	2.6*
Относительная пропускная способность q	0.91667	0.95627	0.97802	0.9891	0.9923	0.99455
Абсолютная пропускная способность A	1.65	1.72128	1.76044	1.78037	1.78615	1.79018

Примечание – символом «*» в таблице отмечены значения интенсивности обслуживания μ, удовлетворяющие условию оптимальности.

По условию оптимальности $q \geq 0.99$, следовательно, при условии полной нагрузки на сервер обработки телеметрии каждый канал должен обеспечивать интенсивность обслуживания

μ не хуже 2.5 кадра/сек. При этом в среднем каждую секунду будет обслуживаться 1,786 телеметрических кадра.

2.4 Подсистема на уровне клиентов обработки телеметрии

Назначение и особенности обслуживающей подсистемы были рассмотрены выше. Такая подсистема решает важнейшие задачи – а) коммутацию телеметрических потоков между всеми доступными НИП и ЦУП и б) обслуживание заявок внутренних и внешних, по отношению к ЦУП, абонентов на получение ТМИ по согласованным протоколам (требование А016). Представлением телеметрической информации в обработанном виде потребителю занимаются клиенты обработки телеметрии, обеспечивая различную степень детализации и обобщения сведений о состоянии КА. Произведём их классификацию. Необходимость классификации клиентов обработки телеметрии обусловлена различиями в решаемых задачах и способах взаимодействия с центральными элементом системы обработки телеметрии, то есть обслуживающей подсистемой.

С одной стороны имеем совокупность подсистем сеансового и внесеансового мониторинга (рисунок 2.2), входящих в состав системы обработки телеметрической информации и призванных обеспечить автоматизированную обработку, детальный анализ и отображение телеметрической информации на средствах ЦУП. Такие подсистемы будем называть *внутренними клиентами телеметрии*.

С другой стороны имеются территориально удалённые по отношению к системе приёма телеметрии абоненты, например центр управления ретранслятором, которые также запрашивают результаты обработки телеметрии у обслуживающей подсистемы и самостоятельно используют полученный результат для дополнительного анализа и выдачи управляющих воздействий. Таких абонентов-потребителей телеметрии будем назвать *внешними клиентами телеметрии*.

Задачи проектирования и разработки клиентов телеметрии выходят за рамки настоящего диссертационного исследования, но их рассмотрение, определение общих требований важно с точки зрения описания интерфейсов информационно-логического взаимодействия с сервером обработки телеметрической информации.

2.4.1 Внутренние клиенты

В подразделе 2.2 отмечалось, что подсистема сеансного мониторинга должна состоять из модуля рабочего места обработки телеметрии (РМТМ), предназначенного для отображения и анализа примитивных объектов (параметры, формуляры, графики), и модуля представления состояния КА в виде мнемонических диаграмм (МПТМ). Кроме того, оценкой накопленной в

БД ЦУП телеметрической информации должна заниматься подсистема внесеансного (офлайн) мониторинга (ПВРТМ).

Сформулируем общие требования, предъявляемые к внутренним клиентам:

1 Разделение функциональных задач между клиентами. Сеансный модуль МПТМ – для представления КА в виде мнемонических диаграмм. Сеансный модуль РМТМ – для детального анализа онлайн телеметрии в виде формуляров, графиков, телеметрических параметров и отдельных байт ТМ-кадра. Внесеансный модуль ПВРТМ – для анализа телеметрии из долговременного архива.

2 Получение телеметрической информации для сеансного анализа через взаимодействие с обслуживающей подсистемой по отдельному информационно-логическому протоколу.

3 Внесеансовый анализ телеметрии должен производиться чтением долговременных архивов напрямую из базы данных.

4 Использование методов, автоматизирующих начало сеанса приёма телеметрии (требование Б002). В существующей системе обработки телеметрии при каждом начале или смене сеанса связи оператор производит ручной ввод параметров, зачастую допуская ошибки ввода. Для обеспечения оперативного начала сеанса и исключения человеческого фактора необходимо внедрить методы автоматизации начала сеанса. Кроме того, такие методы могут быть использованы для переключения между несколькими НИП (требование Б011), принимающими сигнал с одного и того же КА, с целью автоматического выбора источника наилучшего сигнала.

5 Использование методов аутентификации для подключения к серверу (требования А020, Б014) позволит исключить передачу телеметрической информации несанкционированным клиентам и, при необходимости, производить ограниченную передачу на основе системы привилегий.

6 Применение методов обработки телеметрического кадра, используемых в сервере, для исключения дублирования реализации указанных методов.

2.4.2 Внешние клиенты

Состав внешних по отношению к системе обработки телеметрии клиентов определяется задачами при управлении КА и поэтому может уточняться. Тем не менее, все виды заявок внешних клиентов могут быть описаны следующим образом:

1 Текущее значение одного или нескольких телеметрических параметров $Param_i$, причем $j = 1,\ldots,N$, а N – общее количество телеметрических параметров.

2 Значение одного или нескольких телеметрических параметров из оперативного архива на интервале времени $\Delta t = t_2 - t_1$, причём $t_2 \leq t_{тек}$.

3 Последний принятый отчёт бортового компьютера $ObcReport_k$ определённого типа.

Таким образом, для обслуживания заявок внешних клиентов необходима разработка специализированного информационно-логического протокола, обеспечивающего передачу описанных выше видов информации. Функции приёма заявок, обработки и формирования результата будет выполнять обслуживающая подсистема.

2.5 Информационный протокол взаимодействия подсистем

Для обеспечения взаимодействия внутренних клиентов (подсистем онлайн мониторинга) с обслуживающей подсистемой на базе «Протокола взаимодействия САО и СПО управления» [28] был разработан информационный протокол взаимодействия. Подробное описание приведено в приложении А.

Типовая циклограмма взаимодействия обслуживающей подсистемы и клиентов описывается следующими процедурами:

1 Установление подключения

Для установления подключения сетевой клиент отправляет запрос *CONNECT* с указанием IP-адреса и порта слушающего сервера обслуживающей подсистемы. Слушающий сервер выдаёт команду *ACCEPT* и организует канал связи с клиентом.

2 Отправка параметров аутентификации

Для получения разрешения на приём любого вида телеметрической информации клиент сообщает серверу параметры аутентификации в виде директивы *dtAuthentication* с заполнением полей: номер директивы, признаки (требование квитанции), вид информации = 1888, длина информации, причём после сетевого заголовка обязательно следуют имя и пароль учётной записи.

3 Получение таблицы состава сеансов

С целью исключения ручного ввода параметров сеанса связи (требование Б002) обслуживающая подсистема каждые пять секунд отправляет клиенту таблицу текущего состава сеансов в виде директивы *dtActiveSeances* с информацией: номер КА, номер витка, номер сеанса, номер НИП-КТС, скорость поступления телеметрической информации байт/с, состояние передаваемого потока (0 – нет ТМИ, 1 – сбойные кадры, 2 – достоверная ТМИ, 3 сеанс завершён, 255 – сеанс не начат). Клиент использует принятые параметры для выбора интересующего сеанса связи и отправки на сервере команды о начале сеанса.

4 Запрос набора телеметрических параметров

Клиент формирует и отправляет, а обслуживающая подсистема принимает и обрабатывает заявки, описанные в 2.4.2, в виде директивы *dtTmiXml*. Формат заявок соответствует универсальному протоколу передачи состояния космического аппарата, описанному в 2.6.

5 Начало сеанса

Для организации тракта приёма телеметрической информации в реальном масштабе времени клиент сообщает серверу параметры предстоящего сеанса связи в виде директивы *dtStart*.

6 Приём телеметрии

Для непосредственного начала приёма телеметрической информации в виде телеметрических кадров в реальном масштабе времени клиент отправляет на сервер директиву *dtTmiToOtiClient*. В случае если обслуживающая подсистема функционирует исправно и обеспечивает приём телеметрии от источников, телеметрические кадры начинают поступать к клиенту кадр за кадром. В противном случае, клиент становится в ожидание получения телеметрических кадров до устранения причин отсутствия или нарушения приёма телеметрии в обслуживающей подсистеме (внешних источниках телеметрии).

7 Завершение сеанса

Для закрытия тракта приёма телеметрической информации клиент направляет в сервер параметры проводимого сеанса в виде директивы *dtEnd*. В случае аварийного обрыва связи между клиентом и обслуживающей подсистемой все открытые ранее клиентом направления для приёма информации автоматически аннулируются.

2.6 Универсальный протокол передачи состояния космического аппарата

Состояние отдельных датчиков, блоков, модулей и подсистем может быть описано в виде совокупности значений телеметрических параметров, поэтому для обеспечения унифицированного описания состояния космического аппарата был предложен и разработан информационно-логический протокол на основе XML сообщений. Более глубокую и узкоспециализированную оценку функционирования бортовой аппаратуры производят внешние потребители телеметрии, то есть внешние клиенты, а обслуживание их заявок на телеметрические параметры осуществляет обслуживающая подсистема, используя указанный информационно-логический протокол на основе XML сообщений, описанный в приложении Б.

Типовая циклограмма взаимодействия обслуживающей подсистемы и внешних потребителей телеметрии описывается следующими процедурами:

1 Подключение к обслуживающей подсистеме

2 Подготовка транспортного заголовка в виде директивы dtTmiXml

3 Подготовка заявки в виде XML сообщения

4 Отправка пакета, состоящего из транспортного заголовка и заявки

5 Ожидание ответа

6 Отключение от обслуживающей подсистемы

Предложенный универсальный протокол передачи состояния космического аппарата обеспечивает обслуживание следующих видов заявок:

- Запрос текущих значений телеметрических параметров

Заявка содержит номер космического аппарата и перечень запрашиваемых телеметрических параметров.

Ответ содержит совокупность векторов значений запрошенных параметров

- Запрос значений параметров из долговременного архива

Заявка содержит номер космического аппарата, интервал времени и перечень запрашиваемых телеметрических параметров.

Ответ содержит совокупность векторов значений запрошенных параметров

- Запрос отчёта бортового компьютера

Заявка содержит номер космического аппарата и тип запрашиваемых отчётов бортового компьютера, а именно: New – только *новые* отчёты, полученные с момента последнего запроса. Если с момента предыдущего запроса новых отчётов получено не было, запрос вернёт пустое множество. Last – только последний принятый отчёт.

Ответ содержит дату, время, наименование, длину и признаки достоверности последнего принятого отчёта бортового компьютера.

Примечание – В случае ошибки при обработке заявка отклоняется, а обслуживающая подсистема формирует квитанцию с кодом ошибки и подробным описанием.

2.7 Выводы по главе 2

1 Определены ключевые функции системы приёма телеметрической информации

2 Произведена разработка общей структуры системы. Определены элементы системы, задачи, решаемые каждой из подсистем, а также виды внутреннего и внешнего информационного взаимодействия

3 Предложена архитектура обслуживающей подсистемы, состав и назначение её модулей. Произведён расчёт показателей эффективности обслуживающей подсистемы

4 Определён информационный протокол взаимодействия подсистем на уровне внутренних клиентов

5 Предложен универсальный протокол передачи состояния космического аппарата внешним абонентам

Глава 3. Проектирование унифицированных средств описания исходных данных, алгоритмов и методов обработки и анализа телеметрической информации

Специальное программное обеспечение обработки телеметрической информации (СПО ОТИ) разбито на несколько программных комплексов. Схожесть решаемых задач, применяемых алгоритмов расчёта, постоянно растущие требования к платформе космического аппарата диктуют необходимость проектирования унифицированных средств описания исходных данных, алгоритмов и методов обработки и анализа. Общие принципы идеологии унификации были изложены в главе 1. Необходимо выбрать подходящий метод проектирования. В соответствии с выбранным методом определить сущности, их атрибуты, информационные потоки и взаимодействие между сущностями предметной области. Результатом проектирования будет являться библиотека унифицированного описания исходных данных, обработки и анализа телеметрической информации.

3.1 Специальное программное обеспечение автоматизации приёма, обработки и анализа телеметрической информации

Специальное программное обеспечение обработки телеметрической информации является частью специального программного обеспечения управления ЦУП КА и представляет собой комплекс программ, обеспечивающий решение перечня задач телеметрического сектора программно-аппаратного комплекса ЦУП [7], [10], [6]. Для удобства функционирования и эксплуатации СПО ОТИ разбито на несколько программных комплексов:

- проведения сеанса обработки (КП ПСО);
- рабочего места телеметриста (КП РМТМ);
- проведения внесеансных работ (КП ПВРТМ).

Для более эффективного решения локальных задач телеметрического сектора используется клиент-серверная архитектура распределенных вычислений.

Комплекс программ проведения сеанса обработки (КП ПСО), представляющий собой серверную часть в терминах архитектуры клиент-сервер, обеспечивает:

- взаимодействие с внешними программными комплексами (СОТИ, САО-Ц), обеспечивающими взаимодействие с наземными измерительным пунктами в части управления проведением телеметрических сеансов;
- настройку на работу по конкретному изделию, в зависимости от плана работ;

- получение по локальной вычислительной сети и обработку полных (сокращенных) потоков всех видов телеметрической информации;
- оперативное отображение значений ТМ-параметров на экран оператора сектора управления;
- организацию файлов-архивов телеметрической информации на локальном диске.

Комплекс программ рабочего места телеметриста (КП РМТМ). Данный комплекс представляет собой клиентскую часть в структуре клиент-сервер. КП РМТМ обеспечивает:

- получение ТМИ от сервера КП ПСО;
- обработку, анализ полученной информации и представление результатов обработки системным специалистам и операторам управления.

Комплекс программ проведения внесеансных работ с телеметрической информацией (КП ПВРТМ) предназначен для проведения работ, связанных с просмотром и документированием результатов обработки ТМИ, находящихся в архиве ЦУП; а также для просмотра и печати файлов с результатами документирования всех видов телеметрической информации.

Существующая схема взаимодействия СПО ОТИ, представлена на рисунке 3.1.

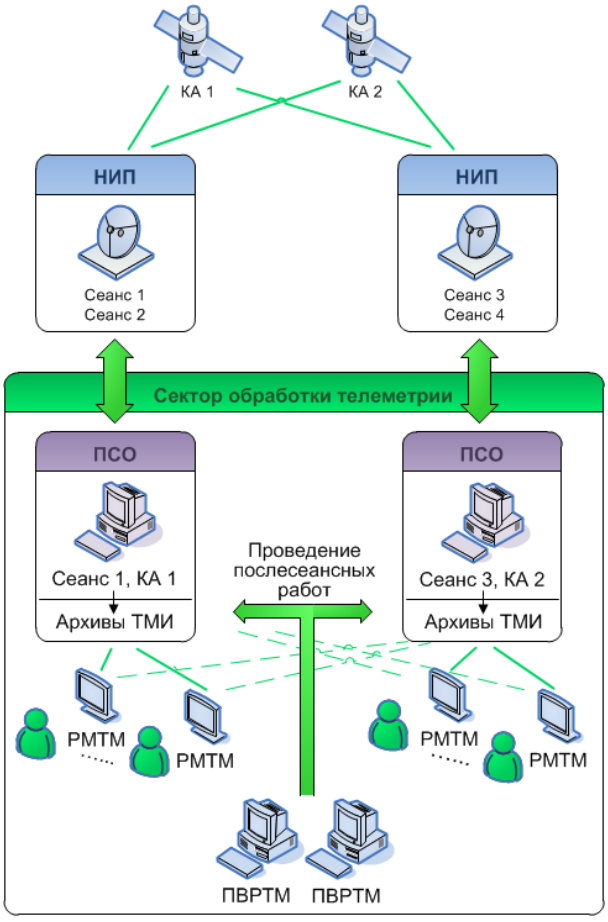

Рисунок 3.1 – Существующая схема взаимодействия СПО ОТИ

3.2 Выбор метода проектирования

Ввиду сложности проектируемой системы, заключающейся в количестве взаимодействующих элементов системы и множестве разнородных циркулирующих потоков данных, требуется применение эффективной, с точки зрения полноты выполнения поставленных задач, методологии анализа и проектирования. Большинство методов проектирования, по мнению Саммервилля, можно разделить на три основные группы [29]:

- метод структурного проектирования сверху вниз;
- метод потоков данных;
- объектно-ориентированный анализ и проектирование.

Структурный подход, не смотря на своё широкое распространение, не позволяет выделить абстракции и обеспечить ограничение доступа к данным; он также не предоставляет достаточных средств для организации параллелизма [30], и, как правило, неэффективен в объектно-ориентированных языках программирования.

В методе потоков система рассматривается как преобразователь входных потоков в выходные. Как и структурный метод, с успехом применяется при решении сложных задач, в которых существуют прямые связи между входными и выходными потоками системы и где не требуется уделять особого внимания быстродействию [30].

В основе объектно-ориентированного проектирования лежит представление о том, что программную систему необходимо проектировать как совокупность взаимодействующих друг с другом объектов. Такой подход отражает топологию новейших языков программирования высокого уровня, таких как Java, C++, C#. Кроме того он позволяет абстрагироваться от деталей реализации и анализировать объект как некоторую обобщённую идеализированную модель.

Анализ литературы показывает, что объектный подход к моделированию предметной области развивается в двух направлениях: объектно-ориентированный анализ и объектно-ориентированное проектирование.

Фундаментальные подходы в объектно-ориентированном анализе заложены в трудах С. Шлеер и С. Меллора [31]. Объектно-ориентированное проектирование как целостная концепция сформулирована Г. Бучем [30].

Объектно-ориентированный анализ – это методология, при которой требования к системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области. Объектно-ориентированное проектирование – это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы. Сравнивая определения объектно-ориентированного анализа и проектирования можно заметить, что обе

методологии используют анализ и абстрагирование в качестве основных методов. Следовательно, эти методологии направлены на одну и ту же сферу моделирования, поэтому можно утверждать, что имеются два разных подхода к моделированию в объектах.

Обобщая, можно сделать вывод, что объектно-ориентированный анализ направлен на описание модели в терминах объектов с внешней стороны. Объектно-ориентированное проектирование направлено на реализацию описанной в терминах объектов модели на языках программирования, т.е. на доработку модели, пока она не станет однозначно реализуемой. Можно сделать вывод, что без применения обоих методологий построить и реализовать объектную модель невозможно.

Рассмотрим содержательные аспекты объектно-ориентированного анализа и проектирования.

Впервые объектно-ориентированный анализ был успешно применен в 1979г в лаборатории Беркли в проекте моделирования физических систем реального времени. Шлеер и Меллор использовали динамические объектные модели. Для их воплощения они стали использовать анализ предметных областей с использованием понятия объектов, что привело к созданию информационных моделей. На этом этапе объектно-ориентированного анализа центральным является абстрагирование сущностей в задаче в терминах объектов и их свойств. Отношения между сущностями формализуются в связях. Объекты, их атрибуты, подобие и связи формализуется в виде графической диаграммы информационной модели. Заметим, что информационная модель по характеру описываемой информации подобна ER-диаграмме предметной области.

В начале 1985 г была создана вторая ступень объектного анализа – модель состояний. Информационная модель позволяла проанализировать отдельные сущности (объекты) и связи между ними, однако она не рассматривала взаимодействия и динамику существования объектов. С этим связан второй этап объектного анализа, анализирующий поведение объектов и их связи во времени. Поведение объекта может быть представлено как набор его состояний, в каждом из которых действует только определенный набор законов-правил. В силу законов состояния, с физическим объектом происходят различные явления, которые с точки зрения наблюдателя рассматриваются как некие действия объекта.

В 1986–1987гг. окончательно оформилась третья ступень объектного анализа – модели процессов. Действительно, модель состояний, рассматривая состояния, не анализировала заключенные в них процессы. С. Меллор и С. Шлеер взяли за основу описания процессов классические диаграммы потоков данных, используемые при функциональном проектировании программного обеспечения. Объектный анализ выделяет три основных элемента

информационного процесса как модели: действия, потоки данных между действиями и свойства объектов изменяемые действиями (архивы данных).

Объектно-ориентированное проектирование проходит все те же три ступени анализа. Отличия состоят в следующем: рассматриваются абстракции, не содержащиеся в исходной модели и имеющие целью упростить ее реализацию (шаблоны классов, метаклассы и др.), на фазе моделирования поведения системы применяются сценарии, как пошаговые описания реакции модели в конкретных состояниях. Усовершенствованная модель, полученная в результате объектно-ориентированного проектирования, может быть реализована на объектно-ориентированных языках программирования.

Таким образом, объектно-ориентированный анализ и проектирование органично дополняют друг друга.

В настоящее время стандартным средством объектно-ориентированного анализа и проектирования является язык унифицированного объектно-ориентированного моделирования UML. Особенности языка позволяют моделировать сложную систему с нескольких различных точек зрения, каждый раз принимая во внимание один аспект моделируемой системы и абстрагируясь от остальных [32], [33], [34]. Он также используется для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. Поэтому для проведения дальнейшего анализа и проектирования унифицированных средств описания исходных данных, алгоритмов и методов обработки и анализа будем использовать семантику и диаграммы языка UML.

Обычно результатами анализа системы являются наборы диаграмм объектов (чтобы выразить поведение системы через сценарии), диаграмм классов (чтобы выразить роли и обязанности агентов по поддержанию заданного поведения системы) и диаграммы состояний и переходов (чтобы показать упорядоченное событиями поведение этих агентов). Проектирование системы, в которое входит разработка ее архитектуры и реализации, порождает диаграммы классов, объектов, компонентов, а также динамические ракурсы этих диаграмм [30].

3.3 Диаграммы классов

Диаграмма классов показывает классы и их отношения, тем самым представляя логическую модель статического представления проекта. На стадии анализа диаграмма классов используется, чтобы выделить общие роли и обязанности сущностей, обеспечивающих требуемое поведение системы. На стадии проектирования диаграмма классов применяется, чтобы передать структуру классов, формирующих архитектуру системы [30].

3.3.1 Общая диаграмма классов

В результате анализа требований и задач обработки телеметрической информации, сформулированных в виде ключевых функций системы (подраздел 2.1) и совпадающих для различных подсистем обработки телеметрической информации, была построена диаграмма классов (рисунок 3.2), предоставляющих библиотеку унифицированного описания исходных данных, алгоритмов и методов обработки и анализа.

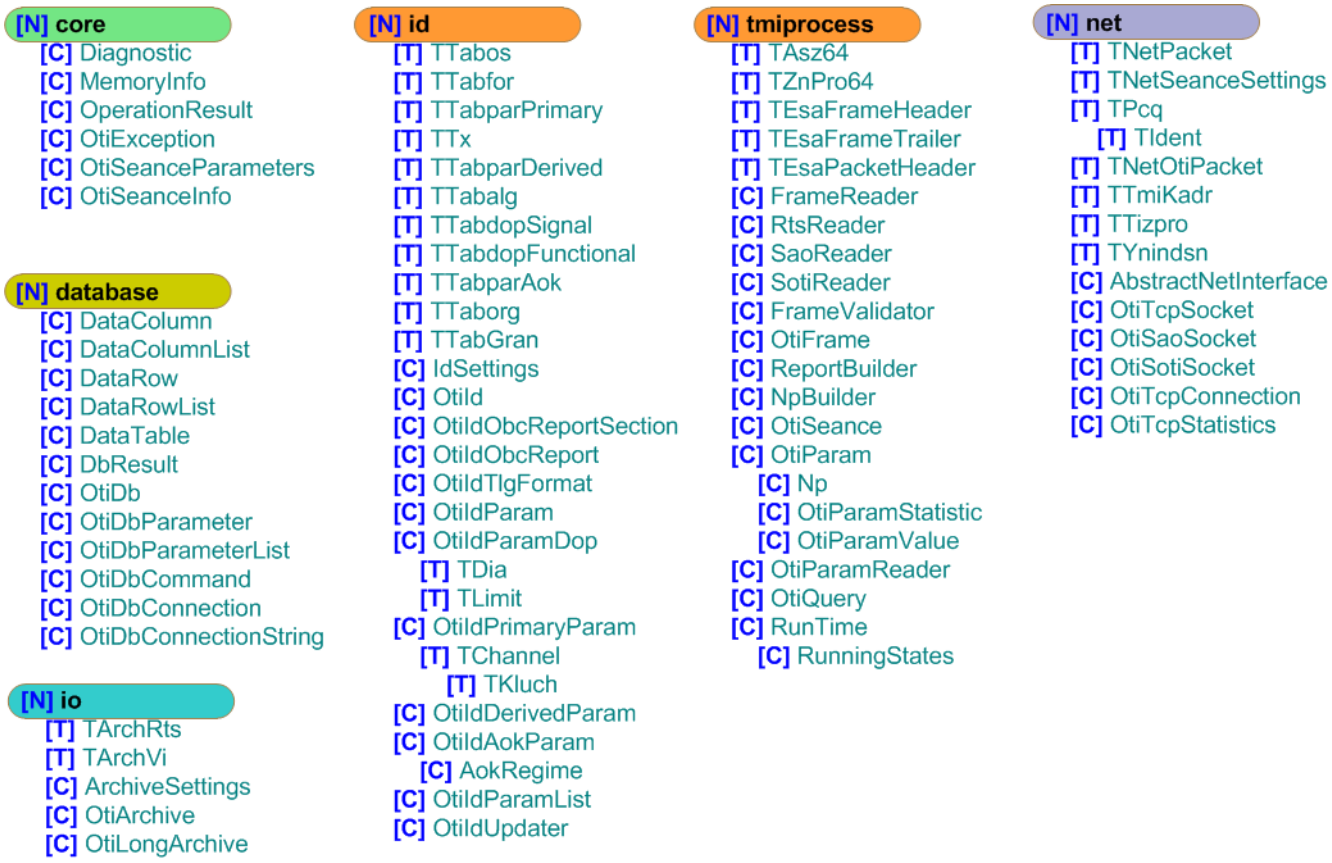

Рисунок 3.2 – Диаграмма библиотеки классов унифицированного описания исходных данных, обработки и анализа телеметрической информации

Множество выделенных классов имеет логическую группировку по характеру решаемых задач. Элементом группировки являются пространства имён, обозначенные символом N (namespace). Каждое пространство имён состоит из классов – C (class) и/или структур – T (struct). Ниже приведено функциональное назначение пространств имён, а также состав и краткое описание входящих в них классов и структур.

core

Пространство имён *core* содержит набор ключевых классов. К нему относятся:

- класс *MemoryInfo* вычисляет объём оперативной памяти, занимаемой приложением;
- класс *Diagnostic* содержит набор методов для протоколирования сообщений;

- класс *OperationResult* содержит результат выполнения некоторой операции;
- класс *OtiException* содержит данные об исключении;
- класс *OtiSeanceParams* содержит параметры сеанса связи с космическим аппаратом;
- класс *OtiSeanceInfo* содержит статистическую информацию о телеметрическом сеансе.

database

Пространство имён *database* содержит набор классов для взаимодействия с базой данных. К нему относятся:

- класс *DataColumn* представляет столбец таблицы;
- класс *DataColumnList* представляет коллекцию столбцов таблицы;
- класс *DataRow* представляет строку таблицы;
- класс *DataRowList* представляет коллекцию строк таблицы;
- класс *DataTable* представляет таблицу с результатами выполнения SQL запроса;
- класс *DbResult* содержит результат последнего выполненного SQL запроса;
- класс *OtiDb* содержит набор SQL методов для манипулирования данными в таблицах базы данных;
- класс *OtiDbParameter* содержит данные параметра, участвующего в SQL запросе;
- класс *OtiDbParameterList* содержит коллекцию параметров OtiDbParameter;
- класс *OtiDbCommand* содержит тело SQL запроса;
- класс *OtiDbConnection* содержит методы для управления физическим подключением к базе данных;
- класс *OtiDbConnectionString* содержит параметры подключения к базе данных.

io

Пространство имён *io* содержит набор классов для манипулирования архивами телеметрии. К нему относятся:

- класс *TArchRts* представляет единичную запись архива телеметрии;
- класс *TArchVi* представляет заголовок отчёта в архиве телеметрии;
- класс *ArchiveSettings* содержит параметры для работы с архивами телеметрии;
- класс *OtiArchive* содержит методы для чтения, записи, перемещения по архивам телеметрии;

- класс *OtiLongArchive* формирует единый телеметрический архив, объединяя данные из нескольких архивов.

id

Пространство имён *id* содержит набор классов для манипулирования исходными данными на обработку телеметрической информации. К нему относятся:

- структура *TTabos* содержит таблицу общих сведений;
- структура *TTabfor* содержит таблицу описания формуляров;
- структура *TTabparPrimary* содержит переменную часть первичного параметра;
- структура *TTx* содержит коэффициенты наклона и смещения тарировочной характеристики;
- структура *TTabparDerived* содержит описание переменной части вторичного параметра;
- структура *TTabalg* содержит таблицу алгоритмов сглаживания параметров;
- структура *TTabdopSignal* содержит дополнительное описание сигнального параметра;
- структура *TTabdopFunctional* содержит дополнительное описание функционального параметра;
- структура *TTabparAok* содержит переменную часть параметра АОК;
- структура *TTaborg* содержит таблицу описания режимов;
- структура *TTabGran* содержит границы функционирования параметра;
- структура *IdSettings* содержит параметры работы с исходными данными;
- класс *OtiId* содержит исходные данные по конкретному космическому аппарату;
- класс *OtiIdObcReportSection* содержит описание участка отчёта бортового компьютера;
- класс *OtiIdObcReport* содержит описание отчёта бортового компьютера;
- класс *OtiIdTlgFormat* содержит описание формата съёма телеграммы;
- класс *OtIdParam* является базовым классом для описания исходных данных отдельного параметра;
- класс *OtiIdParamDop* предоставляет дополнительное описание для первичных и вторичных параметров;
- структура *TDia* содержит описание диапазонов параметра;
- структура *TLimit* содержит расширенное описание границ функционирования параметра;

- класс *OtiIdPrimaryParam* содержит исходные данные первичного параметра;
    - класс *TChannel* содержит описание расположения параметра на различных байтах внутри телеметрического кадра;
        - ❖ класс *TKluch* содержит вторичный адрес и массив значений ключевого параметра;
- класс *OtiIdDerivedParam* содержит исходные данные вторичного параметра;
- класс *OtiIdAokParam* содержит исходные данные параметра АОК;
- класс *AokRegime* содержит описание режима функционирования параметра АОК;
- класс *OtiIdParamList* содержит коллекцию исходных данных параметров;
- класс *OtiIdUpdater* производит проверку обновлений исходных данных и их загрузку.

tmiprocess

Пространство имён *tmiprocess* содержит набор классов для обработки телеметрических кадров, расчёта значений телеметрических параметров и формирования отчётов бортового компьютера. К нему относятся:

- структура *TAsz64* содержит двоичное значение телеметрического параметра;
- структура *TZnPro64* содержит значение телеметрического параметра, используемое в предварительной обработке по дельте существенности и при усреднении;
- структура *TEsaFrameHeader* содержит заголовок телеметрического кадра стандарта пакетной телеметрии ESA PSS-04-106;
- структура *TEsaFrameTrailer* содержит концевик телеметрического кадра стандарта пакетной телеметрии ESA PSS-04-106;
- структура *TEsaPacketHeader* содержит заголовок пакета прикладного уровня стандарта пакетной телеметрии ESA PSS-04-106;
- класс *FrameReader* является базовым классом для чтения телеметрии из потока байт. Обрабатывает поток, выделяет и формирует кадр(ы) в структуре io::TAchRts;
- класс *RtsReader* обеспечивает чтение телеметрии из потока байт в структуре TAchRts;
- класс *SaoReader* обеспечивает чтение телеметрии из потока байт в структуре САО – СПО;
- класс *SotiReader* обеспечивает чтение телеметрии из потока байт в структуре СОТИ – СПО;
- класс *FrameValidator* производит расчёт достоверности телеметрического кадра и запись соответствующих признаков достоврености;

- класс *OtiFrame* является базовым классом для обработчиков псевдокадра;
- класс *ReportBuilder* предназначен для выделения и формирования отчётов бортового компьютера при чтении из телеметрического потока;
- класс *NpBuilder* формирует таблицу непосредственной передачи;
- класс *OtiSeance* обеспечивает управление состоянием телеметрического сеанса;
- класс *OtiParam* содержит методы расчёта значений телемерического параметра и множество расчитанных значений этого параметра;
    - класс *Np* содержит значение для предварительной обработки телеметрического параметра;
    - класс *OtiParamStatistic* содержит статистическую информацию о параметре: минимум, максимум, среднеквадратичное отклонение;
    - класс *OtiParamValue* содержит сведения о единичном значении телеметрического параметра;
- класс *OtiParamReader* обеспечивает чтение телеметрических параметров из архива;
- класс *OtiQuery* содрежит данные телеметрического запроса/ответа от удалённого абонента: перечень телеметрических параметров, перечень отчётов бортового компьютера, команды управления конфигурацией полезной нагрузки;
- класс *RunTime* предназначен для расчёта наработки бортовой аппаратуры;
    - класс *RunningStates* содержит статистику по работе прибора в состояниях откл/вкл.

net

Пространство имён *net* содержит набор классов для чтения информации из сети. К нему относятся:

- структура *TNetPacket* содержит заголовок сетевого пакета. Протокол СПО – САО;
- структура *TNetSeanceSettings* содержит параметры проводимого сеанса. Протокол СПО – САО;
- структура *TPcq* содержит заголовок сетевого пакета. Протокол СОТИ – СПО;
    - структура TIdent содержит идентификатор работы для TPcq;
- структура *TNetOtiPacket* содержит заголовок пакета прикладного уровня, передаваемый между СОТМ – РМТМ;
- структура *TTmiKadr* содержит записи файла «tmi_kadr». Протокол СОТИ – СПО;

- структура *TTizpro* содержит результаты предварительной обработки датчиковой телеметрии при приёме информации с разгонного блока. Протокол СОТИ – СПО;
- структура *TYnindsn* содержит таблицу соответствия условного номера параметра его индексу и сквозному номеру. Протокол СПО – СОТИ;
- класс *AbstractNetInterface* представляет абстракнтый интерфейс для обработки принятого из сети телеметрического кадра;
- класс *OtiTcpSocket* является базовым для всех сетевых сокетов и содержит методы управления подключением к серверу, списком проводимых сеансов;
- класс *OtiSaoSocket* является базовым для сетевых сокетов, обеспечивающих работу по протоколу САО – СПО;
- класс *OtiSotiSocket* является базовым для сетевых сокетов, обеспечивающих работу по протоколу СОТИ – СПО;
- класс *OtiTcpConnection* содержит параметры сетевого подключения;
- класс *OtiTcpStatistics* содержит количество принятых и переданных байт по сети.

3.3.2 Диаграмма классов описания исходных данных

Важным аспектом, обеспечивающим автоматизированную обработку телеметрической информации, является использование унифицированных средств загрузки и описания исходных данных. Группа классов, предназначенных для решения данных задач, представлена на соответствующей диаграмме классов на рисунке 3.3.

