Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ ИМЕНИ Г. ДАУКЕЕВА»
Кафедра _____

«ДОПУЩЕН К ЗАЩИТЕ»
Зав.кафедрой _____
(ученая степень, звание, Ф.И.О.)
_____«___» _____201___г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка методики анализа видов, последствий и критичности отказов программных средств космического назначения
Специальность _____
Выполнил _____ Группа_____
(Ф.И.О.)
Научный руководитель _____
(ученая степень, звание, Ф.И.О.)
Консультанты:
по экономической части: _____
(ученая степень, звание, Ф.И.О.)
_____ «___» _____201___г.
(подпись)
по безопасности жизнедеятельности:_____
(ученая степень, звание, Ф.И.О.)
_____ «___» _____201___г.
(подпись)
Нормоконтролер:_____
(ученая степень, звание, Ф.И.О.)
_____ «____» _____201___г.
(подпись)
Рецензент: _____
(ученая степень, звание, Ф.И.О.)
_____ «____» _____201___г.
(подпись)

Алматы 2020

Аннотация

Темой данной дипломной работы является «Разработка методики анализа видов, последствий и критичности отказов программных средств космического назначения».

Объектом исследования в данной работе являлось программное средство космического назначения.

Актуальность данной темы обусловлена повышением частоты полетов, совершаемых космическими аппаратами, соответственно повышением частоты вероятности возникновения отказов внутри программных средств космического назначения.

Основной целью данной работы является разработка методики выполнения метода анализа видов, последствий и критичности отказов применительно к программным средствам космического назначения.

Содержание работы предусматривает 5 глав, 16 параграфов, и список использованных источников.

В дипломной работе использовано 16 рисунков, 12 таблиц, 29 использованных источников.

Аннотация

Диссертациялық жұмыстың тақырыбы «Ғарыштық бағдарламалық құралдардың сәтсіздіктерінің түрлерін, салдарын сыни тұрғыдан талдау әдістемесін жасау» болып табылады.

Бұл жұмыста зерттеу объектісі ғарыштық мақсаттағы бағдарламалық құрал болды.

Бұл тақырыптың өзектілігі ғарыш аппараттарының ұшу жиілігінің артуымен, сәйкесінше ғарыштық бағдарламалық құралдардың істен шығу ықтималдығы жиілігінің артуымен байланысты.

Бұл жұмыстың негізгі мақсаты - ғарыштық бағдарламалық құралдарда болатын сәтсіздіктердің түрлерін, салдары мен қиын жағдайларын талдау әдістемесін жасау.

Жұмыстың мазмұны 5 тараудан, 16 абзацтан және пайдаланылған дереккөздер тізімінен тұрады.

Диссертацияда 16 сурет, 12 кесте, 29 дереккөз қолданылған.

Annotation

The theme of this graduate work is “Development of a methodology of failure mode, effects, and criticality analysis of space software”.

The object of the study in this work was a space software.

The relevance of this topic is due to an increase in the frequency of flights made by spacecraft, respectively, an increase in the frequency of the probability of failures within space software.

The main goal of this work is to develop a methodology for failure mode, effects, and criticality analysis as applied for space software.

The content of the work includes 5 chapters, 16 paragraphs, and a list of sources used.

In the thesis used 16 figures, 12 tables, 29 sources used.

Содержание

Введение

Для оценки надёжности программных средств космического назначения (ПСКН) недостаточно выбрать или разработать определенный метод. Для практического применения метода оценки надежности необходимо разработать конкретную методику. Методологически понятия метод, методика и процедура связаны следующими соотношениями.

Обычно «метод» определяется как систематизированная совокупность приемов, операций, шагов, которые нацелены на решение определённой задачи или достижение определённой цели.

Методика - это, как правило, описание конкретных способов, процедур, алгоритма достижения поставленных целей для определенной задачи, условий с указанием правил применения. Т.е. методика является конкретным воплощением метода, которое конкретизирует его для конкретной задачи и условий.

Стандартом Европейской кооперации по космической стандартизации ECSS-Q-HB-80-03-12 Space product assurance. Software dependability and safety (Гарантия космического продукта. Надежность и безопасность программного обеспечения) в качестве одного из эффективных методов оценки и анализа ПСКН рекомендуется метод «Анализ видов, последствий и критичности отказов ПО» (SFMECA - Software Failure Modes Effects and Criticality Analysis).

SFMECA представляет собой формализованную, контролируемую процедуру качественного анализа системы, заключающаяся в выделении на некотором уровне разукрупнения ее структуры возможных отказов разного вида, в прослеживании причинно-следственных связей, обуславливающих их возникновение, и возможных последствий этих отказов, а также в качественной оценке показателей критичности отказов и ранжировании по тяжести их последствий.

Основное назначение SFMECA это выявление (идентификация) возможных программных дефектов и отказов, посредством проведения систематического и документированного анализа наиболее вероятных случаев отказа программных

компонентов, определения причин, воздействия на систему, последствий каждого отказа и их тяжести. SFMECA также классифицирует критичность анализируемого ПО на основе анализа тяжести возможных последствий отказов.

Основными задачами выполнения SFMECA являются:

- оценка влияния и последовательностей событий, вызванных каждым идентифицированным видом отказа компонентов по какой-либо причине на различных уровнях функциональной иерархии;

- определение влияния значения или критичности каждого вида отказа на правильность функционирования или работу системы и влияние на надежность и / или безопасность связанных процессов;

- классификация идентифицированных видов отказа относительно их диагностируемости, тестируемости, изменяемости компонентов ПО;

- оценка степени значимости и вероятности отказов, при условии доступности данных.

SFMECA выполняют на основе функционального подхода с учетом критичности выполняемых функций.

SFMECA является одним из самых распространенных методов, применяется в различных отраслях промышленности, для него разработаны и опубликованы специализированные стандарты.

Для конкретного применения SFMECA необходимо разработать методику его выполнения, которая должна включать следующие задачи:

- определение требований к ПСКН;

- разработка схем, диаграмм и других математических моделей, и описаний для формального представления структуры ПСКН и его функционирования;

- выбор уровней анализа ПСКН и документации;

- идентификация потенциальных видов, причин и влияния отказов;

- идентификация методов обнаружения отказов и их локализации;

- описание значимости (тяжести) отказов и мер по их снижению;

- оценка критичности и вероятности отказов;

- разработка рекомендаций и отчета по результатам SFMECA.

Цель дипломной работы: разработка методики выполнения метода анализа видов, последствий и критичности отказов применительно к программным средствам космического назначения.

1 Анализ методов оценки надежности ПС

1. 1 Особенности и показатели надежности ПС

Международный стандарт ISO\IEC 25010 «Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов» является основным стандартом, который описывает и дает регламент по качеству программного обеспечения. В состав данного стандарта входит 4 основные части и 3 компонента, которые характеризует качества, внутренние и внешние метрики и рекомендации по применению данных метрик. В стандарте определяются характеристики качества ПО в виде набора свойств программной продукции, на основе которых происходит описание и оценивание ее качества. Метрика качества ПО определяется стандартом как числовое значение масштаба и метод, используемые для того, чтобы рассчитывать значения признака в конкретном программном изделии [2].

В первой части данного стандарта приводится структура, состоящая из шести пунктов, каждая из которых делится на несколько подпунктов. Общее число подпунктов составляет 21, они используются для всестороннего описания качества ПО (Рисунок 1). Одним из основных свойств можно выделить надежность. Надежность – качество, характеризующее способность готового продукта, либо ПО противостоять различным сбоям и ошибкам в ходе тестирования или эксплуатации. Стоит упомянуть, что виды и последствия отказов программного средства напрямую связанны с дефектами, которые могут быть обнаружены внутри ПО и условиями его функционирования. Одним из самых главных преимуществ ПО является отсутствие износа и старения, причинами снижения надежности в свою очередь являются недочеты при разработке требований к проекту, самого проекта и при его реализации.

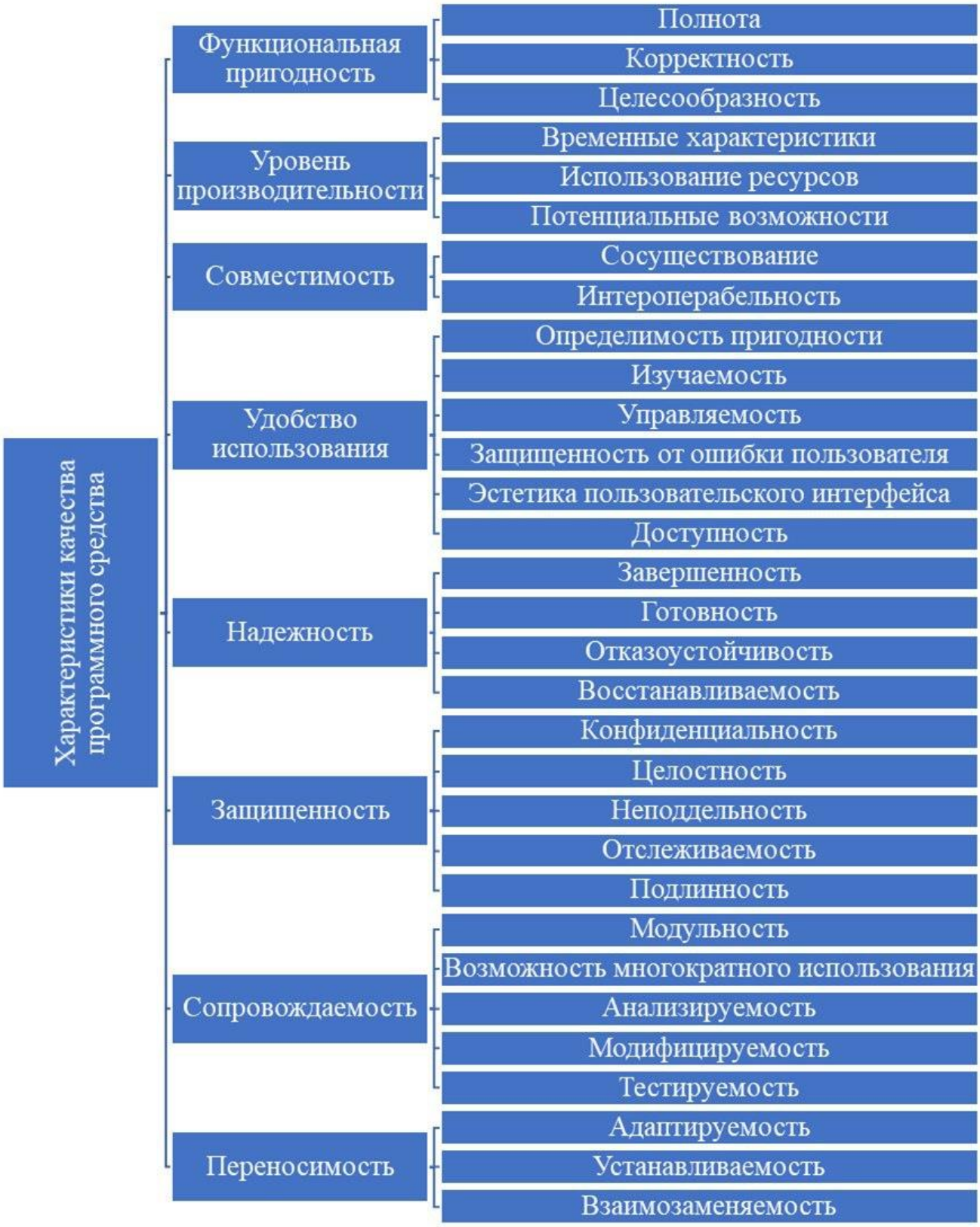

Рисунок 1 – Характеристики качества ПС
Примечание – составлено автором на основании источника [2]

Ниже приведены 5 субхарактеристик характера отказов:

— завершенность оценивает, насколько часто появляется отказы при возникновении ошибок в ПО, измеряется в наработке на отказ при отсутствии средств рестарта;

— устойчивость к ошибкам характеризует возможность ПО оставаться в рабочем режиме при возникновении ошибок, традиционно измеряется в наработке на отказ при наличии средств автоматического рестарта;

— восстанавливаемость означает способность ПО возвращать стандартный уровень качества функционирования либо данных, которые были повреждены при отказе, измеряется длительность восстановления;

— доступность характеризует время нормального функционирования ПО, измеряется вероятностью пребывания в рабочем состоянии;

— соответствие – насколько ПО соответствует стандартам, которые характеризуют надежность.

Согласно стандарту ISO\IEC 25010 необходимо при анализе принимать во внимания четыре субхарактеристики и, максимум шестнадцать числовых атрибутов надежности, включая степень покрытия тестами структуры программ [2]. Наиболее полная характеристика надежности программных средств достигается благодаря следующим неотъемлемым атрибутам: способности функционировать безотказно и способности восстанавливать работоспособность после возникновения сбоя или отказа. На устойчивость же влияют следующие атрибуты: уровень разрешенных дефектов и ошибок, что можно обозначить ка завершенность и способность реагирования программного средства на данные ошибки и дефекты таким образом, чтобы это не влияло на показатели надежности. Качество последних определяется тем, насколько эффективно проходит контроль данных и от эффективности функционирования средств обнаружения аномалий. В реальной жизни, могут быть такие случаи, когда по неизвестным причинам первоначальные данные могут попадают в ту область, где не произошла проверка при разработке и тестировании на требования технического задания, которые могут вызывать сбои в системе и ее отказы. При этом несмотря на подобного рода

недочеты программа может работать идеально. Это говорит о том, что надежность можно назвать динамическим понятием, изменяющимся во времени, однако оно также и статистическое за счет статистической корректности программ.

1.2 Общие требования к оценке надежности ПС

На сегодняшний день требования, выдвигаемые к надежности и безопасности обработки данных при помощи вычислительных систем, вывели задачу оценки качества ПС на новый уровень актуальности.

Известные способы определения качества ПС не удовлетворяют огромной скорости развития техники и не дают гарантий объективного оценивания качества надежности. Использование большинства методов нацелено на оценивание надежности и безопасности уже готового программного обеспечения, что не позволяет изменять разработку на промежуточных этапах для того, чтобы привести к сокращению итоговых затрат. Помимо этого, несмотря на то что невозможно рассматривать показатели качества и комплексное тестирование отдельно друг от друга, эти два этапа не связывают между собой.

