УДК 007.03

Использование информационных технологий для поддержки разработки бортового программного обеспечения

Щелыкалин М.Ю.

Московское опытно-конструкторское бюро «Марс»,

1-ый Щемиловский пер., 16, Москва, 127473, Россия

e-mail: schelikal@gmail.com

Аннотация

Работа посвящена созданию программных комплексов поддержки разработки бортового программного обеспечения (БПО) космическими аппаратами (КА). Рассматривается эволюция систем, используемых при проектировании, планировании, разработке и отладке БПО КА. Продемонстрированы изменения в процессе разработке БПО вызванные внедрением системы поддержки разработки.

Ключевые слова: разработка бортового программного обеспечения, системы поддержки разработки, управление процессом разработки, испытания бортового программного обеспечения, Subversion, Git, Redmine, Bugzilla, Trac.

Рассматриваются вопросы автоматизации процесса разработки бортового программного обеспечения (БПО) космических аппаратов (КА). БПО КА, являясь сложным иерархическим программным комплексом, предназначенным для управления большим количеством разнообразной бортовой аппаратуры в

разнообразных в том числе нештатных, условиях. И одновременно БПО должно удовлетворять высочайшим требованиям по надежности. В условиях ускорения освоения космического пространства увеличивается количество и разнообразие разрабатываемого БПО. Возросшее количество одновременно разрабатываемых систем существенно увеличивает объем экспериментальной отработки и испытаний и, соответственно, требует широкого применения средств автоматизации, основанных на современных информационных технологиях [3].

Для решения поставленных задач необходимо проанализировать существующий процесс разработки БПО и безболезненно модернизировать его путём внедрения информационных систем поддержки разработки БПО.

1. Описание существующего процесса

1.1. Процесс разработки бортового программного обеспечения

Изучим процесс разработки бортового программного обеспечения (БПО) (Рис. 1), сложившийся на МОКБ «Марс» во время разработки бортовых систем управления таких изделий как: космический корабль Буран, разгонный блок «Бриз-М», космический аппарат (КА) «Монитор-М», КА «KazSat», КА «KazSat-2», КА «Электро-Л № 1», КА «Спектр-Р»[1][1].

В работе рассматривается процесс разработки БПО от получения технического задания (ТЗ) до признания БПО удовлетворяющим требованиям ТЗ. После получения ТЗ начинается взаимосвязанная работа по нескольким направлениям разработки:

1. Разработка бортового программного обеспечения (БПО). Включает в себя уточнение схемы деления системы управления – выделение отдельных целевых подсистем, отвечающих за обработку информации с приборов, за стабилизацию объекта управления, ориентацию в пространстве, межпроцессорный и межприборный обмен, телеметрирование информации. Разработка непосредственно алгоритмов управления, их отладка, проверка на испытательных стендах различных уровней детализации.