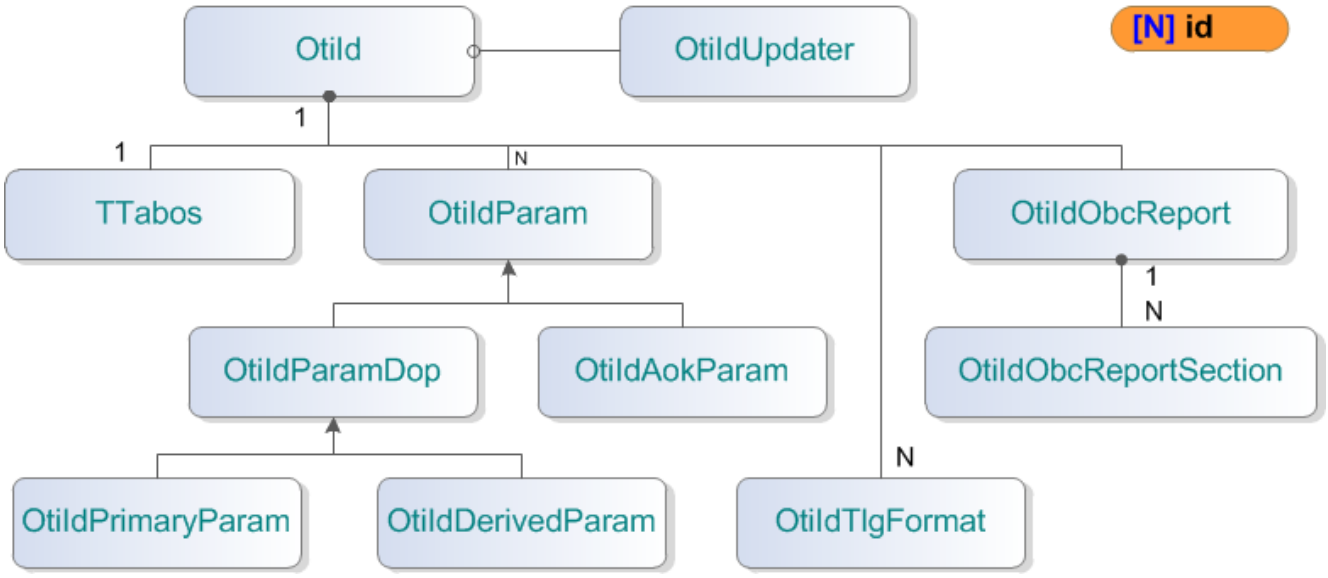

Рисунок 3.3 – Диаграмма классов описания исходных данных

Из диаграммы видно, что класс исходных данных по космическому аппарату *OtiId* взаимодействует с классом проверки и загрузки обновлений исходных данных *OtiIdUpdater*.

Класс *OtiId* является контейнером для хранения различных видов телеметрических объектов: *TTabos* – общие сведения о космическом аппарате, *OtiIdParam* – описания телеметрических параметров, *OtiIdTlgFormat* – форматы для съёма телеграмм, *OtiIdObcReport* – описания отчётов бортового компьютера, в свою очередь содержащие несколько экземпляров класса *OtiIdObcReportSection*.

Мощность отношения между экземпляром класса *OtiId* и экземплярами каждого из классов *OtiIdParam, OtiIdTlgFormat, OtiIdObcReport* – 1:N. Кроме того *OtiId* содержит единственный экземпляр класса общих сведений о космическом аппарате *TTabos*.

Особое внимание уделено описанию исходных данных телеметрических параметров. Так, при обработке используются три вида параметров: первичные, вторичные и параметры АОК.

Первичный параметр – это параметр, формируемый на борту спутника и передаваемый на землю по радиолинии в составе телеметрического кадра.

Вторичный параметр – это параметр, формируемый программно на земле, на основании исходных данных. Представляет собой математическо-логическое выражение, состоящее из первичных, вторичных параметров и операндов, объединяющих между собой параметры выражения.

Параметр АОК – также формируется программно на земле на основании исходных данных. Значение параметра АОК есть некоторый режим функционирования блока, подсистемы или системы спутника, причём один АОК-параметр может принимать разные режимы. Исходные данные каждого параметра АОК содержат следующие сведения: имена допустимых режимов и перечень параметров с указанием граничных условий, однозначно определяющих каждый отдельный режим. Для точного определения нормального / аномального состояния режима параметры режима разделяют на *определяющие* и *сопутствующие* (используются опционально).

Для описания сведений о телеметрических параметрах используется принцип наследования. Каждый из параметров имеет ряд общих характеристик, поэтому базовым классом для параметра каждого типа является *OtiIdParam*. От него наследуются классы *OtiIdParamDop* (дополнительное описание первичного и вторичного параметров) и *OtiIdAokParam*. Класс *OtiIdParamDop* в свою очередь является базовым для класса описания первичного *OtiIdPrimaryParam* и вторичного параметров *OtiIdDerivedParam*.

Множество классов описания исходных данных принадлежит пространству имён ***id***.

3.3.3 Диаграмма классов обработки и анализа

Для обеспечения многопоточного приёма телеметрической информации необходимо уметь организовывать и поддерживать множество одновременных сеансов приёма и обработки информации. В этом случае в качестве объекта управления и сопровождения единичного телеметрического сеанса целесообразно определить класс сеанса *OtiSeance*, структура которого приведена на рисунке 3.4.

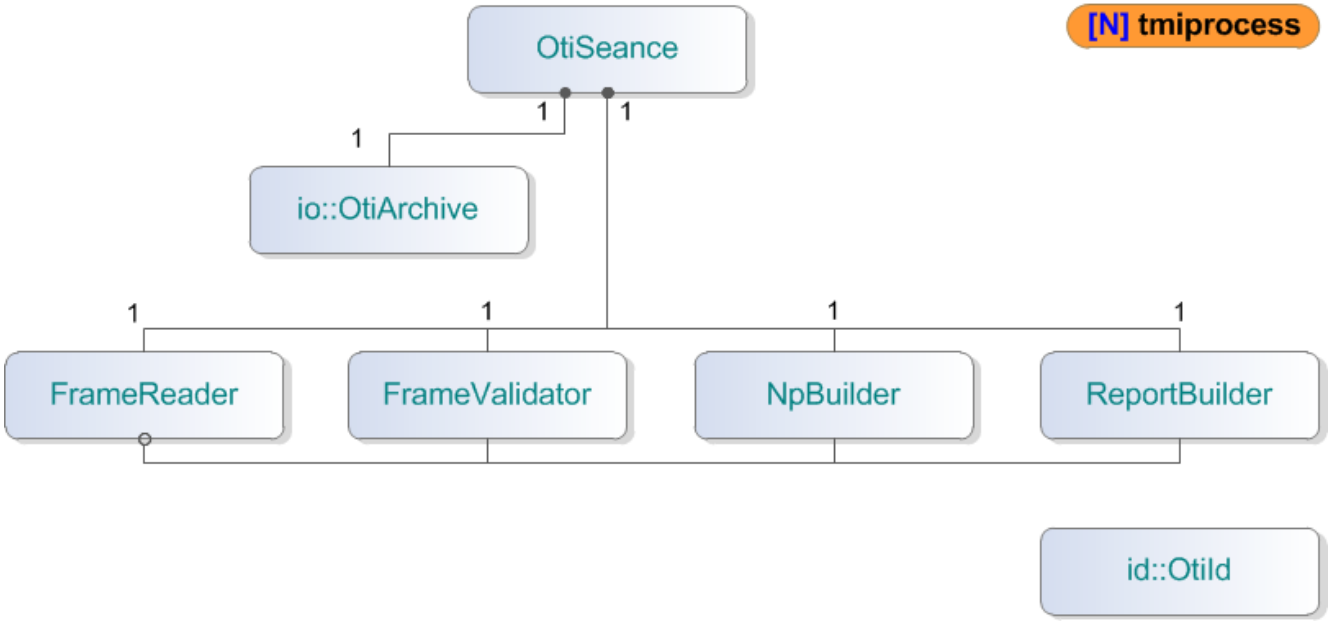

Рисунок 3.4 – Диаграмма классов для сеанса приёма и обработки телеметрической информации

Каждый телеметрический кадр, принимаемый от некоторого источника классом управления состоянием телеметрического сеанса *OtiSeance*, передаётся для предварительной обработки в класс чтения телеметрии из потока байт *FrameReader*, а точнее в один из его наследников *RtsReader, SaoReader, SotiReader*, рассмотренных на рисунке 3.5, который приводит кадр к унифицированному формату *io::TArchRts*.

Контроль достоверности и запись соответствующих признаков для кадра унифицированного формата осуществляет класс расчёта достоверности телеметрического кадра *FrameValidator*. В дальнейшую обработку передаются только достоверные кадры.

В свою очередь класс *NpBuilder* использует результаты расчёта достоверности, обращаясь к *FrameReader*, если кадр не сбойный производит вычисление значений телеметрических параметров, присутствующих в принятом кадре, и формирует таблицу непосредственной передачи, используемую при анализе текущего состояния космического аппарата.

Класс *ReportBuilder* также обращается к *FrameReader* за очередным достоверным кадром, анализирует наличие признаков поступления отчётов бортового компьютера,

производит извлечение и накопление отдельных порций с последующим формированием массива отчётной информации.

Каждый из рассмотренных классов *FrameValidator*, *NpBuilder*, *ReportBuilder* принадлежит пространству имён *tmiprocess* и использует при вычислениях экземпляр класса исходных данных по космическому аппарату *id::OtiId*.

Кроме того, класс *OtiSeance* содержит экземпляр класса *io::OtiArchive*, предоставляющего методы манипулирования архивами телеметрической информации: чтение, запись кадров и отчётов бортового компьютера, перемещение по архивам, что позволяет сохранять информацию по каждому проводимому сеансу связи отдельно.

Иерархия классов, наследуемых от класса чтения телеметрии из потока байт *FrameReader*, представлена на рисунке 3.5.

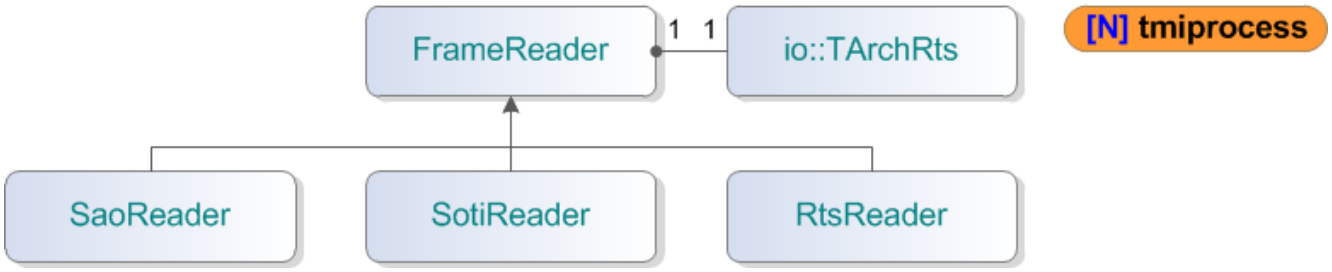

Рисунок 3.5 – Диаграмма классов чтения телеметрии из потока байт

Класс FrameReader определяет общий интерфейс для чтения телеметрических кадров и их преобразования к унифицированному формату *io::TArchRts*. Такой способ проектирования позволяет создавать на основе FrameReader новые классы, способные принимать телеметрию от различных источников, предоставляющих телеметрию в различных форматах. Так, класс *SaoReader* обеспечивает чтение телеметрии из потока байт в структуре САО-Ц, класс *SotiReader* из потока байт в структуре СОТИ, включая потоки от ЕЦУП РБ, а класс *RtsReader* читает телеметические кадры из потока байт в структуре *io::TAchRts*.

Ниже на рисунке 3.6 представлена иерархия классов, наследуемых от базового класса обработчика псевдокадра *OtiFrame*.

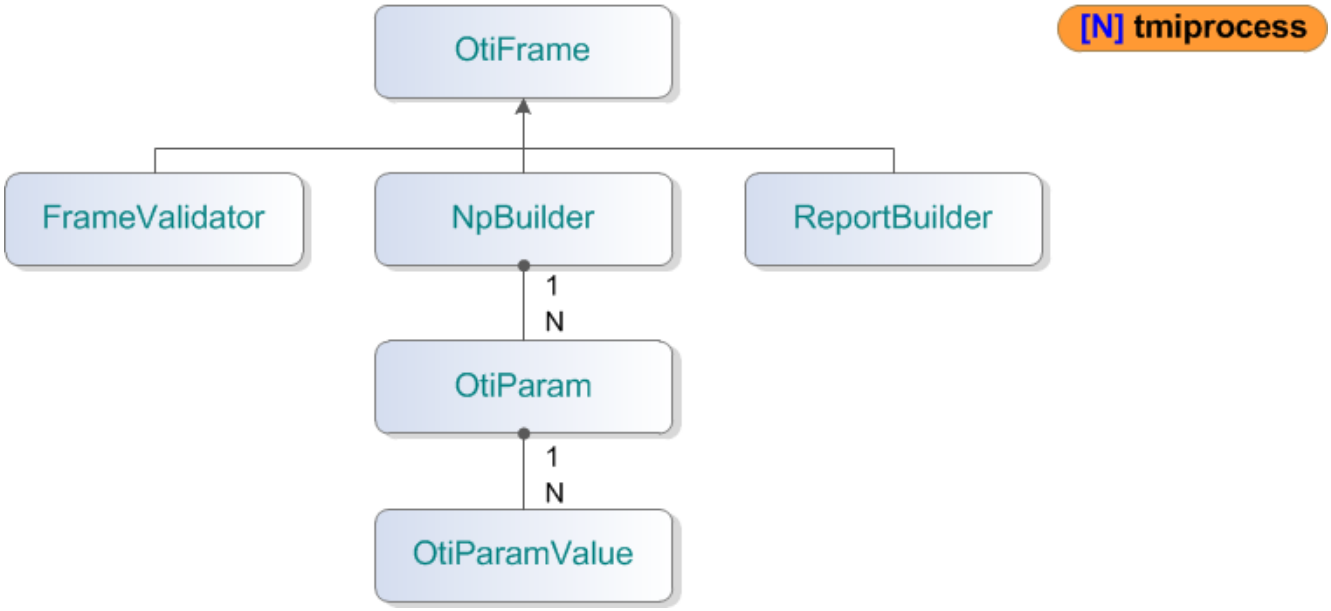

Рисунок 3.6 – Диаграмма обработчиков телеметрического кадра

Как видно из диаграммы от класса *OtiFrame* наследуются уже рассмотренные выше класс контроля достоверности телеметрических кадров *FrameValidator*, класс формирования таблицы непосредственной передачи *NpBuilder* и класс выделения и формирования отчётов бортового компьютера *ReportBuilder*.

В свою очередь класс NpBuilder содержит N экземпляров класса *OtiParam*. Количество экземпляров N класса *OtiParam* определяется исходными данными на обработку космического аппарата и может различаться для разных космических аппаратов. Класс OtiParam содержит методы расчёта и преобразования двоичных значений телеметрического параметра в физические единицы и хранит множество рассчитанных значений этого параметра в виде N экземпляров класса OtiParamValue.

Совокупность классов *FrameValidator*, *NpBuilder*, *ReportBuilder* и *OtiParam* пространства имён *tmiprocess* предоставляет полноценный набор унифицированных алгоритмов и методов обработки и анализа телеметрической информации.

3.4 Диаграммы объектов

Диаграмма объектов показывает существующие объекты и их связи в логическом проекте системы и представляет собой мгновенный снимок потока событий в некоторой конфигурации объектов. При анализе диаграммы объектов используется для показа семантики основных и второстепенных сценариев, которые отслеживают поведение системы. При проектировании диаграммы объектов используются для иллюстрации семантики механизмов в логическом проектировании системы [30], [35].

На рисунке 3.7 представлена диаграмма объектов, участвующих в загрузке исходных данных на обработку телеметричекой информации. Сценарий начинается с вызова объектом *Сеанс (OtiSeance)* операции *загрузить (load(nKA))* из объекта *Исходные_данные (OtiId)*. Далее объект *Исходные_данные (OtiId)* выполняет проверку наличия обновлений в базе данных, используя вызов метода *проверить (check)* объекта *Поиск/загрузка_обновлений_ИД (OtiIdUpdater)*. В случае обнаружения на сервере БД обновлённой версии ИД для конкретного КА осуществляется их загрузка в локальный кэш при помощи операции *обновить (update)* объекта *Поиск/загрузка_обновлений_ИД (OtiIdUpdater)*. Затем объект *Исходные_данные (OtiId)*, всегда обращаясь к актуальной версии ИД, готовит структуры данных для дальнейшего использования, вызывая соответствующие методы загрузки *таблицы общих сведений (prepareCommon), таблицы параметров (prepareParams), таблицы формуляров (prepareForms), таблицы отчётов БЦВК (prepareObcReports)*. В результате выполнения сценария объект *Сеанс (OtiSeance)* получает набор структур ИД, загруженных в память и готовых для использования.

Рисунок 3.7 – Диаграмма объектов загрузки исходных данных

На рисунке 3.8 представлена диаграмма объектов, участвующих в обработке и анализе телеметрической информации. Цель диаграммы – проиллюстрировать сценарий выполнения задач обработки и анализа телеметрической информации.

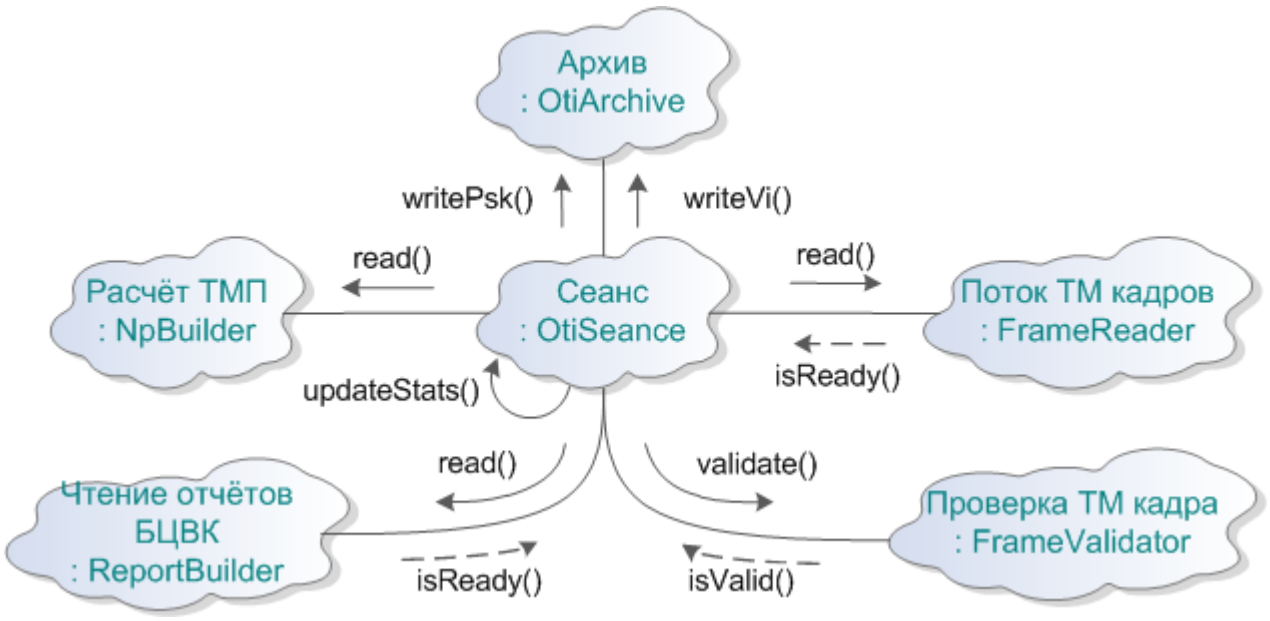

Рисунок 3.8 – Диаграмма объектов обработки и анализа

Сценарий начинается с вызова центральным объектом *Сеанс (OtiSeance)* операции *читать_ТМ_кадр (read)* из объекта *Поток_ТМ_кадров (FrameReader)*, который возвращает результат чтения через метод *готовности_ТМ_кадра (isReady)*. Если данных для формирования телеметрического кадра недостаточно, то есть телеметрический кадр не готов, сценарий завершается. В противном случае, происходит вызов метода *оценить_достоверность (validate)* из объекта *Проверки_ТМ_кадра (FrameValidator)*, а результат оценки достоверности возвращается операцией *достоверность_ТМ_кадра (isValid).* Далее происходит обновление статистики по сеансу, а именно: обновление количества сбойных и достоверных телеметрических кадров посредством вызова метода *обновления_статистики (updateStats)* объекта *Сеанс (OtiSeance).* После чего производится сохранение на жёсткий диск принятого и прошедшего валидацию телеметрического кадра вызовом метода *сохранить_псевдокадр (writePsk)* объекта *Архив (OtiArchive),* причём, если телеметрический кадр не достоверен – сценарий завершается.

На следующем шаге выполнения сценария происходит расчёт значений телеметрических параметров из принятого телеметрического кадра, а именно обращение к методу *чтения (read)* из объекта *Расчёт_ТМП (NpBuilder).* Далее осуществляется попытка чтения очередной порции отчёта БЦВК из принятого телеметрического кадра посредством вызова метода *чтения (read)* из объекта *Чтение_отчётов_БЦВК (ReportBuilder)*. Для определения готовности порции отчёта БЦВК объект *Сеанс (OtiSeance)* обращается к методу *готовности (isReady)* объекта *Чтение_отчётов_БЦВК (ReportBuilder).* Если отчёт БЦВК не сформирован – сценарий завершается. Иначе происходит обновление количества принятых сбойных / достоверных

отчётов БЦВК по сеансу посредством вызова метода *обновления_статистики (updateStats)* объекта *Сеанс (OtiSeance)*. После чего вне зависимости от достоверности принятого отчёта БЦВК производится его сохранение на жёсткий диск посредством вызова метода *сохранить_отчёт_БЦВК (writeVi)* объекта *Архив (OtiArchive)*.

3.5 Диаграммы взаимодействия

Диаграмма взаимодействия используется, чтобы описать поведение взаимодействующих групп объектов во времени, тем самым предоставляя логическую модель системы в динамике [30], [36].

Диаграмма взаимодействия не вводит новых понятий или обозначений. Они использует существенные элементы диаграммы объектов и перестраивает их. На диаграмме взаимодействий имена объектов записываются горизонтально в верхней ее строке. Под каждым из них рисуется вертикальная пунктирная линия. Отправления сообщений показываются горизонтальными стрелками, с тем же синтаксисом и обозначениями синхронизации, что и на диаграмме объектов. Линия, обозначающая посылку сообщения, проводится от вертикали клиента к вертикали сервера [35].

Диаграммы взаимодействий часто лучше диаграмм объектов передают семантику сценариев на ранних этапах жизненного цикла разработки, когда еще не идентифицированы протоколы отдельных классов.

Взаимодействие объектов описания исходных данных во времени представлено на рисунке 3.9. Подробное описание взаимодействующих объектов приведено в 3.4.

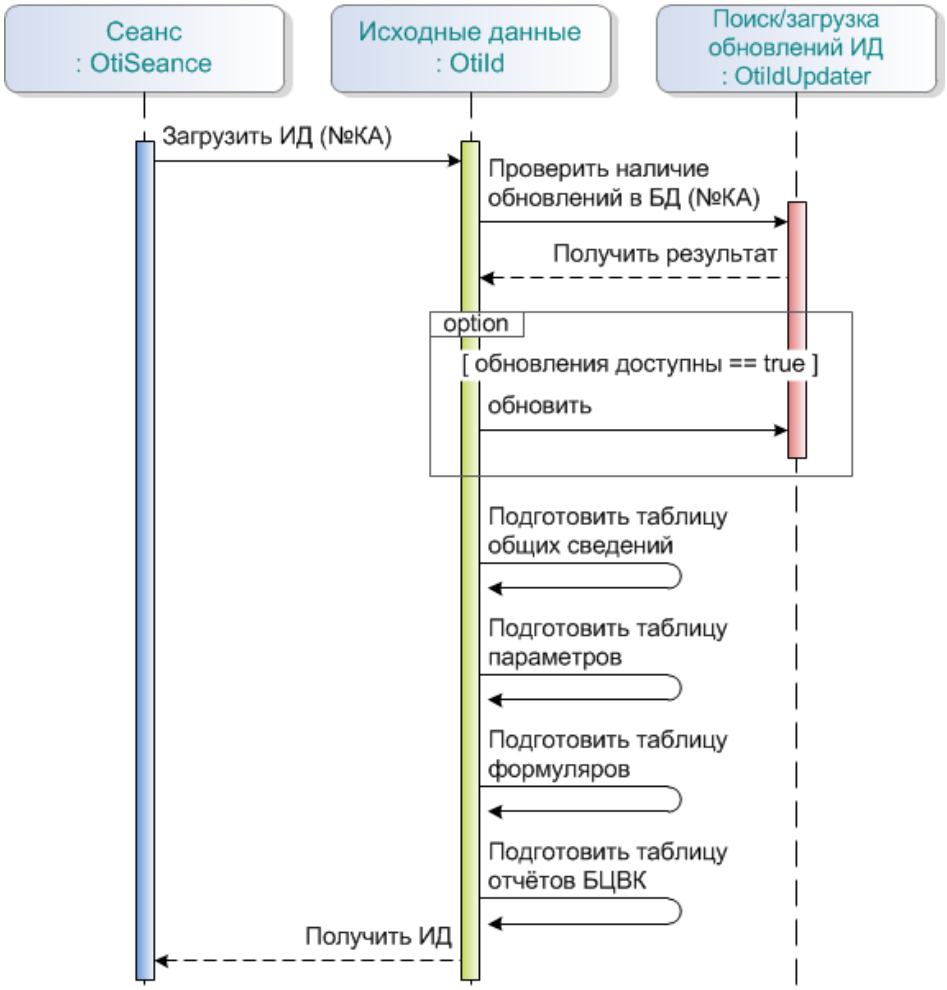

Рисунок 3.9 – Диаграмма взаимодействия объектов описания исходных данных

Взаимодействие объектов обработки и анализа телеметрической информации во времени подробно описанное в 3.4 представлено на рисунке 3.9.

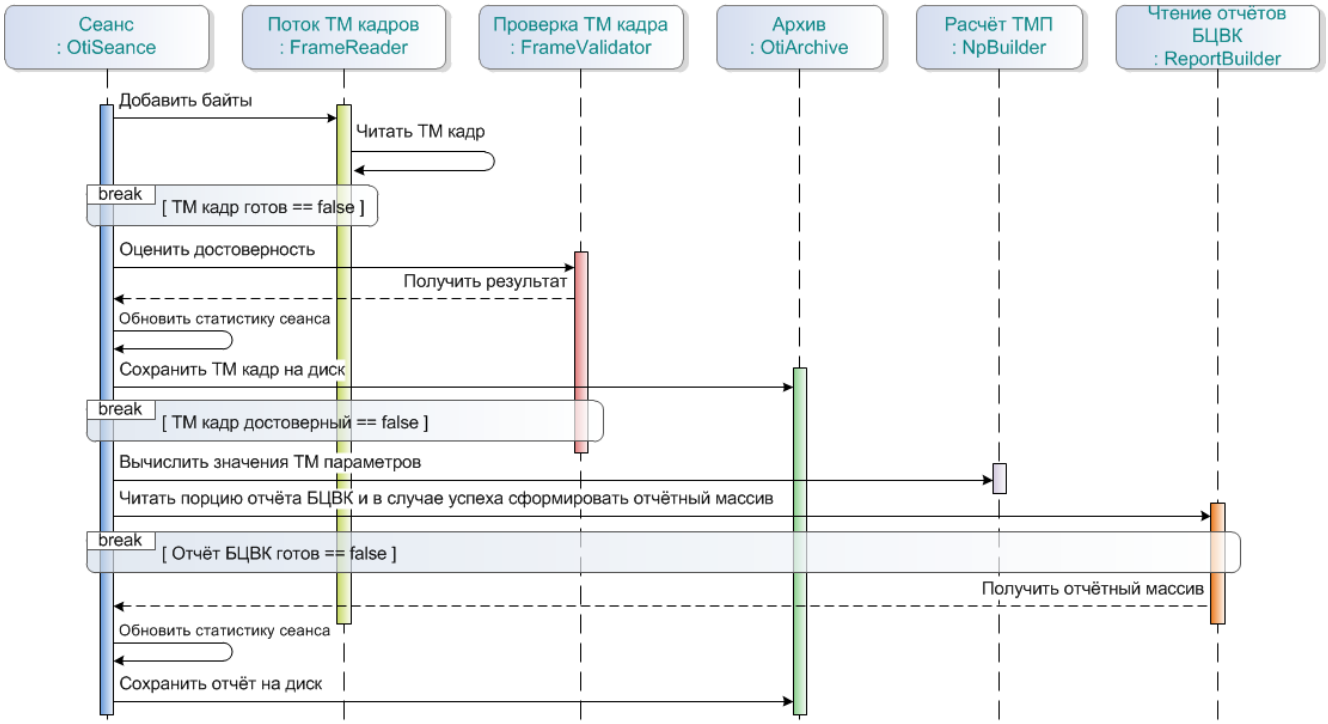

Рисунок 3.10 – Диаграмма взаимодействия объектов обработки и анализа

Очевидное преимущество построенных диаграмм взаимодействий (рисунки 3.9, 3.10) по сравнению с аналогичными диаграммами объектов (рисунки 3.7, 3.8), которые также отражают динамическое поведение системы, заключается в том, что на них легче читается порядок посылки сообщений.

3.6 Диаграмма компонентов

Диаграмма компонентов демонстрирует разбиение системы на структурные компоненты, скрывающие свою реализацию и взаимодействующие друг с другом через интерфейсы. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты [30], [36]. При разработке диаграмма компонентов используется для иллюстрации физического деления архитектуры по слоям и разделам и в отличие от построенных ранее диаграмм описывает особенности физического представления системы. Компонентное представление системы часто называют ядром её архитектуры [37], что подчёркивает её особую значимость. Основными графическими элементами диаграммы компонентов являются компоненты и зависимости между ними.

При построении диаграммы компонентов будем использовать следующую терминологию:

- модуль – законченная программная единица, реализуемая в виде подключаемой библиотеки и предоставляющая набор классов и методов, схожих по функциональному назначению;
- пространство имён – логически сгруппированное именованное множество классов и методов;
- уровень приложений – уровень выполнения прикладных задач;
- уровень библиотек обработки и анализа – совокупность модулей, представляющих средства унифицированного описания исходных данных, реализации алгоритмов и методов обработки и анализа.

Построенная диаграмма компонентов для библиотеки унифицированного описания исходных данных, алгоритмов и методов обработки и анализа представлена на рисунке 3.11.

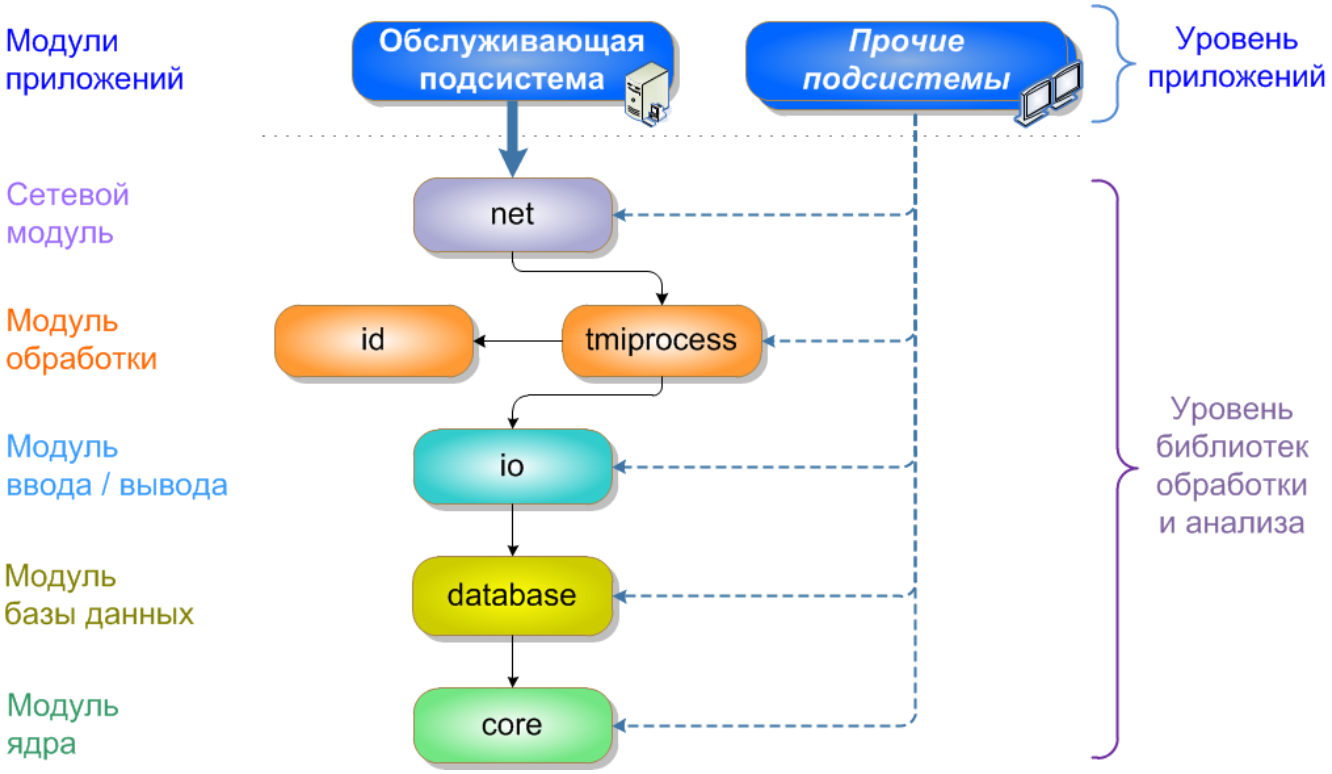

Рисунок 3.11 – Диаграмма компонентов

Как видно из рисунка, построенная диаграмма компонентов представлена в виде иерархии, причём модули нижних уровней предоставляют интерфейсы для модулей верхних уровней.

Модуль ядра, в состав которого входят классы пространства имён *core*, предоставляет интерфейсы ведения журналов протоколирования, ключевые структуры и типы данных, используемые всеми вышележащими модулями.

Модуль базы данных, состоящий из классов пространства имён *database,* использует интерфейсы модуля ядра и содержит прикладные интерфейсы для установки подключения и взаимодействия с базой данных на языке SQL.

Модуль ввода / вывода состоит из классов пространства имён *io*, использует интерфейсы модулей *database* и *core* и реализует интерфейсы для управления архивами телеметрической информации.

В состав модуля обработки входят классы двух пространств имён: *id* и *tmiprocess.* Он использует интерфейсы модулей нижних уровней и предоставляет важнейшие интерфейсы для манипулирования исходными данными на обработку телеметрической информации, обработки телеметрических кадров, расчёта значений телеметрических параметров и формирования отчётов бортового компьютера.

Сетевой модуль, состоящий из классов пространства имён *net*, использует интерфейсы всех нижележащих модулей и предоставляет интерфейсы для организации сетевых подключений, приёма и передачи телеметрической информации от некоторого источника по прикладным протоколам СПО – САО, СПО – СОТИ, реализованных на основе протокола TCP/IP.

Совокупность модулей ядра, базы данных, ввода/вывода, модуля обработки и сетевого модуля относится к уровню *библиотек обработки и анализа* и является унифицированным базисом для решения задач обработки телеметрии. Реализация рассматриваемых модулей в виде подключаемых библиотек с одной стороны позволяет производить независимую и параллельную разработку, тестирование и отладку этих модулей, а с другой – делает возможным решение прикладных телеметрических задач различного рода, относящихся к *уровню приложений* и использующих предоставленные унифицированные интерфейсы обработки и анализа.