Исходя из ранее приведенных недостатков можно предложить методику, которая удовлетворяет следующим требованиям:

— результативность, что означает, что методика помимо объективной и комплексной оценки, должна давать возможность доработать и выдвинуть рекомендации по улучшению ПС;

— практичность, что характеризуется свободной возможностью выбора, замены и расширения показателей качества, а также легкость в их расчёте;

— универсальность – применимость абсолютно на всех стадиях жизненного цикла продукта, как для разработчика, так и для заказчика;

— соответствие всем заранее обговоренным документам по оцениванию качества компьютерных систем [3].

Показатель надежности ПО предполагает точность, корректность и своевременность в выполнении целевых функций продукта. Показатель безопасности обозначает то, насколько хорошо ПС способно защититься от внутренних и внешних факторов, которые могут препятствовать выполнению ее целевых функций. Отсюда следует что показатель надежности определяет качество целевой подсистемы, когда показатель безопасности отвечает за качество подсистемы безопасности программного средства.

Многоуровневая структура является наиболее подходящим способом представления качества целевой подсистемы и подсистемы безопасности. Наиболее часто как иерархия свойств, так и иерархия показателей надежности и безопасности иллюстрируется в виде пятиуровневой структуры (Рисунок 2). Каждый нижний уровень включается в показатель верхнего уровня.

Рисунок 2 - Иерархия свойств и показателей надежности и безопасности программного средства

Примечание – составлено автором на основании источника *[4]*

Общая полезность каждой из подсистем ПС определяется с помощью интегрального показателя. Потребительские свойства, т. е. требования конечных пользователей выражаются эксплуатационными характеристиками. Достижение потребительских свойств происходит благодаря программно-ориентированными свойствами, которые выражаются через программные характеристики. На последнем уровне расположены показатели, которые отвечают за элементарные программные свойства и оцениваются в самом процессе комплексного тестирования. Каждое элементарное свойство имеет метрические характеристики, которые рассчитываются благодаря метрикам, т. е. системам определения величины метрического показателя.

Для того, чтобы оценить надёжность и безопасность программного средства необходимо пройти базовые этапы (Рисунок 3):

1. Первый этап называется этапом анализа, под ним подразумевается выбор характеристик программного средства, их детальное описание, также контроль иерархии свойств. На этом этапе определяются приоритеты в показателях качества на каждом уровне иерархии свойств, также базовые значения для первых характеристик программы. Разрабатываются методики определения величин метрик и их характеристик и составляются плана для комплексного тестирования программы.

2. Второй этап - этап комплексных проверок. На этом этапе начинается тестирование исходных кодов программ, эксплуатационных и проектных документов с помощью статистической методики. После чего необходимо проверить в динамике и оперативно обработать результаты тестирования. Далее производиться сбор всех результатов тестов и их совместная обработка, где определяются все метрические характеристики.

3. Третий этап называется этапом аттестации. Этот этап связан с оценкой показателей, определяются как абсолютные, так и относительные оценки,

рассчитываются величины интегральных показателей. И конечным результатом является аттестация программного средства.

Рисунок 3 – Базовые этапы определения надёжности и безопасности программного средства
Примечание – составлено автором на основании источника [4]

В начале первого этапа эксперты, руководствуясь структурой технического задания, а также существующими стандартами и положениями отбирают и детализируют свойства и показатели для целевой подсистемы и подсистемы безопасности. Далее свойства, которые были выбраны на прошлом этапе проверяются на непротиворечивость. Каждом уровне иерархии также необходимо определить их приоритетные значения, но не включать первый и последний в рассмотрение. Затем, с помощью экспертной оценки определяется методы, которые будут использоваться для определения величин метрических характеристик. Все метрики строятся таким образом, чтобы полученные значения характеристик находились в одном численном интервале, например от нуля до одного. Далее, необходимо определить базовые значения показателей, которые оцениваются надежность и безопасность программ. Они должны быть реально достижимы и характеризовать необходимый уровень качества программного средства. В конце этапа анализа по всех полученным результатам создается план комплексного тестирования и подготавливаются сами тесты.

Во время проведения комплексный проверки рассчитываются величины всех метрических характеристик, в том числе их полнота, согласованность, устойчивость, работоспособность и т. д.

Во время проведения этапа аттестации оценки на каждом вышестоящем уровне рассчитываются на базе нижестоящих. Помимо этого, вычисляются значения абсолютных показателей (Pij) и относительных показателей (Kij), которые являются функцией показателя Pij и базового значения Bij. Характеристика каждого показателя качества программного средства, находящихся между 2-м и 4-м уровней определяется количественным значением и весовым коэффициентом. Величина суммы весовых коэффициент является постоянной и принимается равной единице (формула (1)).

$$\sum_{j=1}^{N} V_{ij} = Const = 1 \tag{1}$$

где V_{ij} – весовой коэффициент;

n - число показателей уровня (l), относящихся к i-му показателю вышестоящего уровня (l-1).

Как говорилось ранее, должны быть выбраны такие метрики, которые обеспечивают значение метрической характеристики от нуля до единицы. Если для того, чтобы определить метрическую характеристику требуется больше одной метрик или данная она будет определяется более комплексно, т.е. при помощи нескольких видов оценки, то в таком случае итоговое значение метрической характеристики будет равно среднему значению всех m_r (формула (2)).

$$m_{kq} = \frac{\sum_{r=1}^{t} m_r}{t} \tag{2}$$

где m_{kq} – метрическая характеристика;

t - число значений метрического показателя;

k - порядковый номер детальной программной характеристики, которой соответствует данный метрический показатель;

q - порядковый номер метрического показателя.

Оценка k-й элементарной программной характеристики j-й первичной определяется по формуле (3):

$$P_{jk} = \frac{\sum_{i=1}^{Q} m_{kq}}{Q} \tag{3}$$

где Q - число оцениваемых элементарных свойств для k-й детальной характеристики.

С помощью следующей формулы (*4*) можно определить абсолютные показатели первичных программных характеристик (P_{ij}) i-й эксплуатационной:

$$P_{ij} = \sum_{k=1}^{N} \left(P_{jk} \times V_{jk}\right) \tag{4}$$

где N - число детальных характеристик Pjk, относящихся к j-й первичной;
V_{jk} - весовые коэффициенты детальных характеристик Pjk.

Относительный показатель (K_{ij}) j-й первичной и i-й эксплуатационной программной характеристики можно вычислить по формуле (5):

$$K_{ij} = \frac{P_{ij}}{B_{ij}} \tag{5}$$

где Pij - абсолютный показатель j-й первичной программной характеристики i-й эксплуатационной;
Bij - базовый показатель j-й первичной программной характеристики i-й эксплуатационной.

Эксплуатационная характеристика качества (K_i) вычисляется по формуле (*6*):

$$K_i = \sum_{j=1}^{N} \left(K_{ij} \times V_{ij}\right) \quad (6)$$

где N - число первичных программных характеристик, относящихся к i-й эксплуатационной;

K_{ij} - относительный показатель j-й первичной программной характеристики i-й эксплуатационной;

V_{ij} - весовые коэффициенты относительных показателей K_{ij}.

Интегральный показатель надежности или безопасности программного средства можно определить по формуле (7), которая использует полученные величины эксплуатационных характеристик:

$$K = \sum_{i=1}^{L} (K_i \times V_i) \quad (7)$$

.

где L - общее число эксплуатационных характеристик;

K_i - эксплуатационные характеристики;

V_i - весовые коэффициенты эксплуатационных характеристик.

Более объективную и детализированную оценку надежности и безопасности ПС можно получить в процессе аттестации за счет использования не только интегральные, но и эксплуатационных характеристик.

1.3 Современные методы оценки надежности ПС

Разделяют два вида моделей надежности ПС: аналитические и эмпирические. Аналитические модели позволяют определить количественные показатели надежности, используя данные полученные в ходе тестирования программы. В основе эмпирических же моделей лежит структурный анализ особенностей программ [5].

Аналитические модели делятся на динамические и статические. В динамических моделях главным фактором, на который обращают внимание является время. В том случае если необходима цельная картина возникновения

отказов во времени можно воспользоваться моделями с непрерывным временем, которые характеризуются фиксацией интервалов отказов. Модели с дискретным временем характеризуются тем, что фиксируется только число отказов, время в свою очередь остается произвольным.

В статистических моделях причины возникновения отказов обуславливают зависимостью числа ошибок от тестовых прогонов, или от вида вводимых данных. Время в данных моделях не играет большой роли.

К аналитические динамическим моделям относятся модель Муса. Методика данной модели заключается в фиксации времени функционирования программы до очередного отказа. Изначально принимается что не каждая ошибка вызывает отказ ПО, поэтому допускается выявление более одной ошибки во время тестирования до очередного отказа.

Считается, что в течение полного жизненного цикла ПО есть вероятность возникновения отказов и при этом будут обозначены все ошибки, которые были выявлены в ПО перед началом теста. Общее число отказов напрямую зависит от первоначального числа ошибок (формула (8)) [6]. После тестирования есть возможность определить коэффициент B, путем деления количества ошибок на количества отказов, выявленных при тестировании ПО.

$$N_0 = B * M_0 \tag{8}$$

где N_0 - первоначальным числом ошибок;
M_0 - общее число отказов;
B — коэффициент уменьшения числа ошибок

Основным показателем надежности, рассчитанным по модели, Муса, является средняя наработка на отказ. Этот показатель определяется как математическое ожидание временного интервала между последовательными отказами.

К аналитические статистические модели относится модель Миллса. Основная концепция данной модели заключается в искусственном внедрении

определенного количества заранее известных ошибок в программу. Ошибки внедряются случайно и заранее фиксируются. Тестировщик не знает сколько было внедрено ошибок и каковы их характеристики до времени, когда производится оценка показателей надежности по модели Миллса. Предполагается, что вероятность проявления искусственно внедренных ошибок равна вероятности проявления естественных ошибок в течение тестирования.

Во время тестирования происходит процесс сбора статистики об ошибках. По стандарту все ошибки могут быть разделены на собственные и искусственные в момент оценки надежности. Оценить количество первоначально выявленных ошибок в программе N позволяет формула Миллса (9) [6] [7].

$$N = \frac{S * n}{V} \tag{9}$$

где S — количество искусственно внесенных ошибок;
n — число найденных собственных ошибок;
V — число обнаруженных к моменту оценки искусственных ошибок.

Количество и сложность межмодульных интерфейсов, а также количество программных модулей характеризует сложность ПО. Программным модулем называется элементарная программная единица, которая выполняет определенную функцию внутри программы, которая в свою очередь тесно связана с другими модулями ПО. Существует несколько видов модели сложности, которые в той или иной степени дают оценку сложности структуры программы, которая прямо пропорциональна ее надежности.

1.4 Задача выбора методов оценки надежности ПС

Исходя из опыта применения моделей, можно сказать, что наиболее выгодными в использовании являются модели оценочного типа, основа которых составляют пуассоновские процессы. К этим моделям относятся модели Мусы, S-образные, Гоэла-Окомото и др. Надежность этих моделей стремится к 1. Однако,

они имеют и недостатки, например, форма кривой интенсивности выявленных отказов, так как она спускается прямо вниз при t>0 возле t=0, что свидетельствует о том, что при проведении теста было сделано недостаточно экспериментов или недостаточное число ошибок когда интенсивность отказов была около 0. В связи с этим, тратиться много времени на поиск ненайденных ошибок.

В следующей Таблица 1 отображаются количество отказов и функции интенсивности отказов для вышеперечисленных и общих моделей. В них величины α и β имеет следующие соотношение (формула (10)) [7]:

$$N = \alpha, \beta = \alpha, b = \beta, \beta^1 = \beta, \alpha_0 = \alpha\beta \tag{10}$$

Таблица 1 - Характеристика моделей надежности Пуассоновского типа *[7]*

№	Название	Функции интенсивности отказов λ (t)	Функции кумулятивного количества отказов μ (t)
1	*2*	*3*	*4*
1	Модель Гоэла-Окумото	$\lambda\,(t) = Nb\exp(-bt)$	$\mu(t) = N(1 - \exp(b - t))$
2	Модель Мусы	$\lambda\,(t) = \beta_0\beta_1\exp(-\beta_1 t)$	$\mu(t) = \beta_0(1 - \exp(\beta_1 - t))$
3	S – подобная модель	$\lambda\,(t) = \alpha\beta^2 t\exp(-\beta t)$	$\lambda(t) = \alpha\{1 - (1 + \beta t)\exp(-\beta t)\}$
4	Модель Шнайдевинда	$\lambda\,(t) = \alpha_0\exp(-\beta t)$	$\mu(t) = \alpha_0/\beta(1 - \exp(-\beta t))$
5	Общая модель пуассоновского процесса	$\lambda\,(t) = \alpha\beta^{n+1}t^n\exp(-\beta t)$	$\mu(t) = \alpha(n^1 - \frac{\sum n\beta^{n-1}}{(n-1)^1 t^n\exp(-\beta t)}))$

Данные α, β, n, задаются системой уравнения (формула (11)) в методе максимального правдоподобия:

$$\begin{cases} \alpha = \dfrac{m}{1 - \sum_{i=0}^{n} \dfrac{n!\,\beta^{n-i} t_m^{n-i}}{(n-i)!} \exp(-\beta t_m)} \\ \dfrac{n+1}{\beta} m = \sum_{k=1}^{m} t_k + \dfrac{m\beta^n t_m^{n+1} \exp(-\beta t_m)}{1 - \sum_{i=0}^{n} \dfrac{n!\,\beta^{n-i} t_m^{n-1}}{(n-i)!} \exp(-\beta t_m)} \end{cases} \quad (11)$$

В данной системе значение параметра n находится в зависимости от процесса тестирования и значений, которые были рекомендуемы:

— n=0, в том случае, когда проект небольшого размера, и разработчик принимает роль тестера (модель Мусы, Гоэло-Окомото и др.);

— n=1, если это средний проект, тестированием и проектированием которого занимаются несколько разработчиков из совместной группы (S- образная модель);

— n=2 имеется у большого проекта. В этом случае тестирование и проектирование осуществляется параллельно;

— n=3, когда проект имеет очень большие размеры и тестирование, и разработка проходит независимо от друг друга.

Для того, чтобы продемонстрировать вид функций μ(t) при разных значениях n=0,1, 2, 3, были получены экспериментальные данные, на базе которых были вычислены функции отказов и интенсивности. Как было доказано, что наибольшее приближение замечается при максимальном n равному 3, а наименьше при минимальном n=0 (Рисунок 4).