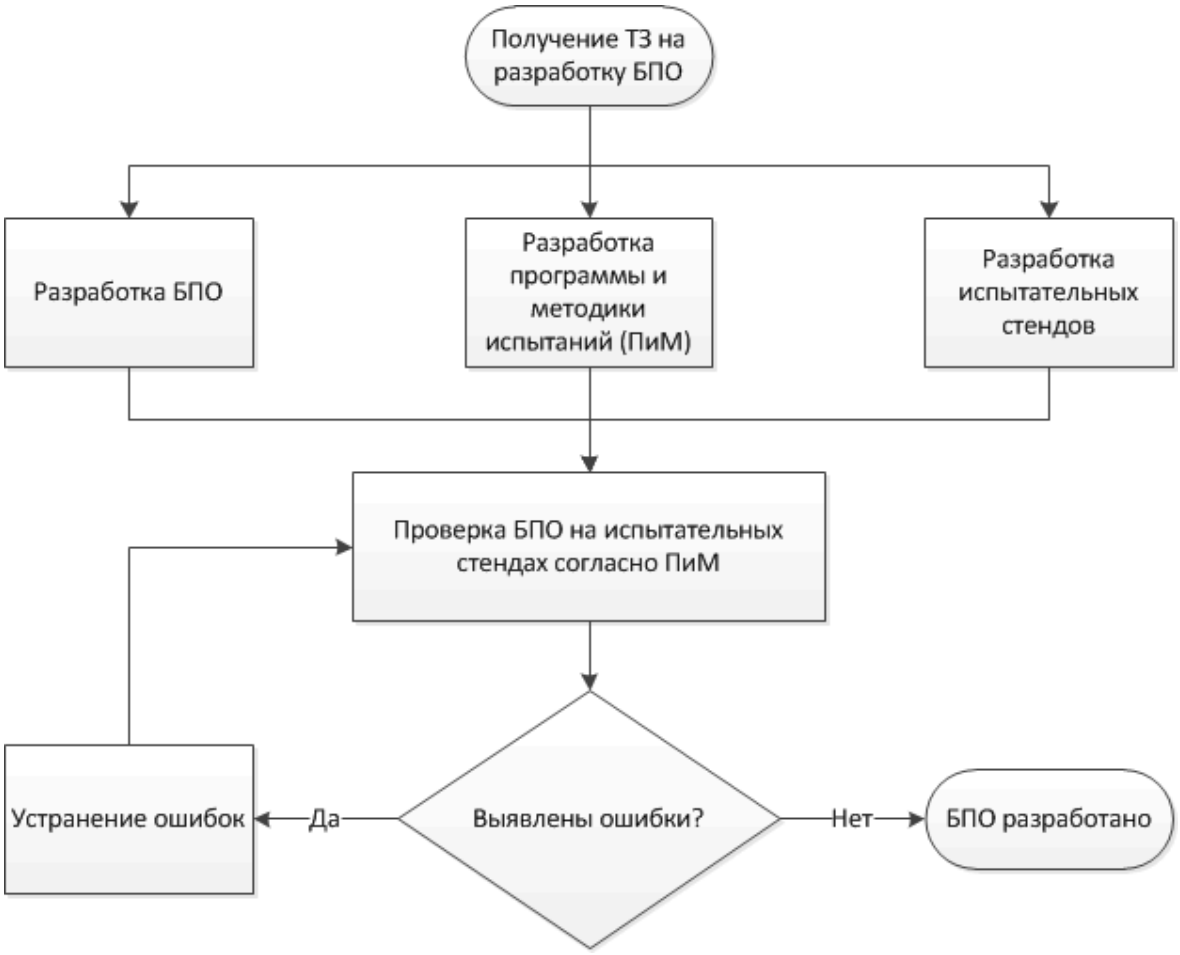

Рис. 1. Укрупнённый процесс разработки БПО

2. Разработка программы и методики испытаний (ПиМ) – выделение из ТЗ перечня режимов работы управляемого объекта, создания перечня исходных данных необходимых для запуска по каждому режиму работы, создания комплексов циклограмм (ЦГ).

3. Разработка испытательных стендов. Включает в себя работы по уточнению исходных данных по приборам. Создание нескольких моделей различного уровня детализации: более подробные для повышения качества проверки и упрощённые, но достаточно достоверные для отладки основных алгоритмов работы. Разработку системного программного обеспечения (СПО), призванного увязать вместе набор моделей различной детализации между собой или с реальными приборами. СПО разрабатывается индивидуально для каждого испытательного стенда.

Проверка работоспособности БПО проводится на испытательных стендах. Рассмотрим особенности данных стендов.

1.2. Испытательные стенды

Важной частью рассматриваемого процесса являются испытательные стенды, предназначенные для проведения имитационного моделирования режимов работы КА входящих в ПиМ. В рамках данной работы затронуты вопросы взаимодействия со следующими стендами:

а) Комплексный математический стенд (КМС) – стенд работающий на персональном компьютере. На стенд поступает задание на моделирование, содержащее циклограмму моделируемого режима, название версии БПО,

командно-программную информацию (КПИ). Результатами работы данного стенда является телеметрическая информация моделируемого процесса. Благодаря тому, что стенд работает на одном компьютере и возможностям компилятора, получается, проводить моделирование существенно быстрее реального времени.

б) Автоматизированный цифровой комплекс (АЦК) – испытательный стенд высокого уровня детализации, содержит в своём составе реальный бортовой вычислитель и математические модели для имитации отсутствующих реальных компонентов КА. Задание на моделирование для данного стенда содержит исходные данные для начала моделирования, циклограмму режима, название версии БПО, командно-программную информацию (КПИ). По результатам моделирования на стенде получается телеметрическая информация. Моделирование идёт в режиме реального времени: среднее время моделирования одного режима составляет примерно 4-8 часов, есть отдельные режимы, длительность моделирование которых больше одного дня.

2. Выявление недостатков в процессе разработки БПО

По результатам анализа было решено обратить внимание на проверку БПО на испытательных стендах согласно ПиМ, выявление ошибок, процесс их устранения (Рис. 1), а именно соответствующие им информационное взаимодействие.

Традиционно обмен данными происходил следующим образом: ответственный сотрудник собирал информацию, необходимую для начала моделирования на

стендах, и передавал на стенды. Он же вёл учёт промоделированных режимов, хранил результаты моделирования и передавал их на анализ разработчикам, после чего собирал результаты и делал заключение о результатах моделирования.

Данные на испытательные стенды поступали следующим образом: приносилась служебная записка с описанием циклограммой режима и носитель информации с исходными данными для начала моделирования. Результаты моделирования со стенда КМС возвращались на том же носителе информации или по внутренней электронной почте. Результаты моделирования со стенда АЦК выкладывались на ftp-сервер. Оповещение разработчиков БПО проводилось по внутренней электронной почте или телефонным звонком. Далее результаты объединялись, и делалось итоговое заключение о нормативности моделирования.

В ходе рассмотрения были выявлены следующие недостатки существующего процесса испытаний БПО:

2.1. При переносе циклограммы режима с бумажного носителя в электронный вид, специфичный для испытательного стенда, происходили ошибки, регулярно приводящие к тому, что 1-2 дня в месяц стенд АЦК работал вхолостую, а разработчики БПО тратили лишнее время на обнаружение ошибок, возникших из-за не верно введённой ЦГ.

2.2. Хранение результатов моделирования в максимально полном виде осуществлялось только на рабочем месте ответственного за испытания. Другие разработчики получали результаты только от ответвлённо за испытания, когда он

есть на месте. Хранение данных осуществлялось согласно логике организованный конкретным ответственным и различалось от человека к человеку.

2.3. Происходили задержки при оповещении участников разработки бортового программного обеспечения (БПО) о необходимости анализа полученной во время моделирования телеметрической информации (ТМИ), задержки при распространении информации через электронную почту. Эти задержки повторяются во время всего процесса анализа результатов телеметрии, сбора и объединения этих результатов.

2.4. Не прозрачность текущего состояния отработки ПиМ для остальных участников отладки. Текущее состояние можно было узнать только из личного общения с ответственным за испытания. Иногда даже ответственный за испытания был не до конца уверен в том, что тот или иной моделируемый режим был отработан на стенде и получил подтверждение нормативности работы от разработчиков БПО именно с последней версией БПО.

2.5. Недостаточный учет выявленных в ходе испытания замечаний и процесса их исправления.

2.6. Хранение полного набора всех версий математических моделей только на компьютере разработчика моделей. Что приводило к трудностям с отслеживанием изменений в моделях и определению последней версии математических моделей.

