Method and tools for verification of cross-platform onboard software

I.V. Kovalev^{1, 2, 3, 4, *}, M.V. Saramud^{1, 3}, V.V. Losev³, A.A. Koltashev⁵

¹ Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russia

² Krasnoyarsk State Agrarian University, 90, Mira pr., Krasnoyarsk, 660049, Russia

³ Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russia

⁴ Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations, 61, Uritskogo street, Krasnoyarsk, 660049, Russia

⁵ JSC “Academician M F Reshetnev Information satellite systems”, 52 Lenin street, Zheleznogorsk, Krasnoyarsk region, 662972, Russia

*E-mail: kovalev.fsu@mail.ru

Abstract. The developed method and tools for verification and confirmation of onboard software are presented, which guarantee its compliance with all established functional and non-functional requirements throughout the entire life cycle of cross-platform onboard software. This approach allows not only to increase the fault tolerance of the control system software during operation, but also allows collecting statistics on the operation of software components in the process of real functioning of all subsystems. This information allows you to identify possible situations in which software failures appear, which allows you to develop more reliable software components in the future. The results of the operation of the version control function of the onboard software in the simulation environment are presented. The process of collecting statistics for identifying faulty versions is described.

Keywords: verification, onboard software, cross-platform, simulation environment

Метод и инструментарий верификации кроссплатформенного бортового программного обеспечения

И.В. Ковалев^{1, 2, 3, 4, *}, М.В. Сарамуд^{1, 3}, В.В. Лосев³, А.А. Колташев⁵

¹ Сибирский федеральный университет, пр. Свободный, 79, Красноярск, 660041, Россия

² Красноярский государственный аграрный университет, Красноярск, Россия, пр. Мира, 90, Красноярск, 660049, Россия

³ Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева, пр. Красноярский рабочий, 31, Красноярск, 660037, Россия

⁴ Красноярский краевой Дом науки и техники Российского Союза научных и инженерных общественных объединений, ул. Урицкого, 61, Красноярск, 660049, Россия

⁵ ОАО «Информационные спутниковые системы имени академика М.Ф. Решетнева», ул. Ленина, 52, Железногорск, Красноярский край, 662972, Россия

*E-mail: kovalev.fsu@mail.ru

Abstract. Представлен разработанный метод и инструментарий верификации и подтверждения бортового программного обеспечения, гарантирующие его соответствие всем установленным функциональным и нефункциональным требованиям в течение всего цикла жизни кроссплатформенного бортового программного обеспечения. Данный подход позволяет не только повысить отказоустойчивость программного обеспечения системы управления в процессе эксплуатации, но также позволяет собрать статистику работы программных компонент в процессе реального функционирования всех подсистем. Эта информация позволяет выявить возможные ситуации, при которых проявляются программные сбои, что позволяет разрабатывать более надежные программные компоненты в дальнейшем. Представлены результаты работы функции контроля версий бортового программного обеспечения в имитационной среде моделирования, описан процесс сбора статистики, обеспечивающий выявление сбойных версий.

Ключевые слова: верификация, бортовое программное обеспечение, кроссплатформенность, имитационная среда моделирования.

1. Введение

При реализации бортового программного обеспечения (БПО) критичных по надежности систем управления, работающих в режиме реального времени, в сборку операционной системы (ОС) FreeRTOS, авторами включены отдельные инструменты, ответственные за верификацию программных версий в процессе эксплуатации системы. В данной операционной системе, которая относится к классу ОС реального времени (OSPB), это необходимо для своевременного выявления «сбойных» программных компонентов, которые

могут негативно повлиять на работу программного комплекса или привести к сбоям и отказам различных подсистем и системы в целом. Полностью исключить вероятность проявления программных ошибок невозможно, однако инструменты, выявляющие и изолирующие подобные модули позволяют существенно повысить отказоустойчивость системы, даже если она содержит несовершенные программные компоненты.

В случае исключения программных компонентов, которые перестали выполнять свою функцию или удовлетворять нефункциональным требованиям (объем потребляемых ресурсов, оценка надежности и другие показатели, не относящиеся непосредственно к исполняемой функции), система продолжает корректно функционировать.

