Программные дефекты, функциональная безопасность и МЭК 61508

С.Н. Зыль (ООО "СВД Встраиваемые Системы")

Статья посвящена теме программных дефектов и их влияния на функциональную безопасность вычислительных систем РВ. Обсуждается вопрос оценки безопасности ПО методом "Доказано-на-практике" стандарта МЭК 61508.

Ключевые слова: отказоустойчивость, надежность, функциональная безопасность, дефекты, реальное время, SIL, МЭК 61508, коэффициент готовности.

Сбой программы – это всегда неприятно. Вы регулярно "сохраняетесь", когда готовите электронный документ? Я делаю это рефлекторно каждые 10...15 мин. И даже чаще – если на что-то отвлекаюсь, то сначала жму кнопку "Сохранить", а потом уже отвлекаюсь. Эта привычка появилась у меня с тех давних пор, как во время подготовки дипломной работы небольшой сбой уничтожил плоды нескольких часов кропотливого набора математических формул в "Лексиконе"... Но было бы хорошо, если бы сбои программ всегда были просто неприятностью. Однако в современном компьютеризированном мире программы – это не только текстовые редакторы и игры, это еще и управление реакторами, турбогенераторами, АБС автомобиля... То есть существуют программы, функциональная безопасность которых весьма критична для нас, о таких программах идет речь в этой статье.

Для начала вспомним, что такое *функциональная безопасность* (по-англ. *functional safety* или просто *safety*). Это отсутствие неприемлемого риска здоровью людей, их собственности или окружающей среде со стороны функционирующей технической системы. С понятием функциональной безопасности тесно связано понятие *система безопасности* (по англ. *Safety-Related System*) – система, выполняющая одну или несколько специальных функций, снижающих риск со стороны функционирующей технической системы до приемлемого уровня. Кстати, эти специальные функции – *функции безопасности* (по англ. *Safety Functions*) могут осуществляться в отношении технической системы как отдельной специальной системой безопасности, так и непосредственно технической системой.

Важно понимать, что в общем смысле функциональная безопасность не имеет ничего общего с надежностью – в зависимости от контекста эти два понятия могут быть прямо противоположными. Например, автомат Калашникова является очень надежной технической системой, но совсем небезопасной. И, напротив, домашний компьютер для игр может "зависать" хоть каждые 5 мин, но при этом не представлять никакой угрозы для жизни людей.

Чтобы эта мысль была полностью понятной, вспомним, что распространенной числовой метрикой надежности является *коэффициент готовности* – выраженное в процентах отношение среднего времени наработки технической системы на отказ к среднему периоду восстановления ее работоспособного состояния. То есть любое сокращение времени наработки на отказ формально снижает надежность технической системы. При этом для системы безопасности своевременное отключение технической системы может являться основной функцией безопасности, например, отключение электропитания объекта во избежание разрушения изоляции при превышении допустимой температуры электрического кабеля.

На высоте десять тысяч метров
слово "надежность" звучит как-то иначе...
(Неизвестный автор)

Совсем другое дело, конечно, если речь идет о надежности системы безопасности. В этом случае функциональная безопасность технической системы прямо пропорциональна коэффициенту готовности системы безопасности. Но важно помнить, что обеспечение надежности и обеспечение функциональной безопасности – это не одно и то же.

Еще одно важное обстоятельство заключается в том, что говорить о надежности, в бытовом смысле этого слова, отдельно взятого ПО, мягко говоря, некорректно. С таким же основанием можно было бы говорить о надежности инструкции по эксплуатации стиральной машины. Ведь любая программа или даже такой сложный программный комплекс как ОС – это всего лишь записанная в машинных кодах инструкция или набор инструкций, указывающая, что нужно делать арифметико-логическому устройству центрального процессора. Программы не ржавеют, не рвутся, не испаряются. В этом смысле можно говорить только о надежности программно-аппаратных комплексов. Но нужно помнить, что, по сути дела, речь идет о надежности аппаратуры во время выполнения ею инструкций-программ.

Теперь вернемся к программным сбоям – сбоям ЭВМ, в которых не виновна аппаратура и при которых аппаратура продолжает корректно функционировать. Программные сбои можно разделить на два типа – "зависание" и "крах". "Зависания" обычно возникают вследствие логических ошибок, например, тупиковых ветвей алгоритма, бесконечных циклов или блокировок по ожиданию не наступающего по каким-то причинам события. А "крах" программы – это завершение процесса ядром ОС, когда процесс нарушит принятые в данной ОС "правила игры", например, пытается "залезть" в чужую область памяти или выполняет деление на нуль. Причинами краха могут быть такие логические ошибки, как переполнение буфера, использование переменной до инициализации, использование указателя после освобождения и т.п.

