Методы обеспечения программно-аппаратной надёжности вычислительных систем

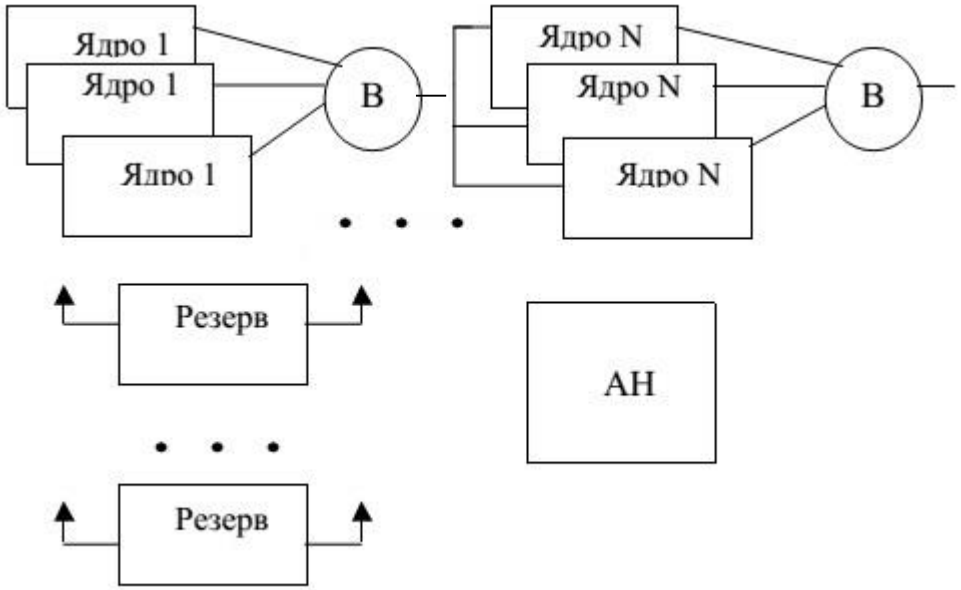
Д.т.н. профессор Чуканов В.О.

К.т.н. доцент Гуров В.В.

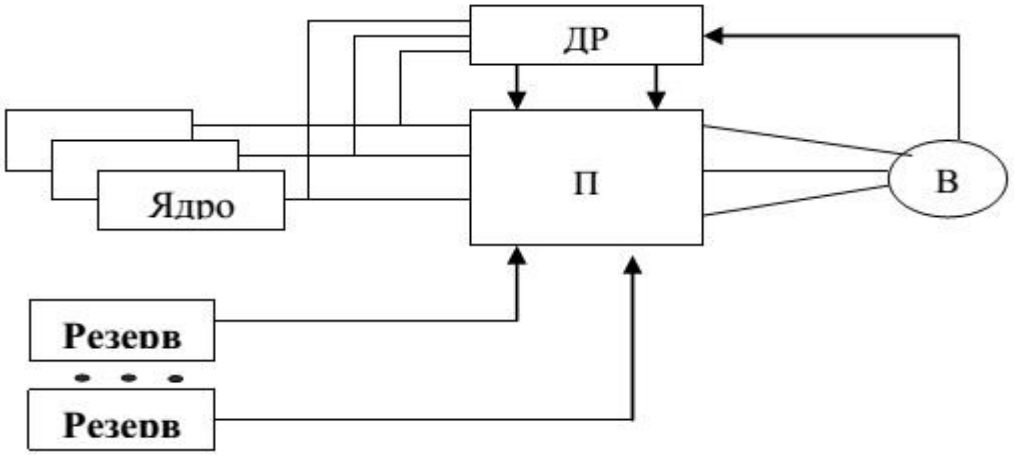
Прокопьева Е.В.

Классификация методов резервирования устройств СВС

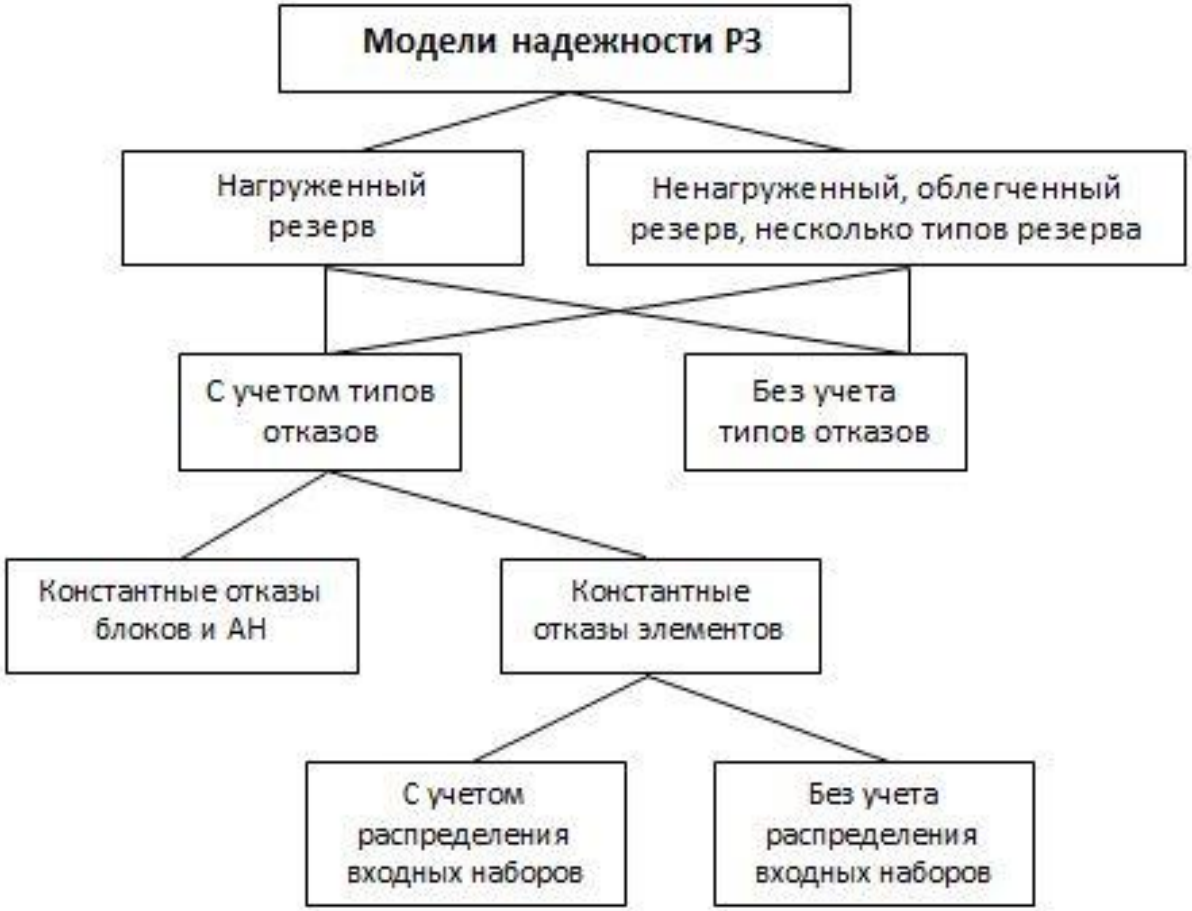
Комбинированное резервирование и три принципа его реализации