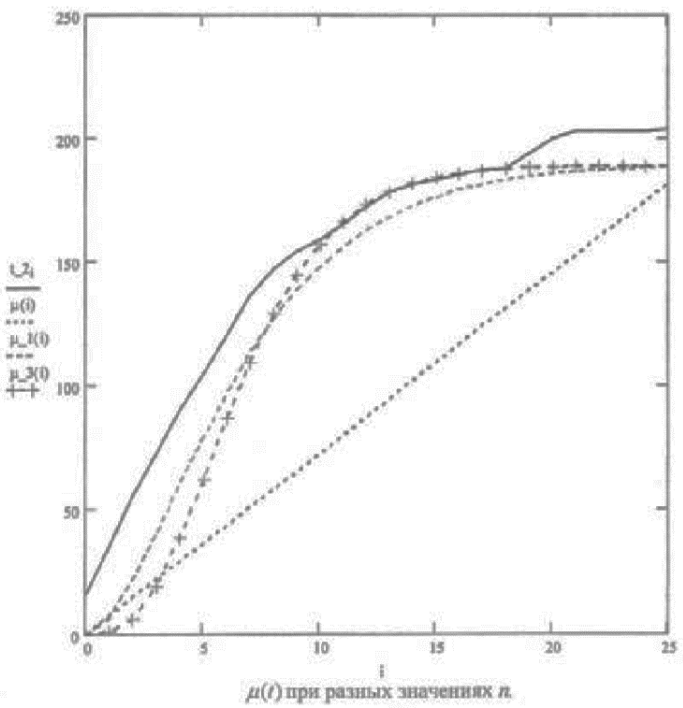

Рисунок 4 - Графики функций μ(t) по моделям Мусы, Окомото и Шнайдевинда *[7]*

Это также можно подтвердить статистическими данным, представленными в Таблица 2, которые демонстрируют разницу между выходными данными и значениями функций при n равной от 0 до 3.

Таблица 2 - Статистические данные функции μ(t) при n=3, 2, 1 и данных t_2 [7]

№	Статистические показатели	Разница функций t_2 - μ_3	Разница функций t_2 - μ_2	Разница функций t_2 - μ_1	Разница функций t_2 - μ
1	*2*	*3*	*4*	*5*	*6*
1	Среднее отклонение	16.13522	16.22889	19.88387	58.93807
2	Медианное отклонение	15.27700	14.11600	16.0000	60.89700
3	Максимум отклонение	33.58100	54.23600	49.10800	88.80200
4	Минимум отклонение	4. 848000	-1.280000	4.175000	15.96200
5	Среднеквадратическое отклонение	8.374089	17.37143	14.07056	23.63765

Итак, можно сделать вывод, что приведенные в таблице данные, характеризующие значения функций μ(t) и λ(t) при n = 3, 2, 1, которые были

вычислены при применении методов оценки надежности Мусы, Мусы-Окомото и Шнайдевинда, соответствуют экспоненциальным кривым и граффики этих функций находятся близко к друг другу из-за близко расположенным значений из заданных моделей [8].

2 Особенности метода анализа видов, последствий и критичности отказов ПС

2.1 Назначение и возможности SFMECA

SFMECA представляет собой формализованную, контролируемую процедуру качественного анализа системы, заключающаяся в выделении на некотором уровне разукрупнения ее структуры возможных отказов разного вида, в прослеживании причинно-следственных связей, обуславливающих их возникновение, и возможных последствий этих отказов, а также в качественной оценке показателей критичности отказов и ранжировании по тяжести их последствий.

Основное назначение SFMECA это выявление (идентификация) возможных программных дефектов и отказов, посредством проведения систематического и документированного анализа наиболее вероятных случаев отказа программных компонентов, определения причин, воздействия на систему, последствий каждого отказа и их тяжести. SFMECA также классифицирует критичность анализируемого ПО на основе анализа тяжести возможных последствий отказов [9].

Основными задачами выполнения SFMECA являются:

— оценка влияния и последовательностей событий, вызванных каждым идентифицированным видом отказа компонентов по какой-либо причине на различных уровнях функциональной иерархии;

— определение влияния значения или критичности каждого вида отказа на правильность функционирования или работу системы и влияние на надежность и / или безопасность связанных процессов;

— классификация идентифицированных видов отказа относительно их диагностируемости, тестируемости, изменяемости компонентов ПО;

— оценка степени значимости и вероятности отказов, при условии доступности данных.

SFMECA выполняют на основе функционального подхода с учетом критичности выполняемых функций.

SFMECA является одним из самых распространенных методов, применяется в различных отраслях промышленности, для него разработаны и опубликованы специализированные стандарты.

Для конкретного применения SFMECA необходимо разработать методику его выполнения, которая должна включать следующие задачи [9]:

— определение требований к ПСКН;

— разработка схем, диаграмм и других математических моделей, и описаний для формального представления структуры ПСКН и его функционирования;

— выбор уровней анализа ПСКН и документации;

— идентификация потенциальных видов, причин и влияния отказов;

— идентификация методов обнаружения отказов и их локализации;

— описание значимости (тяжести) отказов и мер по их снижению;

— оценка критичности и вероятности отказов;

— разработка рекомендаций и отчета по результатам SFMECA.

2.2 Методы анализа надежности, используемые в SFMECA

Существует несколько методов которые применяются в SFMECA. Их можно подразделить на следующие виды: структурные, функциональные, комбинированные.

Структурные методы SFMECA принадлежат к классу индуктивных методов (анализ которых проводится «снизу вверх»). Они используются в тех случаях,

когда структура объектов является достаточно простой, а отказы, происходящие в нем, могут быть четко локализованы, а последствия каждого отказа элементов выбранного начального уровня разукрупнения могут быть прослежены на всех вышестоящих уровнях структуры объекта [10].

Алгоритм проведения SFMECA структурным методом представлен в виде семи основных операций:

— SFMECA начинают с разукрупнения, минимальный уровень которого устанавливается в зависимости от плана.

— производится идентификация всех элементов выбранного уровня разукрупнения основываясь на функциональной блок-схеме объекта;

— составляется список вероятных видов отказов данного элемента для каждого идентифицированного элемента данного уровня на основе имеющихся классификаторов отказов, инженерного анализа, имеющихся априорных данных, опыта и знаний исследователя;

— определяются возможные последствия на данном и последующих структурных уровнях объекта для каждого вида отказа выбранного элемента;

— идентифицируются элементы объекта отказ которых приводит к отказу или понижению качества функционирования самого объекта. После идентификации данных элементов для них проводят оценку категории тяжести последствий отказов (при SFMEA) или рассчитывают показатели критичности (при SFMECA);

— вышеперечисленные операции повторяются последовательно для всех элементов вышестоящих уровней разукрупнения. Последствия отказов элементов нижестоящего уровня рассматривают в виде самостоятельных отказов, так как их нельзя выразить в качестве влияния на функционирование элементов рассматриваемого уровня;

— если степень тяжести последствий отказа или оценка показателей критичности выше, чем установлено в плане, то элементы этих отказов добавляют в список критичных.

Для каждого критичного элемента определяются следующие характеристики, представленные на Рисунок 5. При более тщательном анализе имеется возможность рассмотрения комбинаций из отказов двух или более элементов на каждом уровне разукрупнения.

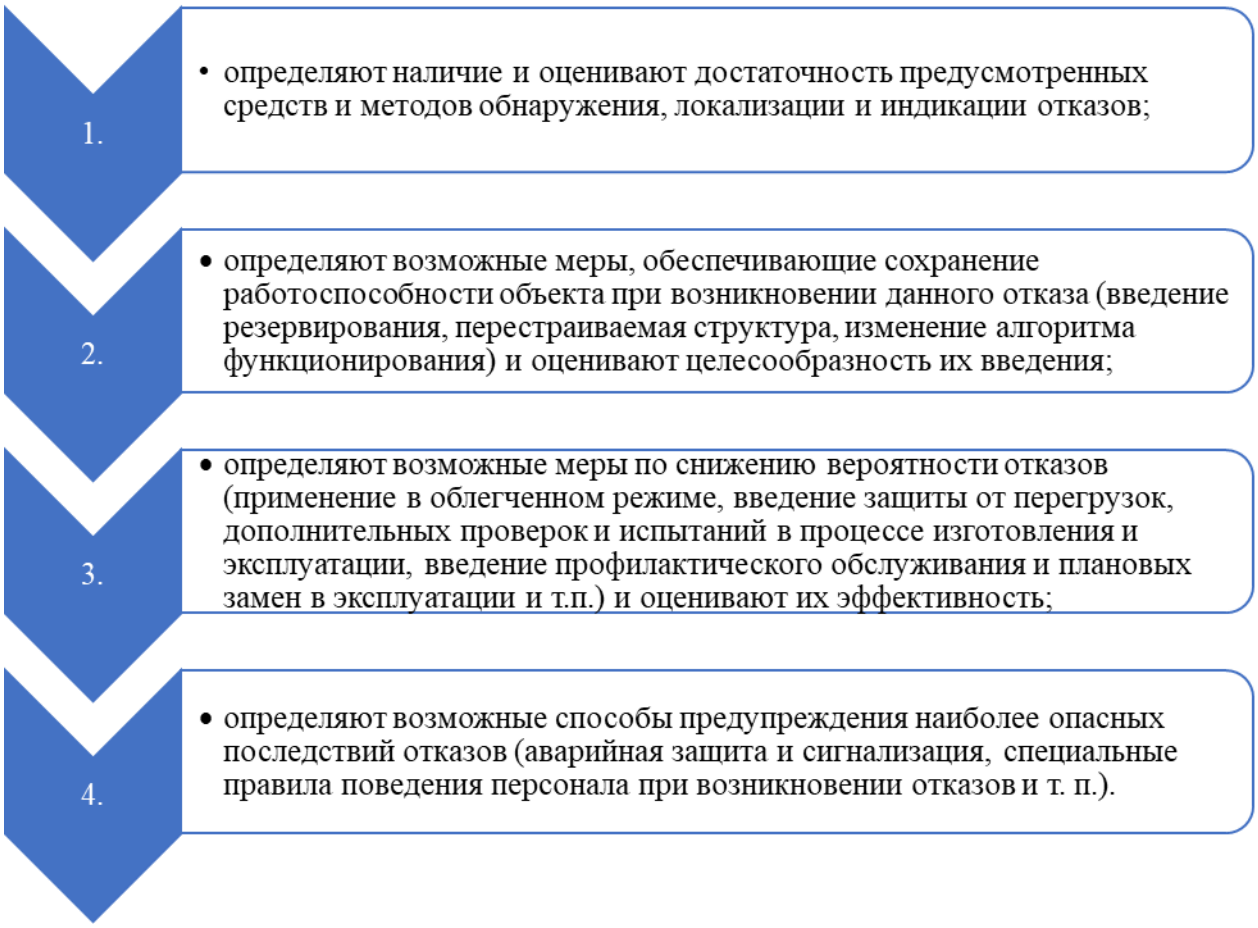

Рисунок 5 - Характеристика критичного элемента
Примечание – составлено автором на основании источника [10]

Функциональные методы SFMECA противоположны структурным методам, так как анализ в этом методе происходит по схеме «сверху вниз» и относится к классу дедуктивных методов. Данный метод применяют для сложно структурированных многофункциональных объектов, в которых крайне сложно локализовать отдельные отказы, так как они могут состоять в сложных зависимостях с другими отказами внутри объекта.

Алгоритм функционального метода SFMECA выглядит следующим образом:

— в начале происходит идентификация всех функций, которые способен выполнять объект анализа;

— на основе изначально известных данных, инженерного анализа, опыта самого инженера-исследователя и других способов определяется список вероятных нарушений данной функции;

— проводят качественную (тяжесть последствий, выявленных при этом нарушении) или количественную (ожидаемый ущерб) оценку для каждой ошибки;

— выявляют отказы функций, тяжесть последствий которых выше пределов в плане;

— строится древо отказов, выполняемое для каждого ранее выявленного критического нарушения (возникновение данного нарушения определяется как «вершинное событие». Данное древо охватывает отказы элементов всех уровней разукрупнения до базисного уровня, который был установлен планом анализа;

— согласно построенному древу, происходит отбор одиночных элементов, которые приводят к критическим сбоям функций исполняемых изделием и группировок элементов, совместные отказы которых становятся причиной указанного нарушения;

— проводят SFMECA, где рассчитывают показатели критичности отказов, также анализируют вероятность появления отказов одиночных элементов и их комбинаций;

— составляется список идентифицированных критичных элементов.

Комбинированный метод проводится в том случае, если объект является сложно структурированным, он сочетает в себе элементы их структурного и функционального методов.

Ниже приведен алгоритм выполнения SFMECA комбинированным методом:

— составляется список задач и описание к ним, которые должен выполнять объект во время его эксплуатации, также дополняются функции, выполнение которых необходимо для реализации поставленных задач. Производится идентификация всех элементов;

— согласно установленной системе кодирования проводится кодирование каждого элемента и функции исследуемого объекта;

— производится описание режимов и условий исполнения каждой задачи, путем составления временных диаграмм и циклограмм нагружения объекта и его элементов;

— в конце составляется структурная схема надежности объекта, а также его функциональная блок-схема [10].

2.3 Порядок анализа надежности в SFMECA

SFMECA проводят в том случае если заинтересованные стороны посчитают это необходимым в момент выработке требований к надежности программного обеспечения.

Ниже приведены характеристики объектов, при которых необходимо проведение SFMECA:

— объекты, отказы которых могут повлечь угрозу безопасности людей, опасное загрязнение окружающей среды, а также огромный экономический или другого рода ущерб;

— объекты, безопасность которых невозможно подтвердить экспериментальным путем;

— объекты, которые должны пройти обязательную или добровольную сертификацию [11].

В том случае если испытатель не располагает достаточными данными для того, чтобы осуществить иной метод анализа надежности объекта или объем и

достоверность этих данных недостаточны на данном этапе жизненного цикла проведение SFMECA считается обязательным.

Проведение SFMECA тесно связано с ПОН (прогнозирование оценок безопасности), в соответствии с чем SFMECA оформляется в виде самостоятельного документа (*Рисунок 6*), который затем включается непосредственно в ПОН.

План проведения SFMECA должен устанавливать:	стадии жизненного цикла объекта и соответствующие им этапы видов работ, на которых проводят анализ (в дальнейшем - этапы анализа или этапы);
	виды и методы анализа на каждом этапе со ссылками на соответствующие нормативные документы и методики. При отсутствии необходимых документов план должен предусматривать разработку соответствующих методик SFMECA (SFMEA) рассматриваемого объекта;
	уровни разукрупнения объекта, начиная с которого (до которого) проводят анализ на каждом этапе;
	сроки проведения анализа на каждом этапе, распределение ответственности за его проведение и реализацию результатов, сроки, формы и правила отчетности по результатам анализа;
	порядок контроля за проведением и реализацией результатов анализа со стороны руководства организации-разработчика и заказчика (потребителя).

