АВТОМАТИКА, ТЕЛЕМЕХАНИКА И СВЯЗЬ

УДК 629.067:004.42

АНАЛИЗ МЕТОДОВ ПОСТРОЕНИЯ ФУНКЦИОНАЛЬНО БЕЗОПАСНЫХ СТРУКТУР НА ЭЛЕМЕНТНОЙ БАЗЕ РАЗЛИЧНОЙ СЛОЖНОСТИ

Е. Н. РОЗЕНБЕРГ, доктор технических наук, Н. Г. ПЕНЬКОВА, С. В. ИПАТОВ, АО «Научно-исследовательский и проектно-конструкторский институт информатизации и связи на железнодорожном транспорте, г. Москва;
К. А. БОЧКОВ, доктор технических наук, Белорусский государственный университет транспорта, г. Гомель

Представлен сравнительный анализ централизованной и распределённой архитектур функционально безопасных систем с учётом эффективности фонового тестирования и количественной оценки безопасности по показателю *PFH* (Probability of Dangerous Failure per Hour). Рассмотрено влияние числа реализуемых функций, сложности программного обеспечения и фактора отказов по общей причине на длительность цикла диагностики и уровень функциональной безопасности. Показано, что, хотя при небольшом числе функций централизованная архитектура может обеспечивать более короткий цикл тестирования, при учёте рисков, связанных с программным обеспечением распределённая архитектура демонстрирует значительные преимущества за счёт изоляции компонентов, снижения объёма и сложности программного кода на узел, а также уменьшения вероятности отказов по общей причине. На основе проведённого анализа предложены критерии выбора архитектуры на ранних этапах проектирования.

Введение. Обеспечение функциональной безопасности ответственных систем требует не только аппаратной надёжности, но и эффективных механизмов постоянного контроля их работоспособности. Одним из важных инструментов для этого является фоновое тестирование, которое позволяет проверять ресурсы вычислительного устройства в периоды простоя основных рабочих, технологических, процессов системы [1]. К основным характеристикам такого метода диагностирования, оказывающим влияние на выполнение требований безопасности, относятся полнота контроля и длительность полного цикла проверки. Данные параметры тестирования напрямую зависят от архитектуры системы. Каждая из архитектур обладает своими ограничениями и преимуществами, с учётом которых разработчиками на стадии планирования системы делается выбор в ту или иную сторону.

В данной статье рассматриваются две архитектуры, использующие элементную базу различной сложности: централизованная и распределённая.

Под «функцией» в рамках данной статьи понимается логически завершённый и автономный модуль системы, реализующий одну конкретную задачу управления или контроля.

В централизованной архитектуре используется одно высокопроизводительное вычислительное устройство, выполняющее сразу все функции системы, включая фоновое тестирование (рисунок 1). Это приводит к линейной зависимости полного времени тестирования от количества выполняемых функций, что при высокой рабочей (технологической) нагрузке может стать критичным для соответствия количественным требованиям безопасности. Интенсивность $\lambda_{\text{ф}}^{\text{оп}}$ опасного накопления скрытых отказов в дублированных каналах системы, связанная с величиной полного цикла тестирования $t_{\text{ф}}$ (плюс времени на восстановление работоспособности t_r) формулой $\lambda_{\text{ф}}^{\text{оп}} = 2\lambda_{\text{du}}^2\left(t_{\text{ф}} + t_r\right)$, с увеличением времени может выйти за допустимые нормы.

Распределённые архитектуры, напротив, используют множество маломощных устройств, параллельно обрабатывающих задачи и проводящих самотестирование (рисунок 2). Такой подход снижает нагрузку на отдельные компоненты, однако делает цикл полного тестирования зависящим от характеристик самого медленного элемента. Важно отметить, что в контексте данной статьи под распределенной архитектурой понимается не географически распределенная система, а архитектура, в рамках которой несколько физически и логически изолированных вычислительных устройств (микроконтроллеров) размещены в едином корпусе или стойке и функционируют параллельно, при этом каждая решает свою подзадачу. Следовательно, первостепенным является именно их аппаратная и программная изолированность.

Цель данной статьи – разработать обоснованный механизм выбора архитектуры функционально безопасных систем (централизованной или распределённой) на ранних этапах проектирования с учётом как эффективности фонового тестирования, так и количественной оценки безопасности. Для достижения этой цели проведён сравнительный анализ обеих архитектур по следующим критериям: зависимость полного времени тестирования от числа функций, рабочей нагрузки и быстродействия элементной базы; влияние сложности программного обеспечения на интенсивность отказов; роль фактора общих причин. На основе полученных результатов сформулированы пороговые условия, при которых распределённая архитектура обеспечивает существенное преимущество в безопасности.

Начнём с централизованной системы, цикл её фонового тестирования схематично представлен на рисунке 1. Основной акцент делается на том, что для завершения цикла системе необходимо протестировать все доступные ей ресурсы один за другим по очереди.

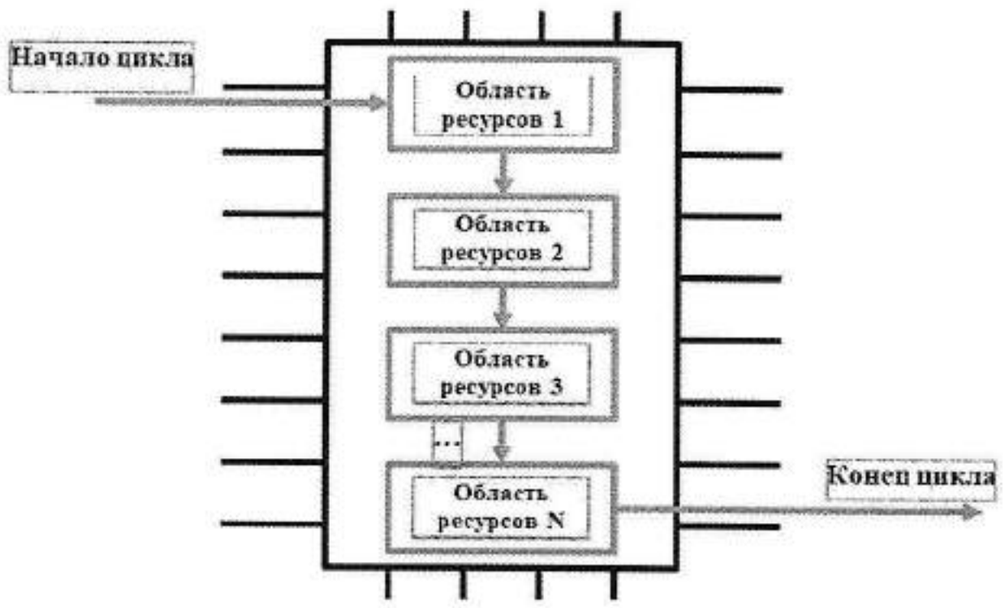

Рисунок 1 – Цикл фонового тестирования централизованной системы

Особенности централизованной системы:

– используется одно высокопроизводительное вычислительное устройство (например, процессор архитектуры Эльбрус);

– тестирование всех ресурсов выполняется этим же устройством в периоды, свободные от основной технологической работы;

– время полного цикла тестирования растет линейно с увеличением объема тестируемых ресурсов;

– объём тестируемых ресурсов прямо пропорционален количеству реализуемых функций N, так как каждая функция требует выделения памяти, процессорного времени и других ресурсов.