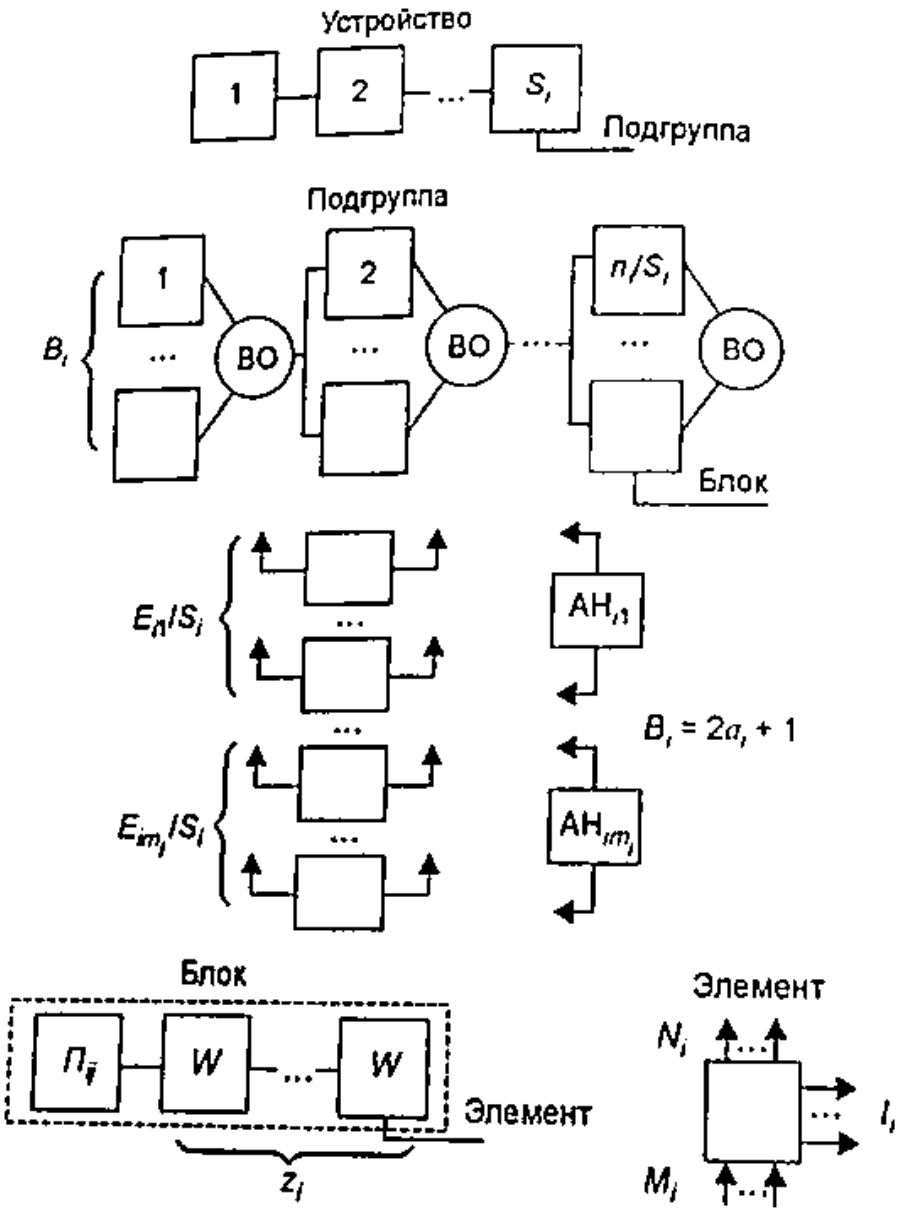
Метод резервирования, основанный на комбинации постоянного резервирования с восстанавливающим органом и скользящим резервированием



