

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное  
учреждение высшего образования  
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Высшая технологическая школа  
«Севастопольский приборостроительный институт»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра информационных технологий и систем

## **КРИТИЧЕСКИЙ КОМПЬЮТИНГ**

**Методические указания**  
к лабораторным работам по дисциплине  
«Критический компьютеринг»  
для студентов направлений подготовки  
09.03.01 – Информатика и вычислительная техника,  
09.03.04 – Программная инженерия  
очной формы обучения

*В двух частях*

Часть 1

Севастополь  
СевГУ  
2026

УДК 004.422.8:004.052(076.5)  
ББК 32.972.1-02я73  
К826

**Р е ц е н з е н т:**

**А. А. Брюховецкий** - канд. техн. наук,  
доцент кафедры информационных систем

*Составители:* Ю. В. Доронина, А. В. Пестрякова

**К826 Критический компьютеринг:** методические указания к лабораторным работам по дисциплине «Критический компьютеринг» для студентов направлений подготовки 09.03.01 – Информатика и вычислительная техника, 09.03.04 – Программная инженерия очной формы обучения. – В 2-х частях. Ч. 1 / Севастопольский государственный университет, факультет Информационных технологий, кафедра информационных технологий и систем ; сост.: Ю. В. Доронина, А. В. Пестрякова. – Севастополь : СевГУ, 2026. – 43 с. – Текст : электронный.

Целью методических указаний является оказание помощи в подготовке к лабораторным работам по дисциплине «Критический компьютеринг», предусматривающим разработку и выполнение различных задач по критическому компьютерингу. Методические указания предназначены для студентов четвертого курса очной формы обучения направлений подготовки 09.03.01 – Информатика и вычислительная техника, 09.03.04 – Программная инженерия.

УДК 004.422.8:004.052(076.5)  
ББК 32.972.1-02я73

Методические указания рассмотрены и утверждены на заседании кафедры информационных технологий и систем, протокол № 4 от 11 ноября 2025 г.

© Доронина Ю. В., Пестрякова А. В., сост., 2026  
© ФГАОУ ВО «Севастопольский  
государственный университет», 2026

## **СОДЕРЖАНИЕ**

|                                                                                                                                                                        |    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ<br>ЛАБОРАТОРНЫХ РАБОТ .....                                                                                                           | 4  |
| 2. Лабораторная работа № 1. ИССЛЕДОВАНИЕ МОДЕЛЕЙ ЖИЗНЕННОГО<br>ЦИКЛА И ТЕХНОЛОГИЙ РАЗРАБОТКИ МНОГОВЕРСИОННОГО<br>ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (КРИТИЧЕСКИЙ КОМПЬЮТИНГ).... | 5  |
| 3. Лабораторная работа № 2. АНАЛИЗ МЕТРИК ДИВЕРСНОСТИ И РАСЧЕТ<br>ПОКАЗАТЕЛЕЙ НАДЕЖНОСТИ МНОГОВЕРСИОННОГО<br>ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ .....                            | 17 |
| 4. Лабораторная работа № 3. ОЦЕНКА ВЕРСИЙ СИСТЕМ КРИТИЧЕСКОГО<br>ПРИМЕНЕНИЯ ПУТЕМ АНАЛИЗА МЕТРИК РАЗЛИЧИЯ .....                                                        | 28 |
| 5. Лабораторная работа № 4. ИССЛЕДОВАНИЕ АСПЕКТОВ<br>КОЛИЧЕСТВЕННОГО АНАЛИЗА КОМПЛЕКСА ТЕХНИЧЕСКИХ СРЕДСТВ<br>КРИТИЧЕСКОГО ПРИМЕНЕНИЯ.....                             | 35 |
| БИБЛИОГРАФИЧЕСКИЙ СПИСОК.....                                                                                                                                          | 42 |

## 1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Каждая работа требует изучения методических указаний и рекомендованной литературы, подготовку текста программы, компиляцию программы некоторое ее исследование.

Все действия, указанные в методических указаниях к отдельной работе обязательно должны быть выполнены до следующего лабораторного занятия.

По выполненной лабораторной работе составляется отчет. Оформление отчета должно соответствовать требованиям к текстовым учебным документам соответствующих ГОСТов [Электронный фонд правовой и нормативно-технической документации <http://docs.cntd.ru>].

Отчет представляется на листах формата А4 и в электронном виде (файл Microsoft Word). Первый лист отчета – титульный. На нем указывается название вуза, название выпускающей кафедры, название работы, ФИО и группа исполнителя, ФИО принимающего работу, город (Севастополь) и год.

Студент допускается к защите отчета по лабораторной работе после того, как он продемонстрирует преподавателю выполнение поставленной задачи на компьютере.