Допустим, представленная модель предполагает последовательное выполнение функций и их тестирования, что является консервативной оценкой для централизованной архитектуры. Безусловно, современные многоядерные процессоры позволяют организовать параллельное выполнение. Однако необходимо учитывать, что:

– даже в многоядерных системах такие ресурсы, как общая память, шины и подсистема ввода-вывода, остаются общими, создавая «узкие места» и не позволяя добиться полного параллелизма;

– фоновое тестирование общих ресурсов (кэшей, межъядерных соединений, контроллеров памяти) является сложной задачей, которая часто требует временного приостановления работы других ядер, что снижает выигрыш от параллелизма;

– многоядерность не уменьшает, а скорее увеличивает фактор отказов по общей причине (β), так как ядра тесно связаны на аппаратном уровне.

Таким образом, хотя учёт параллелизма может улучшить количественные показатели централизованной архитектуры, но при этом серьезно осложнит анализ.

Рассмотрим продолжительность полного цикла тестирования для централизованной архитектуры.

Определение длительности времени тестирования за 1 рабочий цикл в централизованной системе примем по формуле

$$t_{\text{Ц.тест}} = S_{\text{Ц}} t_{\text{рабоч.цикл}}, \qquad (1)$$

где $S_{\text{Ц}}$ – доля рабочего цикла, выделенная на фоновое тестирование в централизованной системе; $t_{\text{рабоч.цикл}}$ – длительность рабочего цикла системы.

При условии последовательного тестирования в централизованной системе получается, что общая длительность времени ($t_{\text{Ц.тест.набора}}$), необходимого на тестирование всех N функций без учета времени рабочей (технологической) нагрузки, определяется по формуле

$$t_{\text{Ц.тест.набора}} = \sum_{i=1}^{N} t_i, \qquad (2)$$

где N – количество функций, выполняемых централизованной системой; t_i – время, необходимое для тестирования одной i-й функции.

Теперь, зная $t_{\text{Ц.тест.набора}}$ и $t_{\text{Ц.тест}}$, можно определить количество рабочих циклов централизованной системы, необходимых для тестирования всех N функций в централизованной системе с учётом времени рабочей (технологической) нагрузки:

$$Q_{\text{Ц.циклов}} = \frac{t_{\text{Ц.тест.набора}}}{t_{\text{Ц.тест}}} = \frac{\sum_{i=1}^{N} t_i}{S_{\text{Ц}} t_{\text{рабоч.цикл}}}. \qquad (3)$$

Таким образом, длительность времени полного цикла тестирования системы с централизованной архитектурой определяется следующим образом:

$$T_{\text{центр}} = t_{\text{рабоч.цикл}} Q_{\text{Ц.циклов}} = \frac{t_{\text{рабоч.цикл}}}{S_{\text{Ц}} t_{\text{рабоч.цикл}}} \sum_{i=1}^{N} t_i = \frac{\sum_{i=1}^{N} t_i}{S_{\text{Ц}}}. \qquad (4)$$

Теперь рассмотрим распределённую систему, цикл её фонового тестирования схематично представлен на рисунке 2. Основной акцент делается на том, что каждый вычислительный модуль является самостоятельной единицей и выполняет тестирование доступных ему ресурсов независимо от остальных.

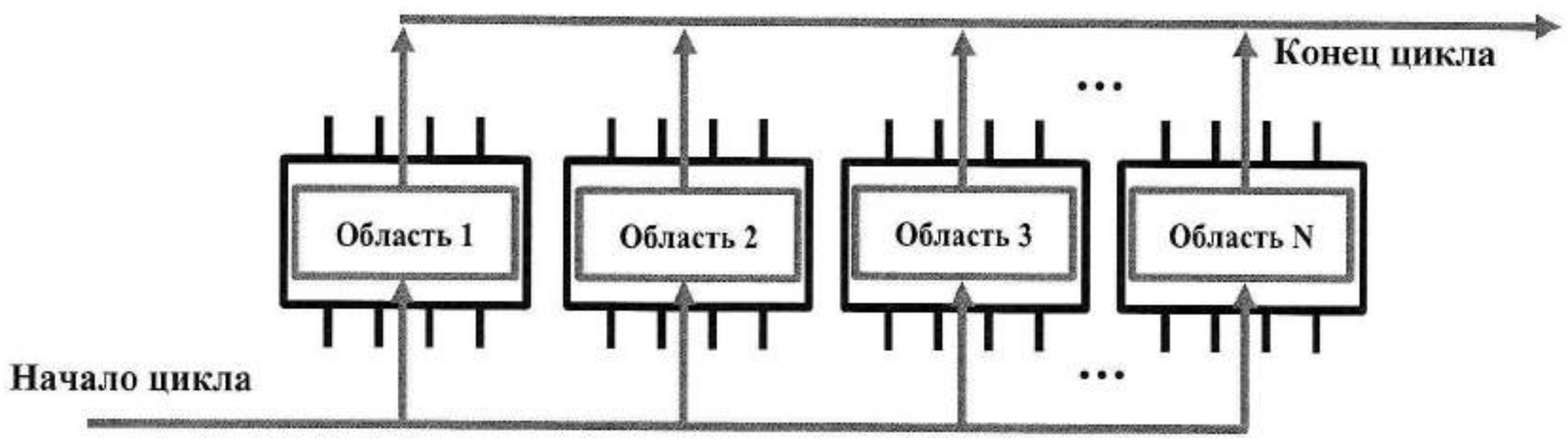

Рисунок 2 – Цикл фонового тестирования распределённой системы

Особенности распределённой системы:

– используется множество специализированных маломощных устройств (например, микроконтроллеры Миландр);

– каждое устройство выполняет фоновое самотестирование параллельно с другими в периоды простоя от своей основной функции;

– время полного цикла тестирования всей системы определяется быстродействием самого медленного устройства (принцип «слабого звена»);

– общее число устройств равно количеству функций N (или пропорционально ему), при этом объём тестируемых ресурсов и сложность ПО на каждом отдельном устройстве остаются фиксированными и небольшими, независимо от общего N.

Рассмотрим продолжительность полного цикла тестирования для распределённой архитектуры.

Определение длительности времени тестирования $t_{\text{Р.тест}}$ за 1 рабочий цикл в распределенной системе примем по формуле

$$t_{\text{Р.тест}} = S_{\text{Р}} t_{\text{рабоч.цикл}}, \quad (5)$$

где $S_{\text{Р}}$ – доля рабочего цикла одного устройства, выделенная на фоновое тестирование в распределенной системе; $t_{\text{рабоч.цикл}}$ – длительность рабочего цикла одного устройства системы.

Учитывая, что в распределённой системе тестирование отдельных элементов происходит параллельно, длительность времени $t_{\text{Р.тест.набора}}$ полного тестирования всех N вычислительных устройств (функций) без учета времени рабочей (технологической) нагрузки определяется по формуле

$$t_{\text{Р.тест.набора}} = \max_{1 \le i \le N} t_i, \quad (6)$$

где t_i – время, необходимое для тестирования одной i-й функции (устройства); N – количество функций (вычислительных устройств) в распределённой системе.