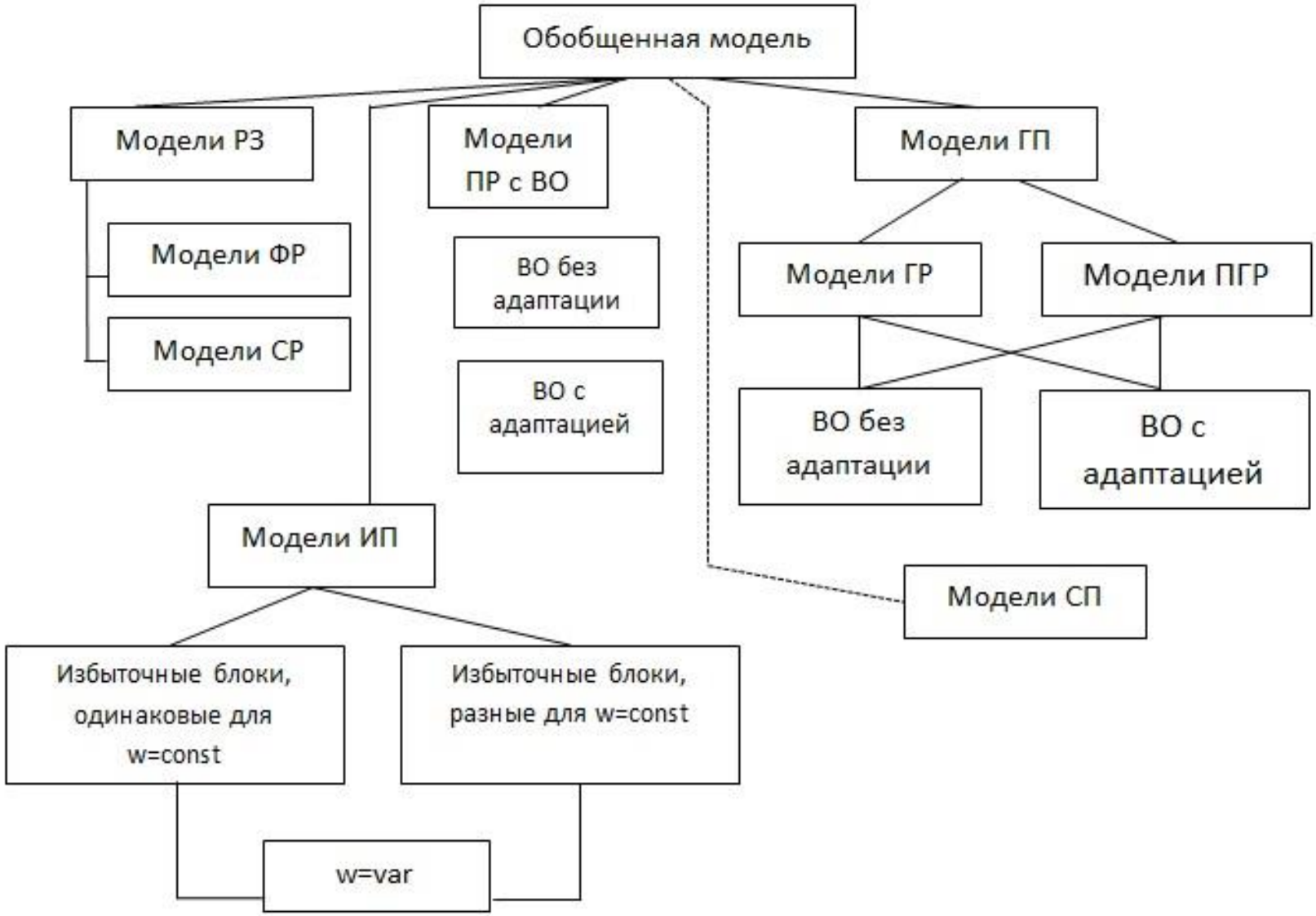
Устройство с гибридным резервированием



Модели надежности с резервированным замещением



Обобщенная логическая схема надежности для этапа эскизного проектирования



Классификация моделей надежности

Задача определения параметров устройств СВС

$$P_{\text{СВС}} = \prod_{k=1}^{N} P_k\left(z_{ki}, n_{ki}, S_{ki}, E_{kij}\right) \rightarrow max;$$

$$C^L_{\text{СВС}} \sum_{k=1}^{N} C^L_K\left(z_{ki}, n_{ki}, S_{ki}, E_{kij}\right) \leq C^L_{\text{зад}}, \qquad L = \overline{1, T},$$

Где $P_k\left(z_{ki}, n_{ki}, S_{ki}, E_{kij}\right)$ - вероятность безотказной работы k-го устройства;

$C^L_K\left(z_{ki}, n_{ki}, S_{ki}, E_{kij}\right)$ – L-й ТЭП k-го устройства;

$C^L_{\text{зад}}$ – заданный L-й ТЭП k-го устройства; T – количество ТЭП.

Коэффициент готовности. Коэффициент готовности *Кг* является самым универсальным показателем надежности восстанавливаемых систем. Он характеризует соотношение времени работоспособного состояния систем и времени ее простоев (отказовых состояний). Если за некоторый промежуток времени *Т* система находилась в рабочем состоянии время *Тр*, а режиме восстановления (простоя) – *Тв*, то коэффициент готовности определится как отношение

$$Кг = \frac{Тр}{Тр + Тв}. \qquad (2.1)$$

Если использовать приближенный метод усреднения параметров, то под значениями *Тр* и *Тв* можно понимать средние значения данных параметров, то есть *Тр* – среднее время безотказной работы системы, а *Тв* – среднее время ее восстановления после отказа.

Заметим, что данный критерий целесообразно использовать для анализа систем, имеющих в течение времени *Т* непрерывный режим работы.

Довольно распространенным в настоящее время является тип систем, работающих в режиме готовности к решению целевой задачи. Такие системы в течение времени *Т* ожидают приход задачи (запроса), а при появлении задачи занимаются ее оперативным решением, как правило, в реальном режиме времени. Для анализа таких систем целесообразно использовать оперативный коэффициент готовности.

Коэффициент простоя (неготовности). Значение данного критерия определяется как $Кнг=1-Кг$.

Оперативный коэффициент готовности. Оперативный коэффициент готовности $Кго$ используется для анализа надежности систем с нерегулярным режимом работы. Его значение можно записать в виде:

$$Кго = Кг\,P(t)\,, \qquad (2.2)$$

где $Кг$ – коэффициент готовности системы, рассчитанный по формуле 2.1, а $P(t)$ – вероятность решения целевой задачи за время t.

Вероятность безотказной работы системы за время *t*. Для восстанавливаемых систем интервал времени отсчитывается от времени последнего восстановления. Если после отказов система восстанавливается полностью, то интервал времени t отсчитывается от времени любого восстановления системы и, в принципе, данный критерий аналогичен вероятности безотказной работы, характеризующей невосстанавливаемые объекты.

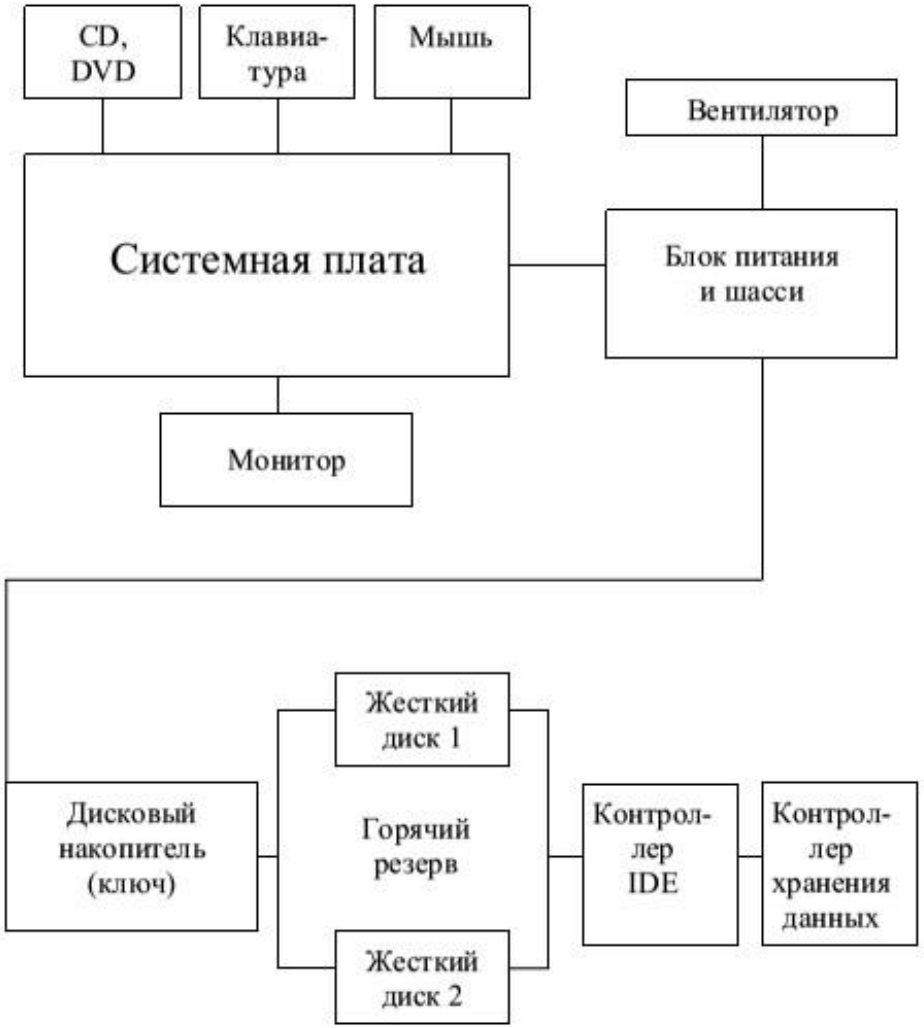
Наработка на отказ. Наработка на отказ определяется как среднее время между восстановлением системы и ее следующим отказом. Если в процессе восстановления система восстанавливается полностью, то данная характеристика аналогична среднему времени безотказной работы для невосстанавливаемых объектов.