Случаи же отказов системы по причине того, что сбойный программный компонент некорректно обращается с памятью, перезаписывая данные, исключены архитектурой ОС FreeRTOS [1-7], поскольку программные компоненты, исполняемые в ней, не работают с памятью напрямую, а взаимодействуют через механизм очередей, в котором содержатся решения для обеспечения отказоустойчивой работы.

В сборку также включена функция журналирования (записи статистики), которая выполняется с меньшим приоритетом, чем основные задачи. Соответственно, ей предоставляются ресурсы только при их наличии, когда все основные задачи выполнены. Также имеется возможность передачи управления функции вручную, при необходимости. Таким образом, функция сбора статистики не может помешать исполнению основного функционала системы.

2. Метод верификации

Для реализации метода верификации в функцию статистики включается дополнительная проверка – не достиг ли вес любой версии из пула взвешенного голосования определенного порогового значения. Допустим – ниже 0,5, что означает, что версия за последние M (глубина стека весов версий) голосований в более чем половине случаев дала ответ, отличающийся от верного. В случае обнаружения подобной ситуации, вызывается сервисная функция *checkversion()*, задача которой – «разобраться» с версией, качество работы которой опустилось ниже необходимого уровня.

В случае, если для версии существует приемочный тест, он проводится, результат его исполнения записывается в журнал, после чего поток версии принудительно останавливается, очищается используемая им память. Далее поток версии запускается заново, происходит выделение памяти. Версия проходит приемочное тестирование повторно.

В случае, если версия не проходит приемочное тестирование успешно, она исключается из пула и далее не участвует в голосовании. Если версия успешно проходит приемочное

тестирование, она возвращается в работу, однако в журнал записывается факт сбоя и соответствующие параметры.

В случае повторного падения веса одной и той-же версии она исключается из пула, даже если успешно проходит приемочное тестирование. Информация об этом факте и параметры системы записываются в журнал и, при возможности, передаются администратору, для подробного изучения причины некорректной работы версии. В дальнейшем, после проведения необходимых манипуляций (корректировка самой версии, набора данных или иных параметров системы), имеется возможность вернуть версию в работу принудительно, а также обнулить счетчик ее сбоев.

3. Инструментарий верификации и подтверждения БПО

Журнал, в который записывается вся статистика работы мультиверсионных модулей представляет собой текстовый файл, с заранее определенной структурой. В случае стандартной записи статистики в файл записывается: символ новой строки, ключ, соответствующий стандартной записи, системное время, имя мультиверсионного модуля, количество версий в текущем пуле, значения весов всех версий, служебные параметры.

В случае обнаружения «сбойной» версии [8] происходит запись символа новой строки, соответствующего данному событию ключа, системное время, имя мультиверсионного модуля и имя конкретной версии, в которой произошел сбой. Также записывается результат прохождения приемочного теста до и после перезапуска версии, а также принятое решение – осталась версия в мультиверсионном пуле [9], или исключена.

Рисунок 1. Журнал событий.

На рисунке 1 представлен пример журнала событий, записанного функцией статистики. В журнале отображена статистика по 3 голосованиям модуля «cr_create» в штатном режиме, далее выявлено падение веса версии «version1» ниже установленного порога в 0,95, однако она проходит приемочное тестирование и остается в пуле. Далее вес той-же версии снова

падает ниже установленного порога, однако в этот раз она не проходит оба приемочных теста и исключается из пула. Это подтверждается последней записью статистики, в которой видно, что теперь в данном модуле в голосовании участвуют 4 версии, а не 5 как было ранее.

Рисунок 2. Модуль генерации функции статистики.

На рисунке 2 представлен интерфейс модуля интегрированной среды разработки – средство генерации функции статистики. На форме задается период, на который функция переходит в режим ожидания, даже при наличии свободных ресурсов, это необходимо для того, чтобы функция статистики, при наличии большого количества свободных ресурсов системы, не записывала данные слишком часто. Задается приоритет, с которым будет запущен поток функции статистики. Формируется список мультиверсионных модулей, для которых будет собираться статистика и задается предельный вес версий для каждого. В случае, если вес любой из версий опуститься ниже заданного значения, будет вызвана дополнительная функция контроля версий. Если необходим только сбор статистики, без дополнительных действий над версиями, можно задать в качестве предельного веса «0».