Зная $t_{\text{Р.тест.набора}}$ и $t_{\text{тест}}$ количество рабочих циклов распределённой системы, необходимых для полного цикла тестирования, определяем по формуле

$$Q_{\text{Р.циклов}} = \frac{t_{\text{Р.тест.набора}}}{t_{\text{Р.тест}}} = \frac{\max\limits_{1 \le i \le N} t_i}{S_{\text{Р}} t_{\text{рабоч.цикл}}}. \quad (7)$$

Таким образом, длительность времени $T_{\text{распр}}$ полного цикла тестирования системы с распределённой архитектурой определяется следующим образом:

$$T_{\text{распр}} = t_{\text{рабоч.цикл}} Q_{\text{Р.циклов}} = \frac{t_{\text{рабоч.цикл}}}{S_{\text{Р}} t_{\text{рабоч.цикл}}} \cdot \max_{1 \le i \le N} t_i = \frac{\max\limits_{1 \le i \le N} t_i}{S_P}. \quad (8)$$

Теперь перейдём к сравнению продолжительности цикла фонового тестирования для рассматриваемых систем.

Рабочий цикл системы определяется как период времени от получения и начала обработки устройством входных данных до выдачи сформированного по ним технологического решения на объект контроля/управления. В течение данного времени система выполняет основные технологические функции (последовательное исполнение набора N функций для централизованной архитектуры и параллельное исполнение набора N функций – одна функция одним из N устройств при распределённой архитектуре) и проводит фоновое тестирование (в свободное от исполнения технологических функций время). Длительность рабочего цикла $t_{\text{рабоч. цикл}}$ выбирается разработчиками системы с учетом специфики реализации взаимодействия с объектом управления/контроля. В рамках текущего анализа $t_{\text{рабоч. цикл}}$ принимается одинаковой для обеих архитектур систем и для всех устройств из состава распределённой архитектуры.

Для упрощения и приведения к единообразию формул (4) и (8) введём следующие обозначения:

$t_{\text{Ц.функ}} = \dfrac{\sum\limits_{i=1}^{N} t_i}{N}$ – среднее время тестирования одной функции в централизованной системе; $t_{\text{Р.функ}} = \max\limits_{1 \le i \le N} t_i$ – максимальное время тестирования одной функции в распределённой системе.

После подстановки введенных обозначений в формулы (4) и (8) и упрощения время полного тестирования для централизованной и распределённой системы определяется по формулам

$$T_{\text{центр}} = \frac{N t_{\text{Ц.функ}}}{S_{\text{Ц}}}; \quad (9)$$

$$T_{\text{распр}} = \frac{t_{\text{Р.функ}}}{S_{\text{Р}}}. \quad (10)$$

Из приведённого сравнения следует, что время тестирования централизованной системы $T_{\text{центр}}$ растёт пропорционально N, а время тестирования распределённой системы $T_{\text{распр}}$ от N не зависит. Это даёт преимущество распределённой архитектуре относительно этого показателя при масштабировании системы (увеличении числа N).

Однако на практике из-за различия производительности устройств (мощный процессор против простого микроконтроллера) время тестирования одной функции в каждой архитектуре несопоставимо. Для корректного сравнения вводится коэффициент k, который показывает, во сколько раз максимальное время тестирования одной функции в распределённой системе отличается от среднего времени тестирования функции в централизованной системе:

$$\max_{1 \le i \le N} t_i = k \frac{\sum\limits_{i=1}^{N} t_i}{N}. \quad (11)$$

Здесь $k = 1$ соответствует равенству максимального времени тестирования функции в распределённой системе и среднего времени в централизованной. Это означает, что одна функция в централизованной системе тестируется в среднем столько же, сколько и самая длительная функция в распределённой системе; при $k < 1$ – распределённая быстрее, чем централизованная. Это означает, что, несмотря на повышенное быстродействие вычислительного устройства централизованной системы, фактор малого объёма тестируемых ресурсов и параллельность процессов в распределённой системе оказывает большее

влияние на снижение длительности тестирования; при $k > 1$ – распределенная медленнее, чем централизованная. Это означает, что за счет повышенного быстродействия вычислительного устройства централизованной системы тестирование одной функции происходит быстрее, чем в распределённой.

В дальнейшем рассматривается диапазон $0{,}5 \le k \le 1{,}5$ как наиболее реалистичный.

Для упрощения анализа представим, что выражение $\frac{\sum_{i=1}^{N} t_i}{N} = 1$ (условная единица).

В результате упрощения формул (9) и (10), получим выражения времени полного тестирования для централизованной и распаренной системы, приведённые к единообразию:

$$T_{\text{центр}} = \frac{N \frac{\sum_{i=1}^{N} t_i}{N}}{S_{\text{Ц}}} = \frac{N}{S_{\text{Ц}}}; \quad (12)$$

$$T_{\text{распр}} = \frac{k \frac{\sum_{i=1}^{N} t_i}{N}}{S_{\text{Р}}} = \frac{k}{S_{\text{Р}}}. \quad (13)$$

Для сравнения продолжительности полного цикла тестирования и выявления параметрических зависимостей воспользуемся наглядным графическим методом. При построении графиков примем следующие допущения и ограничения.

1 Из-за малой производительности устройств распределенная система имеет мало времени на тестирование в одном рабочем цикле вне зависимости от количества функций (устройств). Поэтому всегда принимаем $S_{\text{Р}} = 0{,}1$.

2 Для централизованной системы значение доли времени на тестирование $S_{\text{Ц}}$ является критическим параметром, который на практике сильно зависит от соотношения вычислительной мощности процессора и совокупной трудоемкости реализуемых функций. Поскольку в рамках данного исследования не производится привязка к конкретной аппаратной платформе и типу задач, для анализа выбраны три репрезентативных сценария, покрывающие широкий диапазон возможных состояний системы:

$S_{\text{Ц}} = 0{,}9$ – сценарий с низкой технологической нагрузкой, когда под тестирование выделяется большая доля рабочего цикла;

$S_{\text{Ц}} = 0{,}5$ – сбалансированный сценарий, при котором время на основные функции и тестирование распределено поровну;

$S_{\text{Ц}} = 0{,}1$ – сценарий с высокой технологической нагрузкой, когда для тестирования доступна лишь малая доля цикла.

Выбранные значения $S_{\text{Ц}} = 0{,}1;\ 0{,}5;\ 0{,}9$ можно рассматривать как нижнюю, среднюю и верхнюю границы реалистичного диапазона. Они позволяют провести укрупненную оценку и наглядно продемонстрировать, как изменение загруженности централизованного процессора влияет на длительность полного цикла тестирования. Для более точного анализа при наличии конкретной системы рекомендуется использовать индивидуальное значение $S_{\text{Ц}}$, полученное на основе параметров целевой платформы и требований к функциям.

3 Для графика распределённой системы минимальное значение (нижняя граница) $k = 0{,}5$, а максимальное значение $k = 1{,}5$ (верхняя граница). При демонстрации принятия решения с использованием графика в качестве основного возьмём график распределённой системы, соответствующий $k = 1$.