Среднее время восстановления системы. Данный критерий характеризует усредненное значение времени восстановления системы после каждого ее отказа.

Вероятность функционально безотказной работы. Для автоматизированных систем управления, работающих в реальном времени, предлагается специфический показатель – вероятность функционально безотказной работы системы за заданное время t, позволяющий оценить качество информации о текущем состоянии системы. Значение критерия в данном случае вычисляется как сумма вероятности безотказной работы системы $P(t)$ и произведения вероятности отказа $(1-P(t))$ на коэффициент качества контроля $K(t)$.

При определенных допущениях коэффициент качества контроля можно интерпретировать как условную вероятность того, что в случае отказа системы соответствующие ошибки будут обнаружены (и/или устранены) на заданное время t_0.

Модели надежности сервера типа NT-AKPB-LEN



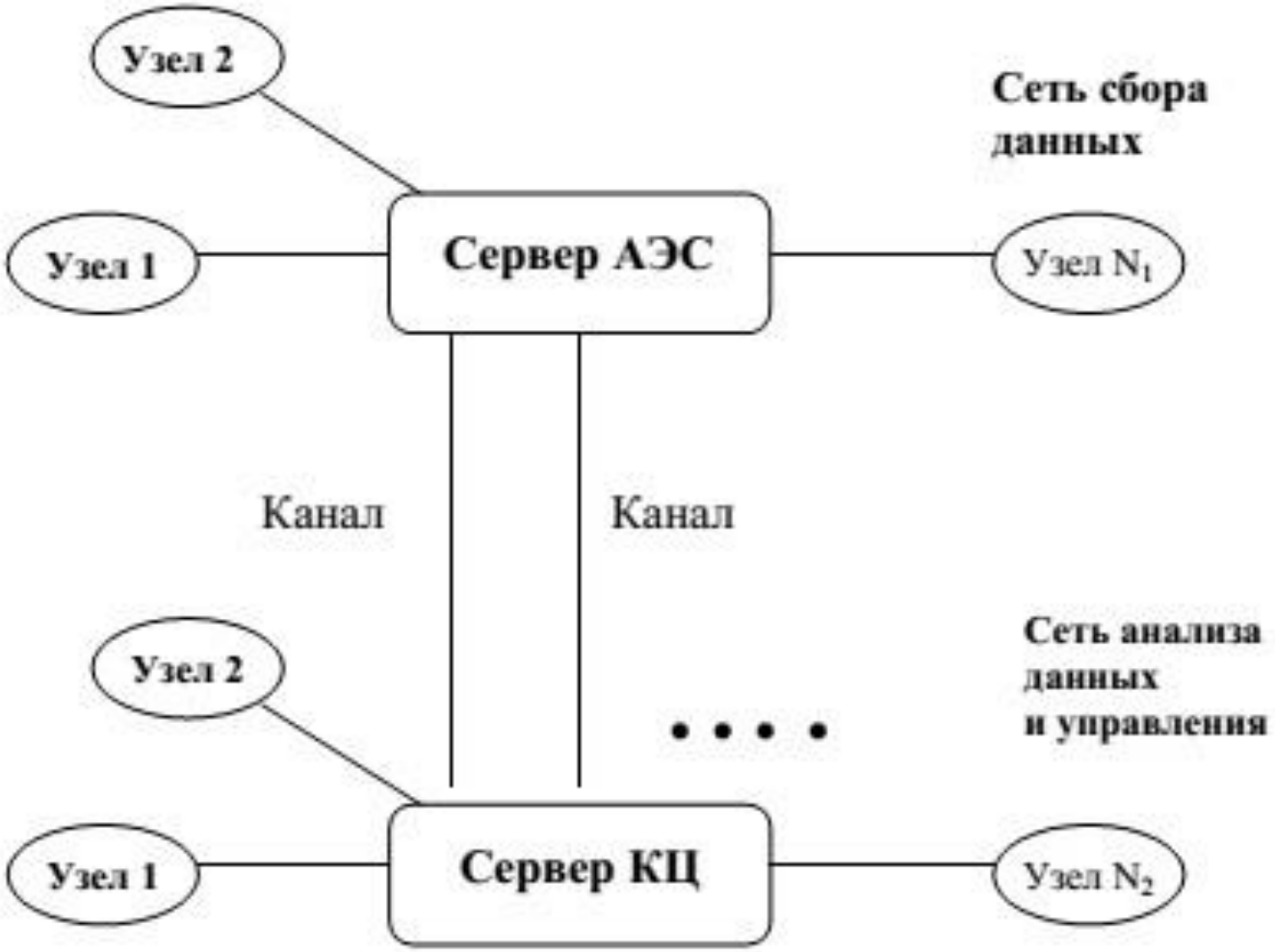


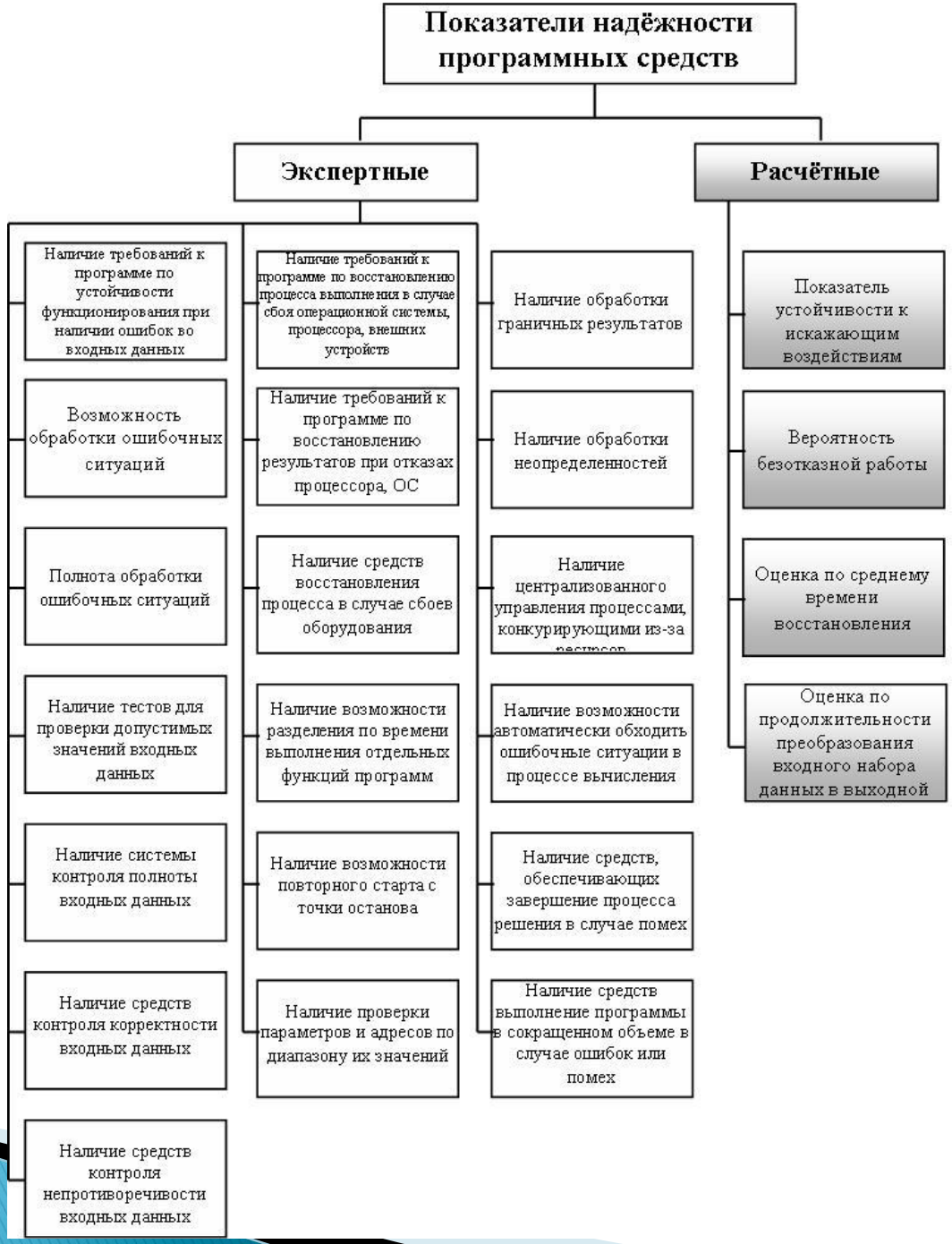
Схема сбора управляющей информации с АЭС

Надёжность программного обеспечения

Надёжность вычислительной системы (ВС) складывается из надёжности аппаратуры и надёжности используемого программного обеспечения. По некоторым оценкам, до 40% отказов в ВС приходится на отказы программного обеспечения.

Нормативные характеристики качества программных средств

ГОСТ 28806-90 "Качество программных средств. Термины и определения"		ГОСТ 28195-89. Оценка качества программных средств. Общие положения		ГОСТ Р ИСО/МЭК 9126-93 Оценка программной продукции. Характеристики качества и руководства по их применению	
Характеристики качества	**Подхарактеристики**	**Показатели качества**	**Комплексные показатели качества**	**Характеристики качества**	**Комплексные показатели качества**
Функциональность	- адекватность - правильность - комплексируемость - нормосоответствие - Защищенность	Показатели корректности	- полнота реализации - согласованность - логическая корректность - проверенность	Функциональные возможности	- пригодность - правильность - способность к взаимодействию - согласованность - защищенность
Надежность	- завершенность - отказоустойчивость - восстанавливаемость	**Показатели надежности**	- устойчивость функционирования - работоспособность	**Надежность**	- стабильность - устойчивость к ошибке - восстанавливаемость
Эффективность	- времяемкость - ресурсоемкость	Показатели эффективности	-уровень автоматизации -временная эффективность - ресурсоемкость	Практичность	- понятность - обучаемость - простота использования
