Современные инновации, системы и технологии // Modern Innovations, Systems and Technologies

2022; 2(3)
eISSN: 2782-2818
https://www.oajmist.com

УДК: 004.582
DOI: https://doi.org/10.47813/2782-2818-2022-2-3-0127-0138

EDN: ITIQTB

Метод блоков восстановления для повышения надежности программного обеспечения: сравнение с мультиверсионным программированием

Д. В. Грузенкин, Д. О. Шаварин

Сибирский Федеральный Университет, г. Красноярск, Россия

Аннотация. На сегодняшний день вычислительные машины применяются в каждой сфере деятельности человека (от научно-исследовательской деятельности и до сферы обслуживания). В данной статье раскрыта тема возрастания актуальности надежности программного обеспечения в связи важностью сохранения надежной и отказоустойчивой работы программного обеспечения в критически важных для человека отраслях науки и техники. В работе описаны такие способы повышения надёжности программного обеспечения и его защиты от влияния ошибок, как метод блоков восстановления и методы, основанные на избыточности, в частности, мультиверсионное программирование. Описан принцип работы, и приведена схема метода блоков восстановления. Проведено сравнение метода мультиверсионного программирования и метода блоков восстановления с последующим теоретическим анализом достоинств и недостатков метода блоков восстановления. Представлены результаты проведения эксперимента по сравнению этих двух подходов.

Ключевые слова: надежность программного обеспечения, метод блоков восстановления, N-версионное программирование, мультиверсионное программирование.

Для цитирования: Грузенкин, Д., & Шаварин, Д. (2022). Применение метода блоков восстановления для повышения надежности программного обеспечения. Сравнение с мультиверсионным программированием. Современные инновации, системы и технологии - Modern Innovations, Systems and Technologies, 2(3), 0127–0138. https://doi.org/10.47813/2782-2818-2022-2-3-0127-0138

Recovery blocks method to improve software reliability: comparison with N-version programming

D. V. Gruzenkin, D. O. Shavarin

Siberian Federal University, Krasnoyarsk, Russian Federation

Abstract. Today, computers are used in every kind of human activity (from scientific research to the service sector). This article covers the topic of software reliability increasing in connection with the importance of maintaining reliable and fault-tolerant software operations in critically important fields of science and technology for humans. The paper describes such methods of software reliability improving and protecting it from the errors influence, such as the method of recovery blocks and N-version programming approach. The constructive principle is described, and a diagram of the method of recovery blocks is given. A comparison of the N-version programming method and the recovery blocks method was carried out, followed by a theoretical analysis of the advantages and disadvantages of the method of recovery blocks. The results of the experiment comparing these two approaches are presented.

Keywords: software reliability, recovery blocks method, N-version programming, multiversion programming.

For citation: Gruzenkin, D., & Shavarin, D. (2022). Application of recovery blocks method to improve software reliability. Comparison with N-version programming. Modern Innovations, Systems and Technologies, 2(3), 0127–0138. https://doi.org/10.47813/2782-2818-2022-2-3-0127-0138

ВВЕДЕНИЕ

На сегодняшний день вычислительные машины являются распространенной частью нашего быта. Во многих сферах деятельности человека требуется высококачественное программное обеспечение (ПО) систем управления. Такое ПО может использоваться, например, в транспортной сфере, медицине, энергетической отрасли, финансовой и банковской сферах, космической и научно-исследовательской отраслях [1,2,3], причём список сфер, где применяются программные системы управления не ограничен лишь приведёнными выше пунктами. Однако существуют такие сферы человеческой жизнедеятельно, как, например, атомная энергетика или космические полёты, где надёжность ПО важна критически, поскольку полезный эффект от его работы за всё время использования может быть несоизмеримо мал по сравнению с негативными последствиями, ставшими результатом ошибки в этом программном обеспечении. Именно поэтому на сегодняшний день существует множество подходов и методов для повышения надёжности программного обеспечения, а само повышение надежности является актуальной задачей.

При безотказной работе аппаратной части причиной нарушения работы различного рода информационных систем могут служить конфликты между исходными данными и их обработчиком, то есть обработчик может некорректно обработать поступившие ему данные, особенно, если сами данные были некорректны [4]. Поэтому повышение надежности зачастую происходит путем усовершенствования программной части, хотя и валидности данных также уделяется большое внимание.