3. Анализ возможностей автоматизации

В ходе анализа недостатков выявленных в разделе 2, было принято решение объединить их:

1.1. Недостатки 2.1 – 2.4 объединить и реализовать для их решения самостоятельную информационную систему автоматизации испытаний (САИ), учитывающую специфику разработки БПО. На систему возлагаются задачи по передаче данных о разрабатываемом БПО, содержании ПиМ на испытательные стенды. Преобразованию этих данных в формат, ожидаемый на конкретном стенде. Получении результатов моделирования со стендов. Создания хранилища максимально полно освещающего прошедшие моделирования на стендах и их результаты.

1.2. Недостаток 2.5 рассмотреть отдельно. Проверить возможность использования для его решения, созданную сторонними разработчиками стандартизированную систему управления ошибками.

1.3. Недостаток 2.6 рассмотреть отдельно. Проверить возможность использования для его решения, готовую стандартизированную систему контроля версий.

4. Создание системы автоматизации испытаний

Для создания системы автоматизации испытаний (САИ) потребовалось формализовать существующий процесс проведения испытаний. Далее ведя итеративную

разработку с одновременным внедрением создаваемой системы и обучением пользователей, были пройдены следующие этапы внедрения САИ:

а) Создание системы, получающей исходные данные от ответственного за назначение испытаний, преобразующей их в формат, ожидаемый на испытательном стенде, и передающей данные на стенд в электронном виде. По результатам её внедрение формат данных получаемых на стендах так же был изменён.

б) Сохранение в системе результатов моделирования и исходных данных с однозначной связью между ними в единой системе для всех разрабатываемых объектов типа КА. Предоставление всем специалистам, участвующим в разработке и отладке БПО, свободного доступа к данной с системе. Для чего разработка САИ как клиент-серверного приложения с разделением прав доступа.

в) Создание перечня исходных данных (ИД) передаваемых на стенд для начала моделирования. Формирование базы ИД, для повторного их использования при назначении новых режимов и подключение ей к ранее созданной системе.

г) Создание системы заказа кодирования (СЗК) командно-программной информации (КПИ). Перенос из САИ данные о КПИ во вновь созданную систему. Отладка взаимодействия САИ и новой системой.

д) Создание в рамках САИ модуля хранение результатов анализа. Доработка получившегося модуля функцией отправки оповещения разработчикам о необходимости проведения анализа моделирования испытательных режимов.

е) Проверка возможности управления ошибками в рамках САИ выявила сложность данной системы. Произведённый анализ последних достижений в раз-

работке ПО показал существование готовых решений в области управления ошибками. Исходя из этого было проведено исследование существующих стандартных систем управления ошибками (СУО). По результатам исследования данных систем в САИ было решено хранить только нормативные разультаты анализа прошедшего на стенде испытания и ссылки на систему управления ошибками. Ссылки образовывать в момент ввода ошибки испытателем после передави в СУО информации об ошибке. Выбор системы управления ошибками основанный на обзоре существующих решений и тестирование подходящих вариантов. Установка, отладка описанного взаимодействия между системой автоматизации испытаний и системой управления ошибками.

ж) Выбор системы контроля версий. Создание хранилища математических моделей на основе системы контроля версий с целью упорядочения изменений в математических моделей. Отладка взаимодействия САИ с системой контроля версий.

Рассмотрим более подробно выбор системы контроля версий и системы управления ошибками.

5. Системы контроля версий

Система контроля версий – это ПО, позволяющее хранить версии элементов и работать с этими версиями, как с самостоятельными элементами. В англоязычных источниках используется термин version control systems, сокращенно VCS. Как правило, в качестве элементов выступают файлы либо каталоги [4].

Для того чтобы определиться с тем, какая именно система будет использоваться в работе МОКБ «Марс», были рассмотрены существующие системы контроля версий с открытым исходным кодом. Среди них выбраны две наиболее популярные среди разработчики и имеющие наибольшие сообщества поддерживающие СКВ: Subversion, Git. Они и были установлены на тестовые машины.

5.1. Система контроля версий Subversion

Subversion это свободная централизованная система управления версиями, официально выпущенная в 2004 году компанией CollabNet[5]. Для изучения данной системы использовалось ПО TortoiseSVN.

TortoiseSVN — это бесплатный Windows-клиент с открытыми исходным кодом для системы управления версиями Apache™ Subversion®[6].

Subversion используется многими сообществами разработчиков открытого программного обеспечения. В их числе такие проекты, как Apache, GCC, Free Pascal, Python, Ruby, FreeBSD, AROS, Blender, Boost, Tor, OGRE.

TortoiseSVN имеет удобный интерфейс для пользователей ОС Windows, встраиваемый в оболочку среды. Имеет русскоязычный дружелюбный интерфейс. Достаточно прозрачную систему работы.

Subversion автоматически присваивает каждому изменённому в её репозитории файлу цифровой номер ревизии.

Благодаря хранению всех изменений на сервере имеется возможность всегда найти последнюю версию проекта в хранилище, просмотреть список всех

разработчиков, участвовавших в доработке проекта.

Основным языком, используемым при разработке Subversion, является язык программирования C[5].

На сегодняшний день Subversion является самой популярной системой централизованного управления версиями.

5.2. Система контроля версий Git

Git это распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux. Первая версия Git выпущена 7 апреля 2005 года. На сегодняшний день его поддерживает Джунио Хамано[7].

Примерами проектов, использующих Git, являются ядро Linux, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki и некоторые дистрибутивы Linux. Программа является свободной и выпущена под лицензией GNU General Public License v2 (GPL).

На сегодня есть несколько вариантов графического интерфейса, представляемых различными группами разработчиков:

https://tortoisegit.org/ - англоязычный проект, посвящённый Git,

https://git-scm.com/ - проект, разрабатываемый на английском языке, с документацией, переведённой на несколько языков, в том числе и на русский.

Эксплуатация данных систем выявила несколько проблем:

- Проблемы с кодировкой русского текста.
- Использование Git более своеобразно за счёт разделения локальных изменений и изменений, отправленных в общий доступ.
- Для различения патчей используется SHA-1 хеш, состоящий из 40 шестнадцатеричных знаков (0-9 и a-f)[11]. То есть номер версии файла в Git имеет примерно такой вид: 24b9da6552252987aa493b52f8696cd6d3b00373.