Удобство использования	- понимаемость - осваиваемость - управляемость	Показатели удобства применения	- легкость освоения - доступность эксплуатационных программных документов - удобство эксплуатации и обслуживания	Эффективность	- характер изменения во времени - характер изменения ресурсов
Сопровождаемость	- анализируемость - модифицируемость - стабилизированность - тестируемость	Показатели сопровождения	- структурность -простота конструкции -наглядность -повторяемость	Сопровождаемость	- анализируемость - изменяемость - устойчивость - тестируемость
Мобильность	-адаптируемость - настраиваемость - заменоспособность	Показатели универсальности	- гибкость - мобильность - модифицируемость	Мобильность	- адаптируемость - простота внедрения - соответствие - взаимозаменяемость


Показатели надёжности программных средств
Экспертные
Расчётные
Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных
Возможность обработки ошибочных ситуаций
Полнота обработки ошибочных ситуаций
Наличие тестов для проверки допустимых значений входных данных
Наличие системы контроля полноты входных данных
Наличие средств контроля корректности входных данных
Наличие средств контроля непротиворечивости входных данных
Наличие требований к программе по восстановлению процесса выполнения в случае сбоя операционной системы, процессора, внешних устройств
Наличие требований к программе по восстановлению результатов при отказах процессора, ОС
Наличие средств восстановления процесса в случае сбоев оборудования
Наличие возможности разделения по времени выполнения отдельных функций программ
Наличие возможности повторного старта с точки останова
Наличие проверки параметров и адресов по диапазону их значений
Наличие обработки граничных результатов
Наличие обработки неопределенностей
Наличие централизованного управления процессами, конкурирующими из-за
Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления
Наличие средств, обеспечивающих завершение процесса решения в случае помех
Наличие средств выполнение программы в сокращенном объеме в случае ошибок или помех
Показатель устойчивости к искажающим воздействиям
Вероятность безотказной работы
Оценка по среднему времени восстановления
Оценка по продолжительности преобразования входного набора данных в выходной

ОШИБКИ В РАЗРАБОТКЕ ПРОГРАММНЫХ СИСТЕМ

Профессиональные программисты со стажем 10 и более лет на 1000 строк кода (СК) допускают в среднем 131,3 ошибки.