Рисунок 6 – Элементы плана проведения SFMECA
Примечание – составлено автором на основании источника [11]

В плане, составленному для анализа SFMECA, должны быть согласованы все объекты по срокам, по их составу и содержанию.

Анализ начинается с самых ранних этапов разработки объекта и повторяется на последующих этапах по мере отработки технологии изготовления объекта и накопления данных для проведения анализа. Во время проведения SFMECA на следующих этапах разработки продукта необходимо предусмотреть

проверку полноты реализации и эффективность проведения мероприятий по доработке продукта, которые были рекомендованы на предыдущих этапах.

В независимости от этапа анализ должен начинаться с SFMECA, в результате чего понимают есть ли необходимость более глубокого количественного анализа и оценки критичности отказов.

Проведения SFMECA (*Рисунок 7*) начинается исходя из уровня разукрупнения объекта. Сам уровень устанавливается в зависимости от результатов, которых требует анализ; степени подготовки необходимой документации; состоянии исходных данных и в зависимости насколько данный объект считается новым. Чем выше уровень отработки продукта и его функционала, и чем лучше испытатель знаком с данным продуктом, тем быстрее и качественнее происходит анализ, и, наоборот, абсолютно новые продукты, содержащие в себе множественные подсистемы ранее не изученные, а также чем новее технологии, с помощью которых был изготовлен данный продукт, тем более углубленный и детализированный анализ может потребоваться.

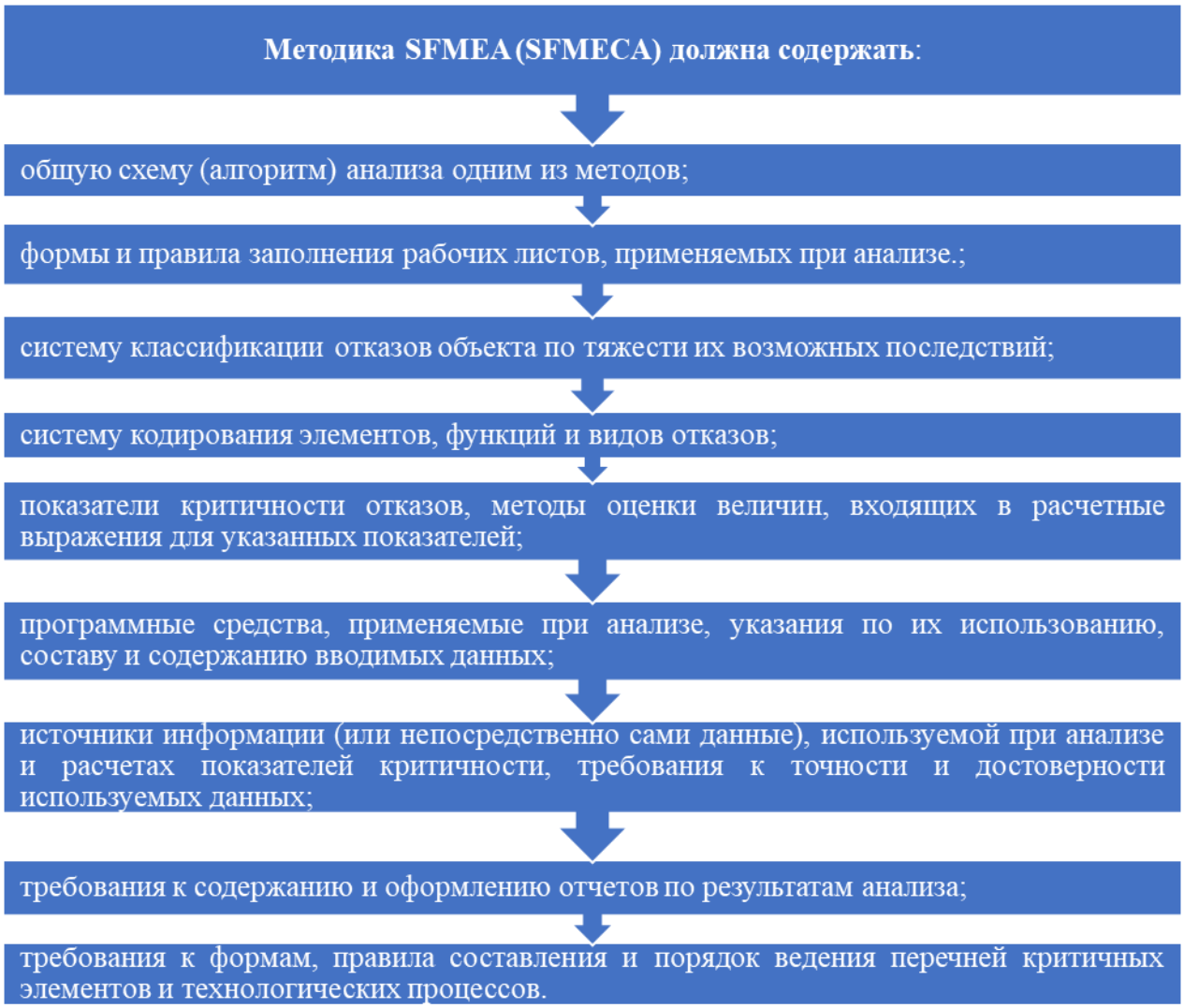

Рисунок 7 – Элементы методики SFMECA

Примечание – составлено автором на основании источника [11]

По результатам, полученным в ходе проведения SFMEA (SFMECA), проведенного на каждом этапе разработки, согласно разработанного плана, должен быть составлен отчет (Рисунок 8).

Рисунок 8 – Элементы отчета в процессе проведения SFMECA
Примечание – составлено автором на основании источника [11]

Также в отчете прилагаются предложения и рекомендации, которые могут быть использованы в последующих этапах разработки. В большинстве случаев предложения и рекомендации касаются изменений в конструкцию изделия или в алгоритмы его функционирования для того, чтобы снизить вероятность появления отказов до определённого уровня или повысить устойчивость объектов в тех случаях, если они возникли. Рекомендации также могут быть связаны с заменой каких-либо материалов в изделии или его комплектующих. Предложения могут быть направлены на изменения задач и процесса экспериментальной отработки изделия, в том числе в случаях проверки на надежность критичных элементов. Также могут потребовать изменить средства контроля, идентификации и диагностирования отказов и проверки технического состояния объекта. Часто предлагают введения специальных мер по предупреждения и устранению отказов,

или последствий критических элементов. Важно также, инструкция по использования объекта, которая включает все правила поведения пользователей и обслуживающего персонала при появлении отказов, направленных на снижения тяжести последствий.

Список критичных элементов (*Рисунок 9*) или технологических процессов составляется сразу же после завершения анализа на начальном этапе разработки продукта, который был заранее предусмотрен планом SFMEA, на последующих этапах продукт подвергается систематическим корректировкам, эффективность доработок по которым подтверждена ранее проведенным анализом, расчетами и данными, полученными в ходе эксперимента.

Рисунок 9 – Характеристика критических элементов
Примечание – составлено автором на основании источника [11]

К критичным элементам можно отнести те элементы, оценка тяжести которых оказалась выше допустимого уровня, также отказ которых приводит к полному отказу объекта. Их качество и надёжность также может быть подвергнута сомнениям, так как отсутствуют достоверные источники информации.

2.4 Установление категории тяжести последствий отказов

Установление категории тяжести и последствий отказов является важным этапом в анализе видов, последствий и критичности отказов. Классификация категорий тяжести отказов может создаваться отдельно по определенному изделию, или осуществляться по разработанному стандарту. Во время разработки классификации тяжести отказов принимают во внимание следующие важные факторы:

— опасность нанесения вреда жизни и здоровья людей, материальному имуществу или окружающей среде;

— влияние последствий отказа на функционирование объекта, качество исполнения им своих функций;

— быстрота появления и распространений последствий отказа.

На рисунке 10 представлено 4 категории тяжести отказов, тяжесть последствий которых усугубляется с возрастанием их порядкового номера:

1. К первой категории можно отнести отказы, не повлекший за собой серьезного ущерба окружающей среде, здоровью людей или материальному имуществу, однако, которые повлияли на качество функционирования объекта, привели к необходимости ремонта и т. п.;

2. Ко второй категории тяжести отказа относятся отказы, которые могут повлечь за собой небольшое ранение либо незначительный материальный ущерб, что повлияет на сроки исполнения задачи и может снизить эффективность объекта.

3. Третья категория тяжести – это отказы, которые наносят большой ущерб самому объекту либо окружающей среде и создает угрозу жизни людей, т.е. может привести к серьезному ранению или повреждению изделия, что, соответственно, приводит к срывам сроков исполнения задачи.

4. Четвертая категория включается в себя отказы, последствия которых с высокой скоростью и вероятностью влекут к значительному материальному ущербу, как для изделия, так и для окружающей среды, также может привести к гибели или тяжелым травмам людей.

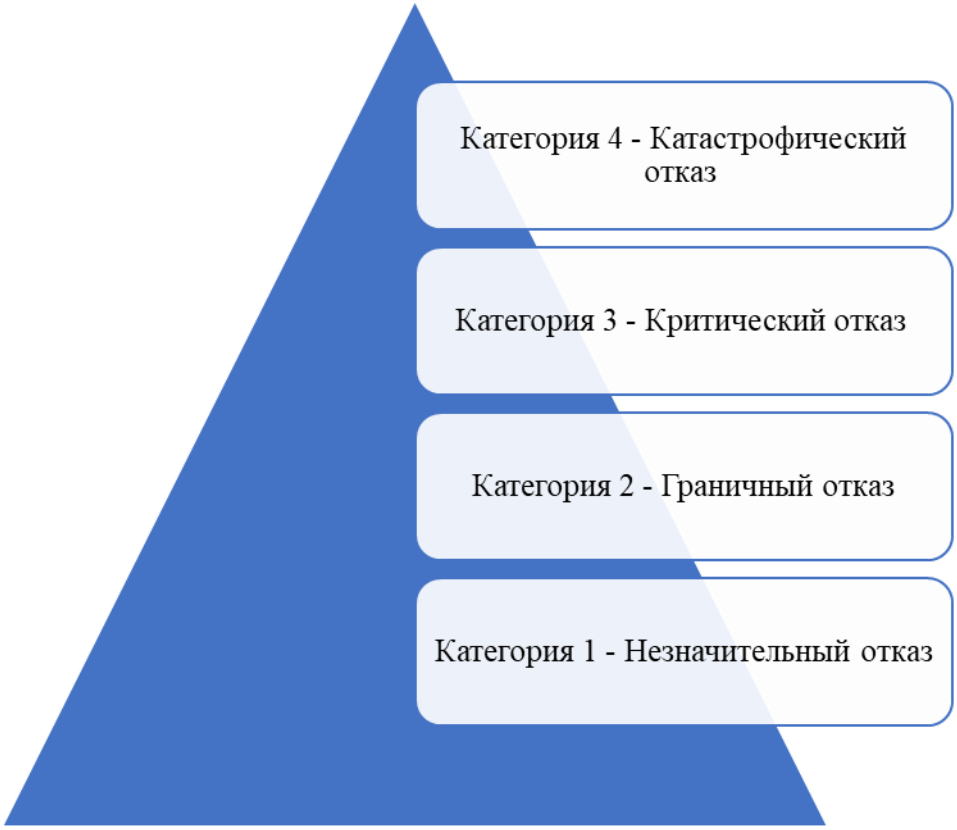

Рисунок 10 - Категории тяжести последствий отказа
Примечание – составлено автором на основании источника []

Отнесения отказов к той или иной категории позволяет наглядно и доступно объяснить последствия отказов разной степени, не вынуждая непосвящённого человека разбираться в полном расчете анализа. Классификация отказов по категориям тяжесть необходимо для проведения глубокого анализа видов, последствий и критичности отказов.

3 Разработка методики выполнения анализа видов, последствий и критичности отказов ПСКН

3.1 Определение требований к ПС для выполнения SFMECA

Требования к программному средству представляют собой документ, который отражают потребности пользователя по отношению к ПС. Они характеризуют общие черты ПС и необходимость его использования. Если

потребности пользователя были поняты неправильно, то они в последствии отразятся в ошибках внешнего описания. С связи с этим, в начале создания ПС необходимо разработать документ, всесторонне отражающий потребности будущих пользователей.

Документ, определяющий требования, включает в себя группу фрагментов, написанных на естественном языке, таблицы и диаграммы. Однако, эта группа должна быть составлена так, что ее могли понимать пользователи, не разбирающихся в программистских терминах. В большинстве случаев в определении требований не имеются такого рода фрагменты, помимо, когда пользователи достаточно подготовлены, тогда описание этих требований служит содержаний задач для коллектива разработчиков.

Причинами неправильного понимания требований самими пользователями, заказчиками или разработчиками являются несогласованность во взглядах на роль программного средства в среде его использования. В связи с этим важно установить контекст, в котором будет использоваться ПС и который должен включать разъяснения по поводу связи между программном средством, людьми и аппаратурой. Наилучшим способом, которым можно представить данный контекст являются графические формы, в которых описывается сущность использованных объектов и связь между ними.

Существует три основных способа, с помощью которых разрабатывается определение требований к программному средству:

— Первый способ – управляемая пользователей разработка. В данном способе требования определяет сам заказчик, который представляет интересы пользователей. Чаще всего, данный способ используется тогда, когда заказчик, т. е. организация потенциальных пользователей, заключает договор с группой разработчиков на разработку программного средства, который включается все выдвигаемые требования. Разработчик в данных требований должен пояснить, насколько ему понятны требования и высказать критику к рассматриваемому

документу. Поэтому, документ может редактироваться несколько раз в процессе того, как будет заключаться договор.

— Второй способ называется контролируемый пользователем. Согласно данному способу, требования к программному средству пишутся разработчиком, но при участии пользователей. В данном способе пользователь должен проинформировать разработчиках о своих требованиях и потребностях и проконтролировать правильность их понимания разработчиком. Требования, которые составил разработчик всегда утверждаются представителем из группы пользователей.

— Третий способ – независимая от пользователя разработка. В данном случае требования к программному средству разрабатываются без участия пользователей. Все ответственность за создаваемый продукт несет разработчик. Данный способ обычно применяют, когда программное средство разрабатывается для широкого спектра пользователей, т. е. на рынке при любых условиях будет способ на новый ПС.