4 Полное время тестирования измеряется в количестве рабочих циклов.

Используя формулы (12) и (13), построим графики продолжительности полного цикла тестирования, в зависимости от количества реализованных в системе функций (рисунок 3).

Рисунок 3 – Графический анализ зависимости длительности тестирования систем

Напомним, что минимизация времени полного тестирования является одним из факторов, который следует учитывать при проектировании безопасной системы. Чем ниже значение, тем более оперативно происходит обнаружение скрытых отказов. Поэтому предпочтение отдаётся архитектуре с более низким (по оси *Y*) значением для конкретного числа функций. Таким образом, если максимальное время тестирования одной функции в распределённой системе сопоставимо со средним временем в централизованной ($k \approx 1$), тогда:

– если тестирование занимает 10 % рабочего цикла $S_Ц$ = 0,1, то при одной функции архитектуры равнозначны, а при наращивании функций распределённая архитектура становится предпочтительнее;

– если тестирование занимает 50 % рабочего цикла, то до 5 функций предпочтение отдается централизованной архитектуре, а при наращивании функций распределённая архитектура становится предпочтительнее;

– если тестирование занимает 90 % рабочего цикла, то до 9 функций предпочтение отдаётся централизованной архитектуре, а при наращивании функций распределённая архитектура становится предпочтительнее.

Исходя из графического анализа можно сделать вывод, что при малом числе функций ($N \leq 5 \sim 9$) и высокой доле времени на тестирование ($S_Ц \geq 0{,}5$) централизованная архитектура обеспечивает более короткий цикл тестирования. Однако при $N > 10$ преимущество переходит к распределённой архитектуре, время тестирования которой не зависит от количества функций. Максимальное количество функций, при котором использование централизованной системы имеет преимущество, составляет $N = 13$ (при $S_Ц = 0{,}1$ и $k = 1{,}5$).

При проектировании ответственных систем основным количественным показателем функциональной безопасности является вероятность опасного отказа в час (PFH – Probability of Dangerous Failure per Hour). Этот параметр характеризует частоту, с которой система может перейти в опасное состояние, и служит основой для определения требуемого уровня полноты безопасности (УПБ) согласно стандартам.

Проведём сравнительный анализ безопасности двух рассматриваемых архитектур с использованием методологии, соответствующей стандартам ГОСТ Р МЭК 61508 и ГОСТ Р МЭК 61511. Анализ выполняется в два этапа.

Базовый анализ – с учётом только аппаратных характеристик и длительности цикла тестирования. Это позволяет выделить и определить влияние архитектурных решений.

Расширенный анализ – с дополнительным учётом влияния сложности программного обеспечения, который, как будет показано, является решающим фактором в современных системах.

Математическая модель расчёта PFH

Для оценки безопасности дублированных систем используется показатель *PFH*. Согласно стандарту ГОСТ Р МЭК 61508, для двухканальной системы с архитектурой 1oo2 (система считается отказавшей только при отказе обоих каналов одной функции) вероятность опасного отказа рассчитывается по формуле

$$PFH_{1oo2} \approx 2\left((1-\beta)\lambda_{du}\right)^2 t_{ce} + \beta\lambda_{du}, \qquad (14)$$

где λ_{du} – интенсивность опасных необнаруженных отказов одного канала 1/ч$^{-1}$; β – фактор общих причин, отражающий долю отказов, одновременно выводящих из строя оба канала (рекомендуемые значения: 0,01–0,1); t_{ce} – среднее время восстановления, ч, т. е. среднее время, в течение которого система находится в неработоспособном состоянии после первого отказа до его обнаружения и восстановления. Для систем с периодическим тестированием принимается

$$t_{ce} \approx \frac{T_{тест}}{2}, \qquad (15)$$

где $T_{тест}$ – длительность полного цикла тестирования.

Следует отметить, что в представленной модели время восстановления работоспособности t_r (время на переключение на резерв, перезапуск и т. д.) явно не выделяется и считается включённым в общее время восстановления t_{ce} В рамках данного сравнительного анализа мы принимаем, что t_r является величиной одного порядка для обеих архитектур и не оказывает решающего влияния на их сравнительное преимущество. Это допущение сделано для упрощения модели и концентрации на анализе длительности цикла тестирования и влияния программного обеспечения. Для систем, где время реакции критично, данный параметр должен быть учтён в уточнённой модели.

Формула (14) учитывает два основных механизма опасного отказа:

– независимые отказы в обоих каналах (первое слагаемое);

– отказы по общей причине (второе слагаемое).

Важно отметить, что даже в рамках базового (аппаратного) анализа распределённая архитектура обладает потенциальным преимуществом, не отражённым в текущей модели, – снижение риска отказов по общей причине. Физическая изолированность узлов, раздельные цепи питания, отсутствие единой операционной системы и межпроцессного взаимодействия значительно уменьшают вероятность того, что один внешний фактор (например, скачок напряжения, электромагнитная помеха или ошибка в драйвере) одновременно выведет из строя несколько каналов. Это позволяет предположить, что в реальных условиях значение фактора β для распределённой системы может быть *существенно ниже*, чем для централизованной, что дополнительно снижает ее *PFH* и повышает безопасность.

Выбор значения фактора β общих причин

Для централизованной системы, характеризующейся наличием единой операционной системы, общих ресурсов (память, шины, драйверы) и высокой сложностью ПО, вероятность отказов по общей причине существенно возрастает. Внешние воздействия (скачки напряжения, ЭМП, ошибки в ОСРВ, конфликты задач) затрагивают всю систему целиком. Согласно методикам стандарта МЭК 61508-6 и эмпирическим данным по отказам встраиваемых систем на базе ОСРВ для таких архитектур обоснованно использовать $\beta_{центр} = 0{,}1$.

Для распределённой системы, где каждая функция реализована на физически изолированном устройстве

с минималистичным ПО без ОС, сведены к минимуму точки отказа. Внешние воздействия, как правило, локализованы и затрагивают только отдельные узлы. Это позволяет снизить фактор общих причин до $\beta_{распр} = 0{,}1$, что соответствует практике применения микроконтроллерных систем в критически важных отраслях (например, в железнодорожной автоматике или АСУ ТП).

Расчёт *PFH* для централизованной архитектуры

В централизованной системе все функции выполняются на едином высокопроизводительном устройстве (микропроцессоре). Среднее время восстановления

$$t_{ce}^{\text{центр}} = \frac{T_{\text{центр}}}{2} = \frac{N}{2S_C}, \qquad (16)$$

где N – количество функций; S_C – доля рабочего цикла выделяемая, на тестирование.