Также задается количество итераций - максимальное количество проверок одной и той-же версии, после которой она будет исключена из пула, не смотря на положительный результат приемочного тестирования. Если вес определенной версии падает ниже заданного значения, она перезапускается, проходит приемочное тестирование и, в случае успеха, возвращается в работу. Однако, вес данной версии может снова продолжить падать, и версия будет проверена повторно. Нет смысла выполнять данную процедуру бесконечно, если версия проходит приемочное тестирование, однако в реальной работе продолжает допускать сбои даже после перезапуска, очевидно, что имеет место быть алгоритмическая проблема на текущем потоке входных данных и версию необходимо исключить, временно или окончательно.

На рисунке 3 представлена блок-схема процесса сбора статистики, которая также включает в себя и выявление сбойных версий.

Рисунок 3. Блок-схема функции контроля версий.

Рассмотрим результаты работы функции контроля версий в имитационной среде моделирования, реализованной в рамках данного подхода.

Рисунок 4. Функционирование в нормальных условиях.

На рисунке 4 представлен график зависимости весов версий от номера итерации. Приведен пример 300 итераций для мультиверсионного модуля [10] с 5 версиями на начало симуляции. На графике представлена «нормальная» работа системы, когда в работе всех версий нет нарушений.

Функция контроля версий периодически запускается, проверяет данный факт, пишет значения в журнал, и, поскольку значения весов версий не достигли установленного ограничения (в данной симуляции установлено значение 0,85), функция более не производит никаких активных действий, снова переходит в режим ожидания.

Журнал событий, записанный функцией контроля версий в описанной выше ситуации представлен на рисунке 5.

На рисунке 6 представлен подобный рассмотренному ранее график, но для случая падения надежности одной из версий. На графике видно, что на 140-й итерации вес первой версии достиг установленного порога. В данном случае представлена ситуация, в которой версия не прошла приемочное тестирование и была исключена из пула. В дальнейшем она уже не принимала участие в мультиверсионном голосовании. Остальные 4 версии продолжили работать в штатном режиме, что является достаточным для осуществления мультиверсионного голосования, следовательно, вся система продолжила корректное функционирование.

На рисунке 7 представлен график весов версий для ситуации, при которой вес первой версии на 170 итерации достигает заданного порога, однако, после перезапуска она проходит приемочное тестирование и возвращается в мультиверсионный пул [11]. Как показывают данные дальнейших итераций – версия после перезапуска восстановила корректную работу, на протяжении всей симуляции ее вес находился в допустимых пределах.

На рисунке 8 представлен график весов версий для ситуации, при которой вес первой версии на 100 итерации достигает заданного порога, однако, после перезапуска она проходит приемочное тестирование и возвращается в мультиверсионный пул. Далее ее вес снова достигает заданного порога на 190 итерации, она снова проходит приемочное тестирование, что записывается в журнал событий, однако все равно исключается из мультиверсионного пула, поскольку это заданное условие работы функции контроля версий (максимальное количество возвратов в пул = 1).

Рисунок 5. Журнал событий при функционировании в нормальных условиях.

Рисунок 6. Сбой одной версии, версия не прошла приемочное тестирование.

Рисунок 7. Сбой одной версии, версия прошла приемочное тестирование.

Рисунок 8. Сбой одной версии, версия прошла приемочное тестирование, далее снова вес достиг установленного значения.

Рисунок 9. Сбой двух версий, первая прошла приемочное тестирование, третья – нет.

На рисунке 9 представлен график весов версий для ситуации, при которой вес третьей версии на 130 итерации достигает заданного порога, она не проходит приемочное тестирование и исключается из мультиверсионного пула. Далее система продолжает работать с 4 версиями. На 150 итерации вес первой версии на достигает заданного порога, однако, после перезапуска она проходит приемочное тестирование и возвращается в мультиверсионный пул. До конца симуляции система продолжает корректно функционировать с 4 версиями.