Защита лабораторных работ состоит из доклада студента о проделанной работе с объяснением содержания отчета. Студент должен ответить на контрольные вопросы к соответствующей лабораторной работе, приведенные в методических указаниях, а также другие вопросы преподавателя, касающиеся поставленной задачи, показать работоспособность подготовленной программы и навыки работы с программной системой. При защите текущей работы возможны вопросы по темам предыдущих лабораторных работ. Результат защиты оценивается по шкале 0 – 100 баллов (ESTC).

Рекомендуется иметь съемный носитель информации (флэш-диск), на котором будут храниться проекты лабораторных работ и соответствующие отчеты.

## 2. Лабораторная работа № 1

### ИССЛЕДОВАНИЕ МОДЕЛЕЙ ЖИЗНЕННОГО ЦИКЛА И ТЕХНОЛОГИЙ РАЗРАБОТКИ МНОГОВЕРСИОННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (КРИТИЧЕСКИЙ КОМПЬЮТИНГ)

**Цель работы:** изучение типов моделей жизненного цикла и технологий разработки программного обеспечения (ПО) и исследование их применимости к созданию многоверсионного ПО.

#### **Задание на лабораторную работу.**

В соответствии с вариантом задания построить алгоритм и функциональную модель процесса создания предложенной системы и обосновать применимость выбранных моделей ЖЦ с учетом возможной критичности компонент системы.

#### **Порядок выполнения лабораторной работы.**

1. Выбрать вариант системы по списку в группе.
2. Построить BPMN – модель функционирования системы с описанием основных процессов.
3. Обозначить (выноской, стрелкой и т.п.) **возможную критичность**<sup>1</sup> компонент системы, исходя из варианта задания.
4. Изучить типы моделей жизненного цикла и технологий разработки программного обеспечения сложных систем.
5. Исследовать применимость той или иной модели ЖЦ для создания предложенной системы.
6. На основе выбранной или выбранных моделей ЖЦ построить WFD – модель создания системы с описанием основных процессов с дополнительными подпроцессами, учитывающими тестирование или дублирование компонент для учета критичности.
7. Подготовить и защитить отчет.

#### **Содержание части отчета по лабораторной работе**

1. Тема, вариант, цель.
2. Краткое описание применимости той или иной модели ЖЦ для создания предложенной системы.
3. Две схемы: BPMN и WFD с учетом выбранной модель ЖЦ.
4. Выводы.

---

<sup>1</sup> **Возможная критичность** в простейшем случае может быть рассмотрена на 3-х уровнях системы: аппаратном (техническое обеспечение), программном (программное обеспечение) и методологическом (модели, методы, результаты анализа данных).

## 2.1. Теоретические сведения о ЖЦ КС и ПО

**Жизненный цикл (ЖЦ) компьютерной системы (КС)** – непрерывный процесс, начинающийся с момента принятия решения о создании КС и заканчивающийся в момент полного изъятия ее из эксплуатации.

**Модель жизненного цикла** – структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, которые осуществляются в ходе разработки, функционирования и сопровождения информационной системы в течение всей жизни системы, от определения требований до завершения ее использования.

Модель ЖЦ включает в себя:

- стадии;
- результаты выполнения работ на каждой стадии;
- ключевые события – точки завершения работ и принятия решений.

### 2.1.1. Модель кодирования и устранения ошибок («модель проб и ошибок»)

Модель кодирования и устранения ошибок – самая простая модель жизненного цикла, алгоритм которой состоит из следующих шагов, рисунок 2.1:

- постановка задачи;
- создание программы;
- тестирование;
- анализ результата тестирования и возможный переход к постановке задачи.

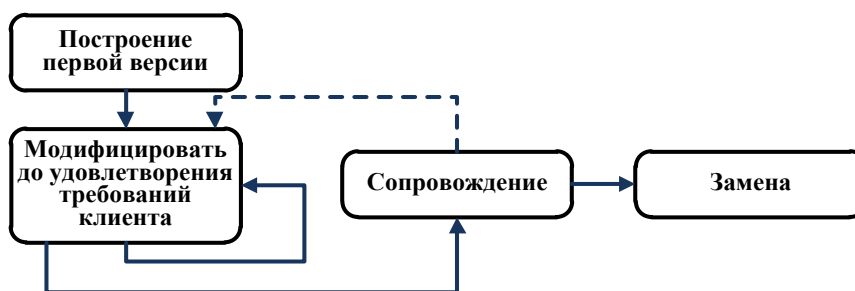


Рисунок 2.1 – Модель кодирования и устранения ошибок

Эта модель не актуальна при профессиональной разработке программного обеспечения. Модель (Build-and-fix) хороша для небольших проектов, которые не требуют сопровождения, но абсолютно непригодна для корпоративных или вообще нетривиальных проектов объемом более 1000 строк.

### 2.1.2. Каскадная модель («водопад»)

Модель предполагает разделение процесса создания программного продукта на последовательные этапы (применялась различными

разработчиками, однако ни количество, ни содержание этапов не унифицируется). Переход с одной стадии на следующую происходит только после того, как будет полностью завершена работа на текущей.