Если брать во внимание обеспечение надежности создаваемого средства, но наилучшим способом будет контролируемая пользователей разработка.

3.2 Формальное представление структуры ПС для применения SFMECA

Формальное представление структуры ПС осуществляется с применение методов структурного анализа по функциональным уровням. Структурный анализ включает в себя различные модели, которые описывают структуру программного обеспечения, последовательность задач, которые оно выполняет, передачу данных между функциональными процессами и отношения между ними [12]. Наиболее распространенными моделями являются:

— модель SADT (Structured Analysis and Design Technique);

— модель IDEF3;

— модель DFD (Data Flow Diagrams).

Первый метод SADT включает в себя совокупность процедур и правил построения функциональных моделей, отображая функциональную структуру объекта, в том числе его действия и связи между ними. Этот метод был основан в 1969 г. Дугласом Россом, который использовал его для создания моделей искусственных систем среднего уровня. Данные модели чаще всего применяются в разработке организационных систем. Однако, метод SADT работает хорошо только в тех случаях, когда бизнес-процессы стандартизированы и хорошо описаны. Это можно наблюдать в зарубежных корпорациях, поэтому в США его приняли в качестве типового. Преимущества применения данного метода в описании бизнес-процессов заключаются в том, что бизнес-процесс максимально полно описан, существуют жесткие требования, которые необходимо соблюдать для обеспечения стандартного вида и соответствие модели стандартам ISO 9000. В большинстве Казахстанских компаний стандартизация бизнес-процессов начала осуществляться сравнительно недавно, в связи с том на данный период необходимо следовать менее жестким моделям [13].

Метод SADT широко применялся в коммерческих, промышленных и военных организациях США, для решения задач, связанных с планированием, автоматизацией производства, управления финансами, снабжением и разработкой программного обеспечения для военной обороны США. В связи с этим он яро поддерживался Министерством обороны США, которое стало заказчиком для создания семейства стандартов IDEF, занимающее важную часть программы интегрированной компьютеризации производства.

Итак, вторым методом структурного анализа является IDEF. Он используют в таких моделях, где важное место занимает последовательность действий и связь между ними. Несмотря на то, что IDEF3 не смог получить звание федерального стандарта США, он стал широко популярным среди аналитиков в дополнении к моделированию методом IDEF0. Основной особенность метода IDEF3 является наличие сценария процесса, который позволяет выделить последовательность действий и подпроцессов анализируемой системы [14].

Третий метод носит название диаграммы потоков данных. Данный метод можно описать как иерархию функциональных процессов, связанных потоками данных. Он позволяет рассмотреть, как происходит преобразования входных данных в выходные в каждом процессе, и связь между ними [15]. Для того, чтобы произвести построения метода DFD использую две системы условных обозначенний, первая система Йордона Демарко, вторая – Гейна Сэрсона. Данные нотации имеют небольшие различия в виде графического изображения символов. В соответствии с данными методами модель системы определяется как иерархия диаграмм потоков данных, описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи потребителю. В большинстве случаев каждый класс системы может быть смоделирован при помощи методов DFD, так как они первоначально были разработаны для проектирования информационных систем и имели больше возможностей в отображении специфики систем, в отличии от метода SADT, который создавался для проектирования систем в целом.

Большую популярность также получила модель ERM, также называемая как «сущность связь». Она была разработана Питером Ченом в 1976 г. Ее часто используют в анализе структуры и проектировании, но, на самом деле она является подмножеством объектной модели предметной области. Модели этой разновидности также применяются в IDEF1X, который включается в семейство стандартов IDEF [16].

В независимости от модели существуют правила, которым должна удовлетворять любая структура модель ПО. Они характеризуют качество данной модели.

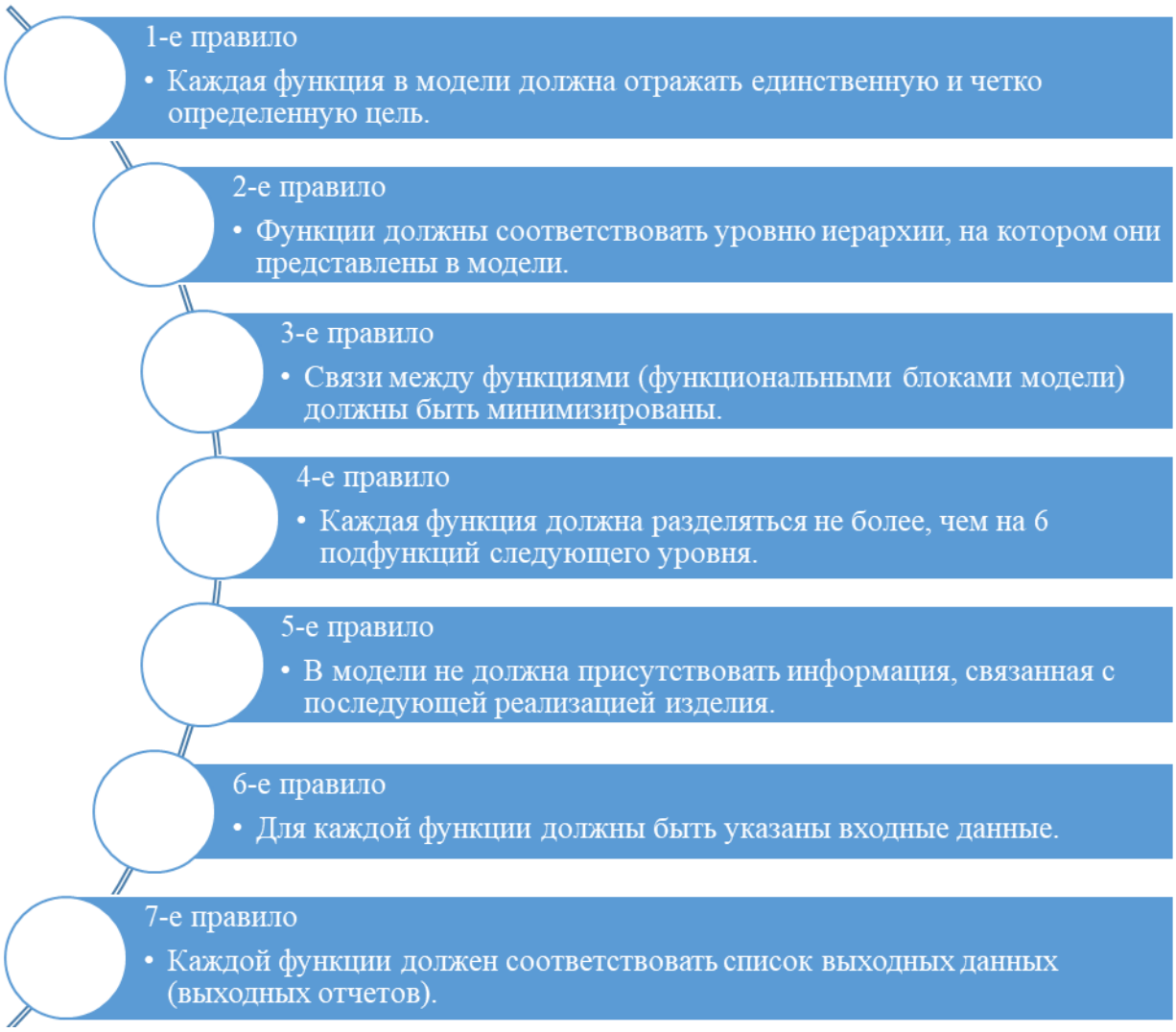

Рисунок 11 - Правила качества структурных моделей
Примечание – составлено автором на основании источника [17]

В целом модели структурного анализа имеют много сходств, например, они одинаковы с точки зрения средств изображения моделирования. Однако, метод структурного анализа необходимо выбирать в зависимости от возможностей аналитика, и степени его владения языком моделирования. В противном случае, в моделях любого метода будет сложно разобраться.

3.3 Идентификация видов, причин и влияния отказов ПС

Функционирования программного средства напрямую зависит от ее критических элементов. В большинстве случаев для того, чтобы оценить работу

ПС необходимо идентифицировать эти элементы. На основе назначения, режима работы системы, требований к использованию, особенностей элементов системы может быть повышена эффективность идентификации видов, причин и последствий отказов. Виды отказов различаются в зависимости от различных типов систем. Однако существуют общие виды отказов. В большинстве случаев каждый вид отказа может быть определен к одному из указанных общих видов (Рисунок *12*). Но, данные общие виды имеют сложность в анализе из-за своего широкого спектра, поэтому оценка объектов, проводимая для идентификации видов отказов, должна быть согласована с целями анализа.

Рисунок 12 - Пример общих видов отказов
Примечание – составлено автором на основании источника [18]

Данные о видах отказов для нового объекта можно получить из разных источников. Могут быть примененные данных о функционировании аналогичных объектов, ранее разработанных. Также используют цели проектирования и анализ функций объекта, данный способ более предпочтительней, так как он основывается на оценки конкретного изделия. Для объектов, которые уже находятся в

эксплуатации, в качестве источников о данных вида отказов могут использовать отчеты по обслуживанию объектов, где прописаны все возникшие ранее отказы и их причины. Способы обнаружения отказов также могут подразделяться в зависимости от метод диагностирования и контроля на органолептические, с использованием встроенных средств контроля и с применением внешних и встроенных средств диагностирования. Важно, чтобы все возможные отказы и их виды были зафиксированы, а результаты испытаний позволили улучшить начальные оценки.

Внешне проявления отказов может быть различно в зависимости от характеристик и параметров системы. Возможные проявления представлены на Рисунок 13.

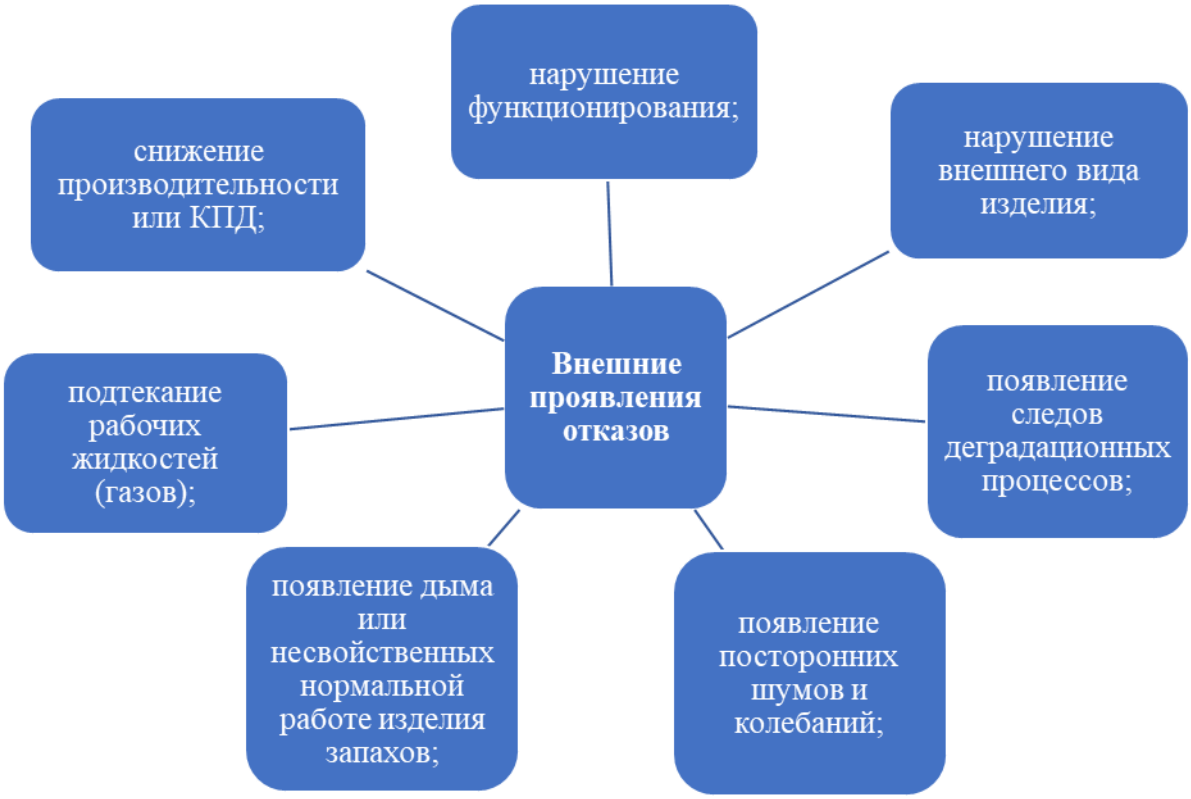

Рисунок 13 - Примеры внешних проявлений отказов
Примечание – составлено автором на основании источника [19]

Каждый вид отказа может иметь несколько причин появления. Наиболее вероятные и часто встречаемые должны быть идентифицированы и должна быть дана их характеристика. В большинстве случаев идентификация причин отказов и меры по их устранению даются на основе анализа тяжести последствий отказов. Чем более серьезней последствия отказов, тем более подробно должна быть описана причина его появления. В противном случае есть вероятность что будет потрачено много времени и средств на идентификацию причин несерьезных отказов, которые не оказывают значительного влияния на функционирование системы.

Причины отказов определяются на основе проведенных испытаний и анализе отказов. В том случае, если проект новых, то они определяются экспертным методом. В целом причины возникновения отказов можно разделить на 2 основные группы:

— причины, возникающие в случае нарушения правил и норм проектирования, техники безопасности в эксплуатации и изготовлении ПС;

— причины, связанные с устареванием объекта, его физическим изнашиванием, при соблюдении всех правил его эксплуатации.

В зависимости от вида деградационного процесса причины возникновения отказов программного средства можно подразделить на физические, химические и комбинированные. После идентификации причин и последствий отказов производят оценку рекомендательных действий.

Совокупность отказов разных видов или нескольких объектов могут стать причинами одного значительного последствия. Все возникающие последствия отказов идентифицируются, описываются и регистрируются. Цель идентификации последствий отказов состоит в том, чтобы оценить условия функционирования объекта и предложить альтернативные методы, снижающие вероятности появления отказов. Последствия на конкретном уровне называются локальными, их оценка производиться для того, чтобы идентифицировать последствия отказов в целом для системы.