Так, например, обслуживающая подсистема, расположенная на уровне приложений, использует все модули библиотеки обработки и анализа. А вспомогательная подсистема внесеансного мониторинга, которая использует архивы телеметрической информации расположенные в БД или на жёстком диске и не получает потоки телеметрической информации по сети в реальном масштабе времени от удалённых источников, может не использовать сетевой модуль. Другим примером является подсистема подготовки исходных данных, обеспечивающая подготовку исходных данных для обработки телеметрической информации (таблицы общих сведений, описания параметров, формуляров и т.д.), которая также не использует сетевой модуль.

3.7 Выводы по главе 3

1 Рассмотрена существующая схема взаимодействия компонентов специального программного обеспечения обработки телеметрической информации

2 Произведено обоснование выбора метода проектирования унифицированных средств описания исходных данных, алгоритмов и методов обработки и анализа

3 В результате проектирования унифицированных средств описания исходных данных, алгоритмов и методов обработки и анализа, в которое входит разработка их архитектуры, были построены диаграммы классов, объектов, компонентов, а также динамические ракурсы этих диаграмм. Между этими диаграммами существует сквозная связь, позволяющая проследить требования от реализации обратно к спецификации. Так, диаграмма компонентов содержит наборы классов и объектов, определения которых находятся на соответствующих диаграммах классов или объектов. А определения отдельных классов указывают на исходные требования

4 Совокупность построенных диаграмм представляет библиотеку классов унифицированного описания исходных данных, обработки и анализа телеметрической информации

Глава 4. Обеспечение качества программной модели автоматизированной системы приёма, обработки и анализа телеметрической информации

Качество программной модели является залогом успешной реализации проекта любой сложности. Однако само понятие качества имеет множество формулировок, оттенков и по-разному воспринимается участниками проектно-производственного процесса. В настоящей главе даётся определение качества программного обеспечения. Производится обоснование применения V-модели жизненного цикла программного обеспечения к созданию многопоточной системы. Такая модель учитывает, что тестирование, как один из способов обеспечения качества продукта, обсуждается, проектируется и планируется на ранних этапах жизненного цикла разработки. Прежде чем перейти к вопросам обеспечения качества рассматривается распространенная многоуровневая модель качества программного обеспечения, представленная в наборе стандартов ISO 9126. Далее формулируются собственные принципы обеспечения качества на основных этапах жизненного цикла разработки. Однако одного обеспечения качества не достаточно, необходим обратный процесс – контроль качества, позволяющий оценить, на сколько система соответствует заявленным требованиям, каков процент завершённых задач. В зависимости от того, обеспечиваются ли заданные характеристики или имеют место недопустимые отклонения от них, управляющие воздействия должны быть направлены соответственно на сохранение фактического состояния процесса или на его корректировку.

4.1 Качество программного обеспечения

При работе над проектом любой сложности, тем более при создании программного обеспечения (ПО), особую роль занимает вопрос обеспечения качества создаваемого продукта [38]. Однако само понятие «качества», а также «качество ПО» является достаточно абстрактным и может иметь различную трактовку у проектировщика, тестировщика или программиста [39]. Рассмотрим определение «качества ПО» в контексте международных стандартов:

- 1061-1998 IEEE Standard for Software Quality Metrics Methodology. Качество программного обеспечения – это степень, в которой ПО обладает требуемой комбинацией свойств.

- ГОСТ 28806–90 «Качество программных средств. Термины и определения». Качество программного средства – совокупность свойств программного средства, которые обусловливают его пригодность удовлетворять заданные или подразумеваемые потребности в соответствии с его назначением.

Обобщая приведённые понятия можно утверждать, что *качество программного обеспечения* (Software quality) — это то, насколько программное обеспечение удовлетворяет предъявляемым к нему требованиям. Выдвигаемые требования могут зависеть от многих критериев, определяемых исходя из сферы применения программного продукта.

Существует набор стандартов ISO 9000, регулирующий общие принципы обеспечения качества во всех отраслях [39]. Наиболее важными стандартами в разработке программного обеспечения являются:

- ISO 9000:2000 Quality management systems — Fundamentals and vocabulary;
- ISO 9001:2000 Quality management systems — Requirements. Models for quality assurance in design, development, production, installation, and servicing;
- ISO 9004:2000 Quality management systems — Guidelines for performance improvements;
- ISO/IEC 9003:2004 Software engineering — Guidelines for the application of ISO 9001:2000 to computer software.

Кроме того в каждой компании могут быть разработаны свои стандарты качества программного обеспечения, отвечающие конкретной специфики работы и соответственно ее требованиям. Можно выделить несколько основных критериев оценки качества программного обеспечения:

1 Качество исходного кода.
   - соответствие кода стандартам;
   - легкость поддержки;
   - малое число предупреждений при компиляции.
2 Качество программного продукта.
   - функциональность;
   - надежность;
   - удобство использования;
   - эффективность;
   - удобство сопровождения
   - портативность.

Становится понятно, что предъявляемые требования должны удовлетворять потребностям, как разработчиков программного обеспечения, так и его пользователей.

4.2 V-модель жизненного цикла программного обеспечения

Среди множества разнообразных моделей жизненного цикла программного обеспечения наиболее удачной в смысле обеспечения надёжности и качества создаваемого продукта для задачи диссертационного проектирования была выбрана V-образная модель [40]. Такая модель жизненного цикла используется с целью помочь работающей над проектом команде в планировании работ по созданию и дальнейшему тестированию системы. В этой модели особое значение придается действиям, направленным на верификацию и аттестацию продукта. Она демонстрирует, что тестирование продукта обсуждается, проектируется и планируется на ранних этапах жизненного цикла разработки [40].

План испытания приемки заказчиком разрабатывается на этапе планирования, а компоновочного испытания системы - на фазах анализа, разработки проекта и т.д. Этот процесс разработки планов испытания обозначен пунктирной линией между прямоугольниками V-образной модели.

V-образная модель по своей сути является разновидностью каскадной модели и поэтому наследует от неё такую же последовательную структуру. Каждая последующая фаза начинается по завершению получения результативных данных предыдущей фазы.

Модель демонстрирует комплексный подход к определению фаз процесса разработки программного обеспечения (рисунок 4.1). В ней подчеркнуты взаимосвязи, существующие между аналитическими фазами и фазами проектирования, которые предшествуют кодированию, после которого следуют фазы тестирования. Пунктирные линии означают, что эти фазы необходимо рассматривать параллельно.

Рисунок 4.1 – V-модель жизненного цикла программного обеспечения

Ниже дано краткое описание каждой фазы V-образной модели, начиная от планирования проекта и требований вплоть до приемочных испытаний [40]:

- планирование проекта и требований – определяются системные требования, а также то, каким образом будут распределены ресурсы организации с целью их соответствия поставленным требованиям (в случае необходимости на этой фазе выполняется определение функций для аппаратного и программного обеспечения);
- анализ требований к продукту и его спецификации – анализ существующей на данный момент проблемы с программного обеспечения, завершается полной спецификацией ожидаемой внешней линии поведения создаваемой программной системы;
- архитектура или проектирование на высшем уровне – определяет, каким образом функции программного обеспечения должны применяться при реализации проекта;
- детализированная разработка проекта – определяет и документально обосновывает алгоритмы для каждого компонента, который был определен на фазе построения архитектуры. Эти алгоритмы в последствии будут преобразованы в код;

- разработка программного кода – выполняется преобразование алгоритмов, определенных на этапе детализированного проектирования, в готовое программного обеспечения;
- модульное тестирование – выполняется проверка каждого закодированного модуля на наличие ошибок;
- интеграция и тестирование – установка взаимосвязей между группами ранее поэлементно испытанных модулей с целью подтверждения того, что эти группы работают также хорошо, как и модули, испытанные независимо друг от друга на этапе поэлементного тестирования;
- системное и приемочное тестирование – выполняется проверка функционирования программной системы в целом (полностью интегрированная система), после помещения в ее аппаратную среду в соответствии со спецификацией требований к ПО;
- производство, эксплуатация и сопровождение – программное обеспечение запускается в производство. На этой фазе предусмотрены также модернизация и внесение поправок;
- приемочные испытания (не показаны на рисунке) – позволяет пользователю протестировать функциональные возможности системы на соответствие исходным требованиям. После окончательного тестирования программного обеспечения и окружающее его аппаратное обеспечение становятся рабочими. После этого обеспечивается сопровождение системы.

При использовании V-образной модели при разработке проекта обеспечивается несколько преимуществ [40]:

- в модели особое значение придается планированию, направленному на верификацию и аттестацию разрабатываемого продукта на ранних стадиях его разработки. Фаза модульного тестирования подтверждает правильность детализированного проектирования. Фазы интеграции и тестирования реализуют архитектурное проектирование или проектирование на высшем уровне. Фаза тестирования системы подтверждает правильность выполнения этапа требований к продукту и его спецификации;
- в модели предусмотрены аттестация и верификация всех внешних и внутренних полученных данных, а не только самого программного продукта;
- в V-образной модели определение требований выполняется перед разработкой проекта системы, а проектирование программного обеспечения — перед разработкой компонентов;

- модель определяет продукты, которые должны быть получены в результате процесса разработки, причем каждые полученные данные должны подвергаться тестированию;
- благодаря модели менеджеры проекта может отслеживать ход процесса разработки, так как в данном случае вполне возможно воспользоваться временной шкалой, а завершение каждой фазы является контрольной точкой;
- модель проста в использовании (относительно проекта, для которого она является приемлемом).

В некоторых случаях V-образная модель может быть модифицирована с целью адаптации к динамическим изменениям требованиям на разных этапах жизненного цикла. Общераспространенная модификация V-образной модели, направленная на преодоление ее недостатков, включает в себя внесение итерационных циклов для разрешения изменения в требованиях за рамками фазы анализа.

4.3 Модель качества программного обеспечения

На данный момент наиболее распространена и используется многоуровневая модель качества программного обеспечения, представленная в наборе стандартов ISO 9126 [41]. На верхнем уровне выделено шесть основных характеристик качества ПО, каждую из которых определяют набором атрибутов [42], [38], [43], имеющих соответствующие метрики для последующей оценки (рисунок 4.2).

Рисунок 4.2 – Модель качества программного обеспечения

Функциональность (Functionality) - определяется способностью ПО решать задачи, которые соответствуют зафиксированным и предполагаемым потребностям пользователя, при заданных условиях использования ПО. Т.е. эта характеристика отвечает за то, что ПО работает исправно и точно, функционально совместимо, соответствует стандартам отрасли и защищено от несанкционированного доступа [41], [43].

Надежность (Reliability) – способность ПО выполнять требуемые задачи в обозначенных условиях на протяжении заданного промежутка времени или указанное количество операций. Атрибуты данной характеристики – это завершенность и целостность всей системы, способность самостоятельно и корректно восстанавливаться после сбоев в работе, отказоустойчивость [41], [43].

Удобство использования (Usability) – возможность легкого понимания, изучения, использования и привлекательности ПО для пользователя [41], [43].

Эффективность (Efficiency) – способность ПО обеспечивать требуемый уровень производительности в соответствие с выделенными ресурсами, временем и другими обозначенными условиями [41], [43].

Удобство сопровождения (Maintainability) – легкость, с которой ПО может анализироваться, тестироваться, изменяться для исправления дефектов, для реализации новых требований, для облегчения дальнейшего обслуживания и адаптироваться к именующемуся окружению [41], [43].

Портативность (Portability) – характеризует ПО с точки зрения легкости его переноса из одного окружения (software/hardware) в другое [41], [43].

Качество программного обеспечения достигается за счёт двух взаимосвязанных процессов: *обеспечение качества* и *контроль качества* – причём оба эти процесса должны быть неотъемлемой частью всех этапов жизненного цикла программного обеспечения. Нельзя начать разрабатывать некачественный программный продукт, и задуматься о его качестве уже перед завершением разработки.

4.4 Принципы обеспечения качества

В работе [44] описаны необходимые условия обеспечения качества программного продукта. В работах [38], [41], [44] также приводятся рекомендации по обеспечению качества.

Обобщая имеющуюся информацию, а также используя личный опыт в проектировании и создании программных систем, опишем подробнее принципы обеспечения качества на различных этапах жизненного цикла программного продукта.

4.4.1 Фаза анализа и проектирования

Принцип: Применять UML для описания проектных требований и идей их реализации

UML — язык графического описания для объектного моделирования в области разработки программного обеспечения, является языком широкого профиля. Кроме того, UML это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью [33], [37], [45].

UML также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. Особенности языка позволяют моделировать сложную систему с нескольких различных точек зрения, каждый раз принимая во внимание один аспект моделируемой системы и абстрагируясь от остальных [32], [45].

Основные преимущества:

- UML позволяет разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (класс, компонент, обобщение, агрегация и поведение) и больше сконцентрироваться на проектировании и архитектуре.
- UML объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на объектно-ориентированных языках;
- Диаграммы UML сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом;
- UML позволяет вводить собственные текстовые и графические стереотипы, расширяя имеющуюся базу семантических примитивов;
- UML имеет широкое распространение и динамично развивается.

4.4.2 Фаза разработки

Принцип: Следовать соглашениям по оформлению кода

Стиль кодирования – набор правил и соглашений, используемых при написании исходного кода на некотором языке программирования. Наличие общего стиля программирования облегчает понимание и поддержание исходного кода, написанного более чем одним программистом, а также упрощает взаимодействие нескольких человек при разработке программного обеспечения [46].

Осознавая важность наличия единых правил написания и оформления программного кода, на основе распространённых соглашений по оформлению кода на C++, таких как Wildfire C++ Programming Style (Wildfire Inc.) [46], Google C++ Style Guide (Google Inc.) [47] , Qt Coding Style (Digia Inc.) [48], [49] и с учётом выбранного инструментария разработки (Qt) был спроектирован собственный набор стилистических правил и рекомендаций по оформлению кода – «Стиль программирования C++ для QT», приведённый в приложении B.

Принцип: Использовать системы управления версиями проектов

Система управления версиями (от англ. Version Control System) — программное обеспечение для облегчения работы с изменяющейся информацией [50]. В более широком смысле представляет комплекс методов, направленных на систематический учёт изменений, вносимых разработчиками в программный продукт в процессе его разработки и сопровождения, сохранение целостности системы после изменений, предотвращение нежелательных и непредсказуемых эффектов, формализацию процесса внесения изменений [51].

Такие системы широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы, что позволяет при необходимости возвращаться к более ранним версиям, определять, кто и когда произвёл изменения. Ярким современным представителем систем управления является свободная Subversion (SVN) [52].

4.4.3 Фаза тестирования и отладки

Тестирование является одним из наиболее устоявшихся способов обеспечения и измерения качества, определения корректности, реальной надежности и безопасности функционирования программного обеспечения на всех этапах жизненного цикла и входит в набор эффективных средств современной системы обеспечения качества программного продукта [53].

Среди множества классификаций видов тестирования следует выделить четырёхуровневую модель, которая наиболее близка структуре применяемой V-модели жизненного цикла разработки программного обеспечения (рисунок 4.3).

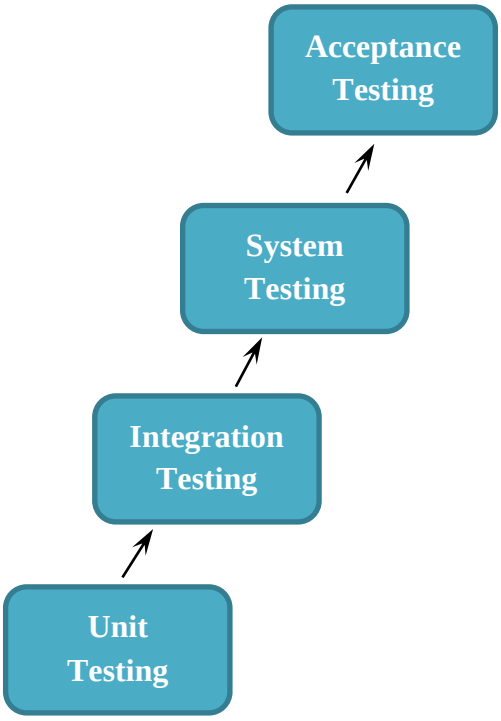

Рисунок 4.3 – Уровневая модель тестирования

Модульное тестирование (Unit-testing) — уровень тестирования, на котором тестируется минимально возможный для тестирования компонент (класс, функция). На этом уровне применяются методы «белого ящика». В современных проектах модульное тестирование осуществляется разработчиками. Модульное тестирование предназначено для проверки правильности отдельных модулей, вне зависимости от их окружения.

Модульное тестирование является важной составной частью отладочного тестирования, выполняемого разработчиками для отладки написанного ими кода.

Интеграционное тестирование (Integration testing) – уровень тестирования, на котором отдельные программные модули объединяются и тестируются в группе. Проводится после модульного тестирования. В качестве входных данных использует модули, над которыми было проведено модульное тестирование, группирует их в более крупные множества, выполняет тесты, определённые в плане тестирования для этих множеств, и представляет их в качестве выходных данных и входных для последующего системного тестирования.

Интеграционное тестирование выполняется разработчиками, используется при отладке, но на более позднем этапе разработки.

Системное тестирование (System testing) – это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям. Системное тестирование относится к методам тестирования «чёрного ящика», и, тем самым, не требует знаний о внутреннем устройстве системы.

Системное тестирование выполняется инженерами по тестированию.

Приемочное тестирование (Acceptance testing) – это тестирование готового продукта конечными пользователями на реальном окружении, в котором будет функционировать тестируемое приложение. Приемочные тесты разрабатываются пользователями, обычно, в виде сценариев.

Поскольку фундаментальные характеристики качества будущего программного продукта обеспечиваются тщательным тестированием на уровнях модульного и интеграционного тестирования и во многом зависят от разработчика, то основной принцип обеспечения качества на стадии тестирования сформулируем следующим образом.

Принцип: Планировать и осуществлять модульное и интеграционное виды тестирования (ответственность разработчика)

Реализация тестирования разделяется на три этапа [53]:

- создание тестового набора (test suite) путем ручной разработки или автоматической генерации для конкретной среды тестирования (testing environment);
- прогон программы на тестах, управляемый тестовым монитором (test monitor, test driver) с получением протокола результатов тестирования (test log);
- оценка результатов выполнения программы на наборе тестов с целью принятия решения о продолжении или остановке тестирования.

Основная проблема тестирования – определение достаточности множества тестов для истинности вывода о правильности реализации программы, а также нахождения множества тестов, обладающего этим свойством.

Несмотря на все сложности и трудозатраты на подготовку и проведение тестирования – тестирование считается основным методом контроля качества.

4.4.4 Фаза внедрения и сопровождения

Принцип: Использовать системы автоматического формирования и доставки отчётов об ошибках

На стадии внедрения и эксплуатации программного продукта его штатное функционирование может прерываться ввиду возникновения критических ошибок, которые приводят к аварийному завершению программы – называемому крэшем. Проанализировать случившееся, выявить причины и устранить неисправность (дефект) разработчику помогают отчёты об ошибках, которые часто включают в себя такую информацию, как: тип крэша, образ стека, версия программы, тип центрального процессора, версия операционной системы, а также протокол работы программы. Отчёт об ошибке создаётся специальной программой – crash reporter. Целью такой программы является сбор данных о произошедшем крэше и отправка этих данных по сети некой третьей стороне, обычно производителю программного обеспечения.

Из множества продуктов создания отчётов об ошибках отдельно стоит выделить библиотеку CrashRpt [54], [55] для Windows. Библиотека CrashRpt позволяет отлавливать исключения в программах, созданных в Microsoft Visual C++ и работающих в Windows. CrashRpt перехватывает необработанные исключения, создаёт файл-минидамп, строит описатель ошибки в формате XML, предоставляет интерфейс с пользователем, и сжимает отчёт и отправляет его группе поддержки приложения. Кроме того библиотека распространяется по «новой» лицензии BSD, что позволяет её бесплатно использовать при разработке собственных приложений.

Принцип: Использовать системы отслеживания ошибок

Система отслеживания ошибок – специальная прикладная программа, разработанная с целью помочь разработчикам программного обеспечения учитывать и контролировать ошибки, найденные в программах, пожелания пользователей, а также следить за процессом устранения этих ошибок и выполнения или невыполнения пожеланий [56]. Наличие системы отслеживания ошибок чрезвычайно важно при разработке программного обеспечения, они также активно используются компаниями по созданию программных продуктов.

К наиболее распространённым системам отслеживания ошибок относятся: Bugzilla [57] и Trac [58] – открытые, Atlassian JIRA [59] и TrackStudio Enterprise [60] – проприетарные.

Особо стоить выделить Atlassian JIRA, поскольку она имеет большое количество возможностей конфигурации: для каждого приложения может быть определен отдельный тип билета с собственным рабочим пространством, набором статусов, одним или несколькими видами представления. JIRA допускает определение для каждого индивидуального JIRA-проекта собственных прав доступа, поведения и видимости полей. Кроме того, JIRA интегрируется с системами управления версиями, такими как Subversion, CVS, Git, Clearcase, Team Foundation Server и другими.

4.5 Контроль качества

Как было сказано выше в 4.3, для контроля качества программных продуктов необходимо проводить оценку качества на всех этапах его жизненного цикла [44], то есть:

- планировать показатели качества программного продукта (допроизводственная стадия);
- контролировать качество на отдельных этапах разработки (техническое задание, технический проект, рабочий проект);
- контролировать качество в процессе производства ПП;
- проверять эффективность модификации ПП на этапе сопровождения (постпроизводственная стадия).

В каждый момент времени нужно сопоставлять информацию о фактическом состоянии процесса жизненного цикла разработки программного продукта с его характеристиками или характеристиками его результата, запланированными в допроизводственной стадии. В зависимости от того, обеспечиваются ли заданные характеристики или имеют место недопустимые отклонения от них, управляющие воздействия должны быть направлены соответственно на сохранение фактического состояния процесса или на его корректировку.

Для оценки качества программного продукта в процессе его *разработки* могут применяться различные методы [39]:

- статический анализ кода;
- динамический анализ кода;
- тестирование (модульное, интеграционное, системное, регрессионное).

4.5.1 Статический анализ

Статический анализ кода – это процесс выявления ошибок и недочетов в исходном коде программ. Статический анализ может рассматриваться как разновидность автоматизированного процесса обзора кода [61].

Инструменты статического анализа находят программные ошибки еще на ранней фазе создания проекта, обычно перед тем, как создается исполняемый файл. Такое раннее обнаружение особенно полезно для проектов больших встраиваемых систем, где разработчики не могут использовать средства динамического анализа до тех пор, пока программное обеспечение не будет завершено настолько, чтобы его было можно запустить на целевой системе [62].

На этапе статического анализа обнаруживаются и описываются области исходного кода со слабыми местами, включая скрытые уязвимости, логические ошибки, дефекты реализации, некорректности при выполнении параллельных операций, редко возникающие граничные условия и многие другие проблемы.

Разработчики могут использовать инструменты статического анализа в любое время на этапе разработки, даже когда написаны лишь фрагменты проекта. Однако чем более завершенным является код, тем лучше. При статическом анализе могут просматриваться все потенциальные пути исполнения кода – при обычном тестировании такое происходит редко, если только в проекте не требуется обеспечения 100%-го покрытия кода [62]. Например, при статическом анализе можно обнаружить программные ошибки, связанные с краевыми условиями или ошибками путей, не тестируемыми во время разработки.

Поскольку во время статического анализа делается попытка предсказать поведение программы, основанное на модели исходного кода, то иногда обнаруживается «ошибка», которой фактически не существует – это так называемое «ложное срабатывание» (false positive). Многие современные средства статического анализа реализуют улучшенную технику, чтобы избежать подобной проблемы и выполнить исключительно точный анализ.

Кроме того, некоторые статические анализаторы позволяют проверять, соответствует ли исходный код, принятому в компании стандарту оформления кода.

Также, инструменты статического анализа производят автоматический подсчёт метрик программного обеспечения, позволяющих дополнительно оценить качество программного обеспечения. Например, приложение SourceMonitor [63] рассчитывает следующие метрики: процент строк с комментариями, количество классов, количество методов в классах, среднее количество выражений на метод, максимальная сложность метода или функции, максимальная и средняя вложенность блока

Достоинства инструментов статического анализа:

1 Используется уже на ранних этапах жизненного цикла программного обеспечения, прежде чем код готов для исполнения и до начала тестирования.

2 Могут анализироваться существующие базы кодов, которые уже прошли тестирование.

3 Средства могут быть интегрированы в среду разработки в качестве части компонента, используемого при «ночных сборках» и как часть инструментального набора рабочего места разработчика.

4 Низкие стоимостные затраты: нет необходимости создавать тестовые программы или фиктивные модули; разработчики могут запускать свои собственные виды анализа.

Недостатки инструментов статического анализа:

1 Могут обнаруживаться программные ошибки и уязвимости, которые не обязательно приводят к отказу программы или воздействуют на поведение программы во время ее реального исполнения. Вероятность «ложного срабатывания»

2 Слаб в диагностике утечек памяти и поиске ошибок параллельных вычислений.

Наиболее распространёнными средствами статического анализа являются [64]: Coverity [65], Cppcheck [66], Clang [67], Klocwork Insight [68], Parasoft C/C++test [69], PC-Lint [70], PVS-Studio [71].

4.5.2 Динамический анализ

Динамический анализ кода – это способ анализа программы непосредственно при ее выполнении. При этом разработчик имеет возможность наблюдать или диагностировать поведение приложения во время его исполнения [72].

Во многих случаях в инструменте динамического анализа производится модификация исходного или бинарного кода приложения, чтобы установить процедуры-перехватчики для проведения инструментальных измерений. С помощью этих процедур можно обнаружить программные ошибки на этапе выполнения, проанализировать использование памяти, покрытие кода и проверить другие условия. Инструменты динамического анализа могут генерировать точную информацию о состоянии стека, что позволяет отладчикам отыскать причину ошибки.

Динамический анализ выполняется с помощью набора данных, которые подаются на вход исследуемой программе. Поэтому эффективность анализа напрямую зависит от качества и количества входных данных для тестирования. Именно от них зависит полнота покрытия кода, которая будет получена по результатам тестирования [72].

С помощью динамического тестирования могут быть получены следующие метрики [72], [62]:

- используемые ресурсы – время выполнения программы в целом или ее отдельных модулей, количество внешних запросов (например, к базе данных), количество

используемой оперативной и дисковой памяти, время задействования центрального процессора и других ресурсов;

- цикломатическая сложность, степень покрытия кода тестами;
- программные ошибки – деление на ноль, разыменование нулевого указателя, утечки памяти, «состояние гонки»;
- наличие в программе уязвимостей.

Динамическое тестирование наиболее важно в тех областях, где главным критерием является надежность программы, время отклика или потребляемые ресурсы.

Последовательность процесса динамического анализа представлена на рисунке 4.4 и состоит из следующих шагов [62]:

- наблюдение – захват ошибок среды выполнения, обнаружение мест утечки памяти и отображение всех результатов в среде IDE;
- корректировка – разработчик проводит трассировку каждой ошибки назад до нарушающей работу строки исходного текста и устраняет проблему;
- профилирование – устранив обнаруженные ошибки и утечки памяти, разработчик анализирует использование ресурсов во времени, включая пиковые ситуации, среднюю загрузку и избыточное расходование ресурсов;
- оптимизация – используя информацию этапа профилирования, разработчик осуществляет анализ использования вычислительных ресурсов программой.

Рисунок 4.4 – Последовательность процесса динамического анализа

Достоинства динамического анализа кода:

- Высокая продуктивность по нахождению ошибок (редко возникают «ложные срабатывания), поскольку обнаружение ошибки происходит в момент ее возникновения в работающей программе;
- Для отслеживания причины ошибки может быть произведена полная трассировка стека и среды исполнения;

- зачастую не требуется исходный код; это позволяет протестировать программы с закрытым кодом.

Недостатки динамического анализа кода:

- полнота анализа ошибок зависит от степени покрытия кода. Обнаруживает дефекты только на трассе, определяемой конкретными входными данными; дефекты, находящиеся в других частях программы, не будут обнаружены;
- не обеспечивает проверку правильности логики функционирования кода;
- требуются значительные вычислительные ресурсы для проведения тестирования, что может приводить к проблемам при работе критическим ко времени программным кодом.

К наиболее популярным средствам динамического анализа относятся: Valgrind [73], IBM Rational Purify [74], Intel Parallel Studio XE Suites [75], DevPartner BoundsChecker [76], Parasoft Insure++ [77].

Каждый из инструментов статического и динамического анализа имеет свои сильные стороны. Разработчики должны использовать оба этих инструмента в ежедневной работе. Например, инструменты статического анализа способны обнаруживать ошибки, которые пропускаются инструментами динамического анализа, потому что инструменты динамического анализа фиксируют ошибку лишь в случае, если во время тестирования ошибочный фрагмент кода исполняется. С другой стороны, инструменты динамического анализа обнаруживают программные ошибки конечного работающего процесса.

4.5.3 Тестирование

Как было сказано в 4.4.3 «тестирование является одним из наиболее устоявшихся способов измерения качества…». С технической точки зрения тестирование заключается в выполнении приложения на некотором множестве исходных данных и сверке получаемых результатов с заранее известными (эталонными), с целью установить соответствие различных свойств и характеристик приложения заказанным свойствам [53].

Для улучшения контроля над процессом тестирования необходимо введение и использование метрик. Согласно международному стандарту ISO 14598, метрика – это количественный масштаб и метод, который может использоваться для измерения.

В литературе [78], [79], [80], [81], [82] представлено огромное количество разнообразных метрик, обеспечивающих оценку разнородных показателей. Для удобства метрики классифицируются на различные категории. Выбор определённых метрик определяется требованиями каждого отдельного проекта. Из множества существующих выделим следующие

группы метрик [83] по типам сущностей, имеющих наибольшую практическую пользу для большинства современных проектов в решении задачи обеспечении качества и тестирования программного продукта.

1 метрики по тестовым случаям (Test Cases) представлены в таблице 4.1;
2 метрики по дефектам представлены в таблице 4.2;
3 метрики по задачам представлены в таблице 4.3.

Таблица 4.1 – Метрики по тестовым случаям

Метрика	**Назначение**
Passed/Failed Test Cases	Метрика показывает результаты прохождения тест кейсов, а именно отношение количества удачно пройденных к завершившимся с ошибками. В идеале, к концу проекта, количество провальных тестов стремиться к нулю
Not Run Test Cases	Метрика показывает количество тест кейсов, которые еще необходимо выполнить в данной фазе тестирования. Имея данную информацию, можно проанализировать и выявить причины, по которым тест не были проведены

Таблица 4.2 – Метрики по дефектам

Метрика	**Назначение**
Open/Closed Bugs	Метрика показывает отношение количества открытых багов к закрытым (исправленным и перепроверенным)
Bugs by Severity	Количество багов по серьезности
Bugs by Priority	Количество багов по приоритету
Reopened/Closed Bugs	Метрика показывает отношение количества переоткрытых багов к закрытым (исправленным и перепроверенным)
Rejected/Opened Bugs	Метрика показывает отношение количества отклоненных багов к открытым

Метрики «Open/Closed Bugs», «Bugs by Severity» и «Bugs by Priority» хорошо визуализируют степень приближения продукта к достижению критериев качества по дефектам. Имея требования к количеству открытых дефектов, после каждой итерации тестирования необходимо сравнивать их с реальными данными, тем самым обнаруживая места, где требуется активизация деятельности для скорейшего достижения цели.

Метрики «Reopened/Closed Bugs» и «Rejected/Opened Bugs» направлены на отслеживание работы отдельных участников групп разработки и тестирования.

Таблица 4.3 – Метрики по задачам

Метрика	**Назначение**
Deployment tasks	Метрика показывает количество и результаты установок приложения. В случае если количество отклоненных командой тестирования версий будет критически высоким, рекомендуется срочно проанализировать и выявить причины, а также в кротчайшие сроки решить имеющуюся проблему
Still Opened Tasks	Метрика показывает количество все еще открытых задач. К окончанию проекта все задачи должны быть закрыты. Под задачами понимаются следующие виды работ: написание документации (архитектура, требования из спецификации, планы), имплементация новых модулей или изменение существующих по запросам на изменения и многое другое

4.6 Выводы по главе 4

1 Произведено обоснование применения V-модели жизненного цикла программного обеспечения к созданию многопоточной системы

2 Рассмотрена наиболее распространенная в настоящее время многоуровневая модель качества программного обеспечения, представленная в наборе стандартов ISO 9126

3 Предложены принципы обеспечения качества, которые основаны на рекомендациях стандарта ISO 9126 и позволяют закладывать качество программ при разработке технического задания и контролировать его на всех этапах жизненного цикла

4 Рассмотрены способы контроля качества и сформулированы рекомендации по проведению статического анализа, динамического анализа и тестирования программного обеспечения

5 Предложенные принципы обеспечения и контроля качества положены в основу создания качественной многопоточной системы обработки телеметрической информации

Глава 5. Информационное и программное обеспечение обслуживающей подсистемы автоматизированного многопоточного приёма, обработки и анализа телеметрии

Ранее в качестве основного метода проектирования системы был выбран объектно-ориентированный подход. Следуя его принципам необходимо спроектировать объектно-ориентированную модель обслуживающей подсистемы, удовлетворяющую ключевым функциям, сформулированным на этапе общего проектирования многопоточной системы приёма телеметрической информации.

Исходя из требований к многопоточной системе также требуется предложить метод адаптивной передачи телеметрии потребителям, метод получения телеметрической информации о состоянии космического аппарата с разгонного блока. Для повышения степени автоматизации задач обработки телеметрии необходимо разработать систему автоматизированной подготовки отчётов о состоянии отдельных параметров и целых групп, с возможностью статистического анализа. Кроме того создаваемая обслуживающая подсистема должна обеспечивать санкционированный доступ клиентов к телеметрической информации.

Одним из требований к многопоточной системе является обеспечение централизованного хранения телеметрической информации. Для этой цели необходимо спроектировать структуру реляционной базы данных.

5.1 Объектно-ориентированная модель

Как было описано выше в 3.1 эффективным средством объектно-ориентированного анализа и проектирования является язык унифицированного моделирования UML. Поэтому для построения объектно-ориентрованной модели сервера обработки телеметрической информации также будем использовать семантику и диаграммы языка UML.

5.1.1 Диаграммы классов

В результате анализа требований, заданных в 2.1, была спроектирована общая структура системы (подраздел 2.2) и отдельно выделены задачи, решаемые обслуживающей подсистемой, архитектура которой описана в 2.3.1.

С учётом задач обработки телеметрической информации и на основании архитектуры обслуживающей подсистемы была построена диаграмма классов, отражающих указанную подсистему в виде объектно-ориентированной модели (рисунок 5.1).

