Факторы надежности программного обеспечения микропроцессорных защит[1]

Dependability factors of microprocessor protection software

Успенский М.И.
Uspensky M.I.
Институт социально-экономических и энергетических проблем Севера, Сыктывкар, Коми НЦ УрО РАН
Institute for Socioeconomic and Energy-Related Problems of the North, Syktyvkar, Komi Science Centre, Urals Division, Russian Academy of Sciences
uspensky@energy.komisc.ru

Успенский М.И.

Резюме. Цель. *Анализ особенностей надежности функционирования программ, характерных для критически важных приложений, работающих в режиме реального времени, применительно к микропроцессорным защитам. К числу основных характеристик функционирования релейных защит относятся показатели надежности. С переходом исполнения таких защит на микропроцессорную основу кроме аппаратной надежности возникла необходимость характеризовать ее функционирование и надежностью программного обеспечения. Важность решаемых ею задач в процессе эксплуатации относит ее к программам, используемым в критически важных приложениях и работающим в реальном режиме. Это, в свою очередь, ужесточает требования к оценке их надежности.* **Методы.** *Сравнительный анализ позволил оценить общность рассмотренных факторов надежности функционирования программного обеспечения рассматриваемых приложений, и в то же время выделить некоторые особенности, характерные для исполнения релейной защиты на микропроцессорах.* **Результаты.** *Приведен пример подобной оценки, показывающий, что при всех сложностях полного тестирования программ микропроцессорной защиты вклад ошибок в программах пока составляет 2,5% от общего вклада.* **Заключение.** *Показано, что программы микропроцессорных защит относятся к программам критически важных приложений, работающих в режиме реального времени, что позволяет использовать опыт и характеристики такого программного обеспечения при решении задач релейной защиты. Тем не менее, приведены некоторые особенности, характерные именно для релейной защиты на микропроцессорной основе. Определены дальнейшие задачи.*

Abstract. Aim. *An analysis of the distinctive features of the functional dependability of software that are common to critical real-time applications in respect to microprocessor protections. Dependability indicators are among the primary operational characteristics of relay protection. As such protections migrate to microprocessor technology, it became necessary to characterise their operation and dependability with software dependability along the hardware dependability. The importance of the tasks it solves in the process of operation puts it into the category of programs used in critical real-time applications. That, in turn, toughens the requirements for their dependability evaluation.* **Methods.** *The comparative analysis allowed evaluating the unity of the examined operational dependability factors of the software as part the examined applications, while identifying certain distinctive features that are common to relay protections that use microprocessors.* **Results.** *The paper provides an example of such estimation that shows that, given all the complexity of testing microprocessor protection programs, the contribution of software errors still is 2,5% from the overall contribution.* **Conclusion.** *It is shown that microprocessor protection programs fall into the category of critical real-time application programs, which allows using the experience and characteristics of such software as part of relay protection. Nevertheless, the paper cites certain distinctive features that are common only to microprocessor-based relay protection. Further tasks are specified.*

Ключевые слова: *программы критически важных приложений, микропроцессорные релейные защиты, надежность программного обеспечения.*
Keywords: *critical application software, microprocessor relay protection, software dependability.*

Для цитирования: *Успенский М.И. Факторы надежности программного обеспечения микропроцессорных защит // Надежность. 2023. №3. С. 73-77. https://doi.org/10.21683/1729-2646-2023-23-3-73-77*

[1] M.I. Uspensky. "SOFTWARE CONTRIBUTION TO THE AVAILABILITY OF MICROPROCESSOR-BASED RELAY PROTECTION" Reliability: Theory & Applications, vol. 17, no. 3 (69), 2022, pp. 31-39. DOI: 10.24412/1932-2321-2022-369-31-39

For citation: *Uspensky M.I. Dependability factors of microprocessor protection software. Dependability 20023;3:73-77. https://doi.org/10.21683/1729-2646-2023-23-3-73-77*