Любое программное решение должно обладать тремя ключевыми особенностями, для того чтобы считаться допустимым:

- Уметь диагностировать ошибку до того, как будет нанесен непоправимый уровень повреждений.
- Уметь отбросить ложную информацию, возникшую впоследствии ошибки, и вернуть систему в функционирующий режим работы.
- Уметь продолжить работу, ожидая, что последующая работа будет выполнена [5].

Существует множество способов повышения надёжности программного обеспечения и его защиты от влияния ошибок, главным образом все эти механизмы используют тот или иной вид избыточности (временная, информационная или программная) [6]. Судя по количеству публикаций и активности авторов [7,8], на данный момент одним из популярных методов повышения надёжности ПО являются мультиверсионное программирование, основанное на программной избыточности [9].

Однако методу блоков восстановления уделяется, по мнению авторов, незаслуженно меньше внимания в научной литературе, хотя и его эффективность также остаётся достаточно высокой. Поэтому в данной работе рассматривается именно метод блоков восстановления для повышения надёжности ПО, а также сравниваются результаты его работы с классическим для мультиверсионного ПО алгоритмом голосования абсолютным большинством [10, 11].

ОПИСАНИЕ МЕТОДА БЛОКОВ ВОССТАНОВЛЕНИЯ

Метод блоков восстановления был предложен в 1974 году Джеймсом Хорнингом для возможности исправления ошибок и автоматизированного восстановления функциональности программы [12].

Суть метода блоков восстановления заключается в том, что для каждого программного компонента задаётся тест, проверяющий корректность его работы после каждого запуска. В случае возникновения ошибки или отказа одного из компонентов ПО, поочерёдно запускаются другие компоненты, функционально эквивалентные данному, то есть, каждый модуль содержит несколько версий, блоков, выполняющих одну функцию, но реализованных по-разному, каждый следующий блок запускается, если предыдущий не справился. Таким образом, отказ ПО с реализованным методом блоков

восстановления возможен в случаях, если тест какого-либо блока даст сбой и пропустит ошибку или если закончатся блоки с алгоритмами.

Более наглядно упрощённая схема работы блоков восстановления представлена на рисунке 1.

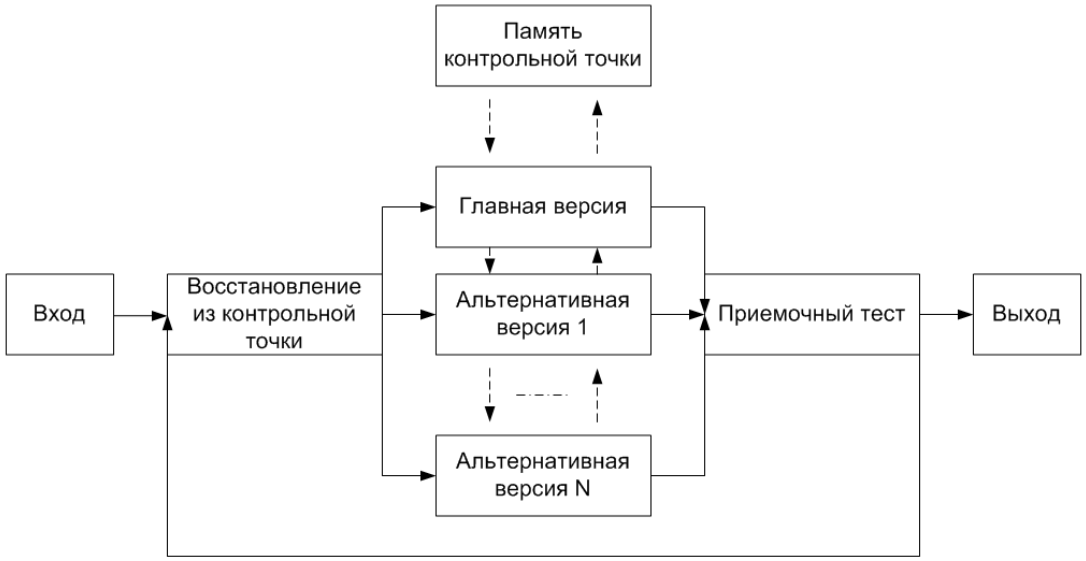

Рисунок 1. Модель блоков восстановления.