Идентификация видов, причин и последствий отказов, так же определения корректирующих действий имеют важное значения для качества системы в целом. Идентификация видов отказов и меры по их смягчению, как правило, носят более серьезный характер чем данные о вероятности их появления. Однако, с помощью проведения анализа критичности можно расставить приоритеты и максимально снизить появления отказов в системе.

3.4 Оценка критичности ПО и вероятности отказов

Определение уровня безопасности ПО задача достаточно сложная, для решения которой необходимо:

— выявить и проанализировать возможные опасности, которые может создавать разрабатываемый программный продукт;

— оценить уровни критичности опасностей (по классификации НАСА: катастрофический, критический, умеренный, незначительный);

— оценить возможности возникновения опасностей (количественные и качественные меры);

— определить приоритеты рисков возникновения опасностей (индекс системного риска);

— оценить риски возникновения опасностей;

— оценить степень, с которой разрабатываемое ПО должно уменьшить риски возникновения опасности (смягчить риск опасности).

В Таблица 3 приведен пример оценки уровня критичности опасности, а также количественные и качественные меры возможности их возникновения.

Таблица 3 - Пример оценки уровня критичности опасности, а также количественные и качественные меры возможности их возникновения

№	Наименование	Оценки уровней критичности/вероятности
1	*2*	*3*
1	Определения уровня критичности	Катастрофический Потеря человеческой жизни или постоянная инвалидность; потеря всей системы; потеря

	(серьезности) опасности	наземного объекта; серьезный ущерб окружающей среде
		Критический Тяжелые травмы или временная нетрудоспособность; ущерб основной системе или окружающей среде
		Умеренный Небольшая травма; незначительное повреждение системы
		Незначительный Нет травм или незначительных травм; некоторое системное напряжение, но не повреждение системы
2	Количественная оценка возможности возникновения опасности	Возможный (весьма вероятный) Событие будет происходить часто, например, более 1 из 10 раз.
		Вероятный Событие произойдет несколько раз в течение срока службы объекта.
3	Качественная оценка возможности возникновения опасности	Возможный в принципе Вероятно произойдет когда-то в течение срока службы объекта
		Мало возможный Незначительная возможность в течение срока службы объекта
		Маловероятный Очень редко, это похоже на выигрыш в лотерею

Пример определения индекс системного риска, основанный на вышеуказанных уровнях критичности и вероятности возникновения, показан в Таблица 4, где 1 - самый высокий приоритет (самый высокий системный риск), 7 - самый низкий приоритет (самый низкий системный риск).

Таблица 4 - Приоритеты опасности - индекс системного риска

№	Уровень критичности опасностей	Возможность возникновения				
		Возможный (весьма вероятный)	Вероятный	Возможный в принципе	Мало возможный	Маловероятный
1	*2*	*3*	*4*	*5*	*6*	*7*

1	Катастрофический	1	1	2	3	4
2	Критический	1	2	3	4	5
3	Умеренный	2	3	4	5	6
4	Незначительный	3	4	5	6	7

Выбор уровня безопасности ПО проводится на основе определенных выше значений индексов системного риска и требований заказчика. Например, в стандарте НАСА установлено 3 основных уровня безопасности критического ПО: «full (полный)», «moderate (умеренный)», «minimum (минимальный)» [NASA-GB-8719.13 NASA Software Safety Guidebook (Руководство по безопасности программного обеспечения НАСА)]. Руководство по безопасности программного обеспечения НАСА содержит рекомендации, которые помогут определить, к каком уровню полноты безопасности относится разрабатываемое ПО. Так, для опасности с самым высоким индексом риска (индекс 1) не допускается разработка ПО. Проект с индексом риска опасности, равным «1», должен быть переработан, чтобы исключить или уменьшить вероятность возникновения опасности и/или уровень его критичности в пределах допустимого диапазона.

Требуемый объем действий по обеспечению безопасности увеличивается с уровнем риска. Наименьшие индексы риска «5» и больше требуют минимального уровня анализа безопасности или контроля. При индексе риска опасности «2» необходимо устанавливать уровень обеспечения безопасности – «полный (full)», для индекса риска «3» - уровень «умеренный (moderate)» и для индексов риска «4» - уровень «Минимальный (minimum)». Системы с индексом «5» находятся между минимальным и необязательным уровнями, для них уровень обеспечения безопасности должен устанавливаться исходя принятой политики безопасности и ограничений по объему ресурсов по обеспечению безопасности. Кроме того, возможно устанавливать смешанную систему уровней безопасности для одной системы. Например, одна подсистема может относиться к уровню «minimum», а другая подсистема может относиться к уровню «необязательный».

В Таблица *5* приведены рекомендации НАСА по выбору уровня обеспечения безопасности в зависимости от значения индекса системного риска.

Таблица 5 - Рекомендации по выбору уровня обеспечения безопасности в зависимости от значения индекса системного риска

№	Индекс системного риска	Рекомендуемый уровень обеспечения безопасности
1	*2*	*3*
1	1	Не применим в данном виде (запрещено)
2	2	Полный (full)
3	3	Умеренный (moderate)
4	4,5	Минимальный (minimum)
5	6,7	Отсутствуют требования (необязательный)

От выбора уровня безопасности ПО зависят требования к его сертификации. Для ПО более высокого уровня безопасности предъявляются более строгие требования не только к показателям, но и к процессам его создания, а также к полноте и независимости сертификации.

Допустимая вероятность и частота отказов для наиболее критичных программных систем очень низкая. Уровень безопасности программных систем, критичных для безопасности, в соответствие со стандартами IEC 61508 задается параметром SIL (Safety Integrity Leve - Уровень полноты безопасности). Под полнотой безопасности (safety integrity) понимается вероятность того, что система связанная с безопасностью, будет удовлетворительно выполнять требуемые функции безопасности при всех заданных условиях в течение требуемого периода времени. Параметр SIL отражает способность системы обеспечивать функции безопасности, чем выше уровень опасности системы, тем выше требования к надежности функций безопасности. В Таблица *6* приведены рекомендуемые IEC 61508 SILs уровни полноты безопасности SIL и соответствующие им допустимые вероятности отказов.

Таблица 6 - Рекомендуемые IEC 61508 SILs уровни полноты безопасности SIL и соответствующие им вероятности отказов

№	Safety Integrity Level (SIL) Уровень полноты безопасности	Probability of dangerous failure per hour (Continuous mode of operation) Вероятность опасного отказа в час (непрерывный режим работы)
1	*2*	*3*
1	SIL 4	>= 10^{-9} до < 10^{-8}
2	SIL 3	>= 10^{-8} до < 10^{-7}
3	SIL 2	>= 10^{-7} до < 10^{-6}
4	SIL 1	>= 10^{-6} до < 10^{-5}

Стандарт IEC 61508 определяет четыре уровня полноты безопасности: SIL 1, SIL 2, SIL 3 и SIL 4 и соответствующие им требования к полноте безопасности для функции безопасности. SIL 4 соответствует самым высоким требованиям безопасности, а SIL 1 – самым низким. Для каждого уровня определены различные степени вероятности отказа, которые не должны превышать способность системы выполнять функции безопасности. Необходимый уровень SIL рассчитывается на основе оценки рисков. Например, для уровня SIL 4 (примерно соответствующей категории «максимум» НАСА) допустимая частота отказов для ПО не должна превышать одного отказа на миллиард часов непрерывной работы (это примерно 114 155 лет).

В качестве показательного примера критически важного для безопасности ПО можно привести данные НАСА о программном обеспечении космического челнока Шаттла. Это ПО имело размерность 420 KLOC (Lines Of Code – количество строк в программном коде) и в каждой из трех последних версий была только одна ошибка. Это самый низкий уровень дефектов в мире, который на порядок меньше, чем в среднем по обычному программному проекту.

4 Экономическая часть

4.1 Основные показатели экономической эффективности ПС

Наиболее важная характеристика, оценивающая качества программного средства, это его экономическая эффективность, которая показывает соотношение затрат и результатов функционирования программного изделия. Основные показатели, которые характеризуют экономическую эффективность: период окупаемости вложений, экономический эффект и коэффициент экономической эффективности инвестиций.

Экономический эффект можно представить в виде экономии от внедрения кого-либо мероприятия в компании, выраженной в денежной форме. Годовой экономический эффект от разработки нового программного средства может быть определен по формуле (1) , где рассчитывается разность между затратами базового варианта и нового в расчете на объем выпуска за год.

$$Э = (З_1 - З_2) \times А_2 \qquad (12)$$

где Э — годовой экономический эффект от производства ПС, тенге;

З1, З2 — приведенные затраты на единицу выпуска ПС по базовому и новому вариантам, тенге.;

А2— годовой объем выпуска в расчетном году, ед.

Существует четыре вида экономического эффекта, которые различаются в зависимости от стадии жизненного цикла ПС и целей расчёта. Первый вид экономического эффекта называется предварительный. Он рассчитывается до начала разработки по данным прогноза использования и технического предложения. Второй вид – потенциальный, он рассчитывается после разработки программного средства на основе технических характеристик и составленных прогнозов о максимальном объеме спроса на ПС на рынке. Этот вид используется при оценке деятельности компании-разработчика программного средства. Третий вид экономического эффекта гарантирует особую величину экономического эффекта для конкретного объекта внедрения, и, соответственно, называется гарантированный. Гарантированный эффект также рассчитывается после

окончания разработки при оформлении договора между предприятием-заказчиком и предприятием- разработчиком и является основой для утверждения обоснованной цены на программное изделие. Четвёртый вид экономического эффекта называется фактический и он рассчитывается на основе учета данных и соотношения результатов и затрат при каждом конкретном случае применения ПС.

Основные источники, за счет которых можно увеличить резервы экономим при использовании нового программного средства отражены на Рисунок 14. Среди них можно выделить снижение трудоемкости производства и внедрения программного средства, а также уменьшения процента постоянных расходов в структуре затрат на ПС.

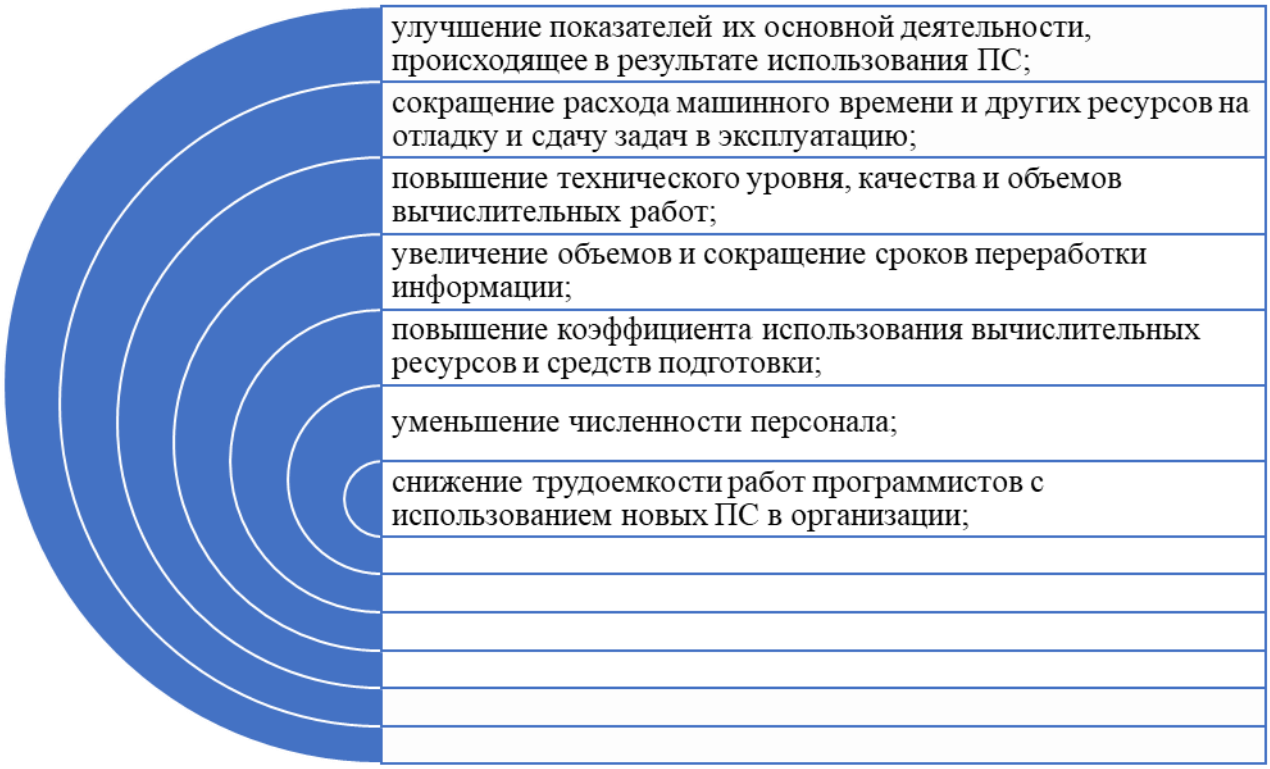

Рисунок 14 - Источники появления экономического эффекта от внедрения нового программного средства *[20]*

Срок окупаемости является обратным показателем экономической эффективности и показывает период, за который, компания возвращает потраченные деньги на разработку и внедрения программного средства за счет полученного эффекта.

Под коэффициентом экономической эффективности вложений понимается значение прироста прибыли за год, которое образуется в результате использования ПС, на один тенге единовременных крупных вложений.

4.2 Анализ и расчет затрат на разработку ПС

Программные средства используются на каждом предприятии. Однако, в некоторых случаях для более эффективной работы необходимо разработать и внедрить новые специализированные программные средства, которые не были ранее использованы в других компаниях, так как они могут повысить показатели работы предприятия и привести к высокому экономическому эффекту. Прежде чем осуществлять данное нововведение, необходимо сделать анализ и оценку его выгоды. Анализ начинается в первую очередь ПС.

Чтобы провести анализ трудовых затрат нужно просчитать трудоемкость всего проекта. Трудоемкость определяется как количество часов, которые были потрачены на разработку одного ПС. Для того, чтобы рассчитать трудоемкость проекта в целом, необходимо определить сколько потребуется времени для каждого этапа проекта. В целом проект состоит из 6 этапов:

1) сбор первоначальной информации, т. е. поставка задач, создание плана работы и т. п. Для этого этапа потребуется 7 рабочих дней, или 56 часов;

2) разработка алгоритма проекта означает создание структуры работы, вследствие которой разрабатывается само средство, на нее потратиться 4 дня или 32 часа;

3) разработка программного средства – это самый длительный процесс, будет длиться около 2-х месяцев, т. е. 336 часов. К окончанию данного этапа должно быть создано само программное средство и все необходимые документы, и файлы для его функционирования;

4) отладка программного средства проводится по мере разработки и после ее окончания для проверки функционирования программы, занимает 24 часа;

5) тестирование проходит после окончания основной работы над проектом, для более глубокой проверки работы программного средства, обнаружения ошибок и их исправления, на него потребуется 2 рабочих дня;

6) разработка описания для потребителей – заключительная стадия, для которой необходимо детальное изучения потребностей пользователя, займет 24 часа.