Логические ошибки называют дефектами программы. Когда мы говори о надежности программы, то имеем ввиду:

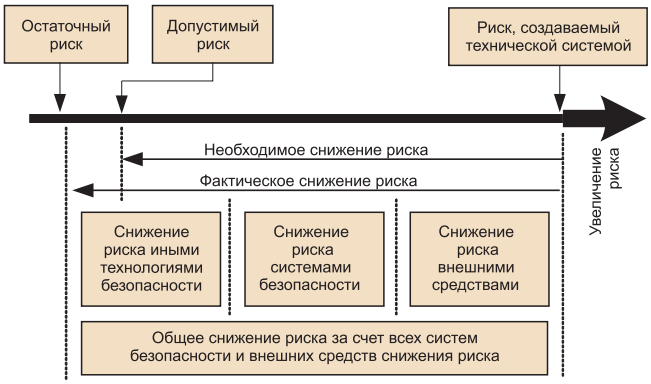

Рис. 1. Концепция снижения риска МЭК 61508

- наличие или отсутствие дефектов, которые могли бы привести к сбою программы;

- способность программы корректно выполнять работу при проявлении дефектов (сия способность называется *отказоустойчивостью*).

О мерах по обеспечению отказоустойчивости – резервировании, квотировании ресурсов ЭВМ, автоматической идентификации сбоев, восстановлении работоспособного состояния – написано уже немало, поэтому в данной статье сосредоточимся на мерах, направленных на минимизацию дефектов.

С точки зрения конечного пользователя, проявления дефектов ПО могут представлять собой от небольшого неудобства до техногенных катастроф. С точки зрения разработчика не выявленные дефекты могут, например, создавать угрозу его репутации. Самое досадное заключается в том, что на сегодняшний день не существует методов, позволяющих гарантировать отсутствие дефектов ни в спецификациях (включая технические задания), ни непосредственно в программах, ни в эксплуатационной документации. Это обстоятельство весьма ярко иллюстрируется тем, что ни одна испытательная лаборатория не даст заключение, что программа не содержит дефектов – она даст заключение, что провела такие-то проверки и дефектов не обнаружила. Улавливаете разницу?

Особую остроту проблема приобретает, когда речь идет о программных дефектах в системах безопасности – ведь они могут привести к тому, что система безопасности не выполнит какую-либо из своих функций безопасности. Что в свою очередь повышает риск того, что техническая система нанесет ущерб здоровью людей, их собственности или окружающей среде.

Введем понятия, связанные с риском, используемые МЭК 61508 – базовым международным стандартом по функциональной безопасности.

Как видно из рисунка, существует *риск, создаваемый технической системой* для здоровья людей, сохранности их собственности или для окружающей среды. Значит, для достижения *допустимого риска* должны быть приняты меры, обеспечивающие *необходимое снижение риска*. Допустимый риск – это риск, допускаемый обществом

Таблица 1. Соответствие уровней SIL значениям фактора снижения риска

Уровень целостности безопасности (SIL)	Значение фактора снижения риска
4	10 000 ... 100 000
3	1 000 ... 10 000
2	100 ... 1 000
1	10 ... 100

по отношению к данной технической системе. То есть в каком-то смысле допустимый риск – понятие субъективное и может определяться на основе юридических требований, директив регулирующих данную отрасль органов, возможной тяжести ущерба окружающей среде, количества подвергаемых опасности людей и т.д. Риск для здоровья людей может быть снижен, например, путем их эвакуации из района, в котором планируется эксплуатация технической системы – это обеспечит снижение риска внешними средствами. Кроме того, в самой технической системе могут быть предусмотрены различные механические, гидравлические, пневматические, электрические и другие средства безопасности – предохранители, расширительные емкости, клапаны, фильтры и т.п., то есть обеспечено снижение риска иными технологиями безопасности. И наконец, системы безопасности должны обеспечить такое снижение риска, чтобы фактическое снижение риска за счет всех систем безопасности и внешних средств снижения риска обеспечило необходимое снижение риска, при котором остаточный риск был бы ниже, чем допустимый риск.