На сегодняшний день Git является самой популярной распределённой системой управления версиями.

5.3. Вывод по системам контроля версии

Таблица 1.

Сравнения характеристик систем контроля версий

Характеристика	**Subversion**	**Git**
Хранение различных версий файлов и комментариев к ним	Да	Да
Отслеживание изменения каталогов	Да	Да
Место хранения репозитория	Сервер	Каждый компьютер
Русскоязычная инструкция	Есть	Есть
Русскоязычный интерфейс	Есть	Есть
Возможность создавать локальные репозитории	Есть	Есть
Различение версий	Номер	40 знаков хеша
Язык разработки	Си	Си

Исходя из стоящих задач, наиболее подходящей была признана система контроля версий Subversion, т.к. она является централизованной и проще в освоении.

6. Системы управления ошибками (СУО)

Системы управления ошибками – это ПО позволяющие фиксировать список выявленных ошибок и видеть, кто нашёл ошибку, кто её исправлял. Позволяющие создавать отчёты и показывать актуальную информацию всем участникам разработки.

Задача регистрации и обработки данных об ошибках, возникших при работе ПО, кажется простой лишь на первый взгляд. В процессе отладки одни ошибки могут заменяться другими, и количество известных ошибок может уменьшаться или увеличиваться. Для контроля ошибок был создан класс ПО – системы управления ошибками или как их ещё называю в англоязычном сообществе Bag Tracker System.

Создание единой открытой информационной системы, прозрачно показывающей всем участникам какие работы сейчас ведутся и предстоят, уже многократно реализована компаниями, являющимися лидерами разработки ПО, такими как Google, Mozilla, Apache.

Создание единого центра отслеживания ошибок позволит использовать подходы по их прогнозированию и сокращению их повторяемости. Многие компании также обнаруживают, что отслеживание ошибок уменьшает издержки, обеспечивая службе поддержки IT систему учета, а всем специалистам - хорошо понятную систему с описанием необычных проблем с системой или программным

обеспечением.

Система управления ошибками может резко увеличить производительность труда и подотчетность работников, обеспечивая автоматизацию работы с документами и положительные отзывы на хороших исполнителей.

Среди существующих СУО были выделены те что распространялись с открытым исходным кодом, для упрощения синхронизации данных с существующими системами и поддерживающих PostgreSQL, используемой на МОКБ. Рассмотрим их более подробно:

6.1. Система управления ошибками Redmine

Redmine является гибким веб-приложение для управления проектами. Управление проектами включает в себя несколько больше возможностей, нежели управление ошибками, но учитывая, что данная система разрабатывалась именно как багтрекер решено рассмотреть её с этой точки зрения. Redmine написан с использованием фреймворка Rubi on Rails, поддерживающее различные платформы и различные базы данных.

Redmine имеет открытый исходный код и распространяется на условиях лицензионного соглашения GNU General Public License v2 (GPL).

Redmine интегрируется с Subversion и Git, поддерживает работу с PostgreSQL, реализует поддержка русского языка [9]

Демо-версию Redmine можно найти по следующему адресу в сети интернет: http://demo.redmine.org/. [9]

Redmine позволяет применять гибкие настройки по организации Workflow[11]. Это позволяет создать единое распределённое рабочее пространство, гибко настроить пути прохождения сообщений, разграничить доступ различных пользователей.

6.2. Система управления ошибками Bugzilla

Bugzilla – это одна из наиболее популярных систем багтрэкинга. В 1998 году Netscape представила первый релиз этой системы. Bugzilla является свободным ПО и распространяется по Mozilla Public License. Разработку этой системы сейчас ведет Mozilla Foundation.

Bugzilla пользуются более 800 компаний по всему миру. Среди них встречаются такие гиганты как NASA, Id Software, Red Hat, Novell и другие [8].

Bugzilla написана на Perl, используется во многих компаниях для разработки собственного программного обеспечения для собственных нужд[12].

Bugzilla может быть легко адаптирована к различным ситуациям. Развернутые инсталляции обеспечивают поддержку очередей обработки запросов в службу поддержки IT, управление внедрением системы менеджмента, отслеживание проблем возникающих с дизайном и разработке чипов (как до, так и после их производства), отслеживание проблем возникающих при разработке программного и аппаратного обеспечения такими грандами, как Redhat, Loki software, Linux-

Mandrake и VA Systems. В связке с такими системами как CVS, Bonsai, или Perforce SCM, Bugzilla обеспечивает мощное, лёгкое в использовании решение для управления конфигурацией и проблемами, возникающими при репликации.

6.3. Система управления ошибками Trac

Trac — средство управления проектами и отслеживания ошибок в программном обеспечении, разработанное Edgewall Software на языке программирования Python[13].

Trac является ПО с открытым исходным кодом, разработанным и поддерживаемым компанией Edgewall Software.

Trac использует минималистичный веб-интерфейс, основанный на технологии Wiki. Позволяет организовать перекрёстные гиперссылки между базой данных зарегистрированных ошибок, системой управления версиями и вики-страницами. Это даёт возможность использовать Trac в том числе и как веб-интерфейс для доступа к системе контроля версий Subversion и Git, а также, через плагины, к Mercurial, Bazaar и другим.

Системой Trac поддерживаются следующие базы данных: SQLite, PostgreSQL, MySQL и MariaDB.

Trac написан на языке программирования Python и в настоящее время распространяется по модифицированной лицензии BSD. В качестве системы HTML-шаблонов веб-интерфейса Trac до версии 0.11 использовал ClearSilver. Новые

версии, начиная с 0.11, используют разработанную в Edgewall систему шаблонов Genshi [13], при этом совместимость с плагинами, использующими ClearSilver, будет оставлена ещё в течение нескольких версий.