Итого на разработку программного средство потребуется 488 часов, или 61 рабочий день, 2 месяца 19 рабочих дней (Таблица 7).

Таблица 7 - Трудоемкость разработки программного средства

№	Наименование этапов	Единицы измерения	Трудоемкость
1	*2*	*3*	*4*
1	Сбор исходной информации	человеко-час	56
2	Разработка алгоритма проекта		32
3	Разработка программного продукта		336
4	Отладка программного продукта		24
5	Тестирование программного продукта		16
6	Разработка описания программного продукта для пользователя		24
Итого (V_{nn}):			488

Второй этап анализа — это расчет заработной платы работников. Для этого нужно определиться с числом рабочих и их часовой ставкой. Для разработки программного средства потребуется один инженер - разработчик. Средняя заработная плата такого инженера на рынке составляется около 250 тыс. тенге в месяц, это означает, что часовая ставка рабочего составляет 1490 тенге за час (Таблица 8). Всего за весь период работы над проектом затраты на оплату труда составят 1 млн 39 тыс. тенге, учитывая премиальные и дополнительные выплаты.

Таблица 8 – Расчет затрат на оплату труда за проект

№	Наименование	Формула	Расчет	Значение
1	*2*	*3*	*4*	*5*

1	Часовая тарифная ставка ($t_{ст}$), тенге/час	-	-	1 490
2	Ставка единого социального налога, %	-	-	9,5
3	Заработная плата по тарифу ($З_т$), тенге/проект	$З_т = V_{пп} \times t_{ст}$	$З_т = 488 \times 1490$	727 120
4	Премиальная зарплата ($З_{прем}$), тенге/проект	$З_{прем} = З_т \times 30\%$	$З_{прем} = 727120 \times 30\%$	218 136
5	Дополнительная зарплата ($З_{доп}$), тенге/проект	$З_{доп} = (З_т + З_{прем}) \times 10\%$	$З_{доп} = (727120 + +218136) \times 10\%$	94 526
6	Отплата труда за разработку программного средства (З), тенге/ проект	$З = З_т + З_{прем} + З_{доп}$	$З = 727120 + 218136 + +94526$	1 039 782
7	Отчисления в социальные фонды (О), тенге/проект	$О = З \times 9{,}5\%$	$О = 1039782 \times 9{,}5\%$	98 779

Следующий этап анализа проекта — это расчет капительных вложений, т.е. необходимого оборудования для создания программного средства. Всего для создания программного средства потребуется один компьютер, мышка, съемный накопитель, стол и стул. Все капительные затраты составят 206 тыс. тенге (Таблица 9).

Таблица 9 - Капитальные вложения в проект

№	Наименование	Количество	Цена за единицу, тенге	Общая сумма, тенге
1	*2*	*3*	*4*	*5*
1	Ноутбук ASUS Laptop 15 M509DA-EJ080	1	169 990	169 990

2	Мышь Logitech M310 Wireless USB	1	7 910	7 910
3	USB Flash карта Kingston Data Traveler 100 G3 128 GB	1	9 615	9 615
4	Стол	1	10 230	10 230
5	Стул	1	8 670	8 670
Итого:		5	206 415	206 415

По мере реализации проекта необходимо проводить амортизационные отчисления. Амортизация представляет собой постепенный перенос стоимость основных средств на стоимость произведенного продукта, в данном случае программного средства. Амортизационные отчисления рассчитываются линейным способом по следующей формуле (*13*):

$$А = КВ \times Н_а \tag{13}$$

где А – амортизационные отчисления;

КВ – общая стоимость капитальных вложений;

$Н_а$ – норма амортизации.

В проекте амортизационные отчисления составили 42 тыс. тенге (Таблица 10). Однако, эта сумма просчитана на год, длительность же проекта составляется меньше 3-х месяцев, поэтому необходимо просчитать амортизационных отчисления на время разработки программного средства по формуле (*14*). На время реализации проекта амортизационные отчисления составили 10 тыс. тенге.

$$А_п = \frac{КВ \times Н_а \times N}{100 \times 12 \times n} \tag{14}$$

где $А_п$ – амортизационные отчисления на время реализации проекта;

КВ – общая стоимость капитальных вложений;

$Н_а$ – норма амортизации;

N – количество рабочих дней в проекте.

n – количество рабочих дней в месяце.

Таблица 10 - Расчет амортизационных отчислений в проекте

№	Наименование	Срок службы оборудования, лет	Норма амортизации, %	Амортизационные отчисления, тенге	АО на время разработки программно

					го средства, тенге
1	*2*	*3*	*4*	*5*	*6*
1	Ноутбук ASUS Laptop 15 M509DA-EJ080	5	20%	33 998	8 230
2	Мышь Logitech M310 Wireless USB	3	33%	2 637	638
3	USB Flash карта Kingston Data Traveler 100 G3 128 GB	4	25%	2 404	582
4	Стол	6	17%	1 705	413
5	Стул	6	17%	1 445	350
Итого:				42 188	10 212

Важным этап в расчетах затрат также выступают оценка энергетических затрат, так как проект непосредственно связан с потреблением электроэнергии. Энергетические расходы включают в себя затраты на освещение помещения, где проводится разработка программного средства и затраты на энергию, которую потребляет компьютер.

При расчете затрат на освещение нужно определить потребность электроэнергии на освещение помещения, которая зависит от квадратуры помещения, нормы и часов работы светильников и рассчитывается по формуле (*15*).

$$P_{\text{о}} = \frac{N_{\text{л}} \times \text{Т}_{\text{ср}} \times \text{К} \times \text{Д}_{\text{р}}}{1000} \tag{15}$$

где $P_{\text{о}}$ - потребность электроэнергии на освещение;

$N_{\text{л}}$ – мощность одной осветительной точки, Вт;

$\text{Т}_{\text{ср}}$ - среднее число часов горения в сутки;

К- число осветительных точек;

$Д_р$- число рабочих дней.

Согласно расчетам, потребность составила 29,28 кВт*ч (Таблица 11). Из этого следует, что можно рассчитать затраты на освещение, умножив потребность на цену одного кВт/ч электроэнергии, которые составили 561,30 тенге.

Таблица 11 - Расчёт энергетических затрат на проект

№	Наименования оборудования	Значение
1	*2*	*3*
1	Мощность, потребляемая ноутбуком в час, кВт	0,17
2	Цена одного кВт·ч электроэнергии, тенге	19,17
3	Календарное время работы период создания программного продукта, часов	376
4	Время планово-профилактических работ, часов	20
5	Эффективное время работы, часов	356
6	Затраты на электроэнергию при работе на ноутбуке, тенге	1 160
7	Мощность одной энергосберегающей лампы, кВт	0,015
8	Количество ламп, шт	4
9	Потребность в освещении, кВт*ч	29,28
10	Затраты на освещение, тенге	561,30
11	Обще энергетические затраты, тенге	1 721,47

Для того, чтобы рассчитать затраты на электроэнергию, затрачиваемую на работу компьютера, необходимо определить эффективное время его работы, путем разности календарного времени работы ноутбука и времени профилактических работ. Оно составило 356 часов (Таблица 11). Исходя из этого можно рассчитать общие затраты на электроэнергию при использовании компьютера по формуле (*16*), которые составили 1160 тенге. Общие энергетические затраты, которые определяются как сумма затрат на освещения и на работы на компьютере составили 1721 тенге за весь период реализации проекта (Таблица 11).

$$Э_к = Т_{эф} \times Р_к \times Ц_э \tag{16}$$

где $Т_{эф}$ – эффективное время работы ноутбука, часов;

$Р_к$,– мощность компьютера в час, кВт;

$Ц_э$ – цена одного кВт·ч электроэнергии, тенге.

.

Помимо вышеперечисленных расчётов, необходимо также учесть затраты на сертификацию и лицензирование, на аренду помещения, на создание дополнительного фонда, на накладные расходы и затраты на внедрение. Затраты на сертификацию составили 68000 тенге, при условии написания 140 строк кода за один рабочий день и общем количестве строк 1000, стоимость одной строки составит 68 тенге (Таблица 12). Стоимость аренду помещения размеров 10 $м^2$, в Бостандыкском районе в административном здании составляет 40 тыс. тенге в месяц. За весь срок реализации проекта эти затраты составят 120 тыс. тенге. Затраты на внедрение программного средства будут заключать в обучении персонала использовать новый программный продукт, они составят 43 тыс. тенге, и увеличатся также социальные выплаты на 4 тыс. тенге. В общем на реализацию данного проекта потребуется 2 млн 269 тыс. тенге.

Таблица 12 - Структура затрат на проект

№	Затраты	Сумма, тенге	Доля от себестоимости, %
1	*2*	*3*	*4*
1	ФОТ	1 039 782	45,8%
2	Социальный налог	102 931	4,5%
3	Оборудование	206 415	9,1%
4	АО	10 212	0,5%
5	Энергетические затраты	1 721	0,1%
6	Арендная плата	40 000	1,8%
7	Лицензирование и сертификация	68 000	3,0%
8	Внедрение	43 701	1,9%
9	Дополнительный фонд	302 552	13,3%
10	Накладные расходы	453 829	20,0%
Итого:		2 269 143	100%

Внедрение данного разработанного программного средства позволит быстро и грамотно выявлять вероятность и критичность отказов, тем самым оно позволит снижать количество сбоев в программах и увеличивать производительность работы в целом. Снижение отказов возрастет на 20%. Время, затрачиваемое на один проекта, снизится со среднего значения в 3 месяца, до 2 месяцев и 8 дней. Вследствие чего увеличится число проектов, осуществляемых за год почти на 26%, и составит 5,04 проекта в год. Средние затраты необходимые осуществления проекта до внедрения нового программного средства составляли 2 млн. 325 тыс. Однако, после внедрения ПО, они снизятся 16% и составят 1 млн 961 тыс. тенге. Итак, согласно формуле (*12*), можно произвести следующий расчёт (формула (*17*)). Из расчетов следует, что годовой экономический эффект составит 1 млн 834 тыс. тенге.

$$Э = (З_1 - З_2) \times А_2 = (2\,325\,138 - 1\,961\,172) \times 5{,}04 = 1\,834\,390 \quad (17)$$

где Э — годовой экономический эффект от производства ПС, тенге;

З1, З2 — приведенные затраты на единицу выпуска ПС по базовому и новому вариантам, тенге.;

А2— годовой объем выпуска в расчетном году, ед.

Экономически, эффективность данных затрат можно рассчитать через срок окупаемость вложений и коэффициента эффективности затрат. Срок окупаемости представляет собой период времени, за которое компания сможет вернуть вложенные деньги в проект через полученную экономию от внедрения. Он рассчитывается по формуле(*18*).

$$Т_0 = \frac{К}{Э} = \frac{2\,269\,143}{1\,834\,390} = 1{,}23 \quad (18)$$

где $Т_о$ – срок окупаемости, лет;

К – затраты, связанные с созданием и внедрением программного продукта, тенге;

Э - годовая экономия, тенге.

Период окупаемости проекта составил 1 год 2 месяц. Коэффициент экономическая эффективность проекта является обратным показателем периода окупаемость, расчет представлен по формуле (19) и составил 0,70, что говорит у том, что каждый один тенге затрат, потраченный на разработку программы, за первый год эксплуатации принесет 80 тиын тенге прибыль.

$$К_э = \frac{Э}{К} = \frac{1\,834\,390}{2\,269\,143} = 0{,}80 \qquad (19)$$

где Кэ – коэффициент экономической эффективности;

Э - годовая экономия, тенге;

К – затраты, связанные с созданием и внедрением программного продукта, тенге.

Подводя итоги проведенного анализа можно сделать вывод, что проект по разработке нового программного изделия, который снизит вероятность появления отказов оборудования и позволит уменьшить количество времени, когда оборудования находится в простое или на ремонты, является эффективным, так как срок его окупаемость составляет чуть больше года, а экономическая эффективность подтверждается что 70% капитальных вложений окупится в течении первого года эксплуатации.

5 Охрана безопасности жизнедеятельности при работе с ПК

5.1 Охрана труда и техника безопасности при работе с ПК

Пользование персональным компьютером, также как и другими электрическими приборами может привести к негативным последствиям. Поэтому во избежание травм, руководство компании должно обеспечить сотрудникам безопасные условия труда путем исполнения норм по охране труда и создания

актов, методических документаций, нормативов, устанавливающие правила безопасности на рабочем месте. Техника безопасности определяется общедоступной инструкцией, которая включает в себя информацию по требованиям к рабочему месту и процессу использования оборудования. Выполнения этих требований контролируется соответствующими органами власти.

При работе с персональным компьютером каждый сотрудник должен иметь рабочее места, площадь которого должна составлять минимум 6 м^2. При этом размещение рабочих мест должно быть организовано так, чтобы расстояния между экраном одного монитора до тыла другого составляло минимум 2 м, а между боковыми поверхностями – 1,2 м. Высота рабочей поверхности должна находиться в пределах 65-85 см (рисунок 1). Рабочий стул должен иметь возможность регулировки наклона спинки и высота, которая должна составлять от 40-52 см [21]. Каждое рабочее место должно быть оборудованы подставкой для ног. Освещение на рабочем месте должно включать как искусственные, так и естественные источники, которые не должны отражаться или отсвечиваться от близлежащих поверхностях. Освещение помещения только за счет потолочного света нежелательно. Все компьютерные провода должны быть изолированы, в частности недопустимо их размещение рядом с отопительной системой. Системный блок должен находиться в пространстве, где есть возможность нормальной вентиляции, т. е. запрещается располагать его в нише стола.