Другими словами, каждая из функций безопасности системы безопасности должна обеспечить необходимое снижение какого-либо из идентифицированных рисков. Поэтому к системам безопасности предъявляются:

- требования к функциям безопасности, определяющие, в чем именно заключаются эти функции;

- требования к целостности безопасности (англ. *Safety Integrity*[1]), понимаемой как вероятность того, что функции безопасности будут выполнены успешно.

Требования к функциям безопасности разрабатываются на основе анализа угроз, возникающих в процессе эксплуатации технической системы. Требования к целостности безопасности задаются исходя из оценки рисков в виде уровня SIL (Safety Integrity Level).

Уровень SIL для компьютерной системы по МЭК 61508 определяется исходя из полученного с помощью какого-либо предлагаемого стандартом подхода значения параметра, получившего название "фактор снижения риска". Влияние величины фактора на уровень SIL показано в табл. 1.

Фактор снижения риска показывает, во сколько раз меры, принятые при спецификации требований, разработке, поставке и эксплуатации программного продукта, снижают вероятность невыполнения функций безопасности по вине этого программного продукта. Например,

[1] Слово integrity в английском языке кроме "целостность" означает так же "честность", "неподкупность", "неприкосновенность". Каждое из этих слов хорошо описывает суть дела.

значение фактора снижения риска, равное 5000, означает вероятность отказа системы безопасности 0,0002%.

Для ПО максимальным уровнем является SIL3. Значение фактора снижения риска для ПО может быть установлено с помощью одного из двух подходов или их комбинации:

- методом сертификации;
- методом "Доказано-на-Практике" (по-англ. Proven-in-Use).

Метод сертификации заключается в исследовании артефактов всех этапов создания ПО. Метод "Proven-in-Use" позволяет установить значение фактора снижения риска на основе опыта реальной эксплуатации ПО в контролируемых, документированных условиях.

Вопрос сертификации ПО на соответствие требованиям МЭК 61508 лежит за пределами статьи. Что касается метода "Proven-in-Use", то для его использования МЭК 61508 требует, что бы ПО имело четко определенную функциональность и конфигурацию, а также документально зафиксированные статистические данные по отказам при использовании данной его функциональности в данной его конфигурации. Документация должна содержать точное обозначение ПО, описание его конфигурации, процедур обнаружения, регистрации и устранения отказов.

Следует иметь в виду, что под словом "конфигурация" понимается не только перечень программных компонентов, их версии, порядок установки, параметры запуска и т.п., но и условия эксплуатации – взаимодействие с другими системами, человеческие факторы и др. факторы, которые могут повлиять на проявление дефектов. Любое изменение конфигурации допустимо только, если доказано, что это изменение не влияет на проявление возможных дефектов.

Такое ПО как ОС относится к классу так называемых систем непрерывного использования. Поэтому требования к нему задаются в виде минимального суммарного времени эксплуатации, как это представлено в табл. 2.

Таблица 2. Суммарное время эксплуатации для соответствия уровням SIL

Уровень целостности безопасности (SIL)	Суммарное время эксплуатации, часы (для доверительной вероятности 95%)
4	3×10^9
3	3×10^8
2	3×10^7
1	3×10^6

Для обеспечения репрезентативности статистических данных по эксплуатации должны выполняться условия:

- наблюдение должно вестись в течение времени не меньшего, чем 1 год;
- под наблюдением должно быть не менее 10 различных систем, имеющих различную реализацию функционального программного обеспечения.

То есть для подтверждения соответствия ОС уровню SIL3 необходимо наблюдать не менее 10 разных компьютерных систем на базе этой ОС в течение не менее 1 года так, что бы суммарное время наблюдения всех компьютерных систем составляло ≥300 млн. часов (чуть больше чем 34246 лет).

Меры по предотвращению или минимизации ущерба от проявления дефектов определяются исходя из экономической целесообразности. То есть сначала необходимо ответить на вопрос: опасность какого уровня может возникнуть при отказе технической системы или системы безопасности в результате проявления дефекта? Другими словами, какие риски для активов конечного пользователя или разработчика могут возникнуть в случае проявления дефектов?

Конечно, тестирование не дает гарантий работоспособности ПО во всех технологических режимах объекта применения. Однако использование проверенных на практике продуктов позволяет с высокой степенью уверенности говорить о безопасности их использования там, где цена отказа очень велика.

Зыль Сергей Николаевич *– технический директор ООО "СВД Встраиваемые Системы".*
Контактные телефоны: (812) 971-71-64, 373-41-17. Http://www.kpda.ru