Из них до 50% выявляется на этапе компиляции (если транслятор имеет развитую систему предупреждений). На этапе тестирования отдельных модулей обнаруживается половина оставшихся ошибок.

Типичный американский проект имеет объем 50 тыс. СК. Его создают 5 программистов, делая при этом 100 ошибок на тысячу СК. 50% ошибок выявляется на этапе компиляции с незначительными расходами времени, устранение ошибок на этапе тестирования занимает 90% времени. Стоимость устранения одной ошибки в готовом продукте оценивается в 4 тыс. долл. (по данным IBM, одна ошибка в ее продуктах обходится в 20 тыс. долл.).

Институт программной инженерии SEI (США)

Для систем, прошедших тестирование, по различным оценкам содержится от 5 до 100 ошибок на 1000 строк исполняемого кода. По мере развития программного обеспечения в каждой его новой версии появляется все больше возможностей (и, соответственно, больший объем кода), и часто новая версия является менее надёжной, чем предыдущая. Показано, что число ошибок на 1000 строк кода стремится к стабилизации по мере роста числа выпущенных версий, но асимптотически этот показатель отличается от нуля.

Для ответственных применений, к которым можно отнести операционные системы и другое системное ПО, к моменту поставки системы клиенту в нем может содержаться 0,04...0,15 ошибки на 1000 строк кода программы, то есть в операционной системе из 5 млн строк кода содержится не менее 200 ошибок.

Для устранения ошибок на заключительных этапах тратится от 10 до 40 человеко-часов на ошибку, т. е. на доведение продукта до потребуется 2000 человеко-часов работы опытных программистов.

ПРИМЕРЫ ПРОГРАММНЫХ ОШИБОК

Ошибка	Причина
Ошибка при выполнении деления чисел с плавающей запятой в процессоре Pentium (1993 г.)	SRT-алгоритм деления использует справочную таблицу для определения частного. Значение в таблице генерируется численно и загружается в программируемый справочный массив. Дефект скрипта привел к тому, что несколько элементов в справочной таблице были пропущены. Ошибочное значение извлекалось, только если делитель содержал шесть последовательных бит, от 5-го до 10-го, установленных в единицу. Таким образом, ошибочные табличные значения могли не извлекаться при тестах, основанных на случайной выборке значения; тест, способный выявить ошибку, должен работать лучше, чем простая случайная выборка. **Тестирование должно не только следовать спецификации, оно для полноты также должно учитывать использованные алгоритмы**
Взрыв при своем первом полете ракеты Ariane 5 через 40 секунд после старта 4 июня 1996 года.	Необработанное исключение при преобразовании 64-битного значения с плавающей запятой в 16-битное целое значение со знаком. Значение с плавающей запятой, вызвавшее исключение, оказалось больше, чем значение, которое может быть представлено 16-битным целым. Программный модуль от Ariane 4 был повторно использован в новой среде, где **условия функционирования отличались от требований программного модуля. Эти требования не были пересмотрены.**
Потеря аппарата для исследования климата Марса (Mars Climate Orbiter) 23 сентября 1999 г.	Программное обеспечение разрабатывалось объединенной командой инженеров и ученых в двух местах. Однако, одна команда использовала метрические единицы, а вторая – английские. Это привело к различию между траекториями, вычисленными космическим аппаратом и наземной станцией, а именно параметры траектории, вычисленные наземной станцией, были в 4,45 раза меньше

Факторы, отличающие разработку модели надёжности ПО от разработки модели надёжности аппаратуры:

– не всегда удается разбить всё ПО на модули с указанием связей между ними,

– трудно определить, какие модули могут выступать в качестве резервных по отношению друг к другу,

– для определения последовательности вызовов программных модулей комплекса требуются специальные инструментальные средства, которые должны быть встроены в программный комплекс на этапе его проектирования,

– каждая программа представляет собой уникальное изделие, для которого, в отличие от аппаратуры, принципиально отсутствуют какие-либо заводские показатели надёжности типа наработки на отказ или интенсивности отказов,

– формулировка отказа системы представляет собой нетривиальную задачу, т.к. надёжность ПО оценивается способностью сохранять свой уровень качества функционирования, определить который не всегда просто;

– большое влияние на оценку надёжности ПО оказывают условия функционирования, в число которых входят помимо характеристик технических средств также и другие программные средства, установленные на данной машине;

– ПО следует рассматривать как систему, имеющую время восстановления, отличное от нуля, так как, в отличие от аппаратуры, здесь нельзя заменить один отказавший модуль на другой, заведомо исправный, а требуется восстановление отказавшего модуля с восстановлением также контекста задачи на момент отказа или ближайший к нему момент контрольной точки;

– программы, обеспечивающие дублирование выполняемых функций, в отличие от дублирующих аппаратных модулей, нельзя рассматривать как два полностью идентичных блока, имеющие одинаковые показатели потоков отказов и восстановлений. Каждая из них является самостоятельным изделием, с присущими только ей ошибками проектирования, которое характеризуются собственными потоками отказов.

Направление работ научной группы «Сигма» в области надёжности ПО

- методика оценки надёжности программ на ранних стадиях их проектирования;
- методы и средства оценки длительности процесса отладки ПО;
- оценка надёжности программного обеспечения, учитывающая особенности высоконадёжных программных комплексов: низкую интенсивность отказов, небольшую статистику по отказам.

МЕТОДЫ И СРЕДСТВА ОЦЕНКИ НАДЁЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА РАННИХ СТАДИЯХ ПРОЕКТИРОВАНИЯ


Модели надёжности ПО
Основанные на закономер-ностях проявления ошибок за заданный интервал времени
Джелинского-Моранды
Шумана
Коркорэна
Шика-Уолвертона
Шнейдевинда
Геометрическая
Дюэна
Распределения Вэйбулла
Простая интуитивная
Основанные на изучении структуры пространства входных данных
Нельсона
Анализ Вальда
Лападулы
Регрессивная IBM
Основанные на искусственном внесении в ПО известных ошибок
Миллса
Бэйзина
Простая эвристическая
Основанные на оценке сложности программы
Холстеда
Хейни
Линейная
Основанные на изучении структуры текста программы
Текстовая
Энтропийная
Салливена

Корреляция между числом ошибок, обнаруженных при тестировании отдельных модулей, и числом ошибок, найденных пользователями в готовом продукте, равна 0,91.