На рисунке 10 представлено содержание журнала событий для ситуации, описанной выше.

Рисунок 10. Журнал событий при проявлении сбоя двумя версиями.

Предложенный метод верификации и подтверждения БПО позволяет не только повысить отказоустойчивость программного комплекса системы управления в процессе эксплуатации, но также позволяет собрать статистику работы программных компонент в процессе реального функционирования всех подсистем. Эта информация позволяет выявить возможные ситуации, при которых проявляются программные сбои, что позволяет разрабатывать более надежные программные компоненты в дальнейшем.

4. Заключение

Предложенный метод верификации и подтверждения БПО, гарантирующий его соответствие всем установленным функциональным и нефункциональным требованиям в течение всего цикла жизни кроссплатформенного бортового программного обеспечения, реализован в качестве программного средства – функции контроля версий, исполняемой в рамках ОСРВ.

Реализованное средство позволяет получить реальные характеристики программных реализаций модулей в среде исполнения. Важно то, что программное обеспечение является исполняемым в рамках реальной физической реализации системы. Данная информация позволит сделать обоснованные выводы о качестве работы программных компонент в реальных условиях, а также принять решение о возможности дальнейшего их использования в критичных по надёжности системах управления, либо о необходимых модификациях и доработках для достижения требуемых характеристик.

Список литературы

- [1] Курниц, А. FreeRTOS. Взгляд изнутри. Алгоритм работы планировщика. Часть 1 / А. Курниц // Компоненты и технологии. – 2013. – № 5(142). – С. 114-122.
- [2] Курниц, А. FreeRTOS. Взгляд изнутри. Алгоритм работы планировщика. Часть 2 / А. Курниц // Компоненты и технологии. – 2013. – № 6 (143). – С. 89-94.
- [3] Курниц, А. FreeRTOS — операционная система для микроконтроллеров / А. Курниц // Компоненты и технологии. – 2011. – № 2(115). – С. 96-100.
- [4] Курниц, А. FreeRTOS — операционная система для микроконтроллеров / А. Курниц // Компоненты и технологии. – 2011. – № 3(116). – С. 109-114.
- [5] Курниц, А. FreeRTOS — операционная система для микроконтроллеров / А. Курниц // Компоненты и технологии. – 2011. – № 7(120). – С. 23-32.
- [6] Курниц, А. FreeRTOS — операционная система для микроконтроллеров / А. Курниц // Компоненты и технологии. – 2011. – № 10(123). – С. 93-100.
- [7] Курниц, А. FreeRTOS — операционная система для микроконтроллеров / А. Курниц // Компоненты и технологии. – 2011. – № 11(124). – С. 99-108.
- [8] Avizienis, A. The N-Version Approach to Fault - Tolerant Software / A. Avizienis // IEEE Transactions on Software Engineering. – December 1985. – № SE-11. – № 12. – P. 1491-1501.
- [9] Gersting, J.A. Comparison of Voting Algorithms for N-Version Programming / J.A. Gersting // Comparison of Voting Algorithms for N-Version Programming, Proceedings of the 24th Annual Hawaii International Conference on System Sciences. – 1991. – № II. – P. 253– 262.

Reprinted in Fault-Tolerant Software Systems: Techniques and Applications, Hoang Pham (ed.), IEEE Computer Society Press. – 1992. – P. 62-71.

- [10] Нечаева, К.О. Реализация разнообразия при разработке мультиверсионного программного обеспечения / К.О. Нечаева // В сборнике: Информационно-телекоммуникационные системы и технологии. Материалы Всероссийской научно-практической конференции. – 2017. – С. 321-323.
- [11] Михалев, А.С. Современные технологии реализации мультиверсионного программного обеспечения / А.С. Михалев, А.Н. Исаев, К.А. Носарев // Новая наука: от идеи к результату. – 2016. – № 12-3. – С. 129-133.