Figure 1. Recovery block model.

Как видно на схеме, данные подаются на вход, производится их запоминание на контрольной точке и последующая обработка на главной версии алгоритма, первым способом. Над итогом вычислений проводится тест и, в случае успешного прохождения теста передается на выход, но в случае, если тест не пройден, данные восстанавливаются на контрольной точке и проходят обработку на альтернативной версии алгоритма. Снова проводится тест и, по его результатам, данные либо подаются на выход, либо снова идут на контрольную точку.

МАТЕРИАЛЫ И МЕТОДЫ

Для проведения эксперимента, который смог бы показать эффективность подхода блоков восстановления, была создана база данных, состоящая из набора таблиц, код создания которых приведён в листинге 1.

Листинг 1. Структура базы данных для эксперимента.

```
CREATE TABLE module (
    "id" integer primary key autoincrement not null,
    "name" varchar(255) not null,
```

```
    "round_to" integer not null,
    "min_out_val" real not null,
    "max_out_val" real null);
CREATE TABLE version (
    "id" integer primary key autoincrement not null,
    "name" varchar(255) not null,
    "reliability" real not null,
    "module" integer null,
    foreign key ("module") references module(id))
CREATE TABLE experiment_data (
    "id" integer primary key autoincrement not null,
    version_id integer not null,
    version_name varchar(255),
    version_reliability real not null,
    version_answer real not null,
    correct_answer real not null,
    module_id integer not null,
    module_name varchar(255),
    module_iteration_num int not null,
    experiment_name varchar(31) not null,
    foreign key ("version_id") references version(id),
    foreign key ("module_id") references module(id));
```

Для проведения эксперимента были сгенерированы тестовые данные, которые имитировали работу двух мультиверсионных модулей, состоящих из 3 и 5 мультиверсий. Соотнесение версий с их априорными надёжностями (вероятностями безотказной работы) приведено ниже:

1. V1 – 0.99;
2. V2 – 0.98;
3. V3 – 0.87;
4. V4 – 0.999;
5. V5 – 0.8.

В модуль, состоящий из 5 версий, входят все перечисленные выше версии, а в модуль, состоящий из 3 версий – только первые 3.

Поскольку тематика данной статьи не затрагивает вопрос получения корректных данных от мультиверсий, то и запрос, позволивший сгенерировать тестовые данные для этого эксперимента в статье, не приводится.

Для проверки гипотезы о не худшей эффективности блоков восстановления по сравнению с мультиверсионным подходом было проведено 2 эксперимента, в ходе которых были сгенерированы данные, выдаваемые описанными выше модулями. Эти данные имитируют реальную работу модулей, определяя, какой ответ будет выдан каждой мультиверсией, при этом определяется и верный ответ для каждой итерации. Каждый из модулей был запущен 1000 раз. В итоге для каждой итерации определялся верный ответ, который будет принят за ответ всего модуля, с помощью двух указанных выше подходов.

SQL-код, реализующий алгоритм голосования абсолютным большинством в контексте мультиверсионной парадигмы, и код, реализующий работу блоков восстановления, приведены в листингах 2 и 3, соответственно.

Листинг 2. Код алгоритма голосования абсолютным большинством.

```
-- Голосование абсолютным большинством
select *, count(module_id) over() whole_answers_count, sum(is_correct_module_answer) over() whole_correct_answers, count(module_answers) over (PARTITION BY experiment_name) experiment_answers_count, sum(is_correct_module_answer) over (PARTITION BY experiment_name) experiment_correct_answers
from (select distinct module_id, module_iteration_num, experiment_name, version_answer, correct_answer, version_answer_num, module_answers , case when CAST(max_version_answer_num as REAL) > CAST(module_answers as real) / cast(2 as real) then 1 else 0 end is_correct_module_answer
    from (select *, max(version_answer_num) over (partition by experiment_name, module_iteration_num) max_version_answer_num
        from (select id, version_id, version_reliability, version_answer, correct_answer, module_id, module_iteration_num, experiment_name, row_number() over (PARTITION BY experiment_name, module_iteration_num, version_answer) version_answer_num, count (version_answer) over (partition by experiment_name, module_iteration_num) module_answers
            from experiment_data
```

```
                where experiment_name in ('M3_I1000', 'M5_I1000')
            ) t
        ) tt
        where max_version_answer_num = version_answer_num
) ttt;
```