Поступила: *20.02.2023* / **После доработки:** *28.06.2023* / **К печати:** *15.09.2023*
Received on: *20.02.2023* / **Revised on:** *28.06.2023* / **For printing:** *15.09.2023*

1. Введение

В последнее время программистами большое внимание уделяется оценке надежности функционирования программ, используемых в критически важных приложениях и работающих в режиме реального времени. Среди целого круга таких программ, выполняющих важные функции на атомных станциях, военном оборудовании и т.п., большая доля их участвует в управлении и защите электрооборудования электроэнергетических систем (ЭЭС). От такого программного обеспечения (ПО) требуется высокая надежность выполнения своих функций.

В инженерном сообществе программные системы имеют репутацию неустойчивых. Известны случаи, когда ошибка программистов приводила к крупным авариям с большими финансовыми ущербами. Так, 14.08.2013 г. на восточном побережье США 55 млн человек оказались без электричества, потому что оператор ЭЭС отключил сработавший сигнал о нарушении, чтобы не мешал работать, исправил режим и забыл включить сигнал, а программа не напомнила об этом, и следующее нарушение режима привело к крупной аварии, поскольку сигала о нем не поступило. Из-за того, что проектировщики забыли о возможности пересечения линии перемены дат, 12 истребителей, направлявшихся из США в Окинаву, потеряли доступ к данным о количестве топлива, датчикам скорости и высоты, частично нарушилась связь. В течение нескольких часов самые современные истребители Америки F-22 Raptor летели через океан совершенно беспомощными. В конце концов, их удалось посадить только благодаря мастерству пилотов. И подобных примеров сотни.

2. Специфика программ критически важных приложений

Почему же инженеры не избегают ПО, коль оно не заслуживает доверия? Обычно отмечают три основных преимущества замены аппаратного обеспечения программным:

1. Технология ПО позволяет встраивать в систему больше логики. Компьютерные системы с программным управлением могут различать большое количество ситуаций и выдавать выходные данные, соответствующие каждой из них. Системы с жесткой логикой не могли бы добиться такого поведения без непомерно большого количества аппаратных средств. Программируемое оборудование стоит дешевле, чем эквивалентная жестко связанная логика, потому что оно имеет регулярную структуру и производится массово. Экономические аспекты ситуации также позволяют системам с программным управлением выполнять больше проверок; надежность может быть увеличена за счет периодического выполнения программ, проверяющих аппаратуру.

С другой стороны, стоимость написания и сертификации действительно надежного ПО очень высока; к тому же необходимо принимать во внимание затраты на сопровождение – опять же такое, которое не подрывает надежности и безопасности. Показательный пример: только сопровождение относительно простого и не очень большого по объему (около 400 тыс. слов) ПО для бортового компьютера, установленного на американском космическом корабле типа Shuttle [1], стоил NASA 100 млн долл. год.

2. Логику, реализованную в ПО, теоретически легче изменить, чем логику, реализованную в аппаратном обеспечении. Многие изменения могут быть сделаны без добавления новых компонентов. Если система тиражируется или находится в труднодоступном физическом месте, гораздо проще внести изменения в ПО, чем в аппаратное обеспечение.

С другой стороны, изменения в программных модулях легко выполнить технически, однако трудно сделать это без внесения новых ошибок. Необходимые для гарантий безопасности верификация и сертификация означают новые большие затраты. К тому же, чем длиннее время жизни программы, тем более возрастает опасность вместе с изменениями внести ошибки – например, потому, что некоторые разработчики с течением времени перестают быть таковыми, а документация редко является исчерпывающей. Между тем, масштабы изменений в ПО могут быть весьма велики. Например, ПО для космических кораблей типа Shuttle за 10 лет сопровождения, начиная с 1980 г., подверглось 14-ти модификациям, приведшим к изменению 152 тысяч строк кода (полный объем ПО – 400 тысяч строк). Необходимость модернизации ПО диктовалась периодическим обновлением аппаратной базы, добавлением функциональности, а также происходило по причине необходимости исправления выявленных дефектов.