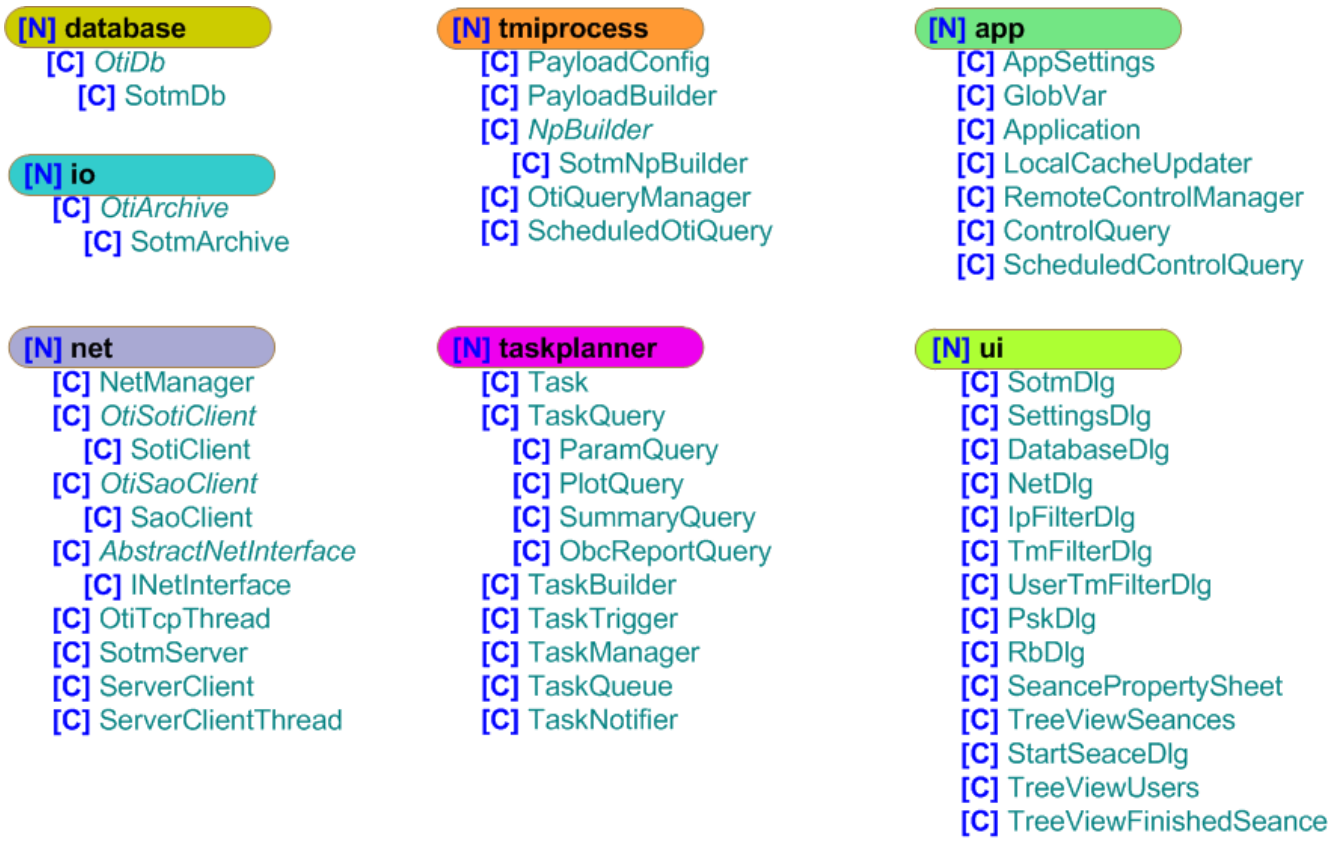

Рисунок 5.1 – Диаграмма классов сервера обработки телеметрической информации

Множество выделенных классов имеет логическую группировку по характеру решаемых задач. Элементом группировки являются пространства имён, обозначенные символом N (namespace). Каждое пространство имён состоит из классов – C (class) и/или структур – T (struct). Курсивом обозначены названия базовых классов, определённых в библиотеке классов унифицированной обработки и анализа. Ниже приведено функциональное назначение пространств имён, а также состав и краткое описание входящих в них классов и структур.

database

Пространство имён *database* содержит класс для взаимодействия с базой данных.

- класс *SotmDb* расширяет функционал базового класса OtiDb методами вывода сообщений об ошибках в протокол и на экран

io

Пространство имён *io* содержит класс для манипулирования архивами телеметрии.

- класс *SotmArchive* расширяет функционал базового класса OtiArchive возможностью автоматического перемещения временного архива в долговременный (на жёсткий диск или в базу данных) через настраиваемый временной интервал.

net

Пространство имён *net* содержит набор классов для чтения информации из сети. К нему относятся:

- класс *NetManager* является менеджером по управлению сетевыми задчами в приложении;
- класс *SotiClient* расширяет функционалность базового класса *OtiSotiSocket* и обеспечивает работу по протоколу СОТИ – СПО;
- класс *SaoClient* расширяет функционалность базового класса *OtiSaoSocket* и обеспечивает работу по протоколу САО – СПО;
- класс *INetInterface* расширяет функциональность базового класса *AbstractNetInterface* и обеспечивает унифицированную обработку телеметрического кадра;
- класс *OtiTcpThread* организует обработку телеметрической информацаии, принимаемого по сети, в отдельном вычислительном потоке;
- класс *SotmServer* управляет подключением внутренних и внешних сетевых клиентов, сохраняя их в общем списке;
- класс *ServerClient* содержит набор методов для взаимодействия с внутренними и внешними сетевыми клиентами;
- класс *ServerClientThread* организует обработку сообщений, передаваемых и/или принимаемых от внутренних и внешних сетевых клиентов, в отдельном потоке.

tmiprocess

Пространство имён *tmiprocess* содержит набор классов для обработки телеметрических кадров, управления конфигурацией полезной нагрузки и обработки запросов от удалённых клиентов. К нему относятся:

- класс *PayloadConfig* содержит ТМ-параметры конфигурации полезной нагрузки;
- класс *PayloadBuilder* содержит методы манипулирования конфигурацией полезной нагрузки КА, а именно: получение текущей конфигурации, а также сохранение, загрузка или удаление из базы данных;
- класс *SotmNpBuilder* расширяет функционал базового класса *NpBuilder* и дополнительно обеспечивает манипулирование конфигурацией полезной нагрузки;
- класс *OtiQueryManager* управляет обработкой запросов типа *OtiQuery* на выбор значений телеметрических параметров от внешних и внутренних клиентов;
- класс ScheduledOtiQuery представляет собой отложенный или регулярный (периодический) запрос *OtiQuery*, помещённый в очередь запросов *OtiQueryManager*.

taskplanner

Пространство имён *taskplanner* содержит набор классов для автоматизированного формирования отчётов по данным телеметрических архивов. К нему относятся:

- класс *Task* содержит параметры запуска задания для формирования отчёта, а также тело задания в формате XML. Тело задания состоит из одной или нескольких секций (подзадач) *TaskQuery*, различающихся форматом представления отчётной информации;
- класс *TaskQuery* содержит описание единичной секции (подзадачи);
- класс *ParamQuery* содержит описание подзадачи на вывод значений параметров в виде списка, или табличном виде;
- класс *PlotQuery* содержит описание подзадачи на построение графиков заданных телеметрических параметров;
- класс *SummaryQuery* содержит описание подзадачи на вычисление обобщённой информации по заданным телеметрическим параметрам;
- класс *ObcReportQuery* содержит описание подзадачи на выбор значений телеметрических параметров из отчётов бортового компьютера;
- класс *TaskBuilder* выполняет задачу и содержит результат выполнения;
- класс *TaskTrigger* содержит дату и время начала выполнения задачи;
- класс *TaskManager* управляет процессом отслеживания новых задач в системе и запуском их на исполнение;
- класс *TaskQueue* содержит очередь задач на исполнение;
- класс *TaskNotifier* контролирует условия начала выполнения задачи и информирует о готовности в случае достижения условий.

app

Пространство имён *app* содержит набор служебных классов приложения. К нему относятся:

- класс *AppSettings* содержит параметры работы приложения, такие как сетевой порт слушателя, параметры подключения к базе данных, язык приложения и многое другое;
- класс *GlobVar* содержит набор глобальных объектов приложения;
- класс *Application* содержит методы для устойчивого функционирования в кластере;

- класс *LocalCacheUpdater* обеспечивает регулярное фоновое обновление локального кэша из базы данных;
- класс *RemoteControlManager* обрабатывает запросы на удалённое управление конфигурацией приложения;
- класс *CotrolQuery* представляет запрос от клиента на удалённое управление конфигурацией приложения;
- класс *ScheduledControlQuery* представляет собой отложенный или регулярный (периодический) запрос *CotrolQuery*, помещённый в очередь запросов *RemoteControlManager.*

ui

Пространство имён *ui* содержит набор графических элементов приложения. К нему относятся:

- класс *SotmDlg* является главным окном приложения;
- класс SettingsDlg обеспечивает конфигурирование настроек параметров приложения;
- класс *DatabaseDlg* обеспечивает добавление, изменение подключений к базе данных;
- класс *NetDlg* обеспечивает добавление, изменение подключений к источникам телеметрической информации;
- класс *IpFilterDlg* обеспечивает конфигурирования IP-фильтра;
- класс *TmFilterDlg* представляет собой окно для конфигурирования фильтра телеметрических параметров;
- класс *UserTmFilterDlg* представляет собой окно для назначения фильтруемых параметров определённым учётным записям пользователей;
- класс *PskDlg* обеспечивает просмотр содержимого телеметрического кадра в необработанном виде;
- класс *RbDlg* обеспечивает просмотр значений телеметрических параметров с разгонного блока на участке выведения;
- класс *SeancePropertySheet* обеспечивает просмотр сведений о проводимом сеансе;
- класс *TreeViewSeances* обеспечивает просмотр статистики о проводимых сеансах;
- класс *StartSeanceDlg* предоставляет процедуру ручного ввода и начала сеанса;
- класс *TreeViewUsers* обеспечивает просмотр сведений о подключенных сетевых клиентах;
- класс *TreeViewFinishedSeances* обеспечивает просмотр истории проведённых сеансов.

Группа классов, предназначенных для обеспечения сетевого взаимодействия между источниками телеметрии и сервером обработки телеметрии с одной стороны и между сервером телеметрии и клиентами телеметрии с другой, представлена на соответствующей диаграмме классов на рисунке 5.2.

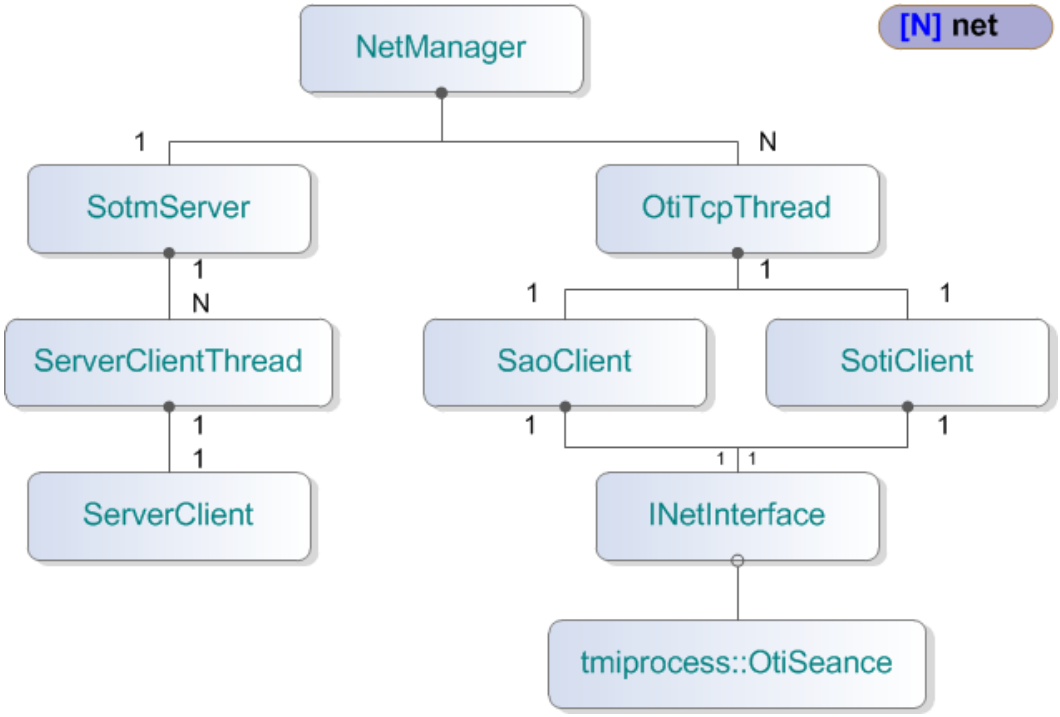

Рисунок 5.2 – Диаграмма классов сетевого взаимодействия

Из диаграммы видно, что класс-менеджер по управлению сетевыми задачами *NetManager* содержит коллекцию вычислительных потоков *OtiTcpThread*, обрабатывающих телеметрическую информации от различных источников телеметрии, принимаемую по сети. Каждый вычислительный поток *OtiTcpThread* содержит экземпляр обработчика телеметрии в формате САО-СПО – класс *SaoClient* или в формате СОТИ-СПО – класс *SotiClient*. Унифицированная обработка отдельных телеметрических кадров каждого из двух форматов достигается за счёт использования класса *INetInterface*, который в свою очередь обращается к методам класса обработки и анализа телеметрического сеанса *OtiSeance*, рассмотренного в 3.3.3.

Кроме классов *OtiTcpThread* класс-менеджер по управлению сетевыми задачами *NetManager* также содержит класс управления подключениями внутренних и внешних сетевых клиентов *SotmServer*. Названный класс для каждого авторизированного клиента организует отдельный вычислительный поток обслуживания *ServerClientThread*, позволяя тем самым эффективнее использовать возможности современных многопроцессорных вычислительных

систем. Каждый вычислительный поток содержит экземпляр класса *ServerClient*, обслуживающего заявки внешних и внутренних сетевых клиентов.

Одной из задач, решаемых системой обработки телеметрической информации, является автоматизирование формирование отчётов о состоянии отдельных параметров и целых групп параметров с возможностью статистического анализа. Группа классов, предназначенных для автоматизированного формирования отчётов по данным телеметрических архивов, представлена на диаграмме классов на рисунке 5.3.

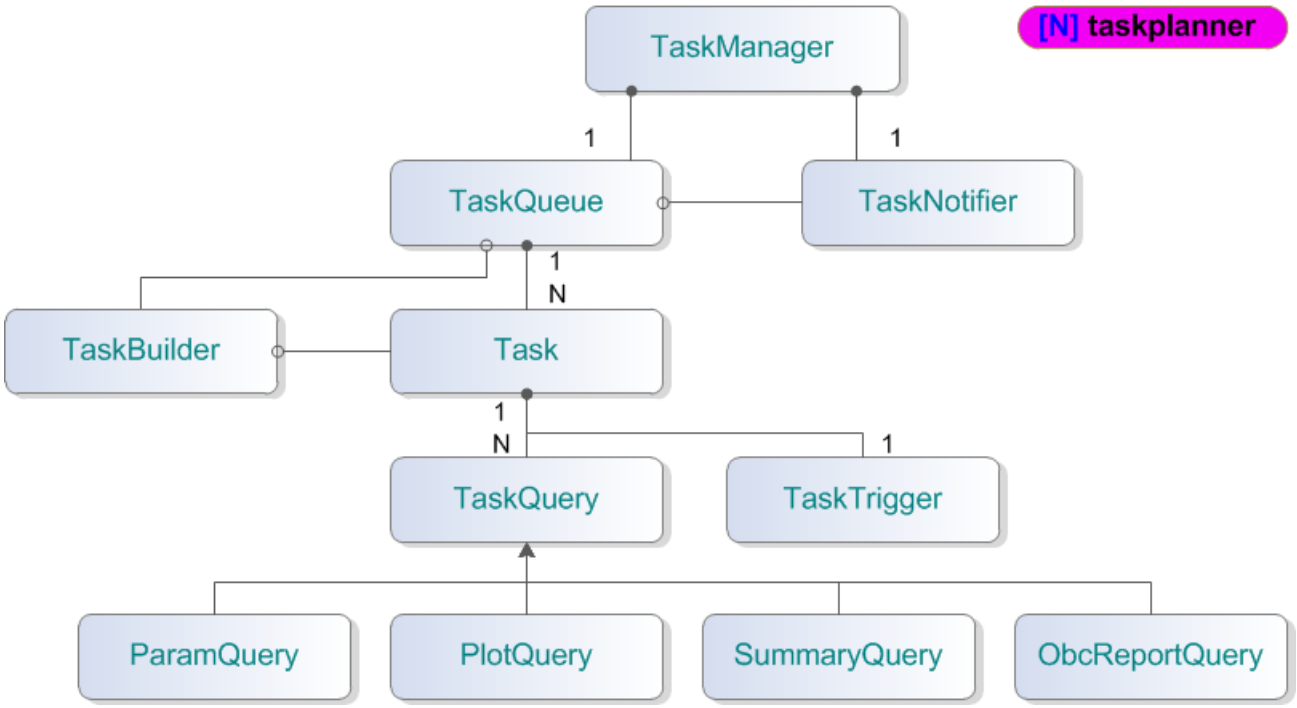

Рисунок 5.3 – Диаграмма классов автоматизированного формирования отчётов

В решении задачи автоматизированного формирования отчётов центральным элементов является класс управления процессом отслеживания новых задач в системе и запуска их на исполнение *TaskManager*. Он содержит экземпляр класса *TaskNotifier*, который обеспечивает проверку достижения условия о начале запуска задачи. В случае если подошло время выполнения задачи класс *TaskNotifier* отсылает соответствующее сообщение очереди задач на исполнение *TaskQueue*, также содержащейся в *TaskManager*. Класс очереди сообщений добавляет очередную задачу *Task* в общий список задач. Класс *TaskQueue* последовательно выбирает из списка задачи *Task* и обрабатывает их, причём обработку каждой задачи обеспечивает класс *TaskBuilder*.

Каждая задача *Task* содержит дату, время начала выполнения в виде экземпляра *TaskTrigger* и тело задания в формате XML, состоящее из одной или N секций (подзадач)

TaskQuery. Для удобства использования отчётной информации определено четыре формата представления результатов: класс *ParamQuery* – подзадача на отображение значений в виде списка или табличном виде, класс *PlotQuery* – подзадача на отображение графиков, класс *SummaryQuery* – подзадача на вычисление обобщённой информации по заданным телеметрическим параметрам и класс *ObcReportQuery* – подзадача на отображение значений параметров из отчётов бортового компьютера. Каждый из четырёх описанных классов наследует общую функциональность подзадачи *TaskQuery*.

Поскольку обслуживающая подсистема будет функционировать на специализированных аппаратно-программных средствах вычислительного комплекса ЦУП, как правило, серверного или кластерного исполнения, мониторинг её состояния и управление конфигурацией должны осуществляться не только через собственный графический интерфейс приложения, но и через программный интерфейс удалённого управления.

Группа классов, предназначенных для обеспечения удалённого управления сервером обработки телеметрии, представлена на соответствующей диаграмме классов на рисунке 5.4.

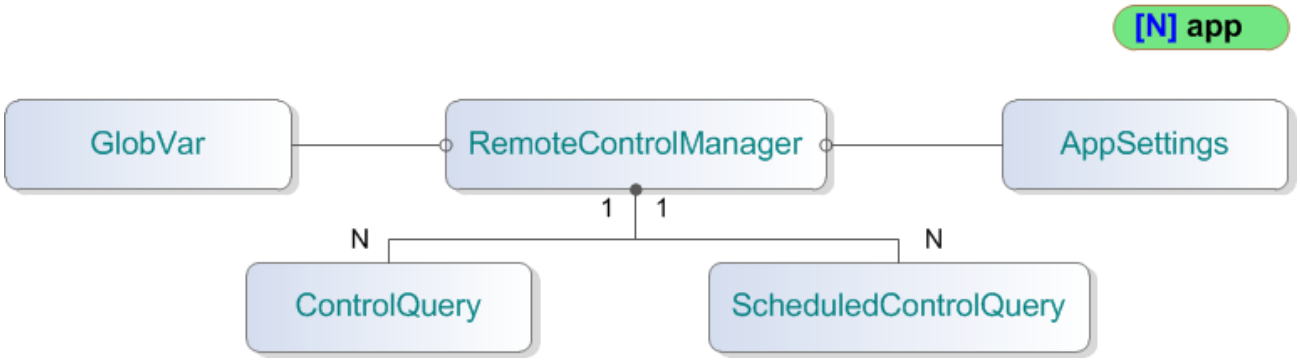

Рисунок 5.4 – Диаграмма классов удалённого управления сервером

На представленной диаграмме показано, что менеджер удалённого управления *RemoteControlManager* содержит несколько экземпляров класса *ControlQuery*, представляющего запрос от клиента на удалённое управление, а также несколько экземпляров класса отложенного или регулярного (периодического) запроса *ScheduledControlQuery*. При обработке принимаемых запросов менеджер удалённого управления *RemoteControlManager* взаимодействует с классом глобальных объектов приложения *GlobVar* и классом конфигурации приложения *AppSettings* для изменения или получения параметров функционирования сервера обработки.

5.1.2 Диаграммы объектов

Описание поведение модели в динамике удобно представлять в виде диаграмм объектов.

На рисунке 5.5 представлена диаграмма объектов, участвующих в сетевом обмене. Сценарий начинается с отправки объектом *Выч.поток_для_источника_ТМИ* (*OtiTcpThread*) команды *начать_сеанс (startSession(nKa, nip))* с параметрами номер КА, номер НИП для объекта *Источник_ТМИ* (*SaoClient* или *SotiClient*, в зависимости от выбранного типа источника ТМИ). Объект *Источник_ТМИ* (*SaoClient, SotiClient*) выполняет попытки подключения к источнику, вызывая метод *подключиться (connect).* В случае успешного подключения производится отправка параметров предстоящего сеанса источнику ТМИ вызовом метода *инициализировать_сеанс (initSession(nKa, nip)).* После этого начинают поступать сетевые сообщения. Каждое сетевое сообщение от источника ТМИ обрабатывается методом *обработать_сообщение (processMessages(nKa, nip)).* Если в принятом сообщении содержится кадр телеметрии вне зависимости от типа источника ТМИ происходит обращение к объекту *Общий_сетевой_интерфейс (INetInterface)* для передачи кадра в обработку посредством метода *кадр_принят (tmiReceived(nKa, nip)).* Объект *Общий_сетевой_интерфейс (INetInterface)* вызывает метод *читать (read)* объекта *Сеанс (OtiSeance),* который осуществляет полную обработку кадра телеметрии и был подробно описан в 3.4.

Когда очередной кадр телеметрии обработан, то есть выполнены расчёт достоверности, вычислены телеметрические параметры и выделены отчёты, метод *кадр (frame)* объекта *Сеанс (OtiSeance)* возвращает кадр телеметрии объекту *Общий_сетевой_интерфейс (INetInterface),* который формирует сообщение о готовности кадра к отправке клиентам *кадр_готов (tmiReceived(nKa, nip)* и отсылает его в объект *Выч.поток_для_источника_ТМИ* (*OtiTcpThread*), а тот в свою очередь пересылает принятое сообщение объекту *Сервер_для_клиентов (SotmServer)* для непосредственной рассылки клиентами телеметрической информации.

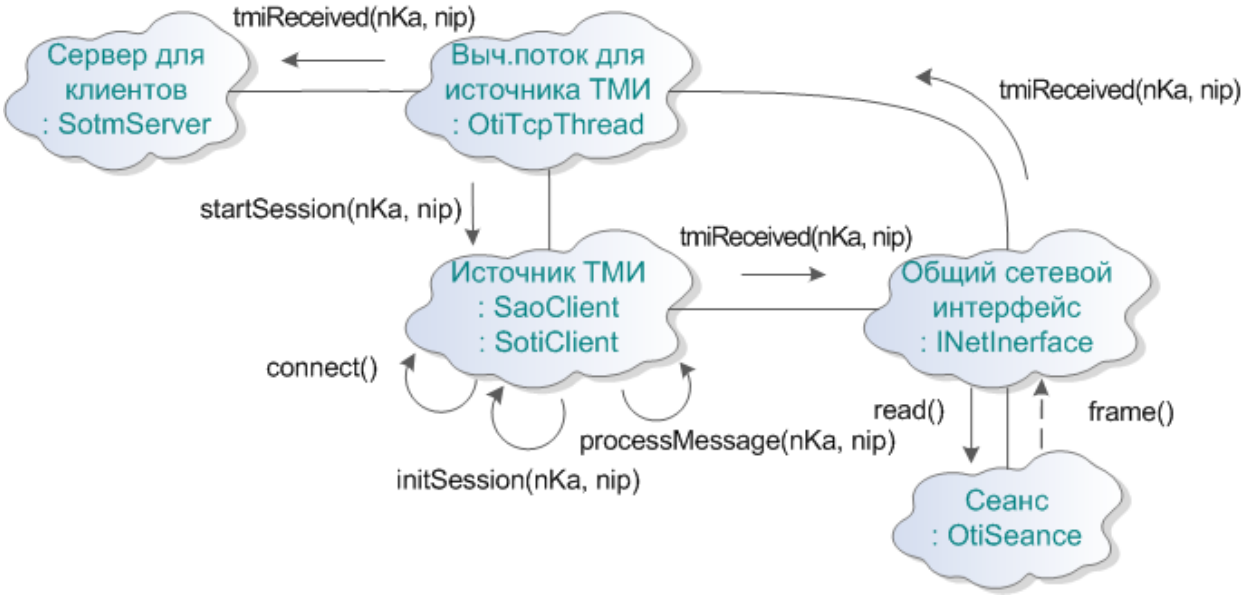

Рисунок 5.5 – Диаграмма объектов сетевого обмена

Для наглядности на этой диаграмме отсутствует описание взаимодействия с внешними и внутренними клиентами.

На рисунке 5.6 представлена диаграмма удалённого управления конфигурацией и мониторинга состояния обслуживающей подсистемы. Сценарий начинается с получения объектом *Менеджер_удалённого_управления (RemoteControlManager)* запроса в виде символьной строки. Далее для формирования из принятой символьной строки объекта *Запрос (ControlQuery) Менеджер_удалённого_управления (RemoteControlManager)* вызывает метод *создать (create).* Перед непосредственной обработкой запроса происходит его синтаксический и логический анализ, формирование внутреннего представления запроса посредством вызовом метода *анализировать (parse)* объекта *Запрос (ControlQuery).* Результат анализа запроса можно получить через обращение к методу *достверность (isValid).* Если проанализированный запрос содержит признак периодичности исполнения, о чём свидетельствует результат true метода *авто_запрос (isAutoQuery)*, то объект *Менеджер_удалённого_управления (RemoteControlManager)* преобразует его в объект *Отложенный_запрос (ScheduleControlQuery),* помещает в очередь отложенных запросов и запрос считается обработанным. В противном случае на основании сформированного внутреннего представления запрос исполняется. При этом происходит обращение к методам *читать (read)* или *писать (write)* объектов *Конфигурация_приложения (AppSettings), Глобальные_объекты (GlobVar)* в зависимости от типа удалённого запроса – читающий или пишущий. После чего запрос считается обработанным.

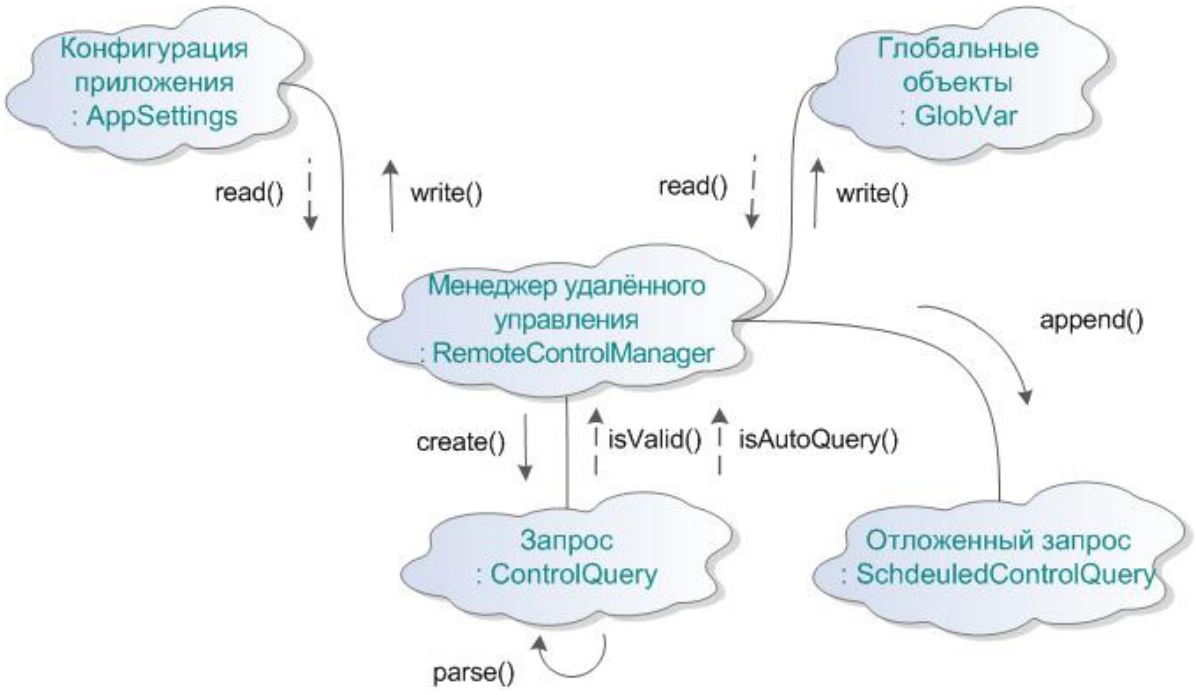

Рисунок 5.6 – Диаграмма объектов удалённого управления сервером

5.1.3 Диаграммы взаимодействия

Стоит ещё раз отметить, что диаграммы взаимодействий лучше диаграмм объектов передают семантику сценариев на ранних этапах жизненного цикла разработки, когда еще не идентифицированы протоколы отдельных классов.

Взаимодействие сетевых объектов во времени представлено на рисунке 5.7. Подробное описание взаимодействующих объектов приведено в 5.1.2.

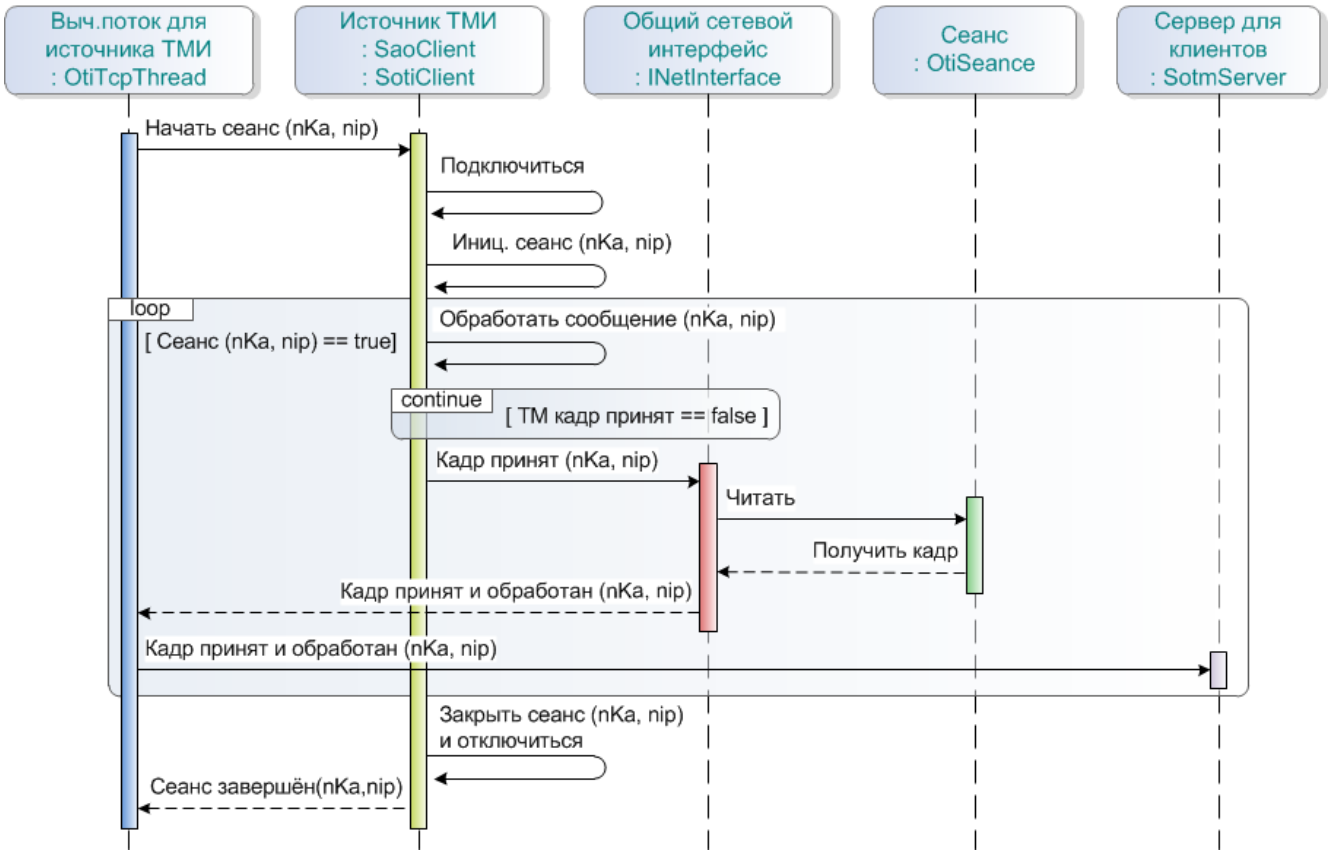

Рисунок 5.7 – Диаграмма взаимодействия сетевых объектов

Взаимодействие объектов удалённого управления сервером и мониторинга его состояния во времени представлено на рисунке 5.8. Подробное описание взаимодействующих объектов приведено в 5.1.2.