6.4. Вывод по системам управления ошибками

В ходе данного исследования рассматривались системы управления ошибками, работающие в качестве веб-приложения и поддерживающие работу с СУБД PostgreSQL.

Таблица 2.

Сравнение характеристик различных систем управления ошибками.

Характеристика	**Redmine**	**Bugzilla**	**Trac**
Поддержка нескольких проектов	Да	Да	Нет
Поддержка подпроектов	Да	Нет	Нет
Возможность регистрации и отслеживания ошибок	Да	Да	Да
Оповещение о наблюдаемых ошибках по электронной почте	Да	Да	Да
Поддержка Subversion	Да	Да	Да
Поддержка Git	Да	Нет	Да
Веб приложение	Да	Да	Да
Поддержка СУБД PostgreSQL	Да	Да	Да
Язык разработки	Ruby	Perl	Python

Исходя из данных представленных в таблице 2 видно, что Redmine обладает наибольшим набором возможностей. Поэтому было решено остановить свой выбор на системе управления ошибками Redmine.

7. Организация взаимодействия средств автоматизации разработки БПО

В результате проведённого анализа в созданную систему автоматизации испытаний были интегрированы: система контроля версий Subversion и система управления ошибками Redmine. Были разработаны и реализованы протоколы взаимодействия данных систем. Так же были реализованы протоколы взаимодействия получившегося программного комплекса с другими разработанными на МОКБ системами:

- Системой заказов на кодирование КПИ
- Системой автоматизации проектирования функционального программного обеспечения (САПР БПО)
- Обработчиками телеметрической информации (ОТМИ).

СЗК КПИ – предназначена для кодирования и хранения КПИ. В состав СЗК КПИ входит система заказа файлов с КПИ, автоматические модули создания файлов и записи их в архив закодированных КПИ.

САПР ФПО – система осуществляет сбор коррекций алгоритмов от разработчиков бортового ПО, так же помогает в создании сборок БПО и хранит их перечень.

ОТМИ – это набор ПО призванного помочь в анализе результатов моделирования. До появления САИ этот класс ПО работал автономно. После ПО данного класс стало интегрироваться в САИ с целью автоматизации анализа результатов моделирования.

Схему взаимодействия указанных разнородных систем можно увидеть на Рис. 2. Рассмотрим более подробно информационный обмен САИ с остальными системами. Во время проведения испытаний БПО САИ:

1.1. взаимодействует с системой СЗК КПИ в части закодированных КПИ. Взаимодействие осуществляется через модуль САИ, проводящий получение перечня закодированных КПИ с указанием сопровождающей информации из специально сформированного в СЗК табличного представления. Выборка данных просидит по средствам SQL-запросов.