3. Компьютерные технологии и гибкость ПО позволяют предоставлять операторам больше информации и предоставлять ее в более полезной форме. Оператору современной системы с программным управлением может быть предоставлена информация, которая была бы немыслима в чисто аппаратной системе. Все это может быть достигнуто при использовании меньшего пространства и мощности, чем использовалось в некомпьютеризированных системах [2].

Рассматривая надежность относительно ПО следует помнить, что:

• самое очевидное различие между программными и аппаратными технологиями – это их *сложность*. Так,

точная документация в достаточно общей нотации для небольших программных систем может заполнить книжный шкаф, тогда как для аппарата достаточно несколько схем и краткого описания. Другим показателем сложности является время, которое требуется программисту для близкого знакомства с системой. Даже в небольших программных системах часто бывает так, что программисту требуется год работы с программой, прежде чем ему можно будет доверить самостоятельное внесение улучшений;

• следующее примечательное свойство ПО – *чувствительность к небольшим ошибкам*. В обычном машиностроении каждое конструктивное и производственное измерение может быть охарактеризовано допуском. От человека не требуется точного соответствия; достаточно быть в пределах установленного допуска на нужное значение. Использование допусков оправдано предположением, что небольшие ошибки имеют небольшие последствия. Хорошо известно, что в ПО тривиальные канцелярские ошибки могут иметь серьезнейшие последствия. Для ПО не известна полезная интерпретация допуска. Одна пунктуационная ошибка может быть катастрофической, даже если фундаментальные оплошности иногда имеют незначительные последствия;

• ПО, как известно, трудно поддается *адекватному тестированию*. Часто бывает, что часть ПО, подвергнутая тщательному и дисциплинированному тестированию, имеет серьезные недостатки. Тестирование аналоговых устройств основано на интерполяции. Предполагается, что устройства, которые хорошо работают в двух близких точках, будут хорошо работать и в промежуточных точках. В ПО такое предположение неверно. Количество случаев, которые должны быть протестированы для того, чтобы вызвать доверие к ПО, обычно очень велико;

• многие из предположений, которые обычно делаются при проектировании высоконадежного оборудования, недействительны для ПО. Разработчики высоконадежного оборудования озабочены производственными отказами и явлениями износа. Они могут проводить свой анализ, исходя из предположения, что отказы не сильно коррелируют, а одновременные отказы маловероятны. Те, кто оценивает надежность аппаратных систем, должны были бы быть обеспокоены ошибками проектирования и *коррелированными отказами*; однако во многих ситуациях влияние других типов ошибок является доминирующим. В ПО мало ошибок, вносимых на этапе производства (компиляции); когда такие ошибки есть, они носят систематический, а не случайный характер. ПО не изнашивается. Ошибки, с которыми должны быть связаны специалисты по надежности ПО, – это ошибки проектирования. Эти ошибки нельзя считать статистически независимыми. Существует множество доказательств того, что даже когда программы для данной задачи пишут люди, не знающие друг друга, они имеют тесно связанные ошибки [3, 4].

Эти свойства являются фундаментальным следствием того факта, что математические функции, реализуемые ПО, являются не непрерывными функциями, а функциями с произвольным числом разрывов. Отсутствие ограничений непрерывности на функции, описывающие программные эффекты, затрудняет нахождение компактных описаний ПО. Отсутствие таких ограничений придает ПО гибкость, но также и сложность. Аналогично, чувствительность к небольшим ошибкам и трудности тестирования можно отнести к фундаментальным математическим свойствам; вряд ли удастся найти чудодейственное лекарство. Для программных систем, критичных к надежности, всегда будет требоваться большая дисциплина и тщательный контроль.