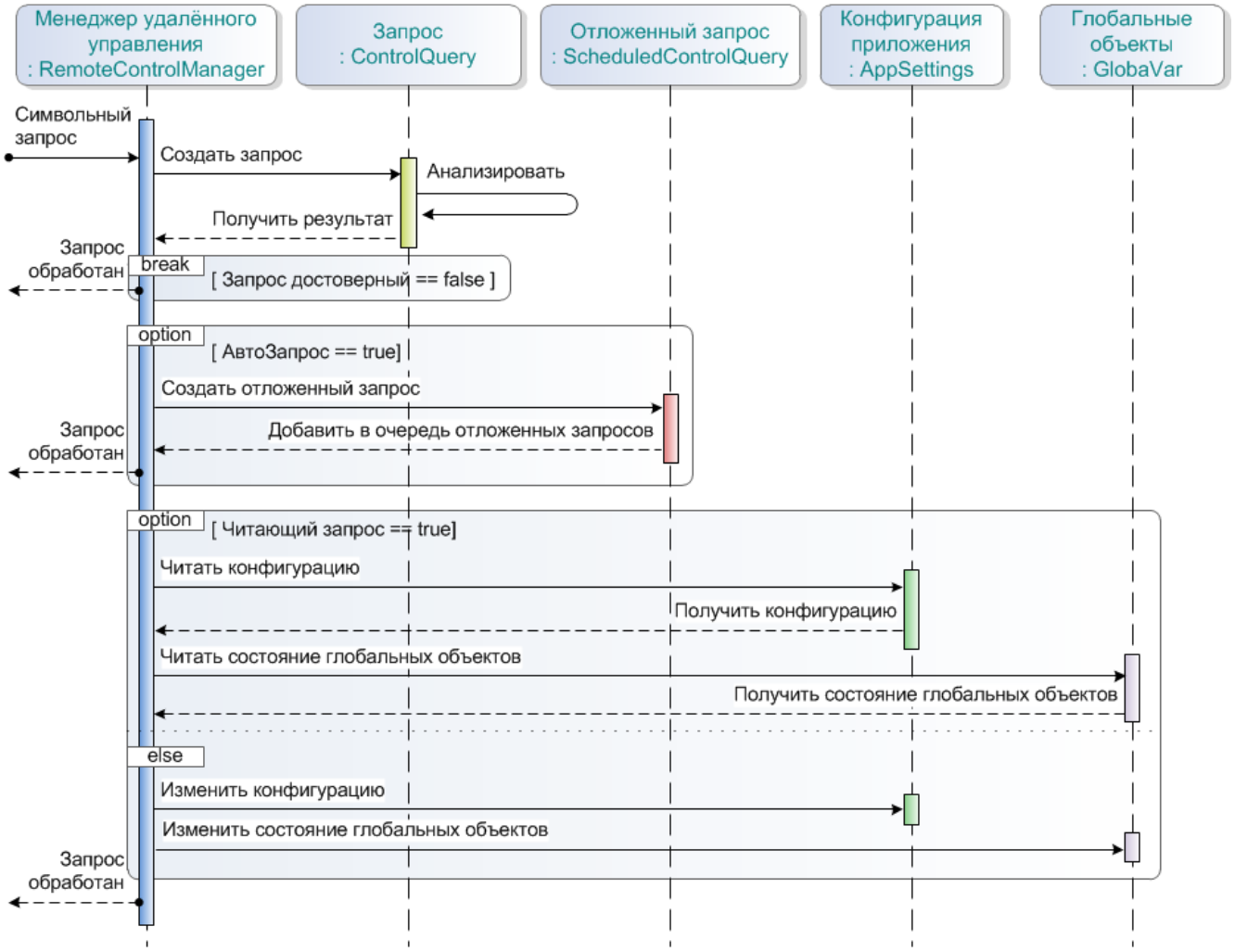

Рисунок 5.8 – Диаграмма взаимодействия объектов удалённого управления сервером

Построенные диаграммы взаимодействий (рисунки 5.7, 5.8) отражают динамику поведения системы и по сравнению с аналогичными диаграммами объектов (рисунки 5.5, 5.6), позволяют легче проследить порядок посылки сообщений.

5.1.4 Диаграмма компонентов

Сервер обработки телеметрии представляет собой законченный программный модуль, использующий библиотеки унифицированного анализа, связь с которыми была рассмотрена в подразделе 3.6 и проиллюстрирована на рисунке 3.11.

5.2 Метод проведения автоматизированного сеанса съёма телеметрии

Организация процесса приёма и обработки телеметрической информации на рабочее место телеметриста в существующей однопоточной системе требует от оператора сектора анализа выполнения последовательности ручных операций, представленных в виде сценария взаимодействия на рисунке 5.9.

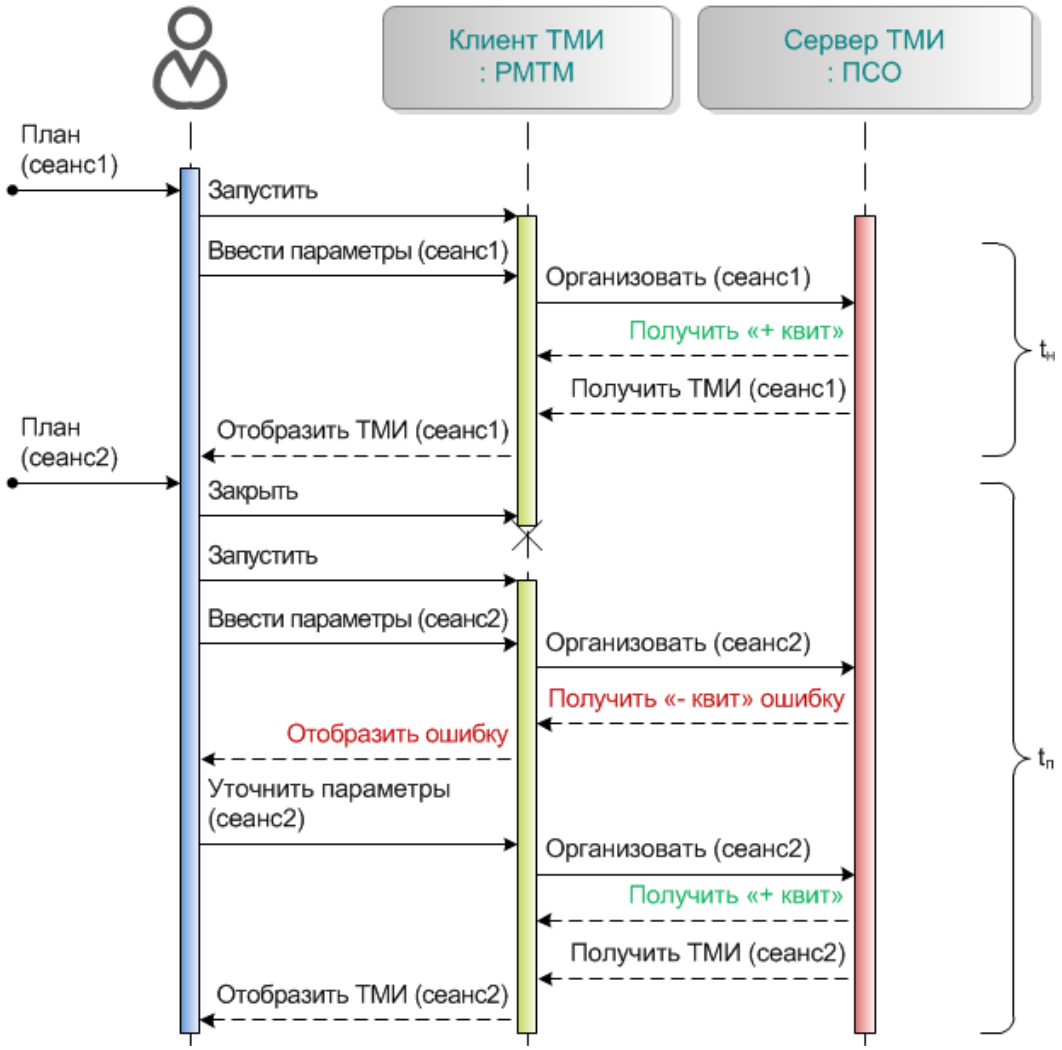

Рисунок 5.9 – Диаграмма взаимодействия компонент при ручной организации сеанса

Сценарий начинается с получения оператором из плана на сутки параметров предстоящего сеанса связи (сеанс$_1$). Затем оператор осуществляет запуск *Клиента_ТМИ (РМТМ)* и производит ручной ввод параметров сеанса$_1$. Программные модули *Клиента_ТМИ (РМТМ)* организуют сеанс$_1$ посредством выдачи набора управляющих команд (директив) в *Сервер_ТМИ (ПСО)*. *Сервер_ТМИ (ПСО)* отправляет в ответ клиенту положительную квитанцию в случае корректность принятых параметров сеанса и начинает выдачу телеметрической информации по сеансу$_1$. *Клиент_ТМИ (РМТМ)* принимает поступающую информацию, производит её обработку и предоставляет результаты анализа оператору на экране монитора.

Однако рассмотрим другую ситуацию, когда во время приёма и анализа телеметрии у оператора сектора анализа может возникнуть необходимость смены сеанса связи, например, при переключении на другой КА или ухудшении качества принимаемой телеметрической

информации с земной станции. Оператор получает параметры очередного сеанса связи (сеанс$_2$). Для смены сеанса в *Клиент_ТМИ (РМТМ)* требуется перезапуск приложения, поэтому оператор закрывает *Клиент_ТМИ (РМТМ)*, запускает его вновь и вводит параметры сеанса$_2$, которые отправляются в *Сервер_ТМИ (ПСО)*. Однако ввиду человеческого фактора оператор может ошибиться. В этом случае *Сервер_ТМИ (ПСО)* вернёт в *Клиент_ТМИ (РМТМ)* отрицательную квитанцию по признаку несоответствия параметров запрашиваемого сеанса параметрам имеющихся сеансов в сервере. Соответствующая ошибка будет отражена оператору на экране. Оператор вынужден проанализировать ошибку, произвести уточнение параметров сеанса$_2$ и предпринять попытку повторной организации сеанса$_2$. И только после того, как параметры сеанса будут указаны корректно, телеметрическая информация начнёт поступать в *Клиент_ТМИ (РМТМ)*.

На рисунке 5.9 параметр tн – время организации нового сеанса, определяемое как интервал времени между моментом получения оператором параметров сеанса до начала отображения телеметрической информации на экране монитора. Параметр tп – время переключения между сеансами, определяемое как интервал между получением команды на смену сеанса до начала отображения телеметрической информации на экране монитора.

Как видно из анализа представленной диаграммы рисунка 5.9 скорость организации нового сеанса связи, а также переключения между сеансами существенно зависит от квалификации и внимательности оператора. Учитывая ежесуточную нагрузку на сектор анализа, до 100 сеансов для группировки космических аппаратов о чём говорилось в 1.4, и имея требования по максимизации степени автоматизации задач управления (требование Б002), необходимо автоматизировать процедуру организации сеанса.

Ниже на рисунке 5.10 приведена диаграмма взаимодействия между внутренними клиентами и обслуживающей подсистемой при автоматизированной организации сеанса, иллюстрирующая решение обозначенной проблемы.

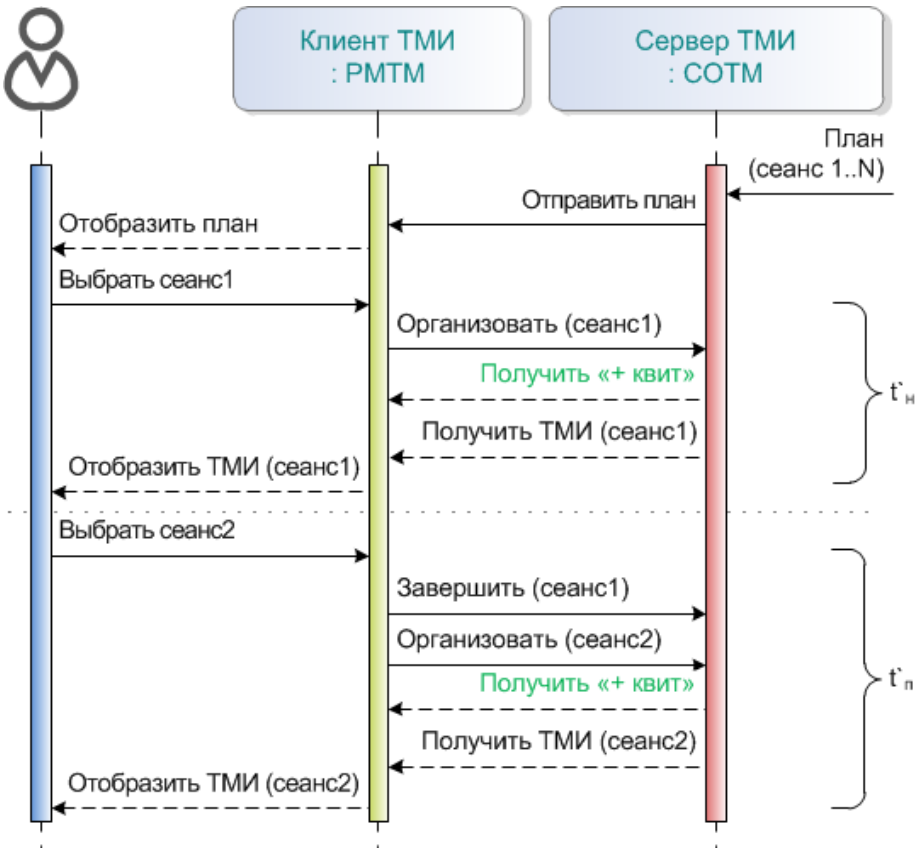

Рисунок 5.10 – Диаграмма взаимодействия компонент при автоматизированной организации сеанса

Обеспечение автоматизации достигается за счёт расширения функций протокола информационного взаимодействия между внутренними клиентами и обслуживающей подсистемой, а именно, введения таблицы состава сеансов, подробное описание которой приведено в приложении А. *Сервер_ТМИ (СОТМ)* формирует таблицу текущих сеансов и сообщает её в *Клиент_ТМИ (РМТМ)* с определённой периодичностью. На экране *Клиента_ТМИ (РМТМ)* оператор выбирает из предлагаемого списка интересующий сеанс и выдаёт команду о его начале. После чего *Клиент_ТМИ (РМТМ)* организует сеанс выдачей соответствующей директивы, *Сервер_ТМИ (СОТМ)* отправляет положительную квитанцию и начинает выдачу потока телеметрии, результат обработки которого отображается оператору на экране *Клиента_ТМИ (РМТМ)*.

Если оператор выбирает из предложенного списка другие параметры сеанса, то *Клиент_ТМИ (РМТМ)* автоматически завершает предыдущий сеанс и выдаёт директивы в *Сервер_ТМИ (СОТМ)* для организации нового сеанса. Далее процесс поступления телеметрии в *Клиент_ТМИ (РМТМ)* продолжается описанным ранее способом.

Предложенная схема автоматизации полностью исключает ручной ввод оператором параметров сеанса связи, а также обязательную процедуру перезапуска приложения для переключения на новый сеанс.

5.3 Метод адаптивной передачи телеметрии потребителям

Другим аспектом обеспечения онлайн анализа телеметрии является стабильность и качество принимаемой телеметрической информации на экраны внутренних клиентов. С внедрением многопоточной обслуживающей подсистемы, поддерживающей одновременный приём телеметрии с нескольких КА через множество земных станций, становится возможной реализация метода адаптивной передачи. В работах [84], [85] рассматривались варианты реализации данного метода. С учётом определённых модификаций был предложен собственный метод адаптивной передачи, суть которого проиллюстрирована на рисунке 5.11 и заключается в следующем.

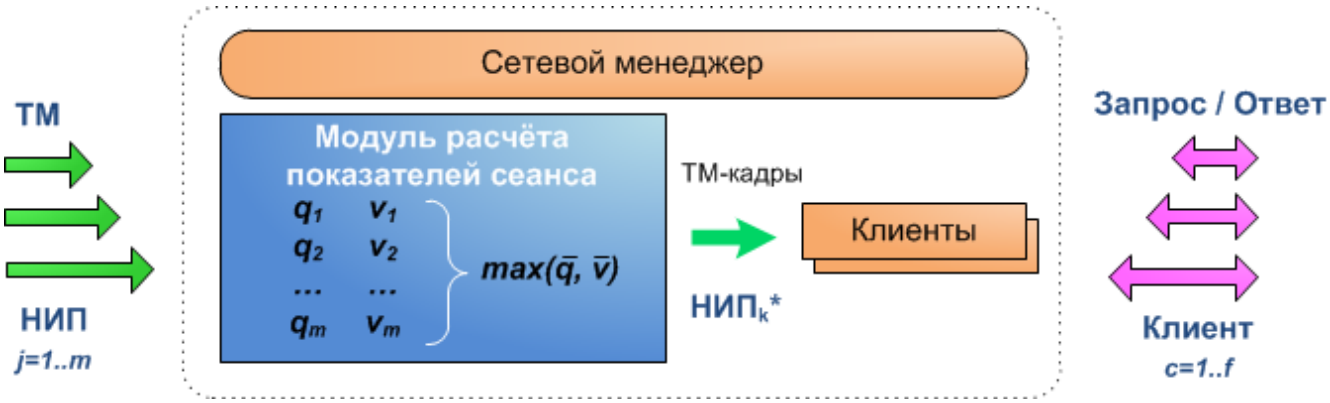

Рисунок 5.11 – Метод адаптивной передачи телеметрии потребителям на примере одного КА

Рассмотрим работу метода в случае приема телеметрической информации с одного КА через несколько земных станций. Обслуживающая подсистема принимает на вход несколько потоков телеметрии. Для каждого потока рассчитываются среднее качество q_j в виде отношения $\frac{n_{valid}}{n_{total}}$, где n_{valid} – количество достоверных кадров, n_{total} – общее количество кадров принятых в единицу времени, и средняя скорость поступления информации v_j. На основе рассчитанных показателей определяется оптимальный с точки зрения максимума среднего качества и средней скорости поток телеметрии $\max(\bar{q}, \bar{v})$, который отмечается соответствующим признаком, а параметры такой земной станции $НИП_k$* запоминаются. При подключении внутренних клиентов к обслуживающей подсистеме указывается только номер интересующего КА без указания конкретных параметров земной станции. Внутренним клиентам отправляются кадры из оптимального потока телеметрии, то есть из оптимальной на данный момент времени земной станции.

Расчёт показателей принимаемых потоков телеметрической информации q_j и v_j производится регулярно для каждого принимаемого кадра, в то время как выбор оптимального потока производится периодически с интервалом раз в три секунды.

В общем случае во время приема телеметрии с нескольких КА телеметрические потоки станций группируются по номеру КА, а при расчёте показателей среднего качества и средней скорости и поиске оптимального потока используются сформированные группы потоков, причём для каждой такой группы возвращается единственный оптимальный поток с параметрами земной станции. Остальная суть описанного выше сценария сохраняется без изменения.

Предложенный метод позволяет повысить стабильность и качество приема телеметрической информации на начальных этапах ориентации КА, а также в условиях нестабильного приёма, избавляя внутренних клиентов от необходимости ручного выбора сеанса.

5.4 Метод получения телеметрической информации о состоянии космического аппарата с разгонного блока

Одной из задач создания многопоточной системы обработки телеметрической информации является обеспечение приёма информации от различных источников телеметрии, в том числе с разгонного блока (требования Б005, Б013). Возможности информационного протокола СОТИ – СПО позволяют передавать соответствующую информацию, поэтому для её использования необходимо разработать метод приёма и обработки результатов средствами обслуживающей подсистемы с возможностью дальнейшей передачи клиентам.

Опишем последовательность шагов метода получения телеметрической информации о состоянии космического аппарата с разгонного блока с учётом специфики протокола СОТИ – СПО.

1 Сформировать комплект исходных данных на обработку параметров КА, принимаемых с разгонного блока.

Комплект исходных данных описывает типы параметров (сигнальные, аналоговые), их полное наименование, единицы измерения и адресацию внутри кадра.

2 Принять от источника таблицу соответствия условных номеров параметров их индексу и сквозному номеру

После установления сетевого соединения между СОТИ и клиентом СПО, то есть обслуживающей подсистемой, с указанием вида информации «Телеметрия с РБ» по протоколу СОТИ – СПО приложение СОТИ высылает названную таблицу с перечнем всех телеметрических параметров, участвующих в оценке состояния КА через интерфейс РБ, и указанием их условных номеров.

3 Принять результаты предварительной обработки датчиковой телеметрии

Каждый принимаемый блок информации содержит множество значений изменившихся телеметрических параметров. Значения передаются в паре с условным номером параметра. По условному номеру через таблицу соответствия условных номером параметров и индексов отыскивается полный индекс параметра и соответствующее ему описание из исходных данных. Затем в общий список значений найденного параметра добавляется очередное значение из принятого информационного блока. Используя исходные данные параметра, производится запись значения параметра в телеметрический кадр. Описанные действия повторяются для всех параметров в принятом информационном блоке, в результате формируется телеметрический кадр.

4 Отправить полученный телеметрический кадр по сети клиентам

Полученный телеметрический кадр рассылается всем внутренним клиентам телеметрии по протоколу «Взаимодействия СОТМ и внутренних клиентов» (Приложение А).

5 Отобразить обработанные значения телеметрических параметров о состоянии КА на экранах обслуживающей подсистемы и клиентов.

Стандартными средствами отображения программных комплексов производится вывод на экран обработанных значений параметров в виде формуляров, отдельных параметров или графиков.

Описанный метод обеспечивает решение одной из задач системы автоматизированного управления космическим аппаратом – анализ состояния космического аппарата средствами ЦУП на этапе вывода космического аппарата на орбиту.

5.5 Система автоматизированной подготовки отчётов о состоянии отдельных параметров и целых групп, с возможностью статистического анализа

В соответствии с дополнительным требованием Б012 система обработки телеметрической информации должна обеспечивать автоматизированное формирование отчётов о состоянии отдельных параметров и целых групп с возможностью статистического анализа, представления результатов в виде графиков, а также осуществлять их долговременное хранение в центральной БД. Решение этой задачи потребует использование модулей описания исходных данных, чтения архивов телеметрии, взаимодействия с базой данных и расчёта значений телеметрических параметров, проектирование которых успешно проведено в главе 3. Совокупность спроектированных модулей предоставляет для системы автоматизированной подготовки отчётов необходимый интерфейс унифицированного описания исходных данных, обработки и анализа телеметрической информации.

Опишем общий алгоритм функционирования системы, проиллюстрированный на рисунке 5.12:

1 Создать задачу

Задача – это задание на формирование автоматизированного отчёта, состоящее из триггерных условий и тела задания в формате XML. Тело задания может состоять из одной или нескольких секций (подзадач). Для удобства использования определено четыре типа секций: значения параметров в виде списка, графики параметров, расчёт обобщённой статистической информации по параметрам, значения параметров из отчётов бортового компьютера. Создание задачи производится извне обслуживающей подсистемы, например, средствами внутренних клиентов.

2 Записать задачу в БД

Созданная задача помещается в базу данных со статусом *Done*, то есть готова для обработки. Задача в базе данных может находиться в одном из следующих состояний: *Done* – готова/выполнена успешно, *Error* – ошибка выполнения, *Running* – запущена на выполнение, *Queued* – поставлена в очередь на исполнение.

3 Выполнять периодическое сканирование БД с целью обнаружения задач в состоянии *Done* или *Error*

4 Проверить триггерные условия для каждой выбранной задачи из предыдущего шага.

Триггерные условия включают в себя дату, время начала и периодичность выполнения задачи, которая может принимать одно из следующих значений: *Once* – однократный запуск, *Multiple* – многократный запуск, *Daily* – ежедневный запуск в указанное время, *Weekly* – еженедельный запуск в указанные дни недели и заданное время, *Monthly* – ежемесячный запуск в указанные дни месяца и заданное время. Мультивариантность триггерных условий обеспечивает достаточную гибкость в планировании формирования автоматизированных отчётов.

5 Если триггерные условия выполняются, установить для задачи состояние *Queued* и добавить её во внутреннюю очередь обработки задач. Иначе вернуться к шагу 3.

6 Выполнять периодическое сканирование внутренней очереди обработки задач. Пока очередь не пуста выбирать задачу.

7 Обработать задачу. Обновить состояние задачи

Для каждой выбранной задачи выполняется разбор XML тела задания, производится расчёт необходимых параметров. При выполнении задания загружаются исходные данные на обработку телеметрической информации в соответствии с заданным номером КА, из базы данных извлекаются телеметрические архивы на требуемом временном интервале. Используя

модуль расчёта телеметрических параметров, предоставляемый библиотекой унифицированной обработки, производится вычисление значений параметров для каждой секции задания.

8 Сохранить результат обработки в указанном формате в центральной БД

Результаты расчёта сохраняются в центральной БД в формате HTML или PDF. Задача переходит в состояние *Done* или *Error* в зависимости от результата выполнения, а её состояние обновляется в базе данных.

9 Удалить задачу из внутренней очереди

Обработанная задача удаляется из очереди обработки задач. Сканирование внутренней очереди продолжается.

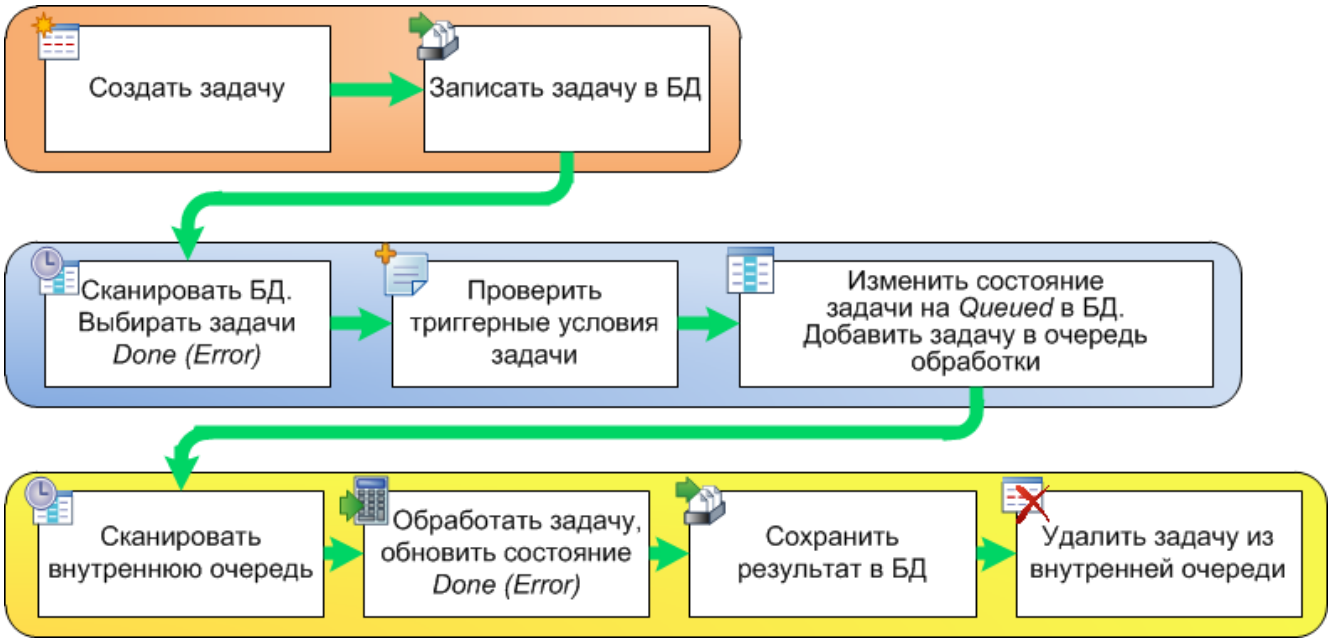

Рисунок 5.12 – Алгоритм функционирования системы автоматизированной подготовки отчётов

Последовательность шагов алгоритма, представленного на рисунке 5.12, разбита на три группы для того, чтобы обозначить асинхронность процессов каждой группы, а именно: подготовка задачи, выбор задачи и выполнение задачи. Такой способ подготовки автоматизированных отчётов позволяет добавлять задачи в систему в произвольные моменты времени, а их дальнейшая обработка будет производиться по мере высвобождения вычислительных ресурсов.

Диаграмма классов описанного алгоритма была предложена и подробно рассмотрена в 5.1.1.

5.6 Подсистема защиты информации

Защита информации является одним из важнейших требований к любой автоматизированной информационной системе. Понятие защиты информации имеет три аспекта: обеспечение требуемого уровня конфиденциальности, доступности и целостности [86].

Конфиденциальность – обеспечение доступа к закрытым сведениям только для авторизованных пользователей.

Доступность – гарантия непрерывного доступа авторизованных пользователей к данным.

Целостность – обеспечение сохранности и непротиворечивости данных.

Для обеспечения требований безопасности к системе обработки телеметрии А020, Б014 в процессе диссертационного исследования особое внимание уделено вопросам конфиденциальности циркулирующей в контуре АСУ КА телеметрической информации и результатов её обработки. Среди множества аспектов конфиденциальности были выделены защита от несанкционированного доступа к приложению и защита циркулирующей информации между обслуживающей подсистемой и внутренними, внешними клиентами.

5.6.1 Защита информации от несанкционированного доступа

Защита информации от несанкционированного доступа подразумевает идентификацию и аутентификации пользователя, а также разграничение доступа к различным участкам информации в зависимости от привилегий пользователя [86], [87].

Общая задача идентификации пользователя сводится к выяснению, имеет ли пользователь право доступа к информации в целом и, как правило, обеспечивается средствами операционной системы. Однако, учитывая повышенные требования к защите информации в ЦУП двойного назначения, идентификации пользователя средствами операционной системы оказывается не достаточно, поэтому предложена дополнительная проверка пользователя при запуске прикладной программы через ввод имени учётной записи и пароля. Контроль правильности ввода осуществляется средствами приложения через сопоставление вводимых данных с данными в центральной базе данных. С каждым авторизированным пользователем соотносится определённая группа функциональных привилегий, что позволяет обеспечивать разграничение доступа к различным видам информации внутри приложения, например: запуск и останов приложения, просмотр и изменение конфигурации, управление сеансом съёма телеметрической информации.

5.6.2 Принципы организации защиты передаваемой информации на основе системы привилегий

Важным вопросом является защита передаваемой информации между обслуживающей подсистемой и внутренними, внешними клиентами. Рассмотрим принципы организации защиты передаваемой информации на основе системы привилегий.

Для обеспечения защиты информации в обслуживающую подсистему внедрён модуль защиты информации, состоящий из двух независимых компонент: Фильтр IP-адресов и Фильтр ТМ-параметров. Каждая из этих компонент может быть отключена или задействована в любой момент времени, тем самым обеспечивая возможность изменения ограничений защиты информации «на лету». Взаимосвязь модуля защиты информации с остальными элементами обслуживающей подсистемы была приведена на рисунке 2.3.

Общая схема обработки телеметрического запроса с использованием модуля защиты передаваемой информации представлена на рисунке 5.13.

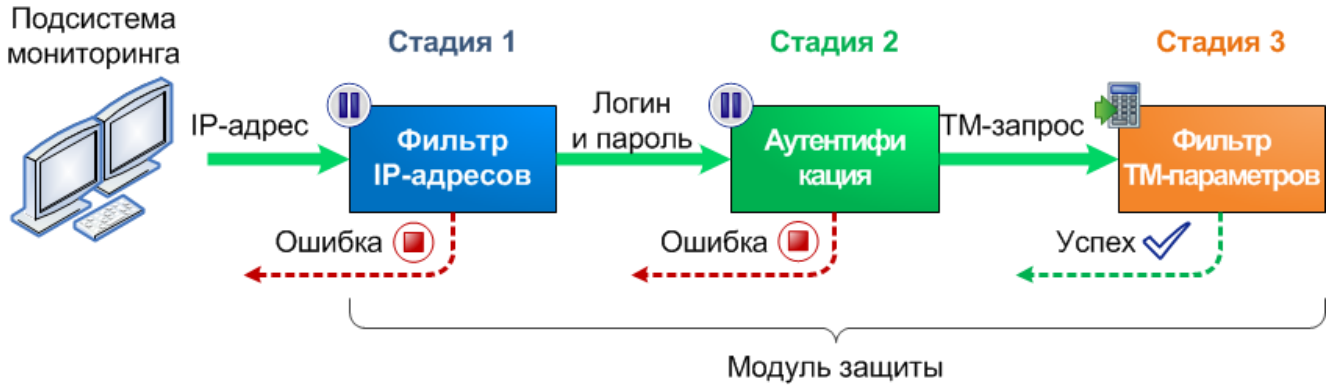

Рисунок 5.13 – Схема обработки телеметрического запроса с использованием модуля защиты передаваемой информации

Опишем алгоритм обработки запроса на получение телеметрической информации.

1 На первой стадии при поступлении запроса от подсистемы мониторинга (клиента) происходит проверка разрешения подключения с IP-адреса клиента. Если IP-адрес находится в списке разрешённых адресов, то переход к Стадии 2, иначе производится автоматический разрыв соединения с клиентом.
Примечание – Если фильтр IP-адресов выключен, алгоритм сразу переходит к Стадии 2.

2 На второй стадии производится проверка учётной записи и пароля подключившегося пользователя. Если проверка выполнена успешно, то переход к Стадии 3, иначе клиенту будет отправлена квитанция об ошибке авторизации.
Примечание – Если фильтр ТМ-параметров выключен, алгоритм сразу переходит к Стадии 3

3 На третьей стадии происходит непосредственная обработка запроса на получение телеметрии и выдача клиенту полного или сокращённого потока в зависимости от

правил фильтрации для авторизованной учётной записи, описываемых ниже при рассмотрении фильтра ТМ-параметров.

Примечание – Стадии 1, 2 выполняются только один раз при первом подключении клиента.

Фильтр IP-адресов

Это компонента блокирования/разрешения подключений клиентов с определённым IP-адресом в сети. Общий вид окна фильтра IP-адресов представлен на рисунке 5.14

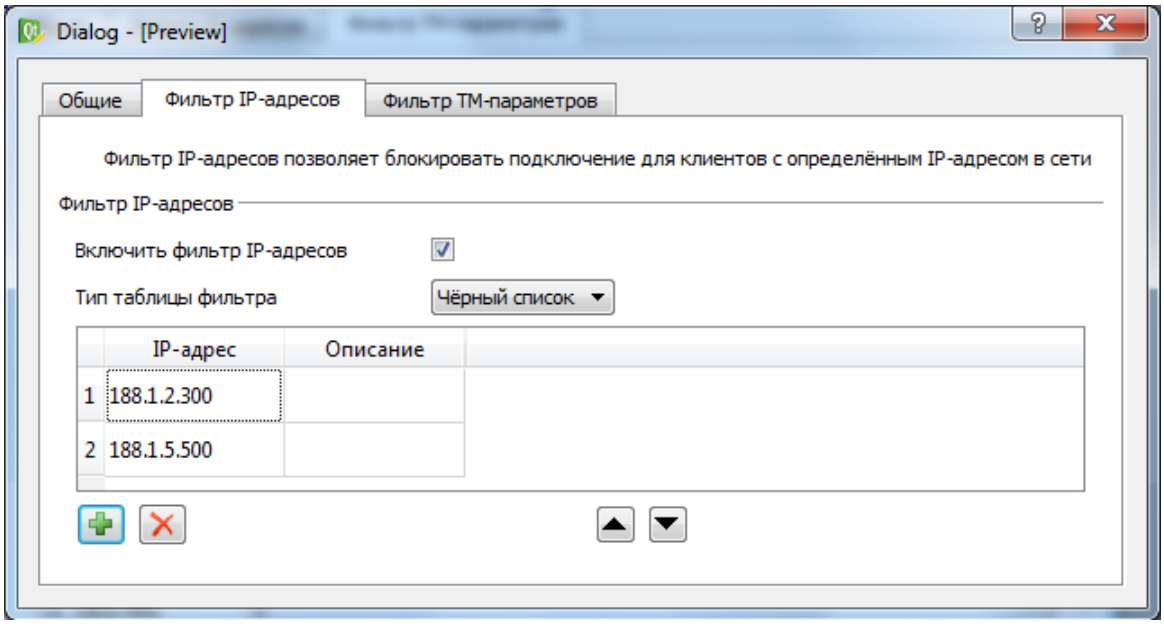

Рисунок 5.14 – Примерный вид окна «Фильтр IP-адресов»

Существует возможность отключить или задействовать фильтр IP-адресов. Также определяется тип таблицы фильтра: Чёрный список или Белый список. Тип «Чёрный список» – означает, что подключение должно быть заблокировано для всех клиентов, чьи IP-адреса перечислены в таблице. Тип «Белый список» означает, что подключение разрешено только для клиентов, чьи IP-адреса перечислены в таблице. Содержимое таблицы фильтра IP-адресов редактируется и сохраняется в базе данных.

Фильтр ТМ-параметров

Это компонента ограничения передачи телеметрического потока для клиентов, подключенных с определённой учётной записью, за счёт переопределения текущего значения телеметрического параметра некоторым «фиктивным» значением. Общий вид окна фильтра ТМ-параметров представлен на рисунке 5.15.