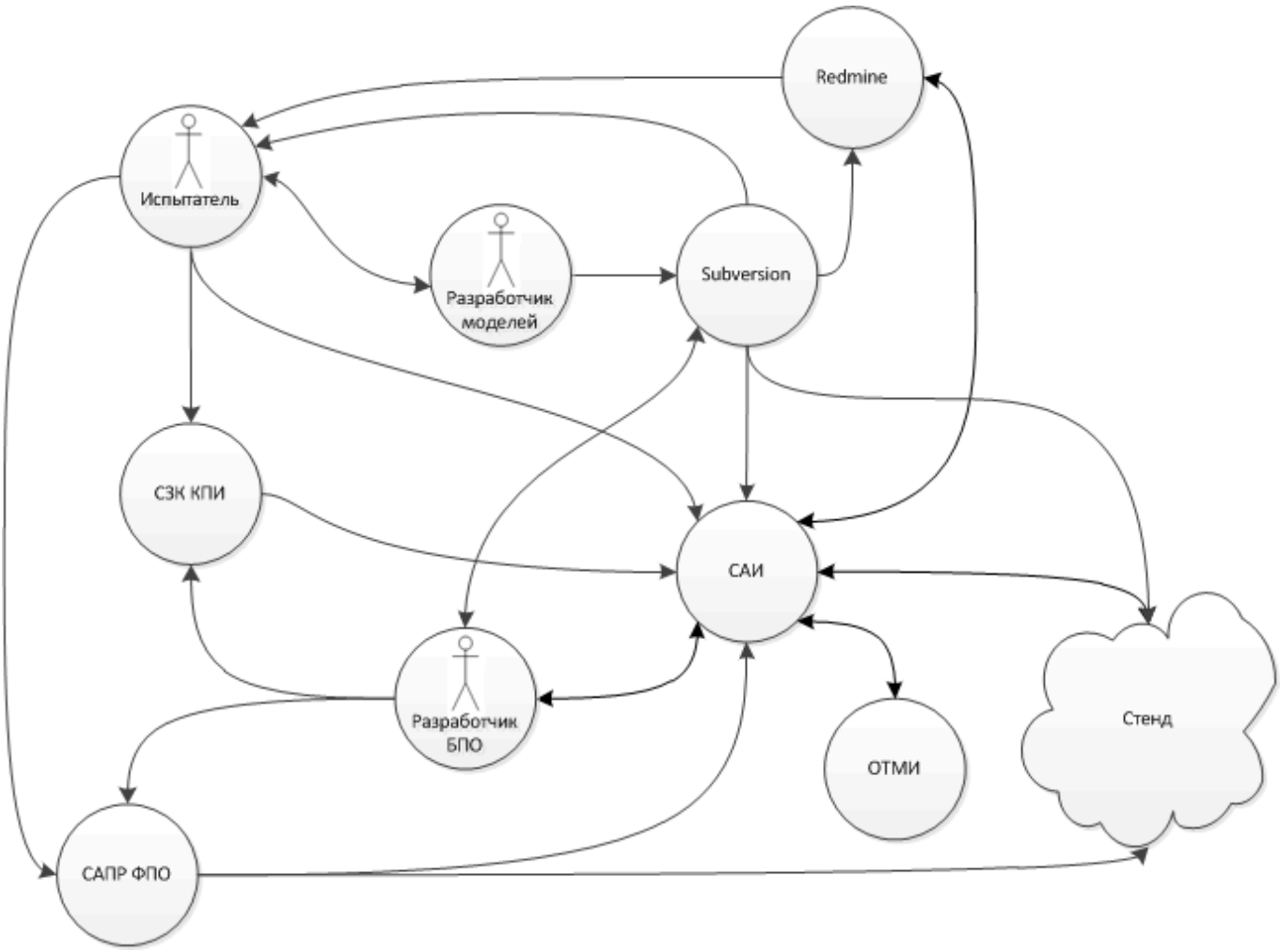

Рис. 2. Структура взаимодействия внедрённых систем

1.2. взаимодействует с испытателями в части получения от них программы и методики испытаний, исходных данных для начала моделирования на стендах. Взаимодействие осуществляется через клиент САИ установленный на рабочем месте ответственных за проведение испытаний.

1.3. взаимодействует с ПО Subversion в части получения номеров версий математических моделей. Взаимодействие осуществляется через модуль САИ, проводящий получение перечня версий при помощи стандартного API Subversion.

Взаимодействие с API Subversion реализовано через консольные команды.

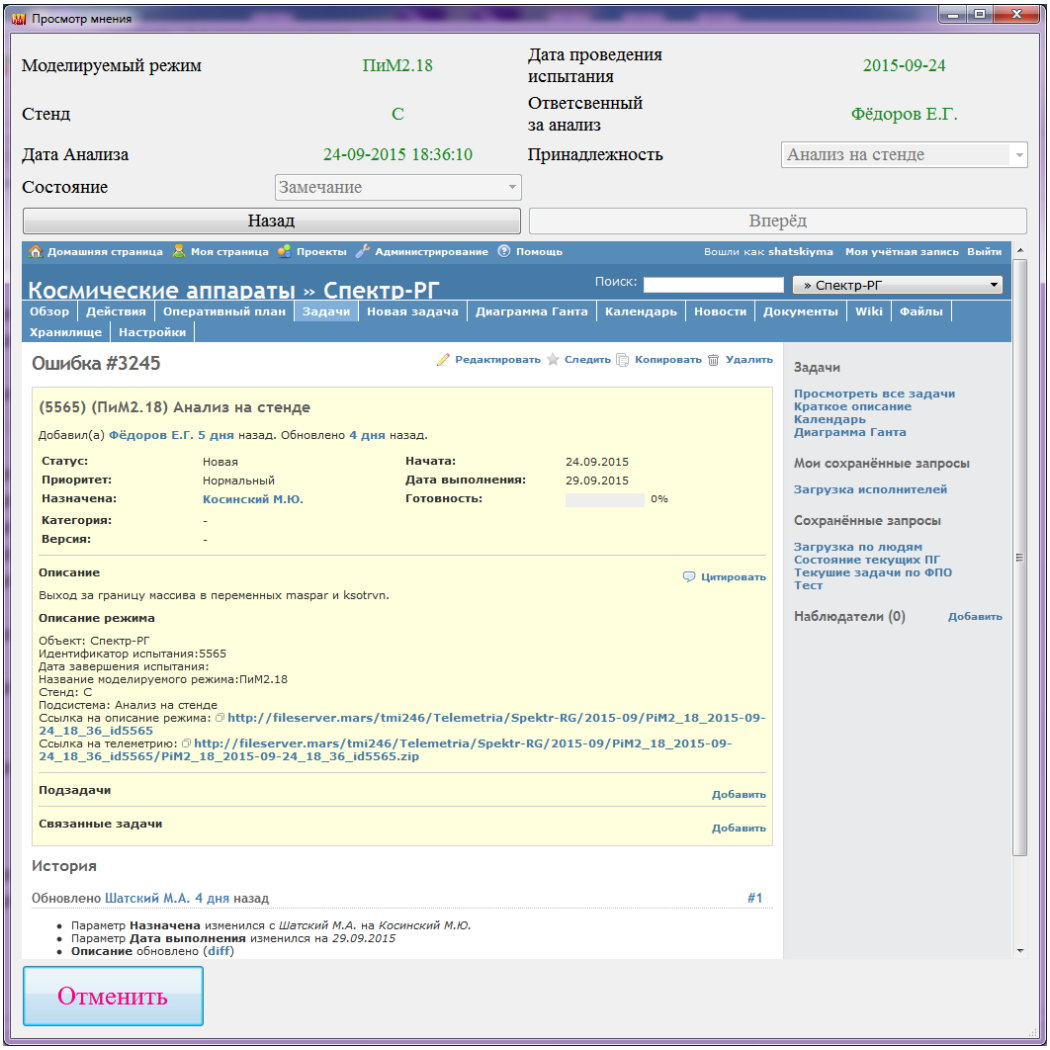

Рис. 3. Пример показа подробной информации в САИ из Redmine

1.4. взаимодействует с ПО Redmine в части получения данных о текущем состоянии ошибок, ранее выявленных в ходе отработки БПО и переданных из САИ в Redmine. Для реализации данных функций была изучена схема данных Redmine, найдены таблицы, хранящие в себе данные о состоянии ошибок. После чего реализован модуль САИ получающий данные по средствам SQL-запросов из таблиц Redmine. Так же в САИ при помощи движка WebKid реализован просмотр полной информации о ходе исправления задачи в Redmine(Рис. 3).

1.5. взаимодействует с испытательными стендами в части получения

результатов моделирования по ранее заказанным режимам. Взаимодействие осуществляется через интерфейс клиентов САИ установленных на испытательных стендах (Рис. 4, Рис. 5).