В отличие от ситуации с аппаратными системами, нельзя добиться более высокой надежности путем дублирования программных компонентов. Просто дублируются ошибки. Даже если программы пишутся независимо друг от друга, ошибки, допущенные одним программистом, часто разделяются другими. В результате нельзя рассчитывать на повышение надежности программных систем только за счет наличия трех компьютеров там, где достаточно одного, хотя и здесь все не так просто.

Уместно напомнить, что в практике разработки и использования ПО огромное значение решению проблемы его безопасности отводится исполнением ряда стандартов серии ГОСТ Р МЭК 61508-*N*-2012 «Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью», где $N = 1, \ldots, 4$. Эта серия детализирует цели и требования при управлении безопасностью функционирования, в том числе, программируемых систем. Но фактическое наполнение этих требований остается за разработчиками ПО.

3. Особенности программ микропроцессорных защит

Рассмотрим приложение перечисленных выше свойств и особенностей к программам микропроцессорной защиты, которые также относятся к программным системам, критичным к надежности.

Прикладные программные модули релейной защиты отличаются сравнительно небольшим объемом программ (в среднем 3-5 тыс. строк в ассемблерном эквиваленте), что определяется, прежде всего, требованиями к быстродействию защиты. С другой стороны, такие объемы создают хорошую обозреваемость текста программы, что снижает появление ошибок при проектировании. Сюда же можно отнести то, что в ряде случаев такие прикладные программы готовятся на языках программируемых логических контроллеров (ПЛК) [5], что также позволяет снизить вероятность программной ошибки. Однако операционная среда пишется на более традиционных для ПО языках, например, Си, Ява и т.п.

Другой характерный момент этих модулей связан с тем, что то или иное нарушение режима компонентов ЭЭС может быть выявлено различными способами на основе одних и тех же входных данных и нередко на разных микропроцессорах. В традиционной защите такой подход известен как резервирование. В микропроцессорной защите он позволяет дублировать ПО при обработке входных данных, поскольку алгоритмы их обработки не повторяются. В результате при такой организации в определенной степени ослабляется ограничение по дублированию программ, поскольку подобная организация их работы – скорее резервирование, чем дублирование, хотя при моделировании надежности они включаются параллельно с точки зрения выполнения задачи (выявления недопустимого режима). Конечно, и здесь каждая из программ может иметь свои не выявленные и не похожие между собой ошибки, но это уже ближе к дублированию аппаратной части защиты.

И, конечно же, важно, чтобы система защиты не загружалась какими-либо вспомогательными фоновыми задачами, которые могут стать источником ошибок и сбойных ситуаций. Она должна быть максимально автономной, если мы считаем ее критически важным приложением. Стремление к объединению нескольких задач защиты на одном процессоре приводит к росту объема ПО, следовательно, к худшей обозреваемости программ, а отсюда и к увеличению вероятности программных ошибок.

Таким образом, различные вероятностные оценки показателей надежности функционирования ПО релейной защиты, подобные оценкам аппаратной части, дают только качественную картину в довольно широком диапазоне. О количественной оценке можно судить только из статистики, которая появляется через годы эксплуатации значительного числа устройств.

Так, по статистическим данным [6] количество микропроцессорных устройств релейной защиты и автоматики (РЗА), находящихся в работе, в 2013 г. составило 274062 устройств, а в 2014 г. – 319912 устройств (табл. 4, [6]). Из данных [7] «Распределение случаев неправильной работы устройств РЗА по видам технических причин и типам устройств РЗА за период с 01.01.2020 по 30.06.2020» известно, что из 727 случаев отказа РЗА 29 случаев связаны с отказом аппаратной части микропроцессорных защит, а 18 случаев – с отказом или сбоем их ПО. Тогда прогнозное число устройств РЗА на 2020 год по отношению к 2013 году (за 7 лет) может составить из формулы

$$a_n=a_1(1+r)^n$$

при $r=(a_2-a_1)/a_1$, где a_1 – число устройств первого года, a_n – число устройств на год n, r – средний годовой прирост устройств,

$$a_7 = 274062\left(1+\frac{319912-274062}{274062}\right)^7 = 809321 \text{ устр.}$$