Тогда, (вероятность опасного отказа в час для централизованной системы)

$$PFH_{\text{центр}} = 2\left(\left(1-\beta_{\text{центр}}\right)\lambda_{du}^{\text{центр}}\right)^2 \frac{N}{2S_C} + \beta_{\text{центр}}\lambda_{du}^{\text{центр}} =$$

$$=\left(1-\beta_{\text{центр}}\right)^2\left(\lambda_{du}^{\text{центр}}\right)^2 \frac{N}{S_C} + \beta_{\text{центр}}\lambda_{du}^{\text{центр}}. \qquad (17)$$

Расчёт PFH для распределённой архитектуры

В распределенной системе каждая функция выполняется на отдельном устройстве (микроконтроллере). Среднее время восстановления

$$t_{ce}^{\text{распр}} = \frac{T_{\text{распр}}}{2} = \frac{k}{2S_R}, \qquad (18)$$

где k – коэффициент скорости тестирования одной функции (учитывает разницу в быстродействии микроконтроллера и микропроцессора); S_R – доля рабочего цикла, выделяемая на тестирование в распределённой системе.

В распределённой системе каждый из N каналов (устройств) работает независимо, но система в целом считается отказавшей, если отказали оба канала в рамках одной функции (предполагается дублирование на уровне функций). Таким образом, общая интенсивность опасных необнаруживаемых отказов системы пропорциональна N, поскольку каждый канал функции вносит свой вклад в риск.

Следовательно, *PFH* для распределенной системы

$$PFH_{\text{распр}} = N\left[2\left(\left(1-\beta_{\text{распр}}\right)\lambda_{du}^{\text{распр}}\right)^2 \frac{k}{2S_R} + \beta_{\text{распр}}\lambda_{du}^{\text{распр}}\right] =$$

$$=N\left[\left(1-\beta_{\text{распр}}\right)^2\left(\lambda_{du}^{\text{распр}}\right)^2 \frac{k}{S_R} + \beta_{\text{распр}}\lambda_{du}^{\text{распр}}\right]. \qquad (19)$$

Примечание – В данной модели предполагается, что отказ (по обоим каналам) любой из N функций приводит к опасному отказу всей системы. Это соответствует сценарию, где все функции являются критически важными для безопасности.

Критерий выбора архитектуры по показателю *PFH*

Рассмотрим соотношение показателя безопасности централизованной архитектуры и распределённой архитектуры. Это соотношение может выступать в качестве критерия для выбора архитектуры на основе показателя *PFH*, так как оно наглядно показывает, какая из них безопаснее.

$$\frac{PFH_{\text{центр}}}{PFH_{\text{распр}}} = \frac{\left(1-\beta_{\text{центр}}\right)^2\left(\lambda_{du}^{\text{центр}}\right)^2 \frac{N}{S_C} + \beta_{\text{центр}}\lambda_{du}^{\text{центр}}}{N\left[\left(1-\beta_{\text{распр}}\right)^2\left(\lambda_{du}^{\text{распр}}\right)^2 \frac{k}{S_R} + \beta_{\text{распр}}\lambda_{du}^{\text{распр}}\right]}. \qquad (20)$$

При равных $\lambda_{du}^{\text{центр}} = \lambda_{du}^{\text{распр}} = \lambda_{du}$ критерий упрощается до:

$$\frac{PFH_{\text{центр}}}{PFH_{\text{распр}}} = \frac{\left(1-\beta_{\text{центр}}\right)^2\lambda_{du}\frac{N}{S_C} + \beta_{\text{центр}}}{N\left(1-\beta_{\text{распр}}\right)^2\lambda_{du}\frac{k}{S_R} + N\beta_{\text{распр}}}. \qquad (21)$$

Интерпретация:

Если $\dfrac{\left(1-\beta_{\text{центр}}\right)^2\lambda_{du}\frac{N}{S_C} + \beta_{\text{центр}}}{N\left(1-\beta_{\text{распр}}\right)^2\lambda_{du}\frac{k}{S_R} + N\beta_{\text{распр}}} < 1,$

то $PFH_{\text{центр}} < PFH_{\text{распр}}$ – централизованная архитектура безопаснее.

Если $\dfrac{\left(1-\beta_{\text{центр}}\right)^2\lambda_{du}\frac{N}{S_C} + \beta_{\text{центр}}}{N\left(1-\beta_{\text{распр}}\right)^2\lambda_{du}\frac{k}{S_R} + N\beta_{\text{распр}}} > 1,$

то $PFH_{\text{центр}} > PFH_{\text{распр}}$ – распределённая архитектура безопаснее.

Данный критерий справедлив *только* при условии равенства $\lambda_{du}^{\text{центр}}$ и $\lambda_{du}^{\text{распр}}$ для обеих архитектур, что является упрощением.

Графическое сравнение безопасности архитектур без учета влияния ПО

Из опыта известно, что микроконтроллеры часто надежнее микропроцессоров. Это связано с их более простой конструкцией. Однако для упрощения базового сравнения архитектур примем следующие допущения:

– интенсивности опасных необнаруживаемых отказов равны: $\lambda_{du}^{\text{центр}} = \lambda_{du}^{\text{распр}} = \lambda_{du} = 10^{-5}$ ч$^{-1}$.

– фактор общих причин для централизованной архитектуры примем $\beta_{\text{центр}} = 0{,}1$, а для распределенной соответственно $\beta_{\text{распр}} = 0{,}01$;

– S_C – доля времени на тестирование в централизованной системе варьируется: [0,1; 0,5; 0,9];

– $S_R = 0{,}1$ – фиксированная доля времени на тестирование в распределённой системе;

– k – коэффициент варьируется: [0,5; 1,0; 1,5].

Подставляя значения в формулы (16) и (18), можно построить графики зависимости *PFH* от количества функций N.

На рисунке 4 представлено сравнение *PFH* в зависимости от N (количества функций) и при вариации параметров S_C и k.