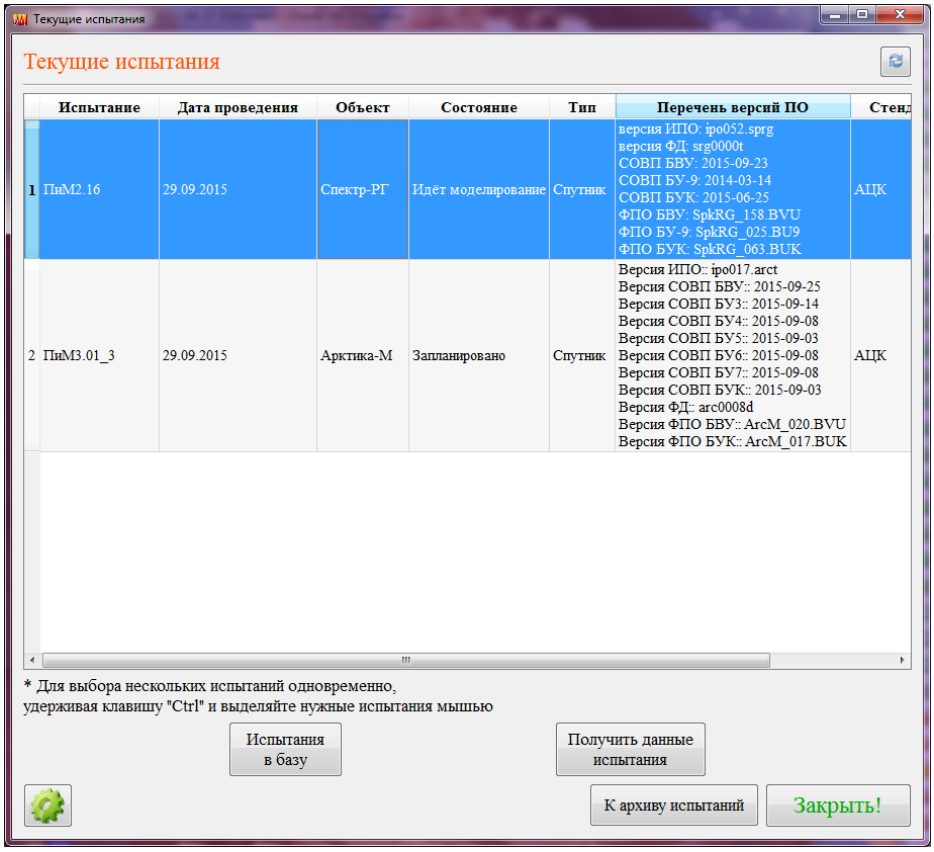

Рис. 4. Пример интерфейса САИ с информацией о переданных на стенд режимах

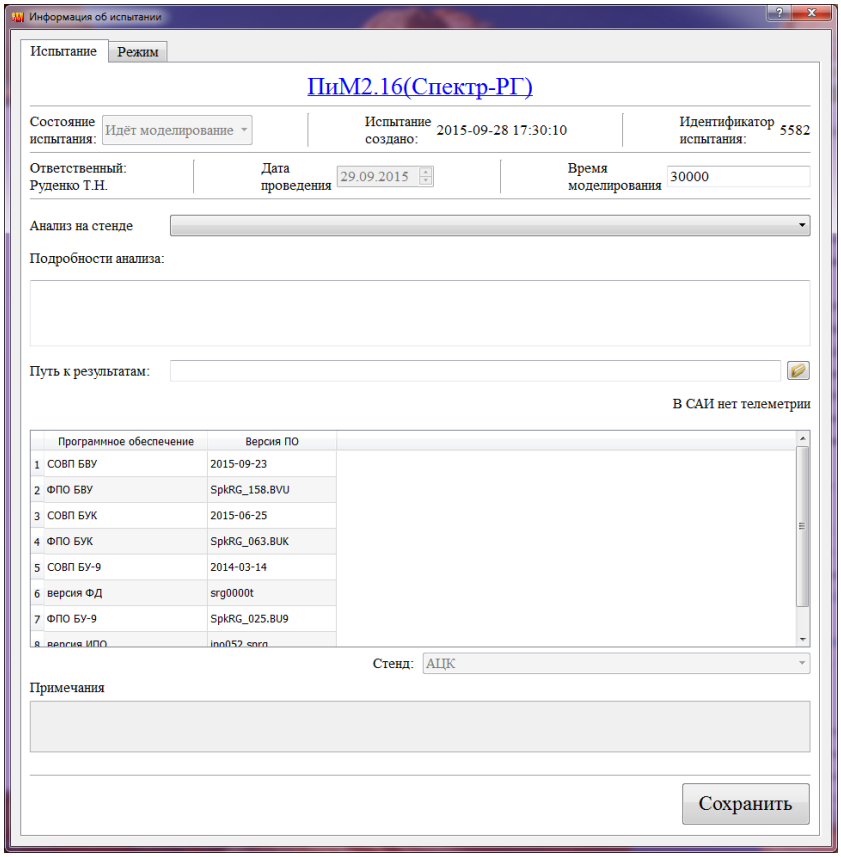

Рис. 5. Форма для ввода результатов моделирования в САИ со стендов

1.6. взаимодействует с ПО ОТМИ в части оценки нормативности моделирования проведшего на стенде. Был разработан универсальный протокол взаимодействия САИ и ОТМИ. Взаимодействие осуществляется через командную строку. ОТМИ, доработанные для взаимодействия с САИ, интегрируются в последнюю как подключаемые модули. В САИ заносится информация об области их применения. После получения ТМИ с испытательных стендов, запускается ОТМИ. Результатом работы ОТМИ является заключение о нормативности работы данного режима или же выявленная в ходе автоматического анализа ошибка. В случае выявления ошибки, данные о ней передаются в Redmine, а ответственному специалисту высылается письмо по электронной почте с описанием выявленной

ошибки и ссылкой на неё в Redmine.