Примем долю РЗА «Россети» в 70 % от всех устройств в [7]. Тогда грубая оценка интенсивности отказов $\lambda=\frac{m}{0,7\cdot Nt}=\frac{18\cdot 2}{0,7\cdot 809321}=6,35\cdot 10^{-5}$ лет$^{-1}$. Здесь m – число отказов устройств из-за ПО, за полгода (2 в числителе), $0,7\cdot N$ – число всех микропроцессорных защит, t – расчетный срок (год). При времени восстановления $t_в$ = 2 ч ($\lambda=\frac{m}{0,7\cdot Nt}=\frac{18\cdot 2}{0,7\cdot 809321}=6,35\cdot 10^{-5}$ лет$^{-1}$) коэффициент готовности ПО $K_{гпо}=\frac{\mu}{\lambda+\mu}=\frac{4380}{6,35\cdot 10^{-5}+4380}=0,999999985$. А средняя наработка до ошибки $t_\Sigma=\frac{1}{\lambda}=15748$ лет. Из соотношения $d_{ПО}=\frac{m_{ПО}}{m_{ПО}+m_{Прч}}\cdot 100\%=\frac{18}{727}\cdot 100\%\approx 2,5\%$, где $m_{ПО}+m_{Прч}=727$ случаев неправильной работы микропроцессорных защит, где $m_{Прч}$ – число случаев неправильной работы защиты, не связанное с программным обеспечением. Отметим, что доля отказов из-за программных ошибок составила 2,5%.

4. Заключение

Программы микропроцессорных защит относятся к программам критически важных приложений, работающих в режиме реального времени, и поэтому от них требуется высокая надежность выполнения своих функций. Учитывая развитие и широкое применение компьютерных технологий и гибкость программного обеспечения микропроцессорных защит, нельзя забывать о сложности их описания, чувствительности даже к малым отклонениям от основного алгоритма, о проблемах тестирования, связанных с отсутствием ограничений непрерывности функций программ, описывающих алгоритмы решений, что не способствует разработке проектов с ошибками, мало влияющими на исполнение требуемых функций.

Есть ряд свойств у этих программ, способствующих улучшению их показателей надежности, таких как краткость, языки программируемых логических контроллеров, но этого недостаточно для решения проблем надежности программ микропроцессорной защиты, как программ, критически важных приложений. С другой стороны, необходима достаточная автономность отдельных устройств защит, ибо стремление к комплексности часто снижает параметры надежности защиты. Здесь следует искать оптимальные решения, учитывающие как функциональные возможности программируемых защит, так и достижение требуемых характеристик надежности исполнения этих функций.

Работа выполнена в рамках темы АААА-А20-120051590026-3 «Модели и методы адаптации систем энергетики в современных условиях».

Библиографический список

1. An Assessment of Space Shuttle Flight Software Development processes. – Committee for Review of Oversight Mechanisms for Space Shuttle Flight Software Development Processes, National Research Council, 1993. 206 p. URL: https://ntrs.nasa.gov/api/citations/19930019745/downloads/19930019745.pdf (дата обращения 15.06.2023).

2. Parnas D.L., Schouwen A.J., Kwan S.P. Evaluation of Safety-Critical Software // Communications of the ACM. 1990, Vol. 33, No. 6. Pp. 636-648.