Следовательно, если на тестирование поступит некачественный продукт, он таким и будет выпущен в продажу. Поэтому надёжность ПО должна закладываться на ранних стадиях его жизненного цикла, а вопрос об оценке надёжности ПО на этапе его разработки является весьма актуальным.

Проведённый анализ существующих моделей надёжности ПО показал, что они ориентированы, в основном, на использование информации о работе программы, получаемой на этапе тестирования и отладки, и не позволяют оценить надёжность ПО на этапе разработки.

С точки зрения решения поставленной задачи наиболее интересной представляется метрика Холстеда, так как она позволяет по минимальным характеристикам программы, известным на раннем этапе её разработки, предсказать количество ошибок в ней.

Параметры программы, учитываемые в метрике Холстеда :

$\eta 1$ – число различных операторов;

$\eta 2$ – число различных операндов;

$\eta = \eta 1 + \eta 2$ – словарь программы;

$N1$ – число появлений операторов;

$N2$ – число появлений операндов.

$N = N1 + N2$ – длина программы.

В качестве меры сложности программы Холстед предложил использовать следующее выражение:

$$E = \frac{\eta_1 \cdot N_2 \cdot N \cdot \log_2 \eta}{2\,\eta_2}$$

Величина E соответствует усилию, затрачиваемому на кодирование и понимание программы.

Эксперименты показали, что число ошибок в неоттестированных программах пропорционально $E^{2/3}$, где E – мера сложности Холстеда:

$$E_0 = K \cdot E^{2/3}, K = \frac{1}{3200}$$

В прошедших стадию тестирования и отладки программах это отношение сохраняется, но коэффициент пропорциональности K принимает меньшие значения.

Рассмотрим оценку сложности по Холстеду программы со следующими характеристиками:

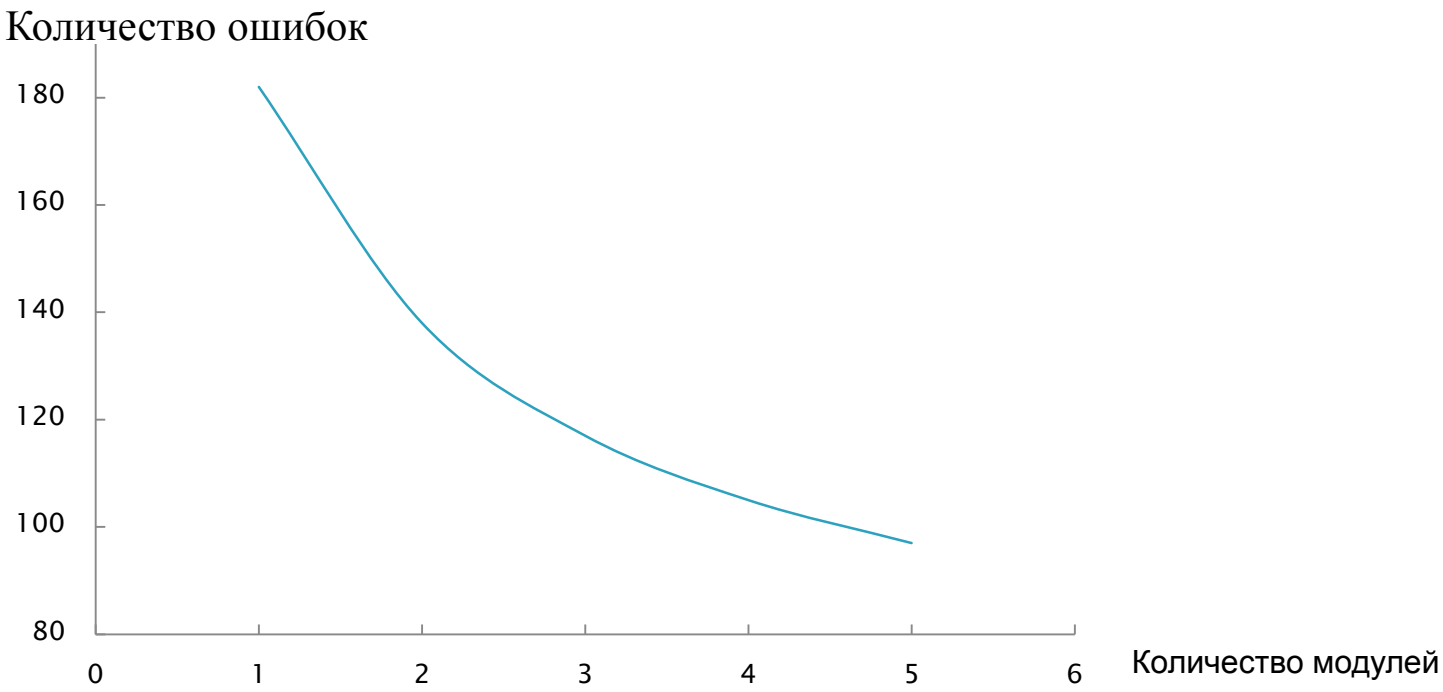
- всего различных операторов : 100;
- общая длина программы: 12 000 операторов ;
- количество операндов – 200;
- количество обращений к операндам 6 000 раз.

при её разбиении на различное число модуле.

Считаем, что количество обращение к операндам внутри модуля пропорционально длине модуля.

Кол-во модулей	Модуль	η_1 – число различных операторов	η_2 – число различных операндов	N_1 – число появлений операторов	N_2 – число появлений операндов	Кол-во ошибок в модуле	Кол-во ошибок в программе
1	M1	100	200	12000	6000	181,972	**182**
2	M1	100	100	6000	3000	68,751	**138**
	M2	100	100	6000	3000	68,751	
3	M1	100	50	4000	2000	38,577	**117**
	M2	100	100	4000	2000	40,040	
	M3	100	50	4000	2000	38,577	
3	M1	100	50	1000	2000	24,302	113
	M2	100	100	2000	2000	30,556	
	M3	100	50	9000	2000	57,786	
3	M1	100	50	500	2000	21,521	110
	M2	100	100	1000	2000	25,224	
	M3	100	50	10500	2000	62,927	
3	M1	100	50	500	2000	21,521	110
	M2	100	100	1000	2000	25,224	
	M3	100	50	10500	2000	62,927	
4	M1	100	50	3000	1500	26,287	106
	M2	100	100	3000	1500	27,284	
	M3	100	50	3000	1500	26,287	
	M4	100	50	3000	1500	26,287	
	M1	100	50	3000	1500	26,287	**105**
	M2	100	50	3000	1500	26,287	
	M3	100	50	3000	1500	26,287	
	M4	100	50	3000	1500	26,287	
5	M1	100	40	2400	1200	19,343	**97**
	M2	100	40	2400	1200	19,343	
	M3	100	40	2400	1200	19,343	
	M4	100	40	2400	1200	19,343	
	M5	100	40	2400	1200	19,343	

График зависимости общего количества ошибок в программе от числа модулей, на которое она разбита



В то же время общее количество ошибок в программе зависит от распределения операндов и операторов в пределах модулей гораздо слабее (см. таблицу для случая разбиения программы на 3 модуля).