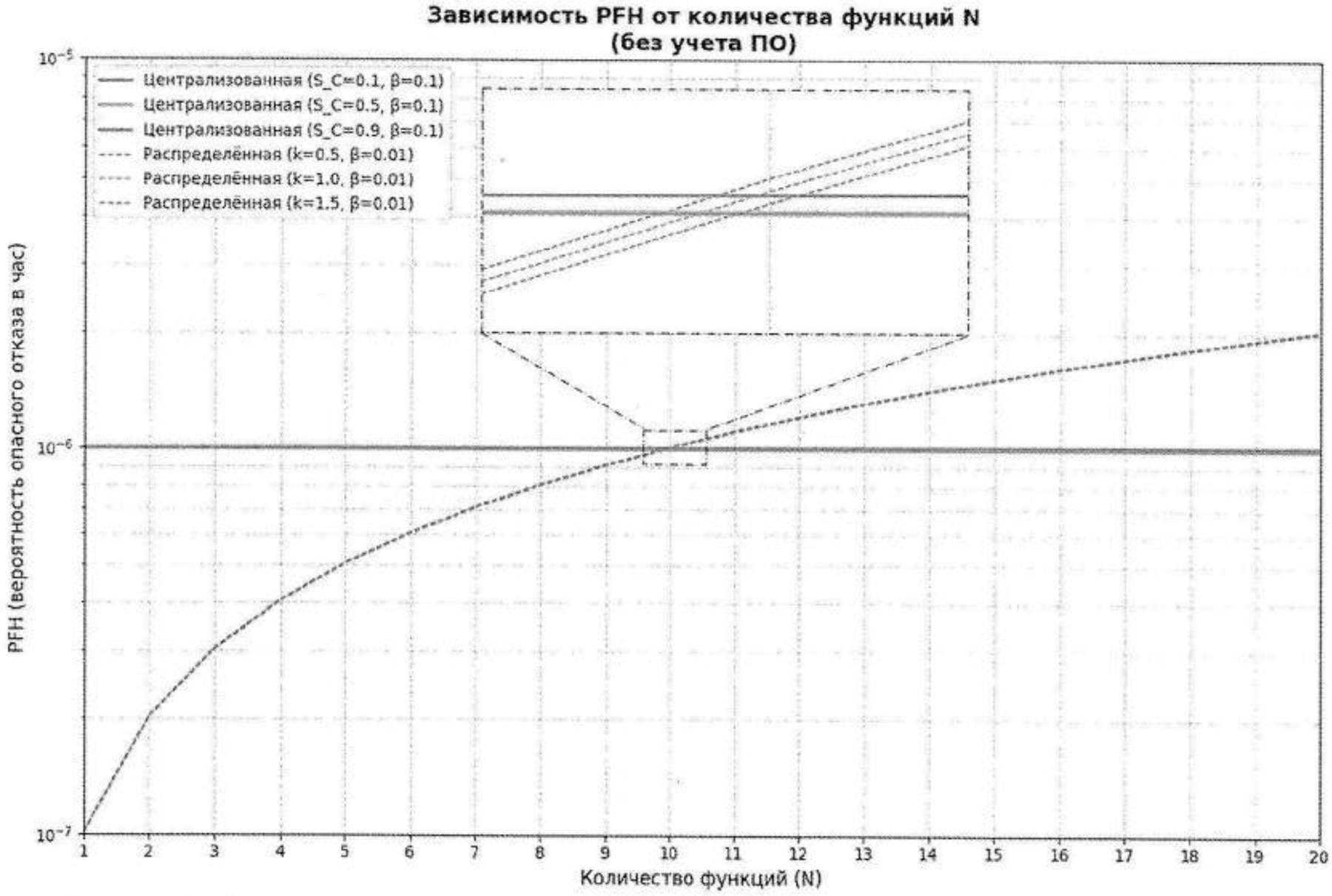

Рисунок 4 – Сравнение безопасности архитектур без учёта влияния программного обеспечения

На рисунке 4 представлено сравнение показателя *PFH* для централизованной и распределённой архитектур при различных значениях доли времени на тестирование S_c и коэффициента скорости тестирования k, без учёта влияния программного обеспечения.

Централизованная система (сплошные красная, желтая, зеленая линии) демонстрирует *практически неизменный уровень PFH* на промежутке от 0 до 20 при увеличении числа функций N. Это связано с тем, что доминирующим слагаемым в формуле (17) является $\beta_{\text{центр}}\lambda_{du}$, которое не зависит от N. Первое слагаемое $(1-\beta)^2\lambda_{du}^2\frac{N}{S_C}$ растёт линейно, но его вклад мал (порядка $10^{-8}\cdot\frac{N}{S_c}$, поэтому график имеет практически горизонтальный характер.

Распределённая система (пунктирные синяя, фиолетовая, коричневая линии) показывает *значительный рост PFH с увеличением N*, поскольку её риск пропорционален количеству функций (формула (19)). При этом наклон линий зависит от коэффициента k: чем выше k, тем больше значение *PFH*.

Все три линии распределённой системы пересекаются с линиями централизованной системы в районе $N \approx 10$. Например, это означает, что *при малых N* (до 10) распределённая архитектура обеспечивает более высокую безопасность (меньший *PFH*). Однако с ростом числа функций ($N > 10$) её преимущество сокращается, и централизованная архитектура, демонстрирующая стабильно низкий *PFH*, не зависящий от N, становится предпочтительнее в рамках данной модели, не учитывающей ПО.

Важное замечание о ограничениях математической модели и корректности выводов

В представленной математической модели было использовано усреднённое значение $\lambda_{du} = 10^{-5}$ ч$^{-1}$ для обеих архитектур. Это упрощение необходимо для более явного выделения влияния архитектурных особенностей на показатель *PFH*. Однако оно не отражает реальной разницы в надёжности компонентов. Микроконтроллеры обычно имеют меньшую интенсивность отказов из-за конструктивной простоты, а микропроцессоры – большую, из-за их сложности.

Более того, и это является основным ограничением данной модели, анализ полностью игнорирует влияние программного обеспечения (ПО). В современных встраиваемых системах, особенно при росте числа функций N, вклад ошибок ПО в общий риск λ_{du} часто превышает вклад аппаратных отказов. Пренебрежение этим фактором может привести к кардинально неверным выводам при выборе архитектуры. Далее будет продемонстрировано, как учет влияния ошибок ПО меняет баланс преимуществ в пользу распределённой архитектуры даже при большом количестве функций.

Учет влияния программного обеспечения на безопасность

В реальных системах интенсивность опасных необнаруживаемых отказов λ_{du} определяется не только аппаратными компонентами, но и программным обеспечением (ПО). Ошибки в ПО (логические, алгоритмические, связанные с управлением памятью или синхронизацией) могут приводить к опасным отказам, особенно если тестирование не покрывает все сценарии поведения системы.

Суммарная интенсивность опасных необнаруживаемых отказов

$$\lambda_{du}^{\text{сумм}} = \lambda_{du}^{\text{апп}} + \lambda_{du}^{\text{ПО}}, \qquad (22)$$

где $\lambda_{du}^{\text{апп}}$ – интенсивность опасных необнаруживаемых отказов аппаратной части; $\lambda_{du}^{\text{ПО}}$ – интенсивность опасных необнаруживаемых отказов, вызванных ошибками программного обеспечения.

Для количественной оценки влияния ПО на безопасность в данной работе используется методология, основанная на модели Джелинского – Моранды [2]. Эта модель связывает интенсивность отказов программного обеспечения с его объёмом и сложностью, предполагая, что ошибки вносятся в процессе разработки и обнаруживаются (или проявляются) в процессе эксплуатации.

На основе данной методологии в работе [3] были получены обобщенные статистические зависимости, позволяющие оценить $\lambda_{du}^{\text{ПО}}$ для различных типов программных комплексов. Эти данные представлены в таблице 1.

Таблица 1 – **Зависимость интенсивности отказов от объёма ПО**

Тип управляющего ПО	Размер процессора	Объём ПО, Мб	$\lambda \cdot 10^{-6}$, $ч^{-1}$
Элементарная управляющая программа	С	0,25–1	0,4
Базовая управляющая программа	С	1,4–4,0	1,5
Расширенная управляющая программа	С	4,0–16	6
Базовая операционная программа	С	16–64	25
Расширенная ОС	С	64–256	100
Управляющая программа	С	4–16	25
ОС с малыми возможностями	М/С	16–64	25
Универсальная ОС	С/Б	256+	100

Из таблицы 1 видно, что с ростом объёма программного обеспечения интенсивность отказов увеличивается экспоненциально. Например, при переходе от элементарной управляющей программы (0,25–1 Мб) к расширенной операционной системе (64–256 Мб) интенсивность отказов возрастает более чем в 250 раз.

В централизованной системе из-за того, что все функции реализуются в едином программном комплексе, объём ПО может достигать десятков или сотен мегабайт, особенно при большом числе функций N. Это соответствует категории «расширенная операционная система».