Листинг 2. Код алгоритма реализации подхода блоков восстановления.

```
-- Блоки восстановления
select *, count(module_id) over() whole_answers_count, sum(is_correct_module_answer)
        over() whole_correct_answers
        , count(version_answer) over (PARTITION BY experiment_name)
        experiment_answers_count, sum(is_correct_module_answer) over (PARTITION BY
        experiment_name) experiment_correct_answers
from (select * , min(tt.version_id) over (PARTITION BY experiment_name,
        module_iteration_num) min_correct_version_id, abs(version_answer - correct_answer)
        < 0.00000001 is_correct_module_answer
        from (select *,
            t.version_reliability > t.test_pass_probability test_passed
            from (select id, version_name, version_id, round(cast(version_reliability * 1000
                as INTEGER)) version_reliability, version_answer, correct_answer,
                module_id, module_iteration_num, experiment_name,
                round(cast(abs(random() % 1000) as INTEGER)) test_pass_probability
                    from experiment_data
                where experiment_name in ('M3_I1000', 'M5_I1000')
            ) t
        ) tt
        where tt.test_passed = 1
) ttt
where ttt.version_id = min_correct_version_id;
```

Стоит отметить, что в коде, реализующем работу блоков восстановления, для имитации прохождения версией проверки после её исполнения используется априорное значение надёжности версии – генерируется случайное число, значение которого сравнивается со значением надёжности версии, если оно не превышает значения

надёжности, то тест считается пройденным, иначе, выход мультиверсии помечается как ошибочный вне зависимости от того, выдала версия правильный ответ или нет.

В качестве СУБД для проведения эксперимента была выбрана SQLite по причине своей легковесности, возможности быстрого развёртывания и переносимости.

РЕЗУЛЬТАТЫ

В результате проведённого выше эксперимента были получены данные, которые отражают количество успешно определённых верных ответов из общего количества обработанных групп ответов (итераций запуска мультиверсий).

Алгоритм голосования абсолютным большинством в эксперименте из 1000 итераций модуля, состоящего из 3 мультиверсий, определил корректно верное решение в 999 случаях, а для модуля, состоящего из 5 версий на выборке той же размерности – в 997 случаях. Итого, из 2000 обработанных итераций алгоритм голосования абсолютным большинством верно определил правильный ответ в 1996 случаях.

Подход повышения надёжности, использующий блоки восстановления показал себя следующим образом: в 996 случаях из 1000 при обработки данных модуля, состоящего из 3 версий, были выданы корректные результаты, а при обработке выборки аналогичного объема, представляющей собой результаты работы модуля, состоящего из 5 мультиверсий, корректно правильный ответ был определён в 997 случаях. Общим итогом работы данного алгоритма можно считать верное определение правильного ответа в 1993 случаях из 2000.

ОБСУЖДЕНИЕ

Из проведённого эксперимента видно, что результат работы алгоритма блоков восстановления сопоставим с результатом алгоритма голосования абсолютным большинством. При этом результат прохождения версией теста определялся случайным образом, а не реальным тестом, что также вероятно снизило показатель количества верных выходов модуля при использовании блоков восстановления.

Также стоит отметить, что при использовании мультиверсионного подхода версии зачастую выполняются параллельно, что может служить причиной взаимного влияния их друг на друга и, соответственно, возникновения взаимозависимых сбоев [13]. Поэтому монопольное исполнение одной версии в один момент времени, как при

использовании блоков восстановления, в данном контексте является наиболее предпочтительным.

ЗАКЛЮЧЕНИЕ

Метод блоков восстановления позволяет значительно сократить количество используемых ресурсов при исполнении программы по сравнению с мультиверсионным подходом, поскольку если все тесты были пройдены успешно у первой версии, использование альтернативных алгоритмов не потребуется. Однако в случае нахождения ошибки временные издержки могут превышать таковые при использовании, например, метода мультиверсионного программирования.