Модель Майерса расчёта сложности программы на основе прочности отдельных модулей программы и сцепления между собой каждой пары модулей:

$$D_{ij} = \begin{cases} 0{,}15 \cdot (S_i + S_j) + 0{,}7 \cdot C_{ij}, & \text{если } C_{ij} \neq 0 \\ 0, & \text{если } C_{ij} = 0 \\ 1, & \text{если } i = j. \end{cases}$$

где D_{ij} - вероятность того, что модуль j придётся изменить, когда изменится модуль i, если рассматривать модули i и j вне контекста всей программы (отношение считается симметричным),

S_i и S_j - прочность этих модулей i и j,

C_{ij} – сцепление модулей i и j.

Вероятность того, что никакой другой модуль не изменится при изменении какого-либо модуля программы, может быть взята в качестве оценки при сравнении надёжности различных конфигураций программной системы, т.к. известно, что при изменении каких-либо модулей довольно высока вероятность внесения в программу новой ошибки (от 20% до 50%).

Здесь же можно принять во внимание и ещё одну интересную особенность, которая была найдена в IBM для оценки числа ошибок в выпусках OS/360 и ее преемников. Исследование данных об ошибках для всех компонент OS/360 показало исключительно хорошее соответствие уравнению:

$$ИЗМ=2\cdot ИМ+23\cdot МИМ,$$

где ИЗМ — полное число исправлений из-за ошибок,

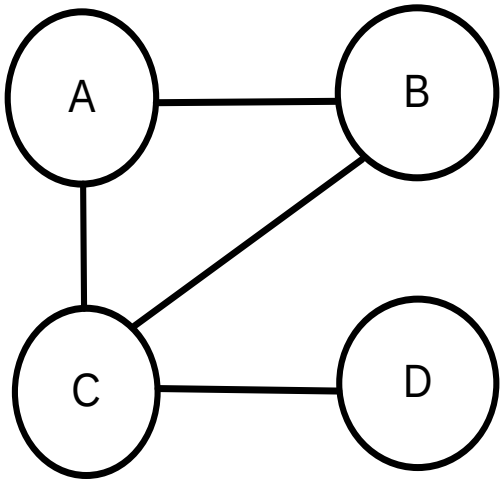
ИМ — количество исправляемых модулей,

МИМ — число многократно исправляемых модулей, которые потребовали 10 и более исправлений.

Величина сцепления изменяется в диапазоне от 0 (сцепление отсутствует) до 1 (сильное сцепление) и определяется на основе экспертных оценок).

Количество ошибок в программе максимально при отсутствии её разбиения на модули. Поэтому прочность модуля можно оценивать как N_i/N, где N – количество ошибок в программе без её разбиения на модули, а N_i – количество ошибок в i-м модуле.

ПРИМЕР РАСЧЁТА



Структура программы, разбитой на 4 модуля

	A	**B**	**C**	**D**
A	1	0,5	0,4	0
B	0,5	1	0,7	0
C	0,4	0,7	1	0,6
D	0	0	0,6	1

Сцепление между модулями программы

Прочность каждого модуля определяем, исходя из данных, представленных в таблице 1:

$$s_a = s_b = s_c = s_d = N_1 | N = 0, 144$$

Полная матрица зависимостей для представленной программы с учётом всех путей для каждой пары модулей

	A	**B**	**C**	**D**
A	1	0,4978	0,4651	0,2323
B	0,4978	1	0,5925	0,2912
C	0,4651	0,5925	1	0,4632
D	0,2323	0,2912	0,4632	1

По полученной матрице можно определить, например, что при изменении модуля A, вероятность того, что изменится модуль B, равна 0,4978.

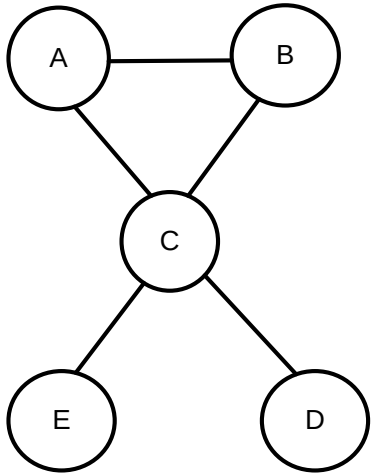
Вероятность того, что ни один модуль не изменится при изменении модуля A, составит:

$$Pa = (1 - 0{,}4978)\cdot(1 - 0{,}4651)\cdot(1 - 0{,}2323) \approx 0{,}2062$$

Вероятность того, что никакой другой модуль не изменится при изменении какого-либо модуля, составит:

$$P \approx 0{,}0320$$

Возможная структура программы, разбитой на 5 модулей



Сцепление между модулями программы

	A	**B**	**C**	**D**	**E**
A	1	0,5	0,4	0	0
B	0,5	1	0,7	0	0
C	0,4	0,7	1	0,6	0,6
D	0	0	0,6	1	0
E	0	0	0	0,6	1

Полная матрица зависимостей для программы из 5 модулей

	A	**B**	**C**	**D**	**E**
A	1	0,4838	0,4501	0,2191	0,2194
B	0,4838	1	0,5800	0,2779	0,2783
C	0,4501	0,5800	1	0,4524	0,4530
D	0,2191	0,2779	0,4524	1	0,2049
E	0,2194	0,2783	0,4530	0,2049	1

Для этой структуры ПО вероятность того, что никакой другой модуль не изменится при изменении какого-либо модуля, составит:

$$P = 0{,}0090$$

Таким образом, по предложенному критерию рассмотренное разбиение программы на 4 модуля оказывается предпочтительнее её разбиения на 5 модулей с указанной связностью между модулями и распределением количества команд по модулям.

Таким образом, по предложенному критерию рассмотренное разбиение программы на 4 модуля оказывается предпочтительнее её разбиения на 5 модулей с указанной связностью между модулями и распределением количества команд по модулям.