1.7. взаимодействует с системой САПР ФПО в части получения номеров версий САПР ФПО. Взаимодействие осуществляется через модуль САИ осуществляющий выбор версий из схемы данных САПР ФПО с использованием шаблонов согласованных при создании протокола взаимодействия систем. Выбор данных осуществляется по средствам SQL-запросов.

1.8. взаимодействует с разработчиками БПО, в части проведения анализа прошедших на стендах испытаний. Разработчикам передаётся информация о прошедших испытаниях БПО через интерфейс (Рис. 6) и внутреннюю электронную почту. От разработчиков получается результат анализа прошедшего испытания через интерфейс САИ.

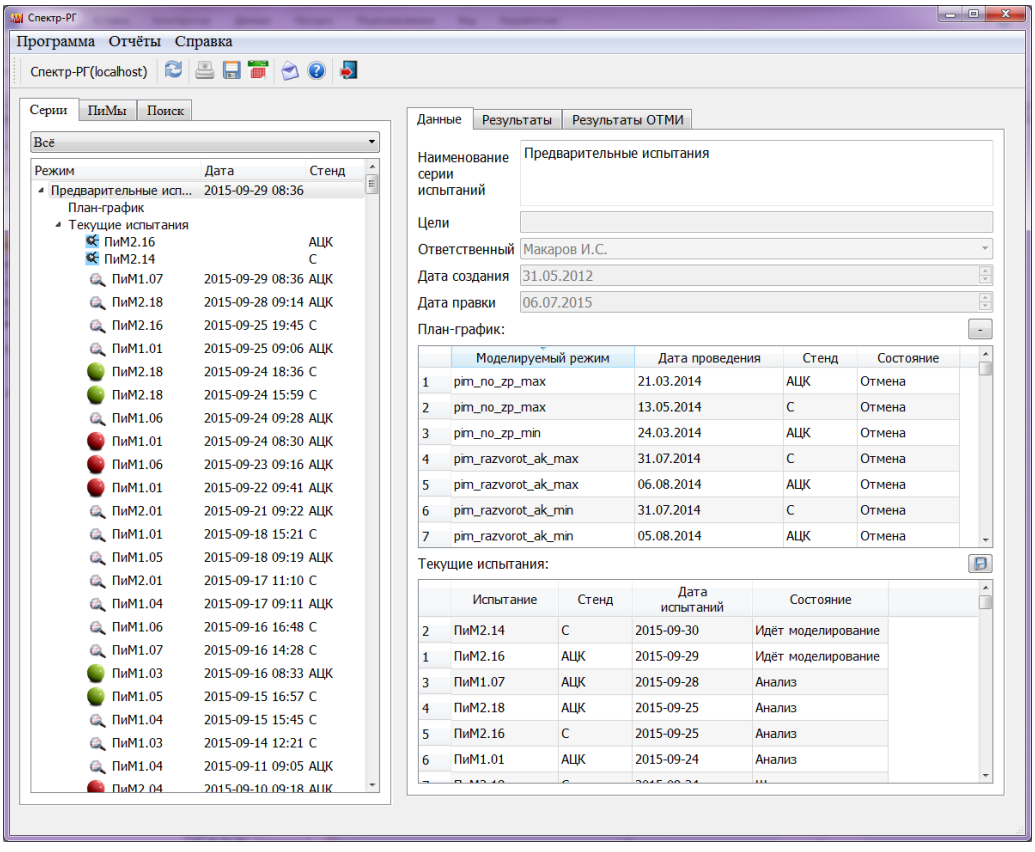

Рис. 6. Интерфейс САИ с информацией о прошедших испытаниях БПО

8. Заключение

Проделанная работа подтвердила возможность использования различных разнородных компонентов сторонних разработчиков, распространяемых с открытым исходным кодом, в системах посвящённых поддержке процесса разработки БПО.

В ходе данной работы была создана система автоматизации испытаний БПО, успешно функционирующая в рамках МОКБ «Марс», проведена интегрирования системы управления версиями Subversion и системы управления ошибками Redmine в рабочий процесс МОКБ. Установлено взаимодействие указанных компонентов с другими системами, разработанными на МОКБ «МАРС» (Рис. 2). Показано сокращение времени на отладку БПО за счёт внедрения указанных технологий.

Список источников

1. Бортовые системы управления космическими аппаратами / Под редакцией А.С. Сырова – М.: Изд-во МАИ-ПРИНТ, 2010. – 304 с.

2. Проектирование и испытание бортовых систем управления / Под редакцией А.С. Сырова. – М.: Изд-во МАИ-ПРИНТ, 2011. – 344 с.

3. Качалин А.М., Задорожная О.Н. Система спутникового контроля авиационных систем (Единая Информационная Система Взаимодействия (SWIM.ru) // Труды МАИ, 2016, №85: http://www.mai.ru/science/trudy/published.php?ID=66220

4. Ачилов Р. SVN с самого начала // Системный администратор. 2015. № 9. С. 67-71

5. Subversion: https://subversion.apache.org/ 2015-09-26

6. Stefan Küng, Lübbe Onken, Simon Large; «TortoiseSVN Клиент Subversion для

Windows: http://tortoisesvn.net/docs/nightly/TortoiseSVN_ru/tsvn-preface.html

7. Scott Chacon, Ben Straub, «Pro Git»: https://git-scm.com/book/ru/v1; 2015-09-26

8. Anton Kolin, «Обзор систем отслеживания ошибок» http://www.teamlead.ru/pages/viewpage.action?pageId=15794279

9. Redmine: http://www.redmine.org/projects/redmine/wiki

10. Andriy Lesyuk. Mastering Redmine. Copyright © 2013 Packt Publishing First published: January 2013, 343 p.

11. Ачилов Р. Установка Redmine и интеграция его с SVN // Системный администратор. 2014. № 4. С. 10-14

12. Bugzilla: http: //mozilla-russia.org/products/bugzilla

13. Trac: http: //trac.edgewall.org/wiki/TranslationRu/TracGuide