Также, как уже было отмечено выше, последовательное исполнение блоков позволяет сократить количество используемых вычислительных мощностей, потребляемых в единицу времени, а также снизить вероятность возникновения взаимозависимых ошибок в исполняемых версиях за счёт отсутствия разделения ими общих ресурсов.

Другой сильной стороной метода блоков восстановления, в отличие от того же метода мультиверсионного программирования, является тот факт, что блокам восстановления не требуется программная разработка по единой спецификации, поскольку тесты для каждого блока пишутся индивидуально, что обеспечивает большую экономическую эффективность данного подхода. Кроме того, зачастую к алгоритмам голосования мультиверсионной среды исполнения предъявляются более жесткие требования, чем к приемочному тесту для блока восстановления [10].

Теоретический анализ метода блоков восстановления, а также проведённый эксперимент, показали, что данный подход обладает значительным количеством преимуществ по сравнению с другими подходами повышения надёжности ПО. Несмотря на то, что публикаций по данной тематике на сегодняшний день выходит относительно мало, по мнению авторов, данный метод имеет хорошие перспективы для дальнейшего развития и применения его для решения практических задач.

СПИСОК ЛИТЕРАТУРЫ

[1] Ковалев И. и др. К вопросу формирования блочно-модульной структуры системы управления беспилотных летательных объектов. Современные инновации, системы и технологии - Modern Innovations, Systems and Technologies. 2021; 1 (3): 48-64.

[2] Бурмистрова Н. И., Кошечкин К. А., Преферанская Н. Г. Оценка результатов внедрения информационной системы управления хроматографическим оборудованием в испытательном центре экспертного учреждения в сфере обращения лекарственных средств. Фармацевтическое дело и технология лекарств. 2021; 1: 42-53.

[3] Durmus M. S. et al. Modular fault diagnosis in fixed-block railway signaling systems. IFAC-PapersOnLine. 2016; 49 (3): 459-464.

[4] Липаев В. В. Надежность программного обеспечения (обзор концепций). Автоматика и телемеханика. 1986; 10: 5-31.

[5] Anderson T., Kerr R. Recovery blocks in action: A system supporting high reliability. Reliable Computer Systems. Springer, Berlin, Heidelberg; 1985. 80-101.

[6] Чигринев М. И., Огородников А. А. Модели и методы повышения надежности программного обеспечения. Проблемы, перспективы и направления инновационного развития науки: Сборник статей по итогам Международной научно-практической конференции. Омск, 24 ноября. 2017; 196.

[7] Ковалев Д. И., Мансурова Т. П., Туев Е. В. Многоатрибутивный анализ отказоустойчивой программной архитектуры систем мониторинга траектории полета воздушных судов. «Модернизация, инновации, прогресс: передовые технологии в материаловедении, машиностроении и автоматизации - MIP: ENGINEERING-IV-2022»: сборник научных статей по материалам IV Международной научной конференции (Красноярск, 28-30 апреля 2022 г.). Красноярск: Красноярский краевой Дом науки и техники. 2022; 57-63.

[8] Штарик Е., Штарик А., Царев Р. Мультиверсионное программное обеспечение. Алгоритмы голосования и оценка надёжности. Litres. 2022.

[9] Chen L., Avizienis A. N-version programming: A fault-tolerance approach to reliability of software operation. Proc. 8th IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-8). 1978; 1: 3-9.

[10] Грузенкин Д. В., Новиков О. С., Суханова А. В. Мультиверсонное ПО и блоки восстановления – два способа защиты от ошибок. Новая наука: от идеи к результату. 2016; 11-2: 72-75.

[11] Черниговский А. С. Сравнение мультиверсионного программирования и блоков восстановления. Системный анализ, управление и программная инженерия. 2016: 74.

[12] Horning J. J. et al. A program structure for error detection and recovery //Conference on Operating Systems. Springer, Berlin, Heidelberg, 1974. 171-187.

[13] Gruzenkin D. V., Chernigovskiy A. S., Tsarev R. Y. N-version software module requirements to grant the software execution fault-tolerance. Proceedings of the Computational Methods in Systems and Software. Springer, Cham, 2017. 293-303.