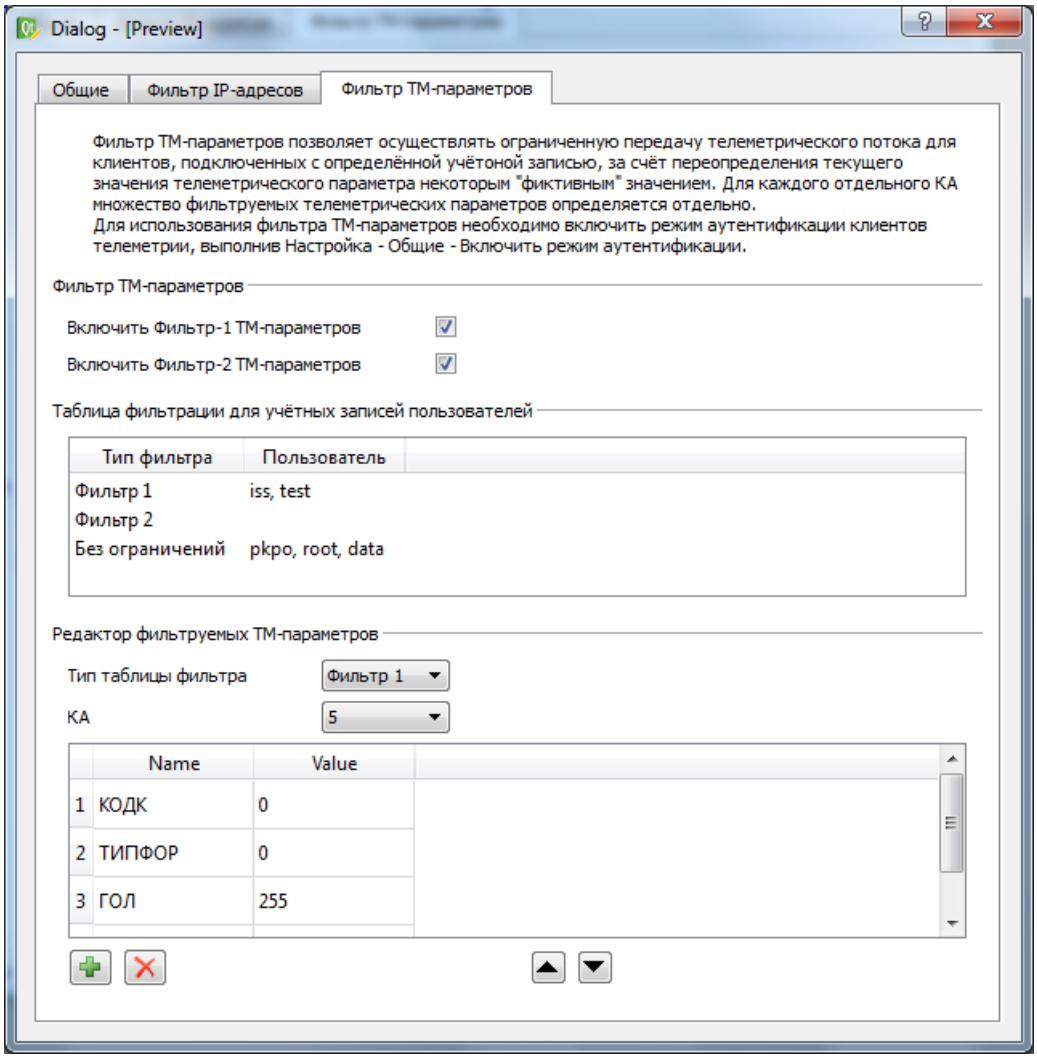

Рисунок 5.15 – Примерный вид окна «Фильтр ТМ-параметров»

Для каждого отдельного КА множество фильтруемых телеметрических параметров определяется отдельно, причём для каждого КА можно определить две различные группы фильтруемых параметров: фильтр-1 и фильтр-2 соответственно. Существует возможность отключить или задействовать любой из двух фильтров.

Фильтрация значений ТМ-параметров производится на основе учётных записей успешно подключившихся клиентов. Поэтому для проверки валидности учётных записей клиентов в обслуживающей подсистеме в обязательном порядке должен быть включен режим аутентификации клиентов телеметрии. В этом случае, любой подключаемый внутренний, внешний клиент телеметрии обязан сообщать параметры своей учётной записи перед началом сеанса приёма телеметрии в составе сетевого запроса.

Таблица фильтрации учётных записей пользователей предназначена для определения соответствия между учётной записью и типом применяемого фильтра (фильтр-1, фильтр-2). По умолчанию все пользователи прошедшие аутентификацию получают телеметрический поток

без ограничений. Для применения ограничений в передаче потока телеметрии к некоторой учётной записи необходимо добавить к соответствующему типу применяемого фильтра требуемую учётную запись. Содержимое таблицы фильтрации учётных записей сохраняется в базе данных. Каждая учётная запись может быть отнесена только к одному типу фильтра

Изменение состава фильтруемых параметров производится в соответствии с выбранным типом таблицы фильтра (фильтр-1, фильтр-2) для каждого КА в отдельности. Редактируемая таблица фильтруемых параметров состоит из двух столбцов: наименование – наименование ТМ-параметра, значение – присваиваемое значение в виде целочисленной константы. Содержимое таблицы фильтруемых ТМ-параметров сохраняется в базе данных в таблице.

5.7 Проектирование реляционной базы данных

Применение базы данных при построении многопоточной системы обработки телеметрии обусловлено с одной стороны необходимостью обеспечения централизованного хранения телеметрической информации (требование Б007), с другой повышенными требованиями к защите и надёжности хранимой информации. Кроме того средствами системы управления базами данных автоматизировано решаются задачи аудита изменений и восстановления базы данных после сбоев.

Анализ предметной области позволяет обозначить следующие виды информации, которые целесообразно хранить в базе данных:

- исходные данные на обработку телеметрической информации;
- архивы телеметрической информации;
- входные и выходные данные системы подготовки автоматизированных отчётов о состоянии телеметрических параметров;
- конфигурация для модуля защиты о фильтруемых IP-адресах, ТМ-параметрах;
- адреса различных источников телеметрии.

В результате проектирования базы данных обработки телеметрической информации была разработана ER-диаграмма, представленная на рисунке 5.16

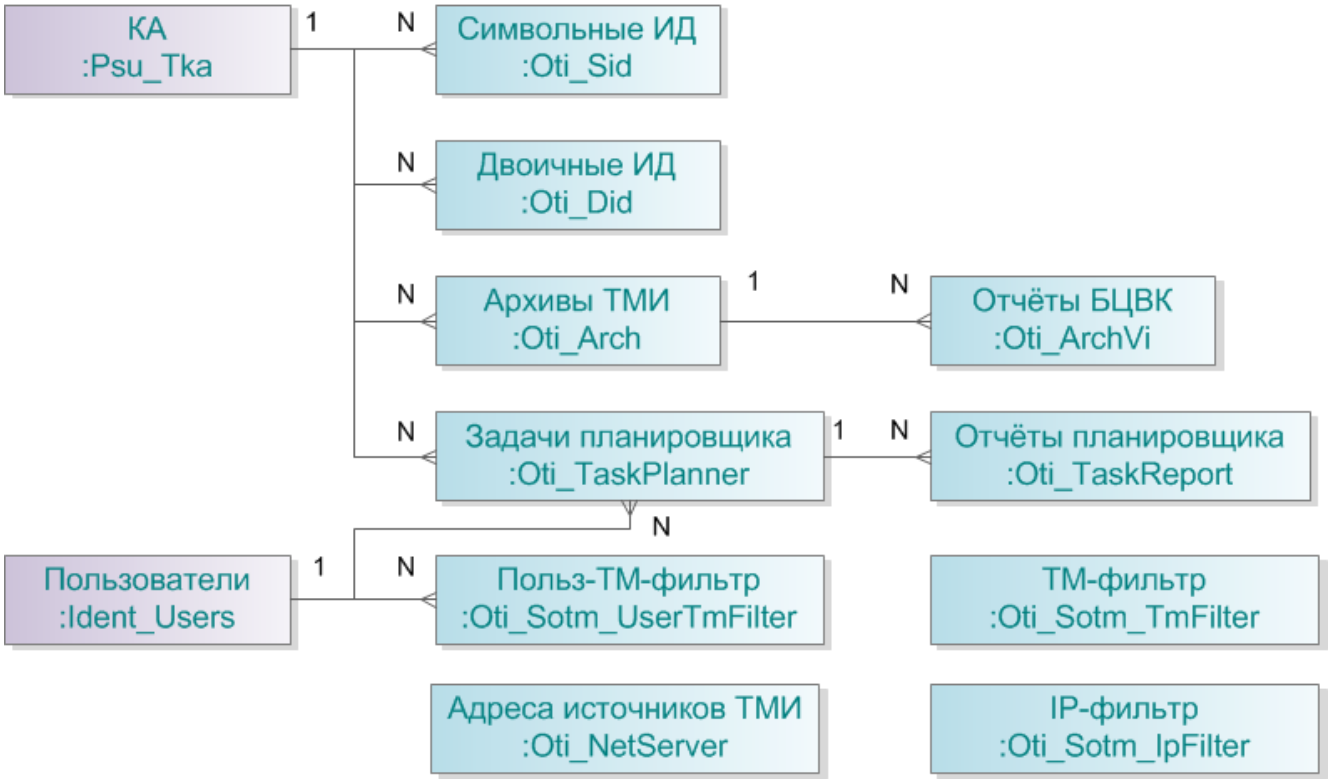

Рисунок 5.16 – ER-диаграмма базы данных обработки телеметрической информации

Множество сущностей базы данных обработки телеметрической информации являются подмножеством всех сущностей базы данных ЦУП КА, поэтому для наглядности на представленной диаграмме синим цветом обозначены сущности, участвующие в задачах обработки телеметрической информации, а фиолетовым прочие служебные сущности центральной базы данных. К выделенным сущностям базы данных обработки телеметрической информации относятся:

- *Oti_Sid, Oti_Did* – комплект символьных и двоичных исходных данных по КА. Их централизованное хранение обеспечивая обслуживающую подсистему и внутренних клиентов единым актуальным комплектом исходных данных. Одному КА соответствует несколько записей символьных и двоичных исходных данных;
- *Oti_Arch* – множество архивов телеметрической информации по КА;
- *Oti_ArchVi* – множество отчётов бортового компьютера, относящихся к определённому телеметрическому архиву;
- *Oti_TaskPlanner* – множество запланированных задач для системы автоматизированной подготовки отчётов о состоянии ТМ-параметров. Каждая задача имеет ссылку на КА (сущность *Psu_Tka*) и идентификатор пользователя (сущность *User_ID*);
- *Oti_TaskReport* – множество отчётов о выполнении задач системы автоматизированной подготовки отчётов. Поскольку каждая запланированная задача

может выполняться многократно, между сущностями *Oti_TaskPlanner* и *Oti_TaskReport* имеется отношение один ко многим;

- *Oti_Sotm_UserTmFilter* – сущность модуля защиты обслуживающей подсистемы, используется для фильтрации учётных записей пользователей. Каждая учётная запись может быть отнесена только к одному типу фильтра: фильтр-1 или фильтр-2. Для учётных записей из сущности *Ident_Users*, отсутствующих в таблице *Oti_Sotm_UserTmFilter*, предоставляется поток телеметрии без ограничений;
- *Oti_Sotm_TmFilter* – сущность модуля защиты обслуживающей подсистемы, используется для определения перечня фильтруемых ТМ-параметров;
- *Oti_Sotm_IpFilter* – сущность модуля защиты обслуживающей подсистем, используется для определения списка фильтруемых IP-адресов;
- *Oti_NetServer* – список IP-адресов, номеров портов различных источников телеметрии.

К служебным сущностям базы данных, используемых на диаграмме 5.16, относятся:

- *Psu_Tka* – список космических аппаратов орбитальной группировки с указанием их ключевых характеристик: дата старта, тип бортового компьютера, номер плоскости и т.д
- *Ident_users* – список разрешённых учётных записей специализированного программного обеспечения в ЦУП КА.

Подробное описание построения реляционной модели базы данных обработки телеметрической информации приведено в приложении Г.

5.8 Обоснование выбора инструментария разработки

Особенности проектируемой системы таковы, что согласно требованию А021, она должна функционировать в двух различных операционных системах: Windows XP (и выше) и МСВС 3.0. Данное требование обусловлено этапностью перехода от существующего центра управления системой Глонасс (ЦУС-У) к модернизированному центру (ЦУС-УМ). На первом этапе все программные комплексы функционируют в операционной системе Windows, используемая СУБД Oracle 7.0. На втором этапе специальное программное обеспечение должно функционировать в ОС МСВС 3.0 и взаимодействовать с СУБД Линтер-ВС.

ОС МСВС (Мобильная Система Вооруженных Сил) — отечественная защищённая операционная система общего назначения на основе GNU/Linux. Разработана Всероссийским научно-исследовательским институтом автоматизации управления в непромышленной сфере

имени В. В. Соломатина (ВНИИНС). Принята на снабжение в Вооружённые силы Российской Федерации в 2002 году [88].

Используется для построения на ее основе защищенных информационных систем. Обладает развитыми средствами управления доступом пользователей к ресурсам ОС, включающими механизмы мандатного, дискреционного и ролевого управления доступом [89].

Функционирует на аппаратных платформах Intel, SPARC (Эльбрус-90микро), IBM S390 и MIPS (комплексы серии Багет производства компании Корунд-М), поддерживает многопроцессорные конфигурации (SMP) [88].

ОС МСВС 3.0 сертифицирована в системе сертификации средств защиты информации по требованиям безопасности информации Министерства Обороны РФ по:

- 2 классу защищенности информации от НСД;
- 1 уровню классификации контроля отсутствия недекларированных возможностей.

СУБД «Линтер-ВС» - это объектно-реляционная защищенная система управления базами данных, функционирующая под управлением ОС МСВС, основанная на ядре PostgreSQL. В отличие от открытой PostgreSQL, СУБД «Линтер-ВС» имеет расширения для обеспечения защиты от НСД (возможность работы с мандатными метками и категориями доступа), и имеет сертификаты по классам защищенности: по 3 классу защищенности от НСД, по 2 уровню контроля недекларированных возможностей. СУБД «Линтер-ВС» 7.0 содержит средства встроенные средства репликации [88].

Учитывая особенности заданных операционных систем и СУБД, наиболее оптимальным средством разработки системы многопоточной обработки телеметрической информации является кроссплатформенный инструмент Qt.

Qt – кросс-платформенный инструментарий для разработки программного обеспечения. Этот инструментарий создан компанией Trolltech и в данный момент принадлежит компании Digia. Qt – это совокупность кросс-платформенной библиотеки классов, реализованной на языке C++, и ряда дополнительных инструментальных средств, включающих Meta Object Compiler (MOC) – объектный предкомпилятор, User Interface Compiler (UIC) – компилятор пользовательских интерфейсов, qmake – средство управления сборкой проектов [90].

Поддерживаются операционные системы Microsoft Windows, Linux, MacOS, а также встраиваемые операционные системы Embedded Linux, Windows CE, Symbian.

В состав Qt входят следующие основные группы классов [91]:

- классы, обеспечивающие разработку оконного графического интерфейса пользователя, включая все основные управляющие примитивы;

- классы, реализующие работу с потоками, объектами синхронизации процессов/потоков;
- классы для работы с 2-х и 3-х мерной графикой, классы реализующие поддержку некоторых графических форматов хранения;
- классы сетевого взаимодействия по протоколам TCP, UDP;
- классы для доступа к базам данных. Поддержка различных типов баз данных через предоставляемы драйвера для SQLite, Postgree, Oracle, ODBC;
- классы для работы с XML;
- классы дополнительных расширений для приложения. Предоставляемая система расширений (plug-ins) позволяет создавать модули, расширяющие функциональные возможности приложений.

Qt 4.x поставляются в составе дистрибутива МСВС 3.0, обеспечивая возможность разработки в интегрированных средах KDevelop, Eclipse. Для ОС Windows имеются средства, позволяющие интегрировать Qt в среду разработки: uic, moc –компиляторы и QtDesigner. При этом возможна интеграция Qt 4.x — в Microsoft Visual Studio 2008/2010/2012. OpenSource Qt 4.x для Windows может быть интегрирована в IDE Eclipse с подключенным компилятором mingw-gcc, а также использоваться совместно с кроссплатформенной IDE QtCreator. Библиотеки для использования могут быть откомпилированы любым компилятором C++, имеющимся в ОС, например для Windows – Microsoft Visual C++ или mingw-gcc, для МСВС 3.0 – gcc.

В библиотеке реализовано автоматическое удаление объектов, являющихся элементами графического интерфейса пользователя [90], [91]. Механизм реализован следующим образом: любой объект Qt является потомком QObject, в состав которого входят средства хранения и позиционирования списка потомков, то есть объектов, при создании которых этот объект указан как parent. Следовательно, при удалении корневого объекта возможно удаление всего дерева объектов-потомков. Новая концепция ведения межобъектного взаимодействия, именуемая «сигналы и слоты», полностью заменяет былую, не вполне надежную модель обратных вызовов. Подробная документация Qt с множеством готовых наглядных примеров упрощает процесс освоения возможностей и особенностей этой богатой библиотеки [91].

В Qt может быть использовано кросс-платформенное средство управления сборкой проектов qmake, посредством которого из .pro-файлов генерируются файлы makefile для конкретной платформы с конкретными компиляторами и компоновщиками.

Описанные достоинства Qt позволили выбрать её в качестве полноценного инструмента разработки заданной системы многопоточной обработки телеметрической информации для последующей эксплуатации как в среде ОС Windows так и ОС МСВС 3.0.

5.9 Выводы по главе 5

1 Произведено проектирование объектно-ориентированной модели обслуживающей подсистемы. Построенные диаграммы классов, объектов, взаимодействия и диаграмма компонентов определяют поведение системы в статическом и динамическом аспектах.

2 Предложен метод проведения автоматизированного сеанса съёма телеметрической информации

3 Предложен метод адаптивной передачи телеметрической информации потребителям, который позволит повысить эффективность проведения сеансов

4 Разработан метод получения телеметрической информации о состоянии космического аппарата с разгонного блока, позволяющий производить оценку состояния космического аппарата на этапе выведения.

5 Предложена структура и принципы функционирования подсистемы защиты информации, состоящей из модуля защиты от несанкционированного доступа и модуля защиты информации на основе системы привилегий и учётных записей пользователей

6 Произведено проектирование реляционной базы данных, обеспечивающей централизованное хранение архивов телеметрической информации, исходных данных на обработку, а также общих сетевых параметров подключения.

7 Выбран программный инструмент для разработки, отладки и тестирования обслуживающей подсистемы с учётом требования кроссплатформенности программного обеспечения

Глава 6. Практическая реализация разработанной системы в составе АСУ ОГ Глонасс

В настоящей главе приводятся характеристики автоматизированной системы приёма, обработки и анализа телеметрической информации системы ОГ КА Глонасс. Рассматриваются результаты практической реализации разработанных методов и моделей для системы Глонасс. Дан сравнительный анализ ключевых функций старой и новой систем приёма телеметрической информации. Для придания большей информативности результатам анализа множество исходных требований сгруппированы по шести базовым характеристикам качества программного обеспечения в соответствии с рекомендациями группы стандартов ISO 9126.

6.1 Характеристики автоматизированной системы приёма, обработки и анализа телеметрической информации орбитальной группировки космических аппаратов Глонасс

Глобальная навигационная спутниковая система (ГЛОНАСС) предназначена для оперативного навигационно-временного обеспечения неограниченного числа пользователей наземного, морского, воздушного и космического базирования. Спутники ГЛОНАСС находятся на средневысотной круговой орбите на высоте 19100 км с наклонением 64,8° и периодом 11 часов 15 минут. Такая орбита оптимальна для использования в высоких широтах (северных и южных полярных регионах), где сигнал GPS ловится плохо. Спутниковая группировка развёрнута в трех орбитальных плоскостях, с 8 равномерно распределёнными спутниками в каждой. Для обеспечения глобального покрытия необходимы 24 спутника, в то время как для покрытия территории России необходимы 18 спутников [92].

Общее количество параметров для КА типа Глонасс-М составляет около 8400 телеметрических параметров, из которых порядка 3100 ТМ-параметра обрабатываются в режиме непосредственной передачи, для КА типа ГлонассК – около 14700 телеметрических параметра, из которых порядка 3800 ТМ-параметра обрабатываются в режиме непосредственной передачи [92].

Полный состав сигнальных ТМ-параметров [93], [94] передаётся за 2 телеметрических кадра, аналоговых ТМ-параметров – за 3-4 телеметрических кадра, температурных ТМ-параметров – за 8 телеметрических кадров, программных ТМ-параметров – за 6 телеметрических кадров. Для защиты от искажений при передаче телеметрической информации каждый телеметрический кадр содержит контрольную сумму БАТС.

Передача отчёта БЦВК максимальной длины (8 КБ) занимает 22 телеметрических кадра. Содержимое отчёта БЦВК защищено от искажений посредством контрольной суммы отчёта. Защита от недостоверности отчёта воспроизведения информации телесигнализации (ВИТС) обеспечивается только контрольной суммой БАТС [93], [95].

Во исполнение опытно-конструкторской работы по модернизации технических и программных средств на 2002-2011 годы производилось переоснащение наземного сегмента системы Глонасс.

6.2 Сравнительный анализ ключевых функций старой и новой систем автоматизированного приёма, обработки и анализа телеметрической информации

Для удовлетворения тактико-технических требований в части обработки полных потоков телеметрической информации ОГ Глонасс, определяемых техническим заданием на НКУ, была спроектирована новая система приёма, обработки и анализа телеметрической информации в ЦУП КА и разработана обсуживающая система, построенная на изложенных в ходе диссертационного исследования методах, принципах и алгоритмах. Общая схема разработанной системы приёма, обработки и анализа телеметрической информации была представлена на рисунке 2.2.

Сравнительные функции существующей системы обработки телеметрии и многопоточной системы, сгруппированные по характеристикам качества группы стандартов ISO 9126, приведены в таблице 6.1:

Таблица 6.1 – Сравнение функций двух систем обработки телеметрии

Описание функции	**Индекс**	**Существующая система**	**Многопоточная система**
1 Функциональность			
• *Функциональная исправность*			
Хранение результатов управления и контроля в течение всего жизненного цикла КА	A001	+	+
Непрерывный автоматизированный контроль, диагностика и отображение состояния КА	A002	+	+
Одновременный мониторинг не менее чем 4 КА	A003	+ (через три доп. раб.м.)	+
Распознавание, выделение, проверка на достоверность и запись ТМИ в оперативный архив	A005	+	+
Допусковый контроль параметров	A006	+	+

Продолжение таблицы 6.1

Отображение значений заявленных оператором ТМ параметров и состояния систем спутника, в том числе в виде мнемосхем	А007	+	+
Отображение каналов псевдокадра ТМИ в необработанном виде	А008	+	+
Построение графиков поведения параметров	А009	+	+
Отображение блоков информации (формуляров)	А010	+	+
Отображение отчетной информации БЦВК с полной обработкой параметров во фразах и событиях	А011	+	+
Сбор, накопление и систематизация информации о состоянии спутника по результатам обработки ТМИ и отчетов БЦВК	А012	+	+
Непрерывный автоматизированный обобщенный контроль состояния спутника по ТМИ, с использованием звуковой и цветовой сигнализации в случае выхода любого параметра за пределы допуска	А013	+	+
Предоставление имитационного режима функционирования (для отладочных и тренировочных целей) с использованием динамического программного имитатора	А014	+	+
Выборку из архива любой группы ТМ параметров на заданном пользователем интервале времени (в пределах срока хранения) и отображение в заданном виде	А015	+	+
Параллельный приём и обработка телеметрической информации не менее чем с 8 НИП	Б004	–	+
Сжатие архивов телеметрической информации в реальном масштабе времени	Б008	–	+
Приём телеметрической информации с разгонного блока на участке выведения	Б013	–	+
• *Функциональная совместимость*			
Взаимодействие с САО-Ц, СОТИ для получения потоков ТМИ с наземных станций управления	А004	+	+
Обслуживание заявок внутренних и внешних, по отношению к ЦУП, потребителей на получение ТМИ по согласованным протоколам	А016	–	+
Обеспечение заданной степени обобщения выходной информации для передачи в смежные модули управления	Б003	–	+
Одновременный приём и обработка телеметрической информации форматов САО-Ц, СОТИ, ЕЦУП РБ	Б005	–	+
Обеспечение активного (приём информации в ЦУП от САО-Ц, СОТИ) и пассивного (приём информации от в РЦУП) режимов обработки телеметрии	Б009	–	+

Продолжение таблицы 6.1

• *Безопасность*			
Использование средств парольной защиты	А020	–	+
Система защиты от несанкционированного доступа	Б014	–	+
2 Надёжность			
Наличие средств контроля входной информации	А017	+	+
Централизованное хранение архивов телеметрической информации	Б007	–	+
Возможность автономного функционирования без базы данных (в случае нештатного разрыва связи с СУБД)	Б010	–	+
Система восстановления после программных сбоев	Б015	–	+
Устойчивость к отказам (работоспособность)	А023	низкая	высокая
3 Удобство использования			
Предварительная подготовка и отладка исходных данных для обработки ТМИ, используемых при летной эксплуатации КА	А019	+	+
Наличие электронной справочной системы	А022	–	+
Обеспечение автоматического проведения сеанса съёма телеметрии в соответствии с планом	Б006	–	+
Автоматизированное формирование отчётов о состоянии отдельных параметров или целых групп	Б012	–	+
Протоколирование событий	Б017	+	+
Расширенное протоколирование событий по типам: сообщения базы данных, сетевого взаимодействия, служебные события, действия оператора	Б018	–	+
4 Эффективность			
Адаптивный приём-передача потока телеметрических кадров клиентам телеметрии	Б011	–	+
Наличие методов автоматизированной организации сеанса съёма телеметрии	Б019	–	+
Использование преимуществ многоядерной аппаратной конфигурации	Б020	–	+
Среднее время организации единичного сеанса $t_н$, сек		28	0,215
Среднее время перехода между сеансами $t_п$, сек		52	1,087
5 Удобство сопровождения			
Масштабируемость СПО ОТИ	Б001	низкая	высокая
Минимизация степени участия человека в управлении системой	Б002	низкая	высокая
Наличие отдельных модулей		–	+
Наличие системы автоматического формирования и доставки отчётов об ошибках	Б021	–	+
Использование системы отслеживания ошибок	Б022	–	+

Продолжение таблицы 6.1

6 Портативность			
Независимость функциональных задач СПО от места их выполнения в сети рабочих станций ЦУП (РЦУП)	А018	+	+
Функционирование программных средств в среде ОС Windows XP и выше, ОС МСВС 3.0	А021	только Windows	+
Наличие конфигурируемой программы установки дистрибутива	Б016	–	+

В части функциональных характеристик новая многопоточная система имеет 9 дополнительных характеристик по сравнению с существующей системой. Параллельный приём и обработка телеметрической информации не менее чем с 8 НИП стали возможными благодаря тщательному проектированию архитектуры обслуживающей подсистемы и сетевого менеджера в её составе. Причём многопоточный приём и обработка обеспечиваются одним рабочим местом с расположенным на нём СОТМ, а не множеством дублирующих рабочих мест, в отличие от существующей системы. Благодаря заложенной расширяемости сетевого менеджера стало возможным создание адаптеров для приёма телеметрической информации от различных источников, включая ЕЦУП РБ. Кроме обеспечения обработки телеметрии внутри ЦУП новая многопоточная система также предоставляет интерфейс для взаимодействия с внешними по отношению к ЦУП потребителями. Особое внимание в новой системе уделено вопросам обеспечения безопасности, для решения которых внедрены средства парольной защиты и система защиты от несанкционированного доступа.

Улучшение характеристик надёжности построенной системы обусловлено рядом усовершенствований. Во-первых, организовано центральное хранилище архивов телеметрической информации в БД ЦУП, для этой цели была спроектирована база данных. Во-вторых, внедрена система восстановления после программных сбоев. Устойчивость к отказам (работоспособность) системы повышена за счёт следования предложенным принципам обеспечения и контроля качества таким как: применение UML для описания проектных требований, следование соглашений по оформлению кода, использование средств статического и динамического анализа, тщательное модульное и системное тестирование и др.

Используя сведения БД ЦУП о запланированных сеансах приёма телеметрии, стала возможной реализация автоматического проведения сеанса средствами обслуживающей подсистемы. Внедрение системы автоматизированной подготовки отчётов о состоянии отдельных параметров и целых групп, а также расширенное протоколирование событий базы данных, сетевого взаимодействия, действий оператора повысили удобство использования

системой в целом. Дополнительно разработанная электронная справочная система оказывает необходимую поддержку при работе операторов.

Характеристики эффективности системы были улучшены за счёт внедрения метода автоматизированной организации сеанса съёма телеметрии. Практическая реализация предложенного метода продемонстрировала, что среднее время организации единичного сеанса и перехода между сеансами сократилось, то есть $t_н > t`_н$, $t_п > t`_п$. Прирост производительности при этом составил $\frac{t_н}{t`_н} \cong 130$, $\frac{t_п}{t`_п} \cong 47$ раз соответственно. Таким образом, достигнутый результат существенно повысил оперативность анализа. Также эффективность новой системы улучшена за счёт применения метода адаптивного приёма-передачи потока телеметрических кадров клиентам, позволяющего в условиях приёма телеметрической информации по одному КА через несколько земных станций автоматически выбирать наиболее качественный с точки зрения достоверности и скорости поступления кадров поток и перенаправлять его клиентам. Кроме того, для наиболее эффективного использования вычислительных ресурсов современных аппаратных комплексов задачи вычислительных и сетевых модулей, сформированных в результате тщательного проектирования архитектуры системы, вынесены в отдельные обслуживающие потоки, которые равномерно распределяются средствами ОС между всеми доступными ядрами вычислительного кластера.

Поскольку в основе новой многопоточной системы лежит библиотека унифицированного описания исходных данных, обработки и анализа телеметрической информации построенная система является легко масштабируемой. На базе спроектированных модулей могут быть построены дополнительные программные комплексы, при этом обеспечивается повторное использование богатых функциональных возможностей библиотеки. Одним из средств повышения качества новой системы стало внедрение системы автоматического формирования и доставки отчётов об ошибках. Использование системы отслеживания ошибок – это другой дополнительный инструмент обратной связи с разработчиком, который также облегчает сопровождение многопоточной системы.

При выборе инструментария разработки решающим фактором стало требование функционирования под ОС различной архитектуры. Выбранная библиотека Qt позволила с успехом реализовать кроссплатформенную многопоточную систему. Для облегчения процедуры развёртывания новой системы на рабочих местах или на серверном оборудовании была написана конфигурируемая программа установки дистрибутива. Названные характеристики, безусловно, демонстрируют бóльшую портативность новой многопоточной системы в сравнении с существующей.

6.3 Результаты внедрения автоматизированной многопоточной системы приёма, обработки и анализа телеметрической информации

Разработанная многопоточная система внедрена и успешно эксплуатируется при решении задачи автоматизированного управления орбитальными группировками в ЦУП ОГ КА Глонасс, Гео-ИК-2 (г.Краснознаменск, Московская область) в интересах Министерства обороны Российской Федерации (МО РФ), а также в ЦУП ОГ КА Экспресс-АМ, Экспресс-АТ (г.Москва, г.Железногорск, Красноярского края) в интересах Министерства связи и массовых коммуникаций РФ, Луч-5В (г.Королёв, Московская область) в интересах Роскосмос, о чём свидетельствуют соответствующие акты внедрения, приведённые в приложении Д.

Специалисты ЦУП КА по обработке и анализу телеметрической информации дают высокую оценку разработанной многопоточную системе, отмечают повышенную надёжность, расширенные функциональные возможности, улучшение удобства использования, увеличение производительности расчётов и сокращение количества регулярных ручных операций, в результате чего снизилось число ошибок операторов и увеличилась оперативность организации сеанса и проведения анализа телеметрической информации. Такая оценка позволяет с уверенностью утверждать о качественном улучшении системы обработки телеметрической информации и повышении эффективности принятия решений при управлении АСУ ОГ КА в целом.

6.4 Выводы по главе 6

1 Приведены основные характеристики систем обработки телеметрической информации ОГ Глонасс.

2 Рассмотрены результаты практической реализации разработанных методов и моделей для системы Глонасс.

3 Проведён сравнительный анализ ключевых функций старой и новой системы приёма телеметрической информации. Прокомментированы достигнутые усовершенствования. По каждой из характеристик качества группы стандартов ISO 9126 (функциональность, надёжность, удобство использования, эффективность, удобство сопровождения, портативность) получены значительные улучшения показателей, что в целом характеризует построенную многопоточную систему как более качественную.

4 Разработанная многопоточная система внедрена и успешно эксплуатируется при решении задачи автоматизированного управления орбитальными группировками в ЦУП ОГ КА Глонасс, Гео-ИК-2 в интересах МО РФ, Экспресс-АМ, Экспресс-АТ в интересах Минкомсвязи РФ, Луч-5В в интересах Роскосмос.

Заключение

В диссертационной работе разработаны и исследованы методы построения современной автоматизированной системы многопоточного приёма, обработки и анализа телеметрической информации. Результаты работы могут быть сформулированы в виде следующих теоретических и практических выводов:

1 Проведено исследование технологических процессов существующей автоматизированной системы приёма, обработки и анализа телеметрической информации в составе АСУ ОГ КА.

2 Выявлены ключевые функции системы приёма телеметрической информации в составе АСУ ОГ КА и проведена их классификация. Предложена архитектура системы телеметрической информации, обеспечивающая ключевые функции системы и позволяющая осуществлять автоматизированный многопоточный приём, обработку и анализ телеметрической информации.

3 Предложен унифицированный сетевой интерфейс для доступа к источникам телеметрии, включающий взаимодействие с САО-Ц, СОТИ, ЕЦУП РБ и позволяющий расширять функции телеметрической системы при введении новых источников телеметрии.

4 Разработан набор протоколов взаимодействия между сервером обработки телеметрии и внутренними / внешними для АСУ ОГ КА клиентами, описывающий значения телеметрических параметров и позволяющий передавать состояния бортовых систем КА в унифицированном виде.

5 Спроектирована и разработана библиотека объектно-ориентированных модулей, включающая унифицированные средства описания исходных данных, алгоритмы и методы обработки и анализа телеметрической информации и позволяющая более эффективно организовывать программное обеспечение новой телеметрической системы.

6 Сформулированы принципы обеспечения и контроля качества, положенные в основу создания качественной автоматизированной системы многопоточного приёма, обработки и анализа телеметрической информации.

7 Предложена и обоснована архитектура обслуживающей подсистемы, включающая поддержку многопоточного приёма, обработки и анализа телеметрической информации и позволяющая обеспечить множественный санкционированный доступ

клиентов обработки и анализа телеметрии и повысить степень доступности телеметрической информации.

8 Разработано кроссплатформенное программное обеспечение сервера многопоточного приёма, обработки и анализа телеметрической информации, функционирующее под управлением ОС Windows, Linux и включающее метод адаптивной передачи телеметрии потребителям и подсистему защиты информации.

9 Разработана реляционная модель базы данных для централизованного хранения архивов телеметрической информации.