В распределённой архитектуре каждое устройство выполняет одну или несколько простых функций, и объём ПО на каждом микроконтроллере обычно составляет от 0,25 до 4 Мб, что соответствует уровню «элементарной» или «базовой» управляющей программы. При этом отсутствие ОСРВ позволяет значительно снизить вероятность ошибок, связанных с конкуренцией за ресурсы, блокировками, переполнением стека и другими проблемами, характерными для многозадачных сред.

Особенности ПО централизованной системы:

– использует мощный процессор с операционной системой реального времени (ОСРВ), драйверами, планировщиком задач и механизмами межпроцессного взаимодействия;

– все N функций реализованы в рамках единого программного комплекса;

– объем и сложность ПО растут нелинейно с увеличением количества функций N (из-за необходимости обеспечения взаимодействия, синхронизации, управления памятью). Для анализа возьмём диапазон (64–256 Мб), что соответствует категории «расширенная операционная система» (см. таблицу 1);

– наличие ОСРВ драйверов, планировщика задач, межпроцессного взаимодействия существенно увеличивает вероятность скрытых ошибок;

– увеличение N ведёт к экспоненциальному росту сложности ПО и, как следствие, к росту $\lambda_{du}^{\text{ПО}}$.

Для анализа возьмём $\lambda_{du}^{\text{ПО}} = 100 \cdot 10^{-6}$ $ч^{-1}$ (см. таблицу 1).

Особенности ПО распределённой системы:

– каждый микроконтроллер выполняет одну (или несколько простых) функций;

– программное обеспечение максимально упрощено, чаще без ОСРВ или с использованием сверхлегкого планировщика;

– объем ПО на каждом узле фиксирован и не зависит от N общего числа функций. Для анализа возьмем диапазон (0,25–4 Мб)), что соответствует категории «базовая управляющая программа» (см. таблицу 1);

– ошибки изолированы в пределах одного узла, что снижает общий системный риск и повышает предсказуемость поведения системы. Для анализа возьмём $\lambda_{du}^{\text{ПО}} = 1{,}5 \cdot 10^{-6}$ $ч^{-1}$ (см. таблицу 1).

Модель расчёта *PFH* с учётом вклада программных ошибок

Для расчёта *PFH* с учетом вклада программных ошибок проведём уточнение интенсивности опасных необнаруживаемых отказов для централизованной и для распределённой системы в соответствии с формулой (22).

С учётом данных таблицы 1 полные интенсивности опасных необнаруживаемых отказов для каждой архитектуры определяются следующим образом.

Для централизованной системы (единое сложное ПО)

$$\lambda_{du}^{\text{центр}} = \lambda_{du}^{\text{апп}} + \lambda_{du}^{\text{ПОцентр}} = 1 \cdot 10^{-5} + 100 \cdot 10^{-6} = 1{,}1 \cdot 10^{-4}. \quad (23)$$

Для распределённой системы (множество простых ПО)

$$\lambda_{du}^{\text{распр}} = \lambda_{du}^{\text{апп}} + \lambda_{du}^{\text{ПОраспр}} = 1 \cdot 10^{-5} + 1{,}5 \cdot 10^{-6} = 1{,}15 \cdot 10^{-5}. \quad (24)$$

Из приведённых расчетов следует, что учет ошибок ПО для централизованной системы увеличивает $\lambda_{du}^{\text{центр}}$ в 11 раз, в то время как для распределённой – $\lambda_{du}^{\text{распр}}$ увеличивается лишь на 15 %. Это наглядно демонстрирует отличие архитектур.

Подставим выражения (23) и (24) в формулу *PFH* для архитектуры 1oo2:

$$PFH_{\text{центр}} = \left(1-\beta_{\text{центр}}\right)^2 \left(\lambda_{du}^{\text{центр}}\right)^2 \frac{N}{S_C} + \beta_{\text{центр}}\lambda_{du}^{\text{центр}} =$$

$$= (1-0,1)^2 \left(1,1\cdot 10^{-4}\right)^2 \frac{N}{S_C} + 0,1\cdot 1,1\cdot 10^{-4} =$$

$$= 9,801\cdot 10^{-9}\frac{N}{S_C} + 1,1\cdot 10^{-5}; \qquad (25)$$

$$PFH_{\text{распр}} = N\left[\left(1-\beta_{\text{распр}}\right)^2 \cdot \left(\lambda_{du}^{\text{распр}}\right)^2 \frac{k}{S_R} + \beta_{\text{распр}}\lambda_{du}^{\text{распр}}\right] =$$

$$= N\left[(1-0,01)^2 \left(1,15\cdot 10^{-5}\right)^2 \frac{k}{S_R} + 0,01\cdot 1,15\cdot 10^{-5}\right] =$$

$$= N\left(1,296\cdot 10^{-10}\frac{k}{S_R} + 1,15\cdot 10^{-7}\right). \qquad (26)$$

Построим графики (рисунок 5).