REFERENCES

[1] Kovalev I. i dr. K voprosu formirovaniya blochno-modul'noj struktury sistemy upravleniya bespilotnyh letatel'nyh ob"ektov. Sovremennye innovacii, sistemy i tekhnologii - Modern Innovations, Systems and Technologies. 2021; 1 (3): 48-64.

[2] Burmistrova N. I., Koshechkin K. A., Preferanskaja N. G. Ocenka rezul'tatov vnedrenija informacionnoj sistemy upravlenija hromatograficheskim oborudovaniem v ispytatel'nom centre jekspertnogo uchrezhdenija v sfere obrashhenija lekarstvennyh sredstv. Farmacevticheskoe delo i tehnologija lekarstv. 2021; 1: 42-53.

[3] Durmus M. S. et al. Modular fault diagnosis in fixed-block railway signaling systems. IFAC-PapersOnLine. 2016; 49 3: 459-464.

[4] Lipaev V. V. Nadezhnost' programmnogo obespechenija (obzor koncepcij). Avtomatika i telemehanika. 1986; 10: 5-31.

[5] Anderson T., Kerr R. Recovery blocks in action: A system supporting high reliability. Reliable Computer Systems. – Springer, Berlin, Heidelberg, 1985; 80-101.

[6] Chigrinev M. I., Ogorodnikov A. A. Modeli i metody povyshenija nadezhnosti programmnogo obespechenija. Problemy, perspektivy i napravlenija innovacionnogo razvitija nauki: Sbornik statej po itogam Mezhdunarodnoj nauchno-prakticheskoj konferencii (Omsk, 24 nojabrja), 2017; 196.

[7] Kovalev D. I., Mansurova T. P., Tuev E. V. Mnogoatributivnyj analiz otkazoustojchivoj programmnoj arhitektury sistem monitoringa traektorii poleta vozdushnyh sudov. «Modernizaciya, innovacii, progress: peredovye tekhnologii v materialovedenii, mashinostroenii i avtomatizacii- MIP: ENGINEERING-IV-2022»: sbornik nauchnyh statej po materialam IV Mezhdunarodnoj nauchnoj konferencii (Krasnoyarsk, 28-30 aprelya 2022 g.). – Krasnoyarsk: Krasnoyarskij kraevoj Dom nauki i tekhniki, 2022; 57-63.

[8] Shtarik E., Shtarik A., Carev R. Mul'tiversionnoe programmnoe obespechenie. Algoritmy golosovanija i ocenka nadjozhnosti. Litres, 2022.

[9] Chen L., Avizienis A. N-version programming: A fault-tolerance approach to reliability of software operation. Proc. 8th IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-8). 1978; 1: 3-9.

[10] Gruzenkin D. V., Novikov O. S., Suhanova A. V. Mul'tiversonnoe PO i bloki vosstanovlenija–dva sposoba zashhity ot oshibok. Novaja nauka: Ot idei k rezul'tatu. 2016; 11-2: 72-75.

[11] Chernigovskij A. S. Sravnenie mul"tiversionnogo programmirovanija i blokov vosstanovlenija. Sistemnyj analiz, upravlenie i programmnaja inzhenerija. 2016; 74.

[12] Horning J. J. et al. A program structure for error detection and recovery //Conference on Operating Systems. Springer, Berlin, Heidelberg, 1974; 171-187.

[13] Gruzenkin D. V., Chernigovskiy A. S., Tsarev R. Y. N-version software module requirements to grant the software execution fault-tolerance. Proceedings of the Computational Methods in Systems and Software. Springer, Cham, 2017; 293-303.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Денис Владимирович Грузенкин, старший преподаватель кафедры информатики института космических и информационных технологий Сибирского федерального университета
e-mail: gruzenkin.denis@good-look.su

Данил Олегович Шаварин, студент 2 курса института космических и информационных технологий Сибирского федерального университета

Denis V. Gruzenkin, senior teacher at Informatics department of School of space and information technologies of Siberian Federal University
e-mail: gruzenkin.denis@good-look.su

Danil O. Shavarin, 2-nd year student at School of space and information technologies of Siberian Federal University

Статья поступила в редакцию 18.06.2022; одобрена после рецензирования 23.08.2022; принята к публикации 23.08.2022.

The article was submitted 18.06.2022; approved after reviewing 23.08.2022; accepted for publication 23.08.2022.