10 Результаты исследования и созданное на его основе специальное программное обеспечение сервера обработки телеметрии внедрено, что подтверждается актами внедрения, и используются следующими организациями:

- центры управления полётами системами Глонасс, Гео-ИК-2 (г.Краснознаменск, Московская область);
- центры управления полётами системами Экспресс-АМ, Экспресс-АТ (г.Москва, г.Железногорск, Красноярского края);
- центр управления полётом системы Луч-5В (г.Королёв, Московская область)
- информационно-вычислительный комплекс генерального конструктора Открытого акционерного общества «Информационные спутниковые системы» имени академика М.Ф. Решетнёва (г.Железногорск, Красноярского края).

Использование результатов исследования для построения системы многопоточного приёма телеметрической информации позволит повысить степень доступности телеметрической информации, обеспечить гибкую модульность программного обеспечения, упростить процесс формирования кроссплатформенного информационно-телеметрического обеспечения, обеспечить полноценное взаимодействие внутри системы и с внешними абонентами, а также повысить качество системы приёма телеметрии и эффективность функционирования АСУ ОГ КА в целом.

Список сокращений и условных обозначений

BSD – Berkeley Software Distribution - система распространения программного обеспечения в исходных кодах

CrashRpt – библиотека для сбора и доставки отчетов об ошибке в операционной системе Windows

HTML – hypertext markup language

IDE – Integrated Development Environment

IEEE – Institute of Electrical and Electronics Engineers

IP – Internet Protocol

ISO – International Organization for Standardization

PDF – portable document format

SQL – Structured query language. Структурированный язык запросов

TCP – Transmission Control Protocol

UML – Unified Modeling Language. Унифицированный язык моделирования

АОК – автоматизированный обобщённый контроль

АСУ – автоматизированная система управления

БАТС – бортовая аппаратура сигнализации

БД – база данных

БКУ – бортовой комплекс управления

БРК – бортовой радиотехнический комплекс

БЦВК – бортовой цифровой вычислительный комплекс

ЕЦУП РБ – единый центр управления разгонным блоком

ИД – исходные данные

КА – космический аппарат

КИС – командно-измерительная система

КОИ – комплекс обработки информации

КП ПВРТМ – комплекс проведения внесеансных работ с телеметрией

КП ПСО – комплекс программ проведения сеанса обработки

КП РМТМ – комплекс программ рабочего места обработки телеметрии

МКПИ – массивы командно-программной информации

НИП – наземный измерительный пункт

НКУ – наземный комплекс управления

ОГ – орбитальная группировка

ОКР – опытно конструкторская работа

ОС – операционная система

ПВРТМ – проведение внесеансных работ с телеметрией

ПК – программная команда

ПО – программное обеспечение

ПЭВМ – персональная электронно-вычислительная машина

РК – разовая команда

РЦУП – резервный центр управления полётами

САО-П – система автоматизированного обмена пункта

САО-Ц – система автоматизированного обмена центра управления полётами

САС – срок активного существования

СЕВ – система единого времени

СОТИ – сектор предварительной обработки телеметрической информации

СПО – специальное программное обеспечение

СПО ОТИ – специальное программное обеспечение обработки телеметрической информации

ССД – сектор сбора данных

ССПД – система связи и передачи данных

СУБД – система управления базой данных

ТМ – телеметрия

ТМИ – телеметрическая информация

ЦУП – центр управления полётами

Список литературы

1. Князькин Ю.М. Методология автоматизированного проектирования бортовых комплексов управления космических аппаратов связи, ретрансляции. – М.: МО, 1992. – 118 с.
2. Соловьев Ю.А. Системы спутниковой связи. – М.: Эко-Трендз, 2000. – 270 с.
3. Пакман Д.Н., Некрасов М.В., Антамошкин А.Н. Проблемы обработки телеметрической информации в контуре автоматизированной системы управления космическими аппаратами // Вестник Сибирского Государственного аэрокосмического университета имении академика М.Ф.Решетнева. – 2009. – № 1(22). Часть 1. – С. 4-9.
4. Назаров А.В. Современная телеметрия в теории и на практике. / А. В. Назаров, Г. И. Козырев, И. В. Шитов, В. П. Обрученков, А. В. Древин, В. Б. Краскин, С. Г. Кудряков, А. И. Петров, С. М. Соколов, В. Л. Якимов, А. И. Лоскутов. – СПб.: Наука и Техника, 2007. – 672 с.
5. Эскизный проект. Построение основного и резервного центров управления полетом орбитальной группировкой космических аппаратов гражданских спутниковых систем связи и вещания государственного назначения. Книга 1. Организационно-техническое построение ЦУП. – Железногорск. – 2002. – 109 с.
6. Космический комплекс «Енисей-А1». Эскизный проект. Часть 1. Космический аппарат «Енисей-А1». Книга 3. Наземный комплекс управления. – Железногорск. – 2012. – 162 с.
7. Спутник «Amos-5». Технический проект. – Железногорск. – 2010. – 92 с.
8. Соловьев В.А., Лысенко Л.Н., Любинский В.Е. Управление космическими полётами. Часть 1. – М.: МГТУ им. Н.Э. Баумана, 2009. – 475 с.
9. Космический аппарат «ЯМАЛ-300К». Инструкция по обработке информации. Часть первая. Информационное обеспечение полета. – Железногорск. – 2011. – 19 с.
10. Изделие 14Ф113. Инструкция по обработке информации. Часть первая. Общие сведения по обработке информации. – Железногорск. – 2011. – 51 с.
11. Космический аппарат «ЯМАЛ-300К». Инструкция по обработке информации. Часть вторая. Методы обработки телеметрической информации. – Железногорск. – 2010. – 18 с.
12. Изделие 14Ф143. Инструкция по обработке информации. Часть первая. Общие сведения по обработке информации. – Железногорск. – 2010. – 18 с.

13. Оуден К., Гино Б. Измерение времени. Основы GPS. – М.: Техносфера, 2002. – 500 с.
14. Шебшаевич В.С. Сетевые спутниковые радионавигационные системы. 2-е изд. – М.: Радио и связь, 1993. – 408 с.
15. Space Segment // Official U.S. Government information about the GPS and related topics. URL: http://www.gps.gov/systems/gps/space/ (дата обращения: 13.01.2013).
16. Control Segment Elements // Official U.S. Government information about the GPS and related topics. URL: http://www.gps.gov/systems/gps/control/#elements (дата обращения: 14.02.2013).
17. Galileo Navigation // ESA. URL: http://www.esa.int/Our_Activities/Navigation/The_future_-_Galileo/What_is_Galileo (дата обращения: 20.01.2013).
18. Galileo FOC Factsheet // ESA. URL: http://download.esa.int/docs/Galileo_IOV_Launch/FOC_factsheet_20111003.pdf (дата обращения: 20.01.2013).
19. Galileo Factsheet // ESA. URL: http://download.esa.int/docs/Galileo_IOV_Launch/Galileo_factsheet_2012.pdf (дата обращения: 20.01.2013).
20. Галилео // Wikipedia. URL: http://ru.wikipedia.org/wiki/Галилео_(спутниковая_система_навигации) (дата обращения: 20.01.2013).
21. Пакман Д.Н., Вершинин А.Б., Некрасов М.В. Построение унифицированной системы обработки телеметрической информации в центрах управления полётами космических аппаратов // Космонавтика и ракетостроение. – 2010. – № 1(58). – С. 124-130.
22. Дальниченко И.А. Автоматизированные системы управления предприятиями. – М.: Машиностроение, 1984. – 360 с.
23. Бахвалов Л.А. Моделирование систем. – М.: Горная книга, 2006. – 290 с.
24. Элементы теории массового обслуживания [Электронный ресурс] // Институт менеджмента, маркетинга и финансов: [сайт]. URL: http://math.immf.ru/lections/206.html (дата обращения: 05.05.2013).
25. Вентцель Е.С. Теория вероятностей. – М.: Высшая школа, 2001. – 576 с.
26. Гмурман В.Е. Теория вероятностей и математическая статистика. – М.: Юрайт, 2013. – 480 с.
27. СМО с отказами: определения и формулы [Электронный ресурс] // Математический форум Math Help Planet: [сайт]. URL: http://mathhelpplanet.com/static.php?p=smo-s-otkazami (дата обращения: 06.05.2013).
28. ОАО «ИСС», ФГУП «РНИИ КП». Протокол взаимодействия САО и СПО управления в

ЦУП, разработанных с использованием БИЗКТ. – 2008. – 54 с.

29. Sommerville. Software Engineering. Second Edition. – Workingham. – 1985. – С. 68-69.
30. Буч Г., Максимчук Р.А., Энгл М.У. Объектно-ориентированный анализ и проектиирование с примерами приложений. – М.: Вильямс, 2010. – 720 с.
31. Шлеер С., Меллор С. Объектно-ориентированный анализ: моделирование мира в состояниях. – Киев: Диалектика, 1993. – 240 с.
32. Новиков Ф.А., Иванов Д.Ю. Моделирование на UML. – СПб.: Наука и техника, 2010. – 640 с.
33. Ларман К. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку. – М.: Вильямс, 2013. – 736 с.
34. Брукс Ф. Проектирование процесса проектирования. Записки компьютерного эксперта. – М.: Вильямс, 2013. – 464 с.
35. Трофимов С.А. CASE-технологии. Практическая работа в Rational Rose. – М.: Бином-Пресс, 2002. – 288 с.
36. Якобсон А., Буч Г., Рамбо Д. Унифицированный процесс разработки программного обеспечения. – СПб.: Питер, 2002. – 496 с.
37. Введение в UML 2.0, часть II [Электронный ресурс] // Национальный Открытый Университет «ИНТУИТ»: [сайт]. URL: http://www.intuit.ru/studies/courses/1041/218/lecture/2992?page=2 (дата обращения: 05.03.2013).
38. Липаев В.В. Проблемы обеспечения качества сложных программных средств // quality.eup.ru. URL: http://quality.eup.ru/MATERIALY4/poksps.htm (дата обращения: 09.06.2013).
39. Качество программного обеспечения // ООО «Program Verification Systems». URL: http://www.viva64.com/ru/t/0077/ (дата обращения: 06.06.2013).
40. ИФМО. Факультет информационных технологий и программирования // ИФМО. 2013. URL: http://fitp.ifmo.ru/shared/files/201211/51_505.doc (дата обращения: 25.10.2012).
41. Качество программного обеспечения // ПроТестинг. URL: http://www.protesting.ru/qa/quality.html (дата обращения: 06.05.2013).
42. Баранюк В.В., Тютюнников Н.Н. Оценка качества электронных словарей и энциклопедий // Программная инженерия. – 2012. – № 8. – С. 29-37.

43. Критерии качества программного средства // Факультет Компьютерных Наук ВГУ. URL: http://fkn.ktu10.com/?q=node/741 (дата обращения: 10.06.2013).
44. Показатели качества программного обеспечения // Качество программного обеспечения. URL: http://85.142.23.53/packages/1C/EE7B2130-6590-483F-5143-DAD5FA193064/1.0.12.20/unpacked/content/index.html (дата обращения: 12.06.2013).
45. Рамбо Д., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. – Питер, 2007. – 544 с.
46. Gabryelski K. Wildfire C++ Programming Style 1997. URL: http://www.literateprogramming.com/wildfire.pdf (дата обращения: 24.01.2012).
47. Google C++ Style Guide URL: http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml (дата обращения: 07.03.2012).
48. Qt Coding Style // Qt Project. URL: http://qt-project.org/wiki/Qt_Coding_Style (дата обращения: 10.02.2012).
49. Qt Coding Conventions // Qt Project. URL: http://qt-project.org/wiki/Coding-Conventions (дата обращения: 21.01.2012).
50. Subversion // Wikipedia. URL: http://ru.wikipedia.org/wiki/Subversion (дата обращения: 30.05.2013).
51. Махоткин А. CVS - система управления версиями [Электронный ресурс] // CIT Forum: [сайт]. URL: http://citforum.ru/programming/application/cvs/ (дата обращения: 02.05.2013).
52. История Subversion // Управление версиями в Subversion. URL: http://svnbook.red-bean.com/nightly/ru/svn.intro.whatis.html#svn.intro.history (дата обращения: 30.05.2013).
53. Основы тестирования программного обеспечения // НОУ "ИНТУИТ". 2012. URL: http://www.intuit.ru/studies/courses/48/48/lecture/728 (дата обращения: 15.03.2013).
54. CrashRpt // Google Developers. URL: https://code.google.com/p/crashrpt/ (дата обращения: 15.10.2013).
55. Доставка и обработка отчетов об ошибках в Windows-приложениях // Использование библиотеки CrashRpt. URL: http://www.rsdn.ru/article/files/libs/crashrpt.xml (дата обращения: 17.03.2013).
56. Bug tracking system // Wikipedia. URL: http://en.wikipedia.org/wiki/Bug_tracking_system (дата обращения: 27.09.2013).
57. Bugzilla // Wikipedia. URL: http://ru.wikipedia.org/wiki/Bugzilla (дата обращения: 15.02.2013).

58. Trac // Wikipedia. URL: http://ru.wikipedia.org/wiki/Trac (дата обращения: 15.02.2013).
59. Atlassian JIRA // Wikipedia. URL: http://ru.wikipedia.org/wiki/Atlassian_JIRA (дата обращения: 16.02.2013).
60. TrackStudio Enterprise // Wikipedia. URL: http://ru.wikipedia.org/wiki/TrackStudio_Enterprise (дата обращения: 14.02.2013).
61. Статический анализ // PVS-Studio. URL: http://www.viva64.com/ru/t/0046/ (дата обращения: 18.01.2013).
62. Использование статического и динамического анализа для повышения качества продукции и эффективности разработки // OCPB QNX. URL: http://www.swd.ru/index.php3?pid=828 (дата обращения: 18.01.2013).
63. Campwood Software // SourceMonitor. URL: http://www.campwoodsw.com/sourcemonitor.html (дата обращения: 10.01.2013).
64. Инструменты статического анализа кода // PVS-Studio. URL: http://www.viva64.com/ru/t/0074/ (дата обращения: 10.01.2013).
65. Coverity [Электронный ресурс] // Coverity: [сайт]. URL: http://www.coverity.com/ (дата обращения: 05.10.2013).
66. CppCheck // SourceForge. URL: http://cppcheck.sourceforge.net/ (дата обращения: 17.06.2013).
67. Clang // The LLVM Compiler Infrastructure. URL: http://clang.llvm.org/ (дата обращения: 15.06.2012).
68. Klocwork Insight // Klocwork. URL: http://www.klocwork.com/products/insight/ (дата обращения: 23.07.2013).
69. Parasoft C/C++ Test // Parasoft. URL: http://www.parasoft.com/cpptest?itemId=47 (дата обращения: 04.12.2012).
70. Gimpel Software [Электронный ресурс] // Gimpel Software: [сайт]. URL: http://www.gimpel.com/html/index.htm (дата обращения: 04.10.2012).
71. Описание PVS-Studio // PVS-Studio. URL: http://www.viva64.com/ru/pvs-studio/ (дата обращения: 18.08.2013).
72. Динамический анализ кода // PVS-Studio. URL: http://www.viva64.com/ru/t/0070/ (дата обращения: 19.01.2013).
73. Valgrind [Электронный ресурс] // Valgrind: [сайт]. URL: http://valgrind.org/ (дата обращения: 15.02.2013).

74. IBM Rational Purify // IBM. URL: http://www-03.ibm.com/software/products/ru/ratpurwin/ (дата обращения: 18.01.2013).
75. Intel® Parallel Studio XE Suites // Intel® Developer Zone. URL: http://software.intel.com/en-us/intel-parallel-studio-xe/ (дата обращения: 03.02.2013).
76. DevPartner Studio Professional Edition // Borland. URL: http://www.borland.com/_images/DevPartner-Studio-Professional-Edition_tcm32-207615.pdf (дата обращения: 21.01.2013).
77. Parasoft Insure++ // Parasoft. URL: http://www.parasoft.com/insure (дата обращения: 14.01.2013).
78. Метрики кода и их практическая реализация в Subversion и ClearCase. Часть 1 - метрики // CM Consult2. URL: http://cmcons.com/articles/CC_CQ/dev_metrics/mertics_part_1/#2.1.1.1 (дата обращения: 10.05.2013).
79. Критерии качества программного средства. Определение качества ПО в стандарте ISO 9126. Многоуровневая модель качества ПО. Оценочные характеристики качества программного продукта // Fkn+Antitotal. URL: http://fkn.ktu10.com/?q=node/741 (дата обращения: 02.05.2013).
80. Горбаченко И.М. Оценка качества программного обеспечения для создания систем тестирования // Фундаментальные исследования. – 2013. – Т. 6. – № 4. – С. 823-827.
81. Модели и метрики оценки качества ПО // Метрики. URL: http://www.met-rix.narod.ru/page2.htm (дата обращения: 18.05.2013).
82. Метрики кода // Net way. URL: http://dimakudr.blogspot.ru/2010/05/blog-post_27.html (дата обращения: 15.05.2013).
83. Метрики по обеспечению качества // ПроТестинг. URL: http://www.protesting.ru/qa/metrics.html (дата обращения: 27.10.2012).
84. Некрасов М.В., Пакман Д.Н., Шмик К.Б. Повышение качества принимаемой телеметрической информации при наличии избыточных источников сигнала // Студент и научно-технический прогресс: XLVIII Международная научная студенческая конференеция. – г. Новосибирск, 10-14 апреля 2010 г. – 2010. – С. 101.
85. Некрасов М.В., Шмик К.Б. Адаптивный алгоритм передачи для систем многопоточной обработки телеметрической информации // Актуальные проблемы ракетно-космического приборостроения и информационных технологий: Материалы III Всероссийской научно-технической конференции. – Москва, 1-3 июня 2010 г. – 2010. – С. 145-146.
86. Грушо А.А., Тимонина Е.Е. Теоретические основы защиты информации. – М.: Яхтсмен,

1996. – 187 с.

87. Вопросы защищённости // CIT Forum. URL: http://citforum.ru/internet/intranet/intra_pr.shtml (дата обращения: 15.02.2013).
88. ВНИИНС [Электронный ресурс] // Всероссийский научно-исследовательский институт автоматизации управления в непромышленной сфере имени В.В.Соломатина: [сайт]. URL: http://www.vniins.ru/ (дата обращения: 10.08.2011).
89. Операционная система МСВС с пакетом офисных и мультимедийных программ // GISTechnik. URL: http://gistechnik.ru/pub/3-publik/65-mcbc.html (дата обращения: 15.08.2011).
90. Самараев Р.С. Программирование с использованием Qt. – М.: МГТУ им. Баумана, 2009. – 55 с.
91. Шлее М. Qt 4.8 Профессиональное программирование на C++. – СПб.: БХВ-Петербург, 2012. – 912 с.
92. ГЛОНАСС: принципы построения и функционирования / Под ред. А. И. Перова, В. Н. Харисова. - 3-е изд., перераб. – М.: Радиотехника, 2005. – 688 с.
93. Изделие 14Ф113.ИДБПО. Исходные данные на бортовое программное обеспечение. – Железногорск. – 2001. – 163 с.
94. Изделие 14Ф113.ДПМ. Программа телеметрических измерений. Приложение 4. Таблицы параметров. – Железногорск. – 1997. – 176 с.
95. Изделие 14Ф113. ИЭ19ч1. Инструкция по обработке информации. Часть первая. Общие сведения по обработке информации. – Железногорск. – 2001. – 23 с.

Приложение А

(обязательное)

Протокол взаимодействия СОТМ и внутренних клиентов телеметрии

1 Протокол транзитного обмена между СОТМ и СПО

Настоящий протокол построен на базе «Протокола взаимодействия САО и СПО управления в ЦУП, разработанных с использованием БИЗКТ» и предназначен для обмена оперативными видами информации между сервером (СОТМ) и клиентами телеметрии в ходе сеанса управления. Передача информации выполняется по каналам ЛВС ЦУП.

1.1 Транспортный уровень

Используемый сетевой протокол — TCP/IP. Клиенты телеметрии при взаимодействии выступают *сетевым клиентом*, СОТМ — *сетевым сервером*. Сетевой порт сервера – 1620, который может уточняться в файле конфигурации.

1.1.1 Действия клиента

Для подключения к серверу клиент выдает сетевую команду «CONNECT». Для разрыва связи с сервером клиент выдает сетевую команду «DISCONNECT».

Клиент и сервер обмениваются сетевыми сообщениями. Клиент должен вычитывать информацию из своего сокета до тех пор, пока не будут выполнены оба следующих условия:

- будет получен блок информации, достаточный для разбора всех полей заголовка сетевого сообщения;
- будет получен весь объем информационной части сетевого сообщения в соответствии с записанным в заголовке значением поля «Длина информации».

Полученное сетевое сообщение должно быть обработано в соответствии с правилами сеансового уровня. Вся информация в сокете, следующая за выделенным сетевым сообщением, должна быть сохранена в оперативной памяти и использована для выделения следующего сетевого сообщения. Данная процедура выделения сетевых сообщений из сокета должна продолжаться до тех пор, пока не будет получен сигнал разрыва связи с сервером или она не будет прервана оператором.

Контроль правильности формирования и передачи в сеть сетевых сообщений возлагается на программные комплексы клиента и сервера.

Клиент самостоятельно отслеживает ситуацию аварийной потери связи с сервером. В случае аварийного пропадания связи клиент должен выполнить одно из следующих действий:

- закрыть рабочий сокет (разорвать связь) и ждать решения оператора;

- выполнить процедуру повторного подключения к серверу (восстановить связь).

В случае, если клиент при отправке (записи в сокет) информации серверу получает код сетевой ошибки (WSA): 10061 или 11001, однако не получил от операционной системы сигнал о разрыве связи с сервером том, то клиент должен считать соединение с сервером потерянным.

1.1.2 Действия сервера

Сервер обеспечивает постоянное прослушивание своего сокета. При получении сигнала «CONNECT» Сервер выдает команду «ACCEPT», организует канала связи между клиентом и сервером. Сервер должен одновременно поддерживать соединение с произвольным числом клиентов.

Рабочий сокет сервера должен быть разрушен только после того, как сервер получит сетевой сигнал «DISCONNECT» или обнаружит ошибку при передаче информации клиенту (аварийный разрыв). По аналогии с действиями клиента сервер:

- следит за правильностью формирования сетевых сообщений;
- выполняет процедуру вычитывания и выделения сетевых сообщений из сетевого сокета.

1.2 Сеансовый уровень

1.2.1 Формат сетевого заголовка

Клиент и сервер обмениваются сетевыми сообщениями с унифицированным сетевым заголовком, формат которого приведён в таблице А.1.

Таблица А.1 – Формат сетевого заголовка

размер, байт	1	1	1	1	2	1	1	2	2	4	2	4	2	1
	ndir	abtk	nkip	nktc	Nka	nvtk	ncc	nform	datn	vrn	datk	vrk	ds	kvit

Поле	**Размер поля**	**Содержание**
ndir	1 байт	номер директивы, принимает одно из значений enum TNetPacket::DirectiveType
abtk	1 байт	Признаки
nkip	1 байт	номер НИП
nktc	1 байт	номер КТС
nka	2 байта	номер КА
nvtk	1 байт	номер витка
ncc	1 байт	номер сеанса
nform	1 байт	вид информации
datn	2 байта	дата начала
vrn	4 байта	время начала
datk	2 байта	дата конца
vrk	4 байта	время конца
ds	2 байта	длина сообщения

kvit	1 байт	код квитанции
data	*ds* байт	необязательное поле. Информация прямого / обратного канала (ИПК/ИОК)

Разряды поля признаков «АВТК» имеют следующие значения:

Бит	**Описание** (при = 1)
A	признак требования квитанций
B	Резервируется
T	Резервируется
K	Резервируется
N	Резервируется
P	Резервируется
M	Резервируется
E	Резервируется

Перечисляемый тип TNetPacket::DirectiveType определяет коды допустимых директив и описывается в таблице А.2.

1.2.2 Виды сетевых сообщений

Все сетевые сообщения делятся на директивы и квитанции.

Директивы имеют инициативный характер и предписывают выполнение получателем заданных действий. Директивы могут быть отправлены как клиентом серверу, так и сервером клиенту. В некоторых случаях директива может не иметь информационной части. Перечень используемых директив приведен в таблице А.2.

Таблица А.2 – Перечень допустимых директив

Константна	**№ Дир**	**Назначение**	**Описание**
dtStart	18	Установить параметры сеанса	Открывает направление. Результат – в коде вторичной квитанции
dtEnd	19	Завершить сеанс	Закрывает направление. Передача любой информации клиенту по данному направлению прекращается.
dtTmiToOtiClient	101	Запросить телеметрические кадры	Запрашивает у сервера выдачу телеметрических кадров. Результат запроса – в коде вторичной квитанции
dtTmiXml	158	Запросить набор телеметрических параметров	Отправляет на сервер заявку получение обработанных телеметрических параметров, или отчётов в виде XML сообщения
dtActiveSeances	200	Запросить состав сеансов	Запрашивает у сервера состав сеансов и их статистику. Результат запроса – в коде вторичной квитанции
dtOtiPacket	202	Отправить служебный телеметрический пакет	Передаёт серверу запрос в виде служебного телеметрического пакета, формат которого определяется отдельным прикладным протоколом ОТИ
dtAuthentication	203	Отправить	Отправляет серверу параметры авторизации

Константна	№ Дир	Назначение	Описание
		параметры авторизации	для дальнейшего приёма любых видов телеметрической информации. Результат запроса – в коде вторичной квитанции

Квитанции отправляются в ответ на директиву, содержащую выставленное требования квитирования (разряд «А» поля признаков в заголовке сетевого сообщения), и могут быть отправлены как клиентом серверу, так и сервером клиенту.

В ответ на каждую директиву с выставленным требованием квитирования получатель должен отправить две квитанции. Первичная квитанция отправляется получателем сразу и сообщает отправителю о том, что:

- директива принята получателем;
- содержит/не содержит ошибок;
- может/не может быть исполнена.

Если директива может быть исполнена, получатель директивы выполняет предписанные действия и после их завершения, при наличии требования квитирования, отправляет вторичную квитанцию.

Вторичная квитанция сообщает отправителю о результате обработки директивы. При отрицательной первичной квитанции вторичная квитанция не выдается.

Для формирования квитанции на директиву необходимо скопировать все поля директивы в соответствующие поля квитанции, за исключением полей: признаков, длины информации, кода квитанции.

Допустимые коды квитанций приведены в Таблице А.3.

Таблица А.3 – Коды квитанций

Код	Описание
00h	директива принята к исполнению (первичная квитанция САО)
01h	неизвестен вид
02h	директива успешно выполнена (вторичная квитанция САО)
03h	директива не понята
07h	ошибка в формате сетевого сообщения
10h	не тот номер КА в признаках сеанса
11h	не соответствуют параметры сеанса
12h	не тот номер НКИС и КТС
13h	не тот номер сеанса в признаках сеанса
14h	не тот номер витка в признаках сеанса
50h	уже есть выдача ТМИ

1.2.3 Открытие направления

После установки сетевого соединения (CONNECT → ACCEPT) клиент открывает на сервере направление приёма телеметрии: это заключается в выдаче клиентом директивы *dtStart* с признаками сеанса и видом информации 1888 (телеметрическая информация).

Положительная реакция сервера на директиву *dtStart* состоит в добавлении строки в таблицу маршрутизации, поддерживаемую сервером.

1.2.4 Обмен информацией

После того, как направление открыто, клиент/сервер может выдавать любые другие директивы по данному направлению. При этом в заголовке директивы необходимо заполнять поля:

- признаков сеанса;
- длины информационной части, если директива содержит информационную часть. В этом случае, информационная часть должна следовать за сетевым заголовком в поле ИПК/ИОК.

Если действие по директиве предписывает выдачу клиенту ИОК, то после выдачи вторичной квитанции, сервер высылает отправителю директивы всю ИОК в виде дополнительных вторичных квитанций на полученную директиву.

1.2.5 Закрытие направления

Перед разрывом сетевой связи с сервером клиент должен закрыть все открытые им направления. Для этого клиент должен сформировать сетевые сообщения аналогичные сообщениям при открытии направления, но с директивой *dtEnd*.

В случае если сервер обнаружил отключение (например, аварийное) клиента, а открытые направления не были закрыты – сервер должен:

- самостоятельно удалить данные направления из своих таблиц маршрутизации;
- закрыть сетевое соединение (выдать сетевую команду «DISCONNECT»).

Клиент может открывать и закрывать направления в любой момент взаимодействия с сервером. Каждый клиент должен вести в своей памяти таблицу открытых им на сервере направлений.

1.3 Уровень представления данных

Все сообщения, циркулирующие между сервером и клиентом, снабжаются стандартным заголовком, формат которого приведен в таблице А.1.

Приложение Б

(обязательное)

Протокол взаимодействия СОТМ и удалённых клиентов телеметрии

1 Транспортный уровень

Используемый сетевой протокол — TCP/IP. Абонент при взаимодействии выступает сетевым клиентом, СОТМ — сетевым сервером. Сетевой порт сервера – 1620 (может уточняться в настроечных файлах СОТМ).

Для отправки запроса сетевой клиент должен сформировать сетевое сообщение, состоящее из сетевого заголовка TNetPacket, формат которого приведён ниже, и информационной части, формат которой приведён в разделе 2.

```
#pragma pack(1)
        // Заголовок сетевого пакета
        struct TNetPacket
        {
                TNetPacket() {
                        memset(this, 0x0, sizeof(TNetPacket));
                }
                BYTE  ndir;       // номер директивы
                BYTE  abtk;       // признаки
                BYTE  nkip;       // номер НИП
                BYTE  nktc;  // номер КТС
                WORD  nka;   // номер КА
                BYTE  nvtk;       // номер витка
                BYTE  ncc;        // номер сеанса
                WORD  nform; // вид информации
                WORD  datn;  // дата начала
                DWORD vrn;   // время начала
                WORD  datk;       // дата конца
                DWORD vrk;        // время конца
                WORD  ds;         // длина сообщения
                BYTE  kvit;       // код квитанции
        // Возвращает значение номера КА в десятичном виде
        WORD GetKa() const {
                return (nka & 0xF) * 1
                  + ((nka & 0xF0) >> 4) * 10
                  + ((nka & 0xF00) >> 8) * 100
                  + ((nka & 0xF000) >> 12) * 1000;
        }
                // Устанавливает значение номера КА
        void SetKa(WORD ka) {
                this->nka = ((ka / 1000) << 12)
                  + (((ka % 1000)/100) << 8)
                  + (((ka % 100)/10) << 4)
                  + (ka % 10);
        }
        };
#pragma pack()
```

Примечание:

1 При описании структуры используются следующие типы данных
BYTE = unsigned char (1 byte)

WORD = unsigned short (2 bytes)

DWORD = unsigned int (4 bytes)

2 Для чтения/установки значения поля nka использовать методы GetKa / SetKa

Для запроса параметров ТМИ у СОТМ в транспортном заголовке должны быть заполнены следующие поля: *НомерДирективы* = 158, *ВидИнформации* = 1888, *НомерКА*, *ДлинаИнформации*. После заголовка должно следовать сообщение-запрос прикладного уровня. Пример запроса приведён в разделе 5.

В ответ на запрос СОТМ формирует стандартный транспортный заголовок, в котором заполняются следующие поля: *НомерДирективы* = 158, *ВидИнформации* = 1888, *НомерКА*, *КодКвитанции* (0x02 – в случае корректности транспортного заголовка). После заголовка следует сообщение-ответ прикладного уровня. Пример ответа приведён в разделе 5. Принимаемые сообщения с кодом директивы отличным от 158 в обработке не использовать.

2 Прикладной уровень

Для запроса значений параметров и получения ответа используется формат XML. Общий вид запроса рассмотрен в разделе 3.

Примечание:

1 Для сообщения XML используется кодировка UTF-8
2 Обработка строковых значений производится *без учёта* регистра символов
3 Запрашиваются только параметры режима НП и конфигурации полезной нагрузки
4 В настоящей версии протокола используется только значения UpdateType=”0”
5 Для значений времени используется следующий формат, если не указано иное: dd.MM.yyyy hh:mm:ss, где:

dd – день месяца с лидирующим нулём (01 до 31)

MM – номер месяца с лидирующим нулём (01 до 12)

yyyy – год в виде четырёхзначного числа

hh – часы в 24 формате с лидирующим нулём (00 до 23)

mm – минуты с лидирующим нулём (00 до 59)

ss – секунды с лидирующим нулём (00 до 59)

3 Общий вид XML сообщения

```
<?xml version="1.0" encoding="utf-8"?>
<SotmDialog BodyType="...">
```

```
<Ka>...</Ka>
<Nip>...</Nip>
<Kts>...</Kts>
<Params ValueType="..." StartTime="..." EndTime="..." Interval="...">
  <Item Index="...">
    <Value State="..." Time="...">...</Value>
    ...
  </Item>
  ...
</Params>
<ViItems ValueType="..." ViCount="..." ViBadCount="...">
  <ViStatus DateTime="..." Name="..." Length="..." Validity="..." />
    ...
</ViItems>
<Error Code="...">...</Error>
</SotmDialog>
```

Подробное описание элементов XML сообщения приведено в разделе 4.

4 Описание элементов сообщения

<?xml version="1.0" encoding="utf-8"?>

Стандартный заголовок сообщения. Его формат должен сохраняться без изменений.

4.1 SotmDialog [Элемент]

SotmDialog [Элемент]

Общий элемент, обозначающий формат диалогового сообщения между севером и удалёнными потребителями телеметрии.

BodyType [Атрибут]

Атрибут для элемента SotmDialog. Содержит тип тела XML сообщения и принимает следующие значения: Query, Response.

4.2 Ka [Элемент]

Ka [Элемент]

Номер космического аппарата, по которому производится запрос параметров.

4.3 Nip [Элемент]

Nip [Элемент]

Номер наземного измерительного пункта, по которому производится запрос параметров. [Значение *по умолчанию* = 0].

Элемент **Nip** используется при работе с виртуальными земными станциями, номера которых превышают 90-00. В случае отсутствия рассматриваемого элемента, запрос будет выполняться по первой обнаруженной станции, с которой ведётся приём телеметрии.

4.4 Kts [Элемент]

Kts [Элемент]

Номер технического средства на НИП, по которому производится запрос параметров. [Значение *по умолчанию* = 0].

Элемент **Kts** используется при работе с виртуальными земными станциями, номера которых превышают 90-00. В случае отсутствия рассматриваемого элемента, запрос будет выполняться по первой обнаруженной станции, с которой ведётся приём телеметрии.

4.5 Params [Элемент]

Params [Элемент]

Открывает группу запрашиваемых параметров. Может содержать атрибуты ValueType, StartTime, EndTime, UpdateType. Количество параметров в группе не ограничено. Если параметры в группе отсутствуют, значит, запрашиваются абсолютно все параметры по указанному космическому аппарату.

ValueType [Атрибут]

Атрибут для элемента Params. Позволяет указать тип запрашиваемых значений параметров, а именно: All – все существенные значения от момента поступления запроса, Last – только последнее значение, Interval – значения параметров за заданный временной интервал, Payload – значения параметров из сохранённой ранее конфигурации полезной нагрузки [Значение *по умолчанию* отсутствует]

При использовании ValueType=”All” должен быть установлен признак автоматической отправки новых значений при помощи атрибута UpdateType, в противном случае результат запроса будет эквивалентен ValueType=”Last”.