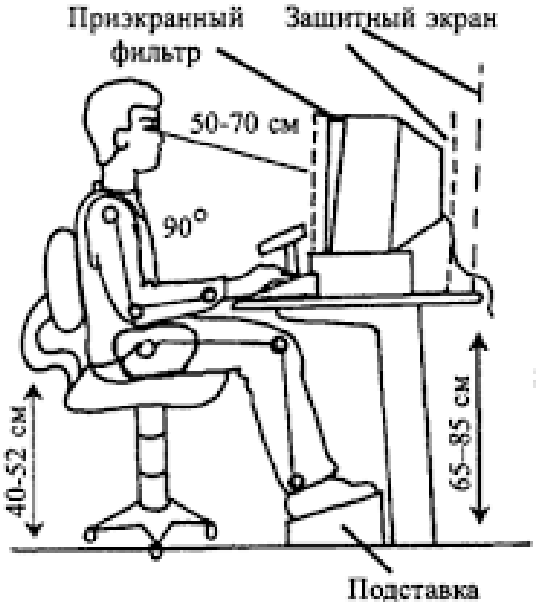

Рисунок 15 -Требования при работе с персональным компьютером *[21]*

Для того, чтобы снизить вероятность появления всевозможных рисков, необходимо соблюдать меры предосторожности и безопасности непрерывно и комплексно, в независимости от этапа пользования компьютером.

Перед началом рабочего процесса каждое рабочее место проверяется специалистом. Перед непосредственным включением компьютера необходимо убедится в отсутствии оголенных проводов, шнуров в зоне рабочего места. При нахождении видимых повреждений начинать работу запрещается, необходимо обратиться в сервисный центр для профессиональной помощи. Предметы на рабочем столе не должны мешать видимости. Дисплей экрана должен быть абсолютно чистым. На системном блоке не должно быть никаких предметов, в частности которые могут помешать системе охлаждения компьютера. Компьютер должен быть подключен к розетке, которая имеет заземляющую шину. Запрещается включения компьютера при повышенной влажности или при открытых источниках влажности в помещении.

В связи с тем, что ПК является электрическим прибором, на него распространяются все правила безопасности, которые используются при работе с

проводниками тока. Запрещается самостоятельно менять расположение проводов, располагать жидкости рядом с модулями компьютера, поэтому кофемашины и кулеры с водой нужно располагать в достаточно отдаленном от рабочих столов месте. Нельзя использовать компьютером влажными руками, проводить очищение поверхностей компьютера, снимать или разбирать составные части в его включённом состоянии. Ремонт компьютером должен проводить только специализированный сотрудник. Запрещается курить или употреблять пищу в помещении с компьютерами. При ощущении запаха гари необходимо сразу выключить компьютером и обратиться к ответственному за его техническое состояние.

Перед окончанием работы нужно заключить все программы и окна, извлечь все активные носители информации. Выключение начинается по цепочке: системный блок, периферия, общее питание.

Соблюдение всех перечисленным правил необходимо во избежание проблем, так как неправильная работа с персональным компьютером может привести к множественным угрозам для здоровья человека.

5.2 Анализ опасных и вредных производственных факторов при работе на ПК и их последствия

Несмотря на то, что компьютерные технологии считаются безопасными в эксплуатации, соблюдение норм безопасности при работе с ними необходимо, так как существует многочисленные факторы, которые оказывают серьезное влияние на состояния человека и его здоровья (Рисунок *16*). Согласно ГОСТу к производственным факторам, возникающим при долгом пользовании персональным компьютером на рабочем месте относятся следующие понятия, приведенный в Рисунок *16*

Рисунок 16 - Опасные и вредные производственные факторы при выполнении работ на персональном компьютере

Примечание – составлено автором на основании источника *[22]*

Данные факторы можно разделить на несколько групп:

—К физически вредным факторам относятся факторы, связанные с изменением температуры воздуха, влажности, выделением вредных веществ и излучением;

—Химически вредные факторы связаны с выделением химически опасных веществ;

—Психофизически вредные факторы отражаются непосредственно на органах человека и влияют на его психологическое состояние.

В независимости от группы, рабочим необходимо соблюдать меры безопасности во избежание последствий, к которым может привести чрезмерное использование ПК. В многих развитых странах вопрос о безопасности использования технологий с дисплеем решается на национальном уровне, так как, например в Германии, профессии, связанные с использованием дисплеев, входят в

список 40 наиболее опасных профессии. Данная статистика имеет место быть, так как факторы, влияющие на людей во время работы с компьютерами, имеет серьезные последствия.

Наиболее часто встречаемая проблема здоровья людей, возникающая из-за длительного пользования ПК, связанна со зрением. Длительная нагрузка на глаза приводит к снижению зрения, появлению близорукости, переутомлению глаз, возникновению головной боли и повышению нервного напряжения. Для снижения влияния данного фактора, при выборе компьютера необходимо обращать внимание на характеристики технического состояния компьютера, т.е. на яркость, контрастность, разрешающую способность и частоту мелькания, которые способны снизить последствие высокого напряжения глаз. Также, необходимо грамотно размещать оборудование относительно источников света, так чтобы снизить многочисленные источники прямой и отраженной блескости, неравномерное распределение яркости, в связи с тем, что освещение рабочих мест является важным фактором, влияющим на степень зрительного утомления. Защита от избыточного сине-фиолетового света и повышение четкости изображения являются одними из важных мер в профилактике ухудшения зрения, так как не только ультрафиолетовое излучением, но также и избыточный сине-фиолетовый свет могут привести к помутнению оптических сред глаз. Многие ученые утверждают, что если во время работы за компьютером происходит большое воздействие на органы зрения, то от чрезмерной нагрузи и лечение глаз будет происходит за счет энергии из других органов человека, что в последствие приведет к болезням головного мозга, почек, сердца и т. д.

Помимо жалоб на органы зрения, многие сотрудники, работа которых связаны с использованием компьютеров, замечают боли в других частях тела, а именно у 57,7% обследованных операторов были жалобы на общее психическое состояние. У них наблюдалась повышенная утомляемость, частые головные боли, бессонница и снижение активности и работоспособности. У 40,3% работающих с ЭВМ были выявлены нервные нарушения, связанные с повышенной

раздражительностью, депрессией и необоснованным чувством беспокойства [23]. Данные последствия работы с ПК могут быть снижены только за счет сокращения времени препровождения перед дисплеем и обязательным отдыхом и физическими нагрузками.

Работа с ПК в большинстве случаев предполагает долговое пребывание в показах, которое приводит к статистического напряжению мышц тела, что сказывается на уровне утомляемости и частоты появления жалоб на здоровья. Согласно исследованию, 52% операторов отмечают чувство окаменелости и онемения мышц шеи и плечевого пояса, боль в позвоночнике появляется у 42,9%, одеревенелость рук и ног возникает у 15,2% [23]. Боли в мышцах разных частей тела возникают вследствие того, что работники долгое время находятся в одном положении, мышцы сокращаются, и в них нарушается правильное кровообращение. Это, в свою очередь, отражается на том, что мышцы не достаточно быстро получают необходимые питательные вещества, и в них накапливаются продукты распада, которые и являются причинами боли. Для того, чтобы снизить данные последствия работы с ВДТ, необходимо приобретать правильную мебель и грамотно располагать рабочее место, так как многие исследователь уверены, что зачастую на предприятия мебель не соответствует антропометрическим характеристикам человека. Также неудачная конструкция клавиатуры вынуждает использовать усилия, чтобы кисти располагались параллельно клавиатуре, что также приводит к перегрузке мышц. Физические активности – одно из наиболее эффективных способов, чтобы снизить вероятность данных заболеваний.

До недавнего времени использование низкочастотных ЭМП считалось абсолютно безопасными, однако в настоящее время выяснилось, что они обладают интересным свойством, которое не разделяют рентгеновские лучи: электромагнитные поля могут влиять и способствовать биологическим сдвигам в клетках, вплоть до изменения синтеза ДНК. Данные говорят о том, что слабые ЭВМ могут вызывать тошноту, мигрень, усталость и даже аллергию. Это подтвердили

исследования, которые доказали, что магнитные поля с частотой 50 Гц и интенсивностью в 0,2-0,3 А/м вблизи 30-50 см от компьютеров могут стать причиной образования злокачественных заболеваний, в том числе в крови и мозге. Поэтому согласно данным, у операторов ЭВМ чаще чем у других специальностей встречается опухоль мозга [23]. Снижения вероятности данное последствие возможно только за счет снижения времени работы за компьютером.

Кроме выше представленных вредоносных причин и их последствий, связанных до этого только со зрительными и эмиссионными параметрами компьютеров, на работника при использовании компьютерам могут также воздействовать шум от работы ЭВМ и другого оборудования в комнате, тепловыделения и выделение различных вредоносных препаратов в воздух рабочей среды. Не считая этого, если плохо соблюдать условия пользования ПК, постоянно имеется возможная угроза поражения электрическим током, так как устройство подключено к питанию электрической энергией, ежели никак не соблюдаются строго верховодила техники сохранности. Также, имеется угроза возгорания вследствие перегрузки, если была неверная эксплуатация и включение нескольких электроприборов к источникам питания.

Как и любое оборудование персональный компьютером имеет свои нормы использования, регламентируемые охраной труда и безопасность жизнедеятельности людей. Данные правила необходимо соблюдать с целью снижению вероятности негативных ситуаций на рабочем месте. Также длительная работа за техническим устройством может оказать более глубокое влияние на состояния здоровья человека. Несмотря на то, что вышеперечисленные факторы, соблюдение норм безопасности позволит снизить вероятности появления данных негативных последствий и улучшить условия работы сотрудников предприятий.

Заключение

Целью данной работы являлась разработка методики выполнения метода анализа видов, последствий и критичности отказов применительно к программным средствам космического назначения.

В ходе работы был проведен тщательный анализ методов оценки надежности ПС. Были обозначены особенности и показатели надежности программных средств, названы общие требования к оценке надежности программных средств, а также изучены современны методы оценки надежности ПС.

При определении особенностей метода анализа видов, последствий и критичности отказов ПС были изучены следующие характеристики:

— Назначение и возможности SFMECA
— Методы анализа надежности, используемые в SFMECA
— Порядок анализа надежности в SFMECA
— Установление категории тяжести последствий отказов.

Разработка методики выполнения анализа видов, последствий и критичности отказов ПСКН была приведена к общему теоретическому виду. Были определены требования к программным средствам для выполнения SFMECA, затем была формально представлена структура ПС для применения SFMECA, были идентифицированы возможные виды, причины и влияния отказов ПС. В конце была дана оценка критичности и вероятности отказов программных средств космического назначения.

Список использованных источников

1. Исмаил Е. Е. Конспект лекций по дисциплине «Надежность систем управления летательных аппаратов».
2. ISO/IEC 25040:2011 Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Evaluation process (Системная и программная инженерия. Требования к качеству и оценка системы и программного продукта (SQuaRE).
3. Жаднов В. В., Лазарев Д. В. Качественные и надежностные показатели и характеристики программного обеспечения // Труды Международного симпозиума «Надежность и качество», 2005. - т. I.
4. Балыбердин В. А., Белевцев А. М., Степанов О. А. Вопросы оценки и обеспечения надёжности программных средств АСУ специального назначения // Известия Южного федерального университета. Технические науки, 2014. - №4 – с.153
5. Василенко Н. В., Макаров В. А. Модели оценки надежности программного обеспечения // Вестник Новгородского государственного университета им. Ярослава Мудрого, 2014. -№ 28.
6. Поначугин А. В. Определение надёжности программного обеспечения в структуре современной информационной системы // Кибернетика и программирование, 2019. - № 2. - с. 65-72.
7. Пакулин Н. В., Лаврищева Е. М., Рыжов А. Г., Зеленов С. В. Анализ методов оценки надежности оборудования и систем. Практика применения методов. – М:Труды ИСП РАН, 2018. -с. 99–120.
8. Мартишин С. А. Основы теории надежности информационных систем. - М: Инфра-М, Форум, 2018.
9. Исмаил Е. Е. Показатели надежности программных средств космического назначения // Вестник Алматинского университета энергетики и связи, 2016. - №3. - с. 27-35

10. Иванов Б.В., Кане М.М., Корешков В.Н., Схиртладзе А.Г. Системы, методы и инструменты менеджмента качества: Учебник для вузов. – С-П: Издательский дом "Питер", 2008.

11. Шищиц И. Комплексное прогнозирование оценок безопасности при захоронении радиоактивных отходов. -Litres, 2017.

12. Иванова Г.С. Технология программирования: Учебник для вузов. - М: Издательство МГТУ им. Н.Э. Бауман, 2002.

13. Марка Д.А., МакГоуэн К. Методология структурного анализа и проектирования. - М: МетаТехнология, 1993.

14. Черемных С.В. Структурный анализ систем: IDEF-технологии. - М: Финансы и статистика, 2001.

15. Калашян А.Н., Калянов Г.Н. Структурные модели бизнеса: DFD-технологии. - М: Финансы и статистика, 2003.

16. Конноли Т., Бегг К. Базы данных: проектирование, реализация и сопровождение. Теория и практика. 3-е издание. - М: Вильямс, 2003.

17. Орлов С. А. Технология разработки программного обеспечения. - С-Пг: Питер 464, 2002.

18. ГОСТ Р 51901.12-2007. *Метод анализа видов и последствий отказов.* - Москва: Стандартинформ, 2008.

19. РД 50-699-90. *Методические указания. Надежность в технике. Общие правила классификации отказов и предельных состояний.* -М., 1991.

20. Колесниченко С. И., Рудакова О. В. Показатели экономической эффективности в современных информационных системах // *Вестник ОрелГИЭТ,* 2010. -№ 2. -с. 12

21. Шумилин В. К., Осипов В. И. Типовая инструкция по охране труда для пользователей ПЭВМ в электроэнергетике. РД 153-34.0. 03.2. 98, 2000.

22. ГОСТ 12.0.003 – 74 ГОСТ 12.0.003-74 ССБТ. Опасные и вредные производственные факторы.

23. R. E. Nieuwenhuijsen. Health behavior change among office workers: an exploratory study to prevent repetitive strain injuries // *Work, 2004.* - № 3., pp. 215-224.

24. ECSS-Q-HB-80-03-12 Space product assurance. Software dependability and safety.

25. ГОСТ 27.310-95 Анализ видов, последствий и критичности отказов.

26. ГОСТ Р 51901.5-2005 (МЭК 60300-3-1:2003) Руководство по применению методов анализа надежности.

27. 1633-2016 - IEEE Recommended Practice on Software Reliability (стандарт IEEE 1633-2016 Рекомендуемая практика IEEE по надежности программного обеспечения).

28. An Introduction to Software Failure Modes Effects Analysis (SFMEA).

29. Harish Sukhwani, Javier Alonso, Kishor S. Trivedi1, and Issac Mcginnis Software Reliability Analysis of NASA Space Flight Software: A Practical Experience.