3. Bishop P.G., Pullen F.D. Error Masking: A Source of Failure Dependency in Multi-Version Programs / 1991, January. 17 p. DOI: 10.1007/978-3-7091-9123-1.3

4. Eckhardt D.E. et al. An experimental evaluation of software redundancy as a strategy for improving reliability / D.E. Eckhardt A.K. Caglayan J.C. Knight L.D. Lee D.F. McAllister J.P.J. Kelly // IEEE Transactions on Software Engineering. Vol.17, Is.7, 1991. Pp. 692-702. DOI: 10.1109/32.83905

5. Лившиц Ю. Е. Программируемые логические контроллеры для управления технологическими процессами. Минск: БНТУ, 2014. Ч. 1. 206 с.

6. Концепция развития релейной защиты и автоматики электросетевого комплекса // Приложение № 1 к протоколу Правления ОАО «Россети» от 22.06.2015 № 356пр. М., 2015. 49 с. URL: https://mig-energo.ru›wp-content/uploads/2015/12/rza-fsk.pdf (дата обращения 15.06.2023).

7. Распределение случаев неправильной работы устройств РЗА по видам технических причин и типам устройств РЗА за период с 01.01.2020 по 30.06.2020. URL: https://www.so-ups.ru/fileadmin/files/company/rza/rza_rez_info/rza_rez_vid_teh_1-2k2020.xl (дата обращения 15.06.2023).

References

1. An Assessment of Space Shuttle Flight Software Development processes. Committee for Review of Oversight Mechanisms for Space Shuttle Flight Software Development Processes, National Research Council; 1993. (accessed 15.06.2023). Available at: https://ntrs.nasa.gov/api/citations/19930019745 /downloads/19930019745.pdf.

2. Parnas D.L., Schouwen A.J., Kwan S.P. Evaluation of Safety-Critical Software. *Communications of the ACM* 1990;33(6):636-648.

3. Bishop P.G., Pullen F.D. Error Masking: A Source of Failure Dependency in Multi-Version Programs; 1991. DOI: 10.1007/978-3-7091-9123-1.3.

4. Eckhardt D.E., Caglayan A.K., Knight J.C., Lee L.D., McAllister D.F., Kelly J.P.J. An experimental evaluation of software redundancy as a strategy for improving reliability. *IEEE Transactions on Software Engineering* 1991;17(7):692-702. DOI: 10.1109/32.83905.

5. Livshits Yu.E. [Programmable logic controllers for process control. Part 1]. Minsk: BNTU; 2014.

6. [Concept for the development of relay protection and automatic equipment of the integrated power grid. Annex 1 to the Minutes of meeting of the Rosseti Executive Board no. 356 dated 22.06.2015].

7. [Distribution of RPAE malfunctions by the types of technical causes and types of RPAE devices for the period between 01.01.2020 and 30.06.2020]. (accessed 15.06.2023). Available at: https://www.so-ups.ru/fileadmin/files/company/rza/rza_rez_info/rza_rez_vid_teh_1-2k2020.xl.

Сведения об авторе

Успенский Михаил Игоревич – кандидат технических наук, ведущий научный сотрудник лаборатории электрических систем Института социально-экономических и энергетических проблем Севера Коми научного центра Уральского отделения Российской академии наук, ул. Коммунистическая, д. 26, Сыктывкар, Республика Коми, Российская Федерация 167982, e-mail: uspensky@energy.komisc.ru

About the author

Mikhail I. Uspensky, Candidate of Engineering, Lead Researcher, Laboratory for Electrical Systems, Institute for Socioeconomic and Energy-Related Problems of the North, Komi Science Centre, Urals Division, Russian Academy of Sciences, 26 Kommunisticheskaya St., Syktyvkar, 167982, Komi Republic, Russian Federation, e-mail: uspensky@energy.komisc.ru.

Вклад автора

Вклад автора состоит в приложении понятий критически важного программного обеспечения к системам микропроцессорной защиты и автоматики с выделением особенностей последних для правильного использования опыта эксплуатации критического программного обеспечения в устройствах релейной защиты и автоматики.

Конфликт интересов

Автор заявляет об отсутствии конфликта интересов.