При использовании ValueType=”Interval” должен быть определён временной интервал при помощи атрибутов StartTime и EndTime, а использование атрибута UpdateType запрещено

StartTime [Атрибут]

Атрибут для элемента Params. Задаёт начальную точку временного интервала в оговоренном формате, для которого будет осуществляться выбор значений параметров. [Значение *по умолчанию* отсутствует]

Этот атрибут задаётся только при ValueType=”Interval” и является обязательным.

EndTime [Атрибут]

Атрибут для элемента Params. Задаёт конечную точку временного интервала в оговоренном формате, для которого будет осуществляться выбор значений параметров. [Значение *по умолчанию* соответствует текущему моменту времени, при этом будут выбраны все значения со временем от StartTime до настоящего момента времени.]

Этот атрибут задаётся только при ValueType=”Interval”.

Interval[Атрибут]

Атрибут для элемента Params. Управляет режимом автоматической отправки значений для запрашиваемых параметров и определяет интервал выдачи результата [Значение *по умолчанию* = 0]. Принимает значения:

- 0 – однократный запрос;
- > 0 – разрешить автоматическое повторное исполнение (минимальное значение перезапроса составляет 5000 мс);
- -1 – отменяет запрос об автоматической отправке новых значений (для сессии с параметрами Ka, Nip, Kts) для указанной группы параметров или для всех параметров, если группа не определена;
- иначе – сообщение об ошибке.

4.5.1 Item [Элемент]

Item [Элемент]

Элемент группы Params, содержащий описание параметра. Открывает группу значений параметра и содержит перечисления элементов Value. Если для запрашиваемого параметра отсутствуют значения, будет сформирована пустая группа Item.

Index [Атрибут]

Атрибут для элемента Item. Содержит текстовый индекс запрашиваемого параметра.

4.5.1.1 Value [Элемент]

Value [Элемент]

Элемент группы Item, содержащий значение параметра в текстовом виде, обработанное на сервере по типу выхода.

Этот элемент задаётся только при BodyType=”Response”.

State [Атрибут]

Атрибут для элемента Value. Содержит состояние параметра и принимает одно из следующих значений: 0 – параметр в норме, 1 – внимание, 2 – тревога (параметр вне диапазона), -1 – параметр не сформирован.

DateTime [Атрибут]

Атрибут для элемента Value. Содержит дату и время формирования параметра в оговоренном формате.

4.6 ViItems [Элемент]

ViItems [Элемент]

Открывает группу запрашиваемых отчётов бортового компьютера. Может содержать атрибуты ValueType, ViCount, ViBadCount.

ValueType [Атрибут]

Атрибут для элемента ViItems. Позволяет указать тип запрашиваемых отчётов, а именно: New – только *новые* отчёты, полученные с момента последнего запроса. Если с момента предыдущего запроса новых отчётов получено не было, запрос вернёт пустое множество. Last – только последний принятый отчёт. [Значение *по умолчанию* отсутствует]

ViCount [Атрибут]

Атрибут для элемента ViItems. Содержит общее количество отчётов внутри группы ViItems.

Этот атрибут задаётся только при BodyType=”Response”.

ViBadCount [Атрибут].

Содержит количество сбойных отчётов внутри группы ViItems.

Этот атрибут задаётся только при BodyType=”Response”.

4.6.1 ViStatus [Элемент]

ViStatus [Элемент]

Элемент группы ViItems, содержащий описание отчёта в виде атрибутов: DateTime, Name, Length, Validity.

Этот элемент задаётся только при BodyType=”Response”.

DateTime [Атрибут]

Символьное. Атрибут содержит дату и время получения отчёта.

Name [Атрибут]

Символьное. Атрибут содержит наименование отчёта.

Length [Атрибут]

Целое число. Атрибут содержит длину отчёта в байтах.

Validity [Атрибут]

Целое число. Атрибут содержит признаки достоверности отчёта.

4.7 Payload [Элемент]

Payload [Элемент]

Запрос на выполнение действий с конфигурацией полезной нагрузки. Содержит атрибуты Name, Action.

Этот элемент задаётся только при BodyType=”Query”.

Name [Атрибут]

Атрибут для элемента **Payload**. Содержит обязательное имя для конфигурации полезной нагрузки. [Значение *по умолчанию* отсутствует]

Action [Атрибут]

Атрибут для элемента **Payload**. Позволяет указать тип действия, производимого над конфигурацией полезной нагрузки, а именно: Save – требует выполнить сохранение текущей конфигурации полезной нагрузки, Remove – требует выполнить удаление ранее сохранённой конфигурации полезной нагрузки. [Значение *по умолчанию* отсутствует]

4.8 Error [Элемент]

Error [Элемент]

Элемент формируется сервером на запрос клиента в случае, если при обработке запроса на выбор параметров произошла ошибка. Содержит текстовое пояснение ошибки.

Этот элемент задаётся только при BodyType=”Response”.

Code [Атрибут]

Атрибут для элемента Error. Содержит код ошибки.

5 Примеры использования протокола

5.1 Пример 1

Задача: Запросить последние значения параметров BD0141 (КодК), AD0076 (Рыск)

- *Запрос*

Ниже приведён состав запроса в двоичном виде. Жёлтым цветом выделен транспортный заголовок (соответствующий структуре TNetPacket), синим – пакет прикладного уровня (XML запрос).

```
9E 00 00 00 45 09 00 00  60 07 00 00 00 00 00 00
00 00 00 00 00 00 DE 00  00
3C 3F 78 6D 6C 20 76 65  72 73 69 6F 6E 3D 22 31
2E 30 22 20 65 6E 63 6F  64 69 6E 67 3D 22 75 74
66 2D 38 22 3F 3E 0A 3C  53 6F 74 6D 44 69 61 6C
6F 67 20 42 6F 64 79 54  79 70 65 3D 22 51 75 65
72 79 22 3E 0A 20 20 20  20 3C 4B 61 3E 39 34 35
3C 2F 4B 61 3E 0A 20 20  20 20 3C 50 61 72 61 6D
73 20 56 61 6C 75 65 54  79 70 65 3D 22 4C 61 73
74 22 20 55 70 64 61 74  65 54 79 70 65 3D 22 30
22 3E 0A 20 20 20 20 20  20 20 20 3C 49 74 65 6D
20 49 6E 64 65 78 3D 22  42 44 30 31 34 31 22 20
2F 3E 0A 20 20 20 20 20  20 20 20 3C 49 74 65 6D
20 49 6E 64 65 78 3D 22  41 44 30 30 37 36 22 20
2F 3E 0A 20 20 20 20 3C  2F 50 61 72 61 6D 73 3E
0A 3C 2F 53 6F 74 6D 44  69 61 6C 6F 67 3E
```

Транспортный заголовок

ndir = 158
nform = 1888
nka   = 0x4509
ds   = 222

Пакет прикладного уровня

```
<?xml version="1.0" encoding="utf-8"?>
<SotmDialog BodyType="Query">
  <Ka>945</Ka>
  <Params ValueType="Last" UpdateType="0">
    <Item Index="BD0141" />
```

```
  <Item Index="AD0076" />
 </Params>
</SotmDialog>
```

- *Ответ сервера обработки телеметрии*

Ниже приведён состав ответа сервера обработки телеметрии в двоичном виде. Жёлтым цветом выделен транспортный заголовок (соответствующий структуре TNetPacket), синим – пакет прикладного уровня (XML ответ).

```
9E 80 00 00 45 09 00 00  60 07 00 00 00 00 00 00
00 00 00 00 00 00 7D 01  02
3C 3F 78 6D 6C 20 76 65  72 73 69 6F 6E 3D 22 31
2E 30 22 3F 3E 0A 3C 53  6F 74 6D 44 69 61 6C 6F
67 20 42 6F 64 79 54 79  70 65 3D 22 52 65 73 70
6F 6E 63 65 22 3E 0A 20  20 20 20 3C 4B 61 3E 39
34 35 3C 2F 4B 61 3E 0A  20 20 20 20 3C 50 61 72
61 6D 73 20 56 61 6C 75  65 54 79 70 65 3D 22 4C
61 73 74 22 20 55 70 64  61 74 65 54 79 70 65 3D
22 30 22 3E 0A 20 20 20  20 20 20 20 20 3C 49 74
65 6D 20 49 6E 64 65 78  3D 22 42 44 30 31 34 31
22 3E 0A 20 20 20 20 20  20 20 20 20 20 20 20 3C
56 61 6C 75 65 20 53 74  61 74 65 3D 22 30 22 20
44 61 74 65 54 69 6D 65  3D 22 32 33 2E 31 32 2E
32 30 31 31 20 30 30 3A  31 34 3A 32 31 22 3E 31
30 31 35 3C 2F 56 61 6C  75 65 3E 0A 20 20 20 20
20 20 20 20 3C 2F 49 74  65 6D 3E 0A 20 20 20 20
20 20 20 20 3C 49 74 65  6D 20 49 6E 64 65 78 3D
22 41 44 30 30 37 36 22  3E 0A 20 20 20 20 20 20
20 20 20 20 20 20 3C 56  61 6C 75 65 20 53 74 61
74 65 3D 22 30 22 20 44  61 74 65 54 69 6D 65 3D
22 32 33 2E 31 32 2E 32  30 31 31 20 30 30 3A 31
38 3A 32 34 22 3E 30 2E  34 3C 2F 56 61 6C 75 65
3E 0A 20 20 20 20 20 20  20 20 3C 2F 49 74 65 6D
3E 0A 20 20 20 20 3C 2F  50 61 72 61 6D 73 3E 0A
3C 2F 53 6F 74 6D 44 69  61 6C 6F 67 3E
```

Транспортный заголовок

ndir = 158
nform = 1888
nka   = 0x4509
ds   = 381
kvit = 2

Пакет прикладного уровня (XML ответ)

```
<?xml version="1.0"?>
<SotmDialog BodyType="Responce">
 <Ka>945</Ka>
 <Params ValueType="Last" UpdateType="0">
  <Item Index="BD0141">
   <Value State="0" DateTime="23.12.2011 00:14:21">1015</Value>
  </Item>
  <Item Index="AD0076">
   <Value State="0" DateTime="23.12.2011 00:18:24">0.4</Value>
  </Item>
 </Params>
```

```
</SotmDialog>
```

5.2 Пример 2

Задача: Запросить последний принятый отчёт

Запрос

```
<?xml version="1.0" encoding="utf-8"?>
<SotmDialog BodyType="Query">
  <Ka>945</Ka>
  <ViItems ValueType="New" />
</SotmDialog>
```

Ответ сервера

```
<?xml version="1.0" encoding="utf-8"?>
<SotmDialog BodyType="Responce">
  <Ka>945</Ka>
  <ViItems ValueType="New" ViCount="1" ViBadCount="0">
    <ViStatus DateTime="23.04.2000 23:18:33" Name="C" Length="1036" Validity="6" />
  </ViItems>
</SotmDialog>
```

5.3 Пример 3

Задача: Запросить сервер обработки телеметрии выполнить сохранение текущей конфигурации полезной нагрузки.

Запрос

```
<?xml version="1.0" encoding="utf-8"?>
<SotmDialog BodyType="Query">
  <Ka>945</Ka>
  <Nip>99</Nip>
  <Kts>1</Kts>
  <Payload Name=”Идеальная конфигурация” Action="Save" />
</SotmDialog>
```

Ответ сервера

Если запрос не содержит ошибок, сервер его выполнит и произведёт сохранение текущей конфигурации полезной нагрузки в базе данных. В противном случае, вернёт код ошибки.

Приложение В

(обязательное)

Стиль программирования C++ для QT

Введение

C++ является основным языком разработки многих современных объектно-ориентированных проектов. Каждый программист знает, что этот язык имеет мощные возможности, которые, однако, несут в себе определённую сложность, что может наполнить программный код ошибкам и сделать его сложночитаемым и трудносопровождаемым.

В настоящем руководстве описываются «за» и «против» при использовании возможностей языка C++ во время написания программного кода. Эти правила существуют для поддержания кода в контролируемом состоянии, в то же время позволяя разработчикам эффективно использовать возможности C++ и работать в коллективе над общими проектами.

1 Заголовочные файлы

Как правило, каждый *.cpp* файл должен иметь соответствующий *.h* файл. Однако существует некоторое общее исключение – при написании юнит-тестов *.cpp* файлы могут состоять из единственной функции *main()*.

Правильное использование заголовочных файлов может существенно влиять на читаемость, размер и производительность вашего кода.

Директива #define

Все заголовочные файлы должны иметь директиву *#define* для исключения повторного включения. Формат имени должен быть следующим <PROJECT>_<PATH>_<FILE>_H_. Например, для файла foo/src/bar/baz.h в проекте foo должны быть следующие директивы:

```
#ifndef FOO_BAR_BAZ_H_
#define FOO_BAR_BAZ_H_
…
#endif // FOO_BAR_BAZ_H_
```

Раннее объявление

Допускается использовать раннее объявление классов для исключения лишних *#include*-ов.

Inline функции

Объявляйте функции как *inline* только если они не большие, содержат до 10 строк кода.

Порядок параметров в функции

При определении функции порядок параметров должен быть: входные, затем выходные.

Параметры функции C/C++ являются либо входными, либо выходными, либо теми и другими. Входные параметры – обычно значения или константная ссылка, в то время как входные\выходные параметры должны быть не константными указателями. При определении порядка следования параметров располагайте все входные параметры перед выходными. В особенности не добавляйте новые параметры в конец лишь потому, что они новые; размещайте новые входные параметры перед выходными

Наименования и порядок подключения Include-ов

Используйте следующий стандартный порядок для улучшения читабельности и иллюстрации зависимостей: *C* библиотеки, *C++* библиотеки, *.h* файлы других библиотек, *.h* файлы ваших проектов.

2 Переменные

Локальные переменные

Правила объявления переменных приведены в следующем списке:

- дождитесь, пока переменная будет нужна, и только тогда объявляйте ее;
- производите инициализацию переменных при объявлении;
- объявляйте каждую переменную на отдельной строке.

Статичные и глобальные переменные

Статичные или глобальные переменные типа класс – запрещены: они приводят к сложно-обнаруживаемым ошибкам из-за неоднозначного порядка конструирования и разрушения объектов.

Статичные и глобальные переменные типа POD (Plain Old Data), такие как int, char, float, указатели могут использоваться без ограничений.

3 Классы

Выполнение задач в конструкторах

Избегайте выполнения сложной инициализации в конструкторах (в особенности при инициализации, которая может завершиться ошибкой, или которая требует вызова виртуальных функций)

Инициализация

Если в вашем классе объявлены поля-переменные, вы обязаны инициализировать средствами класса значение каждого такого поля или написать конструктор по умолчанию. Если вы не объявите каких-либо конструкторов, тогда компилятор создаст конструктор по умолчанию за вас, который может оставить некоторые поля неинициализированными или

инициализировать их некорректными значениями.

Явные конструкторы

Используйте ключевое слово *explicit* языка C++ для конструкторов с одним аргументом.

Конструктор копии

Добавляйте конструктор копии и оператор присваивания только при необходимости.

Структуры или классы

Используйте *struct* только для пассивных объектов, хранящих только данные; в остальных случаях используйте *class*.

Наследование

Метод включения зачастую наиболее предпочтителен и удобен, чем наследование. Если используется наследование, используйте *public*.

Множественное наследование

Используйте множественное наследование только при наследовании чистых виртуальных интерфейсов.

Перегрузка операторов

Не перегружайте операторы за исключением редких случаев.

Контроль доступа

Объявляйте поля данных как *private*, а для доступа к ним определяйте соответствующие функции доступа при необходимости.

Порядок объявления

Используйте следующий порядок при объявлении внутри класса: *public* перед *private, методы* перед *переменными*, и т.д.

Определение класса должно начинаться с раздела *public*, затем идет раздел *protected* и в завершение раздел *private.* Если один из разделов пуст, то его можно опустить.

Внутри каждого из разделов общие определения должны быть в следующем порядке:

1) Типы (types) и перечисления (enums);
2) Константы (static const data members);
3) Конструкторы;
4) Деструкторы;
5) Методы, включая статические методы;
6) Поля данных (except static const data members)
7) Определения типа *friend declaration* должны объявляться в разделе *private,* макросы запрета копирования (DISALLOW_COPY_AND_ASSIGN) должны определяться в последней строке раздела *private.*

Определения методов в соответствующем файле *.*cpp* должны располагаться в той же последовательности, что и в определении класса.

Пишите короткие функции

Сосредоточьтесь на написании небольших функций.

4 Именование

Наиболее важными правилами являются правила именования. Стиль имени мгновенно информирует нас о природе именованного объекта: тип, переменная, функция, константа, макрос и т.д. – без необходимости поиска описания этого объекта.

Общее правило именования

Имена функций, переменных и файлов должны быть понятным, самодостаточными. Избегайте использования аббревиатур и сокращений.

Имена файлов

Имена файлов должны состоять из букв нижнего регистра, могут содержать символы «_», или «-».

Названия типов

Названия типов должны начинаться с заглавной буквы и иметь заглавную букву для каждого нового слова, без нижнего подчёркивания: MyPerfectClass, MyFavoriteEnum.

Аббревиатуры должны быть в PascalCase (например, QXmlStreamReader вместо QXMLStreamReader).

Названия публичных классов должны начинаться с буквы ‘Q’ (QRgb), затем заглавная буква имени.

Имена переменных

Переменные должны быть в *camelCase*. Переменные класса на конце должны иметь суффикс в виде нижнего подчёркивания.

Имена констант

Константы должны начинаться с буквы *k*, за которой следует имя в *PascalCase.*

Имена функций

Функции также как и переменные должны быть в *camelCase*. Кроме того публичные функции должны начинаться с буквы ‘q’ (qRgb);

5 Комментарии

Несмотря на то, что написание комментариев требует сил и времени, они являются абсолютно необходимыми для поддержания читаемости кода. Следующие правила описывают что и как необходимо комментировать. Но помните: хотя комментарии очень важны, лучший код – это самодокументируемый код. Давая понятные имена типам и переменным вы

вкладываете больший смысл, чем используя неясные наименования, которые затем приходится объяснять в комментариях.

Стиль комментариев

Используйте синтаксис // или /* */ для комментариев, однако // является наиболее распространённым.

Комментарии в файлах

В начале каждого файла приводите лицензионный шаблон, за которым должно следовать описание содержимого.

Комментарии классов

Каждое объявление класса должно иметь сопутствующий комментарий, который описывает его назначение и способы использования.

Комментарии функций

Комментарий в объявлении должен описывать назначение функции; комментарии к реализации – описывать выполняемые операции.

Комментарии к переменным

В общем случае, само имя переменной уже должно содержать описание назначения. В некоторых ситуациях допускается использование явных комментариев.

Комментарии в реализации

Комментарии в реализации должны относится к сложным, неочевидным, интересным или важным участкам вашей программы.

Пунктуация, правописание и грамматика

Обращайте внимание на пунктуацию, написание и грамматику. Правильно написанные комментарии читаются легче плохо написанных.

6 Форматирование

Длина строки

Каждая строка текста вашего кода не должна превышать 100 символов. Обрывайте их, если это необходимо.

При переносе строк операторы следует переносить в начало новой строки, поскольку оператор в конце строки очень просто не увидеть, если ваш редактор слишком узкий.

Символы не-ASCII

Символы в кодировке не-ASCII должны использоваться крайне редко и должны быть в кодировке UTF-8.

Пробелы или табуляция

Для выделения блоков программного кода используйте отступ в 4 пробела. Не

используйте табуляцию. Кроме того:

- используйте пробелы для группировки отдельных сегментов кода;
- для разделения можно использовать только одну пустую линию;
- всегда используйте только один, и только один пробел после ключевого слова и фигурной скобки;
- для указателей и ссылок, всегда используйте один пробел между типом и ‘*’ или ‘&’, но никогда не ставьте пробел между ‘*’ или ‘&’ и названием переменной;
- окружайте бинарные операторы пробелами;
- никаких пробелов после приведения типов.

Объявление и реализация функций

Возвращаемый тип должен располагаться в одной строке с именем функции, параметры следует также располагать в одну строку, если они умещаются.

Вызовы функций

Располагайте в одну строку, если они умещаются. Иначе переносите аргументы.

Фигурные скобки

Скобки, как и пробелы, играют существенную роль во внешнем виде и читабельности кода. В качестве основного правила, левая фигурная скобка должна находиться на той же строчке, что и оператор, кроме реализации функций и объявления классов, где левую скобку всегда располагают на новой строке.

- используйте фигурные скобки, когда тело условия содержит более одной линии, и если однострочный оператор — что-то комплексное;
- исключение 1: Также используйте фигурные скобки, если родительский оператор занимает несколько линий или оберток;
- исключение 2: Также используйте фигурные скобки в *if*-*else* блоках где *if*-код или *else*-код занимает несколько линий;
- используйте фигурные скобки когда тело оператора — пустое.

Условия

Предпочтительно не использовать пробелы внутри скобок. Ключевое слово *else* должно располагаться на новой строке.

Логические выражения

Если у вас есть выражение, длина которого превышает максимальный размер строк, будьте внимательны, когда разделяете выражение на несколько строк.

Возвращаемые значения

Не используйте без необходимости круглые скобки вокруг возвращаемого выражения.

Директивы препроцессора

Символ #, который обозначает директиву препроцессора, всегда должен располагаться в начале строки.

Форматирование класса

Наименование секций *public*, *protected* и *private* должно располагаться на одном уровне под ключевым словом *class*.

Ссылки

1 http://www.literateprogramming.com/wildfire.pdf - Wildfire C++ Programming Style

2 http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml - Google C++ Style Guide

3 http://qt-project.org/wiki/Qt_Coding_Style - Qt Coding Style

4 http://qt-project.org/wiki/Coding-Conventions - Qt Coding Conventions

Приложение Г

(обязательное)

Реляционная модель базы данных обработки телеметрической информации

Результаты проектирования базы данных обработки телеметрической информации представлены в виде реляционной модели на рисунке Г.1.

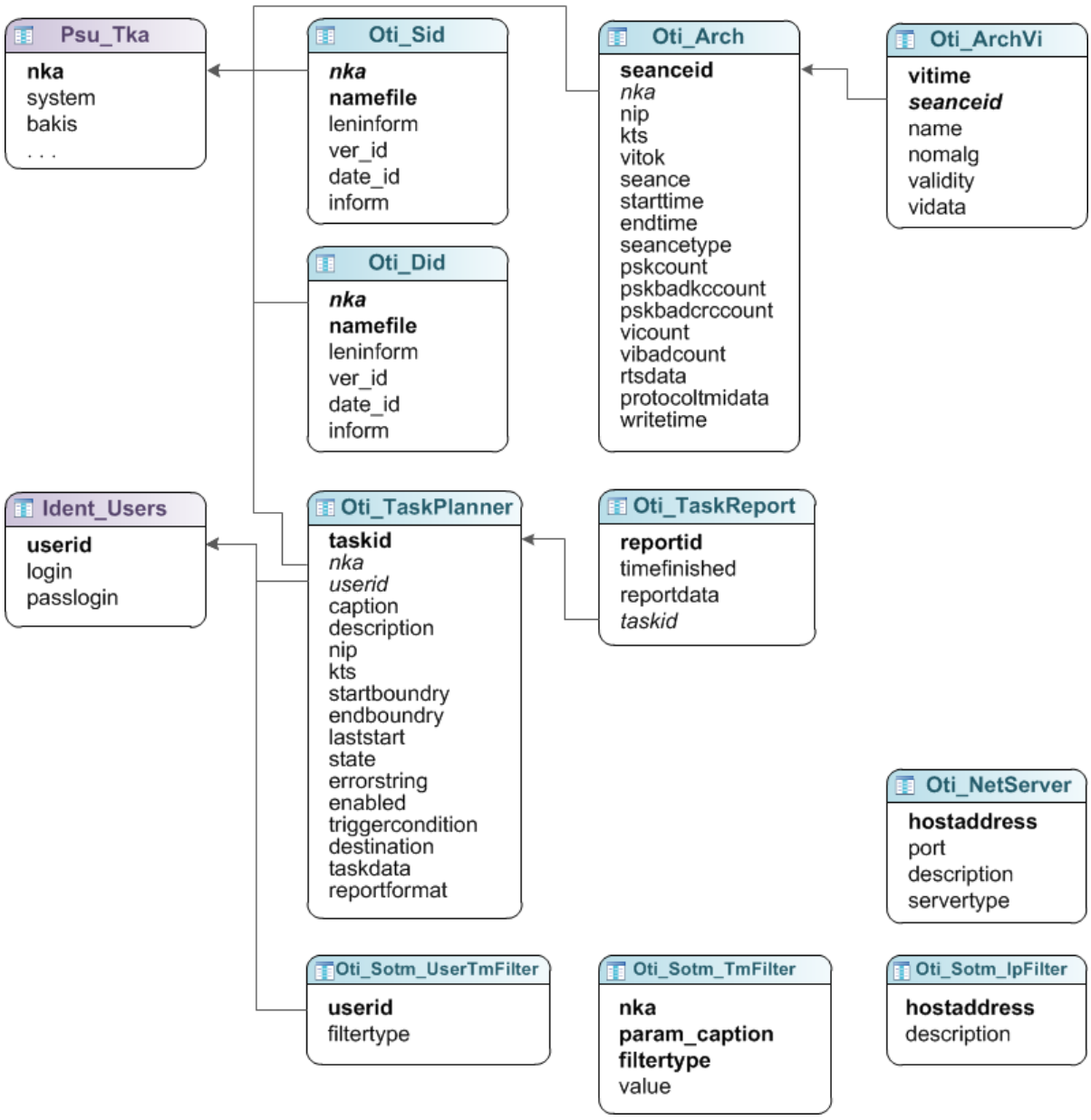

Рисунок Г.1 – Реляционная модель базы данных обработки телеметрической информации

На приведённом рисунке синим цветом обозначены сущности, участвующие в задачах обработки телеметрической информации, а фиолетовым служебные сущности центральной базы данных. Ниже приведены подробные описания каждой выделенной сущности.

Таблица Г.1 – Таблица космических аппаратов Psu_Tka (служебная таблица)

Поле	**Тип данных**	**Ограничение**	**Описание**
nka	number(4)	Primary key	Номер КА
system	number(4)	Not null	Индекс системы
bakis	char(20)		Тип бортовой КИС

Таблица Г.2 – Таблица описания символьных исходных данных Oti_Sid

Поле	**Тип данных**	**Ограничение**	**Описание**
Nka	Number(4)	Primary key	Номер КА
Namefile	Varchar2(250)		Наименование файла
Leninform	Number(8)	Not null	Размер файла в байтах
ver_id	Number(4)	Not null	Версия
date_id	Date	Not null	Дата формирования файла
Inform	Blob	Not null	Текстовые данные файла

Атрибут *nka* также является внешним ключом для связи с таблицей *Psu_Tka*.

Таблица Г.3 – Таблица описания символьных исходных данных Oti_Did

Поле	**Тип данных**	**Ограничение**	**Описание**
Nka	Number(4)	Primary key	Номер КА
namefile	Varchar2(250)		Наименование файла
leninform	Number(8)	Not null	Размер файла в байтах
ver_id	Number(4)	Not null	Версия
date_id	Date	Not null	Дата формирования файла
inform	Blob	Not null	Двоичные данные файла

Атрибут *nka* также является внешним ключом для связи с таблицей *Psu_Tka*.

Таблица Г.4 – Таблица телеметрических архивов Oti_Arch

Поле	**Тип данных**	**Ограничение**	**Описание**
seanceid	Number(12)	Primary key	Идентификатор сеанса связи
nka	Number(4)	Foreign key	Номер КА
nip	Number(4)		Номер НИП
kts	Number(4)		Номер КТС
vitok	Number(8)		Номер витка
seance	Number(2)		Номер сеанса
starttime	Date	Not null	Время начала сеанса
endtime	Date	Not null	Время окончания сеанса
seancetype	Number(2)	Not null	Тип сеанса (0 – Сао, 1 – Соти)
pskcount	Number(6)	Not null	Количество кадров
pskbadkccount	Number(6)		Количество сбоев по командному слову
pskbadcrccount	Number(6)		Количество сбоев по контрольной сумме
viscount	Number(6)	Not null	Количество отчётов БЦВК
vibadcount	Number(6)		Количество сбойных БЦВК отчётов
rtsdata	Blob	Not null	Телеметрический архив
protocoltmidata	Blob		Исходный телеметрический поток принятый из сети
writetime	Date	Not null	Время записи архива в базу

			данных

Атрибут *nka* является внешним ключом для связи с таблицей *Psu_Tka.*

Таблица Г.5 – Таблица отчётов БЦВК (с привязкой к архиву) Oti_ArchVi

Поле	**Тип данных**	**Ограничение**	**Описание**
vitime	Date	Primary key	Время приёма отчёта
seanceid	Number(12)		Идентификатор сеанса
name	Varchar2(10)	Not null	Наименование
nomalg	Number(5)	Not null	Номер алгоритма обработки
validity	Number(5)	Not null	Признаки достоверности
vidata	Blob	Not null	Тело отчёта

Атрибут *seanceid* также является внешним ключом для связи с таблицей *Oti_Arch.*

Таблица Г.6 – Таблица учётных записей пользователей Ident_Users (служебная таблица)

Поле	**Тип данных**	**Ограничение**	**Описание**
userid	Number(9)	Primary key	Идентификатор пользователя
login	Varchar2(20)	Not null	Имя учётной записи
passlogin	Varchar2(20)	Not null	Пароль

Таблица Г.7 – Таблица запланированных задач для системы автоматизированной подготовки отчётов о состоянии ТМ-параметров телеметрических архивов Oti_TaskPlanner

Поле	**Тип данных**	**Ограничение**	**Описание**
taskid	Number(4)	Primary key	Идентификатор задачи
userid	Number(4)	Foreign key	Идентификатор пользователя
nka	Number(4)	Foreign key	Номер КА
caption	Varchar2(255)	Not null	Название задачи
description	Varchar2(255)		Описание
nip	Number(4)		Номер НИП
kts	Number(4)		Номер КТС
startboundry	Date		Время начала
endboundry	Date		Время окончания
laststart	Date		Время последнего запуска
state	Number(1)		Состояние
errorstring	Varchar2(255)		Текст ошибки
enabled	Number(1)	Not null	Признак включенности
Triggercondition	Varchar2(255)		Триггерные условия
destination	Number(1)	Not null	Адрес сохранения результатов
taskdata	Blob	Not null	Тело задачи
reportformat	Number(1)	Not null	Форм отчёта (0 - html; 1 - pdf; 2 – xml)

Атрибут *userid* является внешним ключом для связи с таблицей *Ident_Users*, атрибут *nka* является внешним ключом для связи с таблицей *Psu_Tka.*

Таблица Г.8 – Таблица отчётов о выполнении задач системы автоматизированной подготовки отчётов Oti_TaskReport

Поле	**Тип данных**	**Ограничение**	**Описание**
reportid	Number(4)	Primary key	Идентификатор отчёта
timefinished	Date	Not null	Время окончания обработки

reportdata	Blob	Not null	Тело отчёта
taskid	Number(4)	Foreign key	Идентификатор задачи

Атрибут *taskid* является внешним ключом для связи с таблицей *Oti_TaskPlanner*.

Таблица Г.9 – Таблица фильтрации учётных записей пользователей Oti_Sotm_UserTmFilter

Поле	**Тип данных**	**Ограничение**	**Описание**
userid	Number(9)	Primary key	Идентификатор учётной записи
filtertype	Number(1)	Not null	номер назначенного фильтра

Атрибут *userid* также является внешним ключом для связи с таблицей *Ident_Users.*

Таблица Г.10 – Таблица фильтрации учётных записей пользователей Oti_Sotm_TmFilter

Поле	**Тип данных**	**Ограничение**	**Описание**
nka	Number(4)	Primary key	Номер КА
param_caption	Varchar2(20)		Индекс ТМ-параметра
filtertype	Number(1)		номер фильтра
value	Number(8)	Not null	Устанавливаемое значение ТМ-параметра

Атрибут *nka* также является внешним ключом для связи с таблицей *Psu_Tka.*

Таблица Г.11 – Таблица фильтруемых IP-адресов Oti_Sotm_IpFilter

Поле	**Тип данных**	**Ограничение**	**Описание**
hostaddress	Varchar2(30)	Primary key	IP-адрес
description	Varchar2(50)		Описание фильтруемого узла

Таблица Г.12 – Таблица IP-адресов, номеров портов различных источников телеметрии Oti_NetServer

Поле	**Тип данных**	**Ограничение**	**Описание**
hostaddress	Varchar2(30)	Primary key	IP-адрес источника
port	Number(5)	Not null	Порт источника
description	Varchar2(50)		Описание источника
servertype	Number(5)	Not null	Тип источника (0 – САО, 1 – Соти, 2 - СОТМ)

Приложение Д

(обязательное)

Акты внедрения результатов научной работы

«УТВЕРЖДАЮ»
Главный конструктор управления
и эксплуатации КА и систем
ОАО «Информационные спутниковые системы»
имени академика М.Ф. Решетнёва
В.П. Ковалев
11.03 2011 г

АКТ ВНЕДРЕНИЯ

результатов научной работы Некрасова М.В.

Настоящий акт составлен в том, что результаты научной работы «Создание многопоточной системы обработки и анализа телеметрической информации в центрах управления полётами космических аппаратов», аспиранта Некрасова М.В. используются при решении задачи автоматизированного управления орбитальными группировками в центрах управления полётами космических аппаратов Глонасс, Гео-ИК-2.

Разработанный комплекс программ многопоточного приёма, обработки и анализа телеметрической информации (Свидетельства №2010611073, №2010611073, №2010614028) внедрён в информационно-вычислительный комплекс генерального конструктора ОАО «Информационные спутниковые системы» имени академика М.Ф. Решетнёва и позволил повысить эффективность принятия решений при управлении космическими аппаратами.

И.о. начальника отдела
Проектирования автоматизированных
систем управления и эксплуатации
орбитальной группировки КА С.В. Юксеев

«УТВЕРЖДАЮ»
Главный конструктор разработки
космических комплексов (систем)
координатно-метрического назначения,
наземных комплексов управления и
баллистического обеспечения
ОАО «Информационные спутниковые
системы»
имени академика М.Ф. Решетнёва
_____ С.В. Сторожев
«__» 03. 2014 г

АКТ
об использовании результатов
диссертационного исследования Некрасова М.В.

Настоящий акт составлен в том, что результаты диссертационного исследования «Автоматизированная система многопоточного приёма, обработки и анализа телеметрической информации» аспиранта Некрасова М.В. используются при решении задач автоматизированного управления орбитальными группировками (ОГ) в центрах управления полётами (ЦУП) КА Экспресс-АМ, Экспресс-АТ, Луч-5В.

Специалисты ЦУП КА по обработке и анализу телеметрической информации дают высокую оценку разработанной многопоточной системе, отмечают повышенную надёжность, расширенные функциональные возможности, увеличение производительности расчётов и сокращение количества регулярных ручных операций, в результате чего снизилось общее число ошибок операторов.

Разработанный комплекс программ многопоточного приёма, обработки и анализа телеметрической информации позволил качественно улучшить систему приёма, обработки и анализа телеметрической информации и повысить эффективность принятия решений при управлении АСУ ОГ КА в целом.

Начальник отдела
проектирования автоматизированных
систем управления и разработки наземных
комплексов управления КА Д.Н. Рыжков