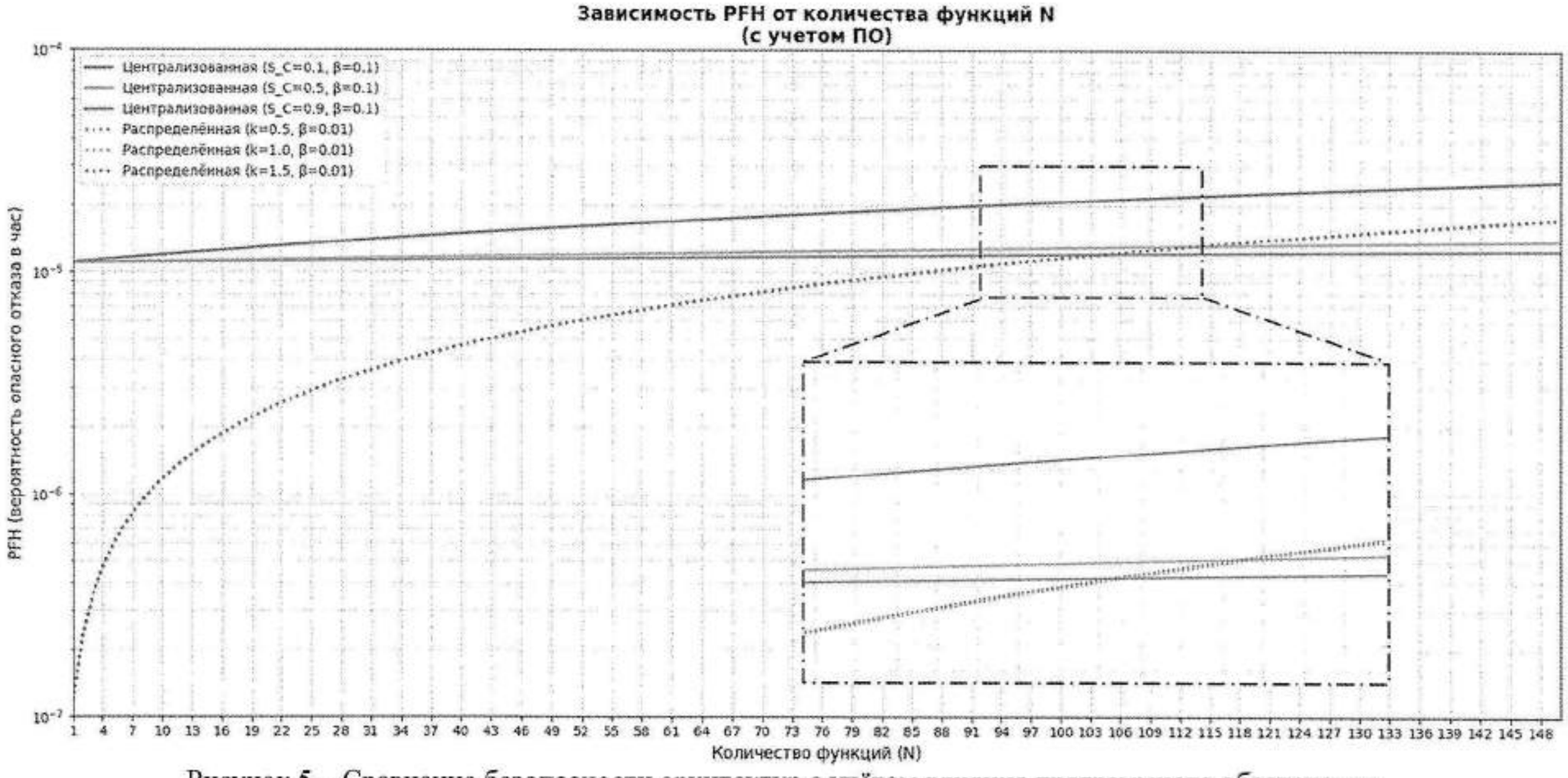

Рисунок 5 – Сравнение безопасности архитектур с учётом влияния программного обеспечения

На рисунке 5 показаны графики *PFH* при учёте влияния ошибок программного обеспечения. Здесь разница между архитектурами становится уже значительной.

Централизованная система (сплошные красная, желтая, зеленая линии) имеет *значительно более высокий базовый уровень PFH* из-за большого вклада ПО ($\lambda_{du}^{\text{центр}} = 1,1\cdot 10^{-4}$). Различия между графиками при $S_c = 0,1$ (красная линия) и при $S_C = 0,9$ (зелёная линия) становятся очевидными.

Красный график демонстрирует более сильный рост *PFH*.

Распределённая система (пунктирные синяя, фиолетовая, коричневая линии) начинается с очень низкого уровня *PFH* (порядка 10^{-7} при $N = 1$) и растёт до тех пор, пока не пересекается с графиками централизованной системы в районе $N > 100$.

Анализ графиков показывает, что пересечения происходят на гораздо больших значениях *N*. Графики распределённой системы пересекают централизованную с $S_c = 0,9$ (зеленая линия) при $N \approx 104$; $S_c = 0,5$ (оранжевая линия) при $N \approx 114$, а с $S_c = 0,1$ (красная линия) **пересечение в выбранном диапазоне до** $N = 150$ **не наблюдается**, и, вероятно, произойдёт только при $N >> 150$.

Заключение. В настоящей работе проведён сравнительный анализ централизованной и распределённой архитектур функционально безопасных систем с точки зрения эффективности фонового тестирования и количественной оценки безопасности (показатель *PFH*). Установлено, что хотя при малом числе функций ($N \leq 10$) централизованная архитектура может обеспечивать более короткий цикл тестирования, её безопасность оказывается существенно ниже при учёте влияния программного обеспечения и фактора отказов по общим причинам.

Важным результатом исследования является демонстрация доминирующей роли сложности программного обеспечения в формировании риска опасных отказов. В системах с высокими требованиями к функциональной безопасности, таких как железнодорожная автоматика, промышленные АСУ ТП, предпочтение следует отдавать распределённой архитектуре на базе специализированных микроконтроллеров (например, Миландр), поскольку она обеспечивает изоляцию ошибок, минимизацию объёма ПО на узел и снижение вероятности отказов по общей причине.

Однако несмотря на продемонстрированные преимущества в функциональной безопасности (*PFH*), распределённая архитектура имеет и существенный недостаток – более низкую общую надежность (Availability). Это напрямую следует из теории надежности: система, состоящая из N последовательно соединенных с точки зрения общей работоспособности элементов, имеет общую интенсивность отказов, равную сумме интенсивностей отказов ее компонентов.

Поскольку распределенная система содержит больше аппаратных компонентов (микроконтроллеров, блоков питания, разъемов), вероятность того, что хотя бы один из них откажет (приведя к простою всей системы или ее части), выше, чем в централизованной системе.

Таким образом, выбор архитектуры представляет собой классический компромисс: распределённая архитектура предлагает высокую функциональную безопасность (меньшая вероятность опасного отказа), в то время как централизованная может обеспечить более высокую общую надежность (меньшая вероятность полного простоя). В системах, где требования к безопасности менее критичны и доминируют экономические ограничения (стоимость аппаратуры, сложность интеграции и сопровождения), централизованная архитектура на базе высокопроизводительных процессоров может быть целесообразной.

Следовательно, выбор архитектуры на ранних этапах проектирования должен основываться на комплексной оценке, учитывающей не только производительность и время тестирования, но и приоритеты проекта в отношении функциональной безопасности, а также общей надежности и экономической эффективности.

Список литературы

1 **Розенберг Е. Н.** О влиянии полноты диагностирования на показатели функциональной безопасности / Е. Н. Розенберг, Н. Г. Пенькова // Интеллектуальные транспортные системы : материалы II Междунар. науч.-практ. конф., Москва, 25 мая 2023 г. – М. : РУТ (МИИТ), 2023. – С. 631–636.

2 **Al turk L.** Jelinski-Moranda Software Reliablity Growth Model : A Brief Literature and Modification / L. Al turk, E. Alsolami // International Journal of Software Engineering & Applications. 2016. – Vol. 7. – P. 33–44. – DOI: 10.5121/ijsea.2016.7204.

3 Предложение по определению эксплуатационной надежности программного обеспечения сложных технических систем / А. С. Белов, М. М. Добрышин, А. Н. Горшков, Д. Е. Шугуров // Известия Тульского государственного университета. Технические науки. – 2022. – № 9. – С. 143–148. – DOI: 10.24412/2071-6168-2022-9-143-148.

Получено 05.09.2025

E. N. Rozenberg, N. G. Penkova, S. V. Ipatov, K. A. Bochkov. Analysis of methods for constructing functionally safe structures using elements of varying complexity.

This paper presents a comparative analysis of centralized and distributed architectures for functionally safe systems, considering both the efficiency of background testing and quantitative safety assessment based on the Probability of Dangerous Failure per Hour (PFH). The study examines the influence of the number of implemented functions, software complexity, and the common cause failure (CCF) factor on diagnostic cycle duration and functional safety integrity. It is shown that while centralized architectures can offer shorter test cycles for a small number of functions, distributed architectures provide significant safety advantages when software-related risks are considered. These advantages stem from physical and logical isolation of components, reduced software size and complexity per node, and a lower likelihood of common cause failures. Based on the analysis, practical criteria for selecting an appropriate architecture at the early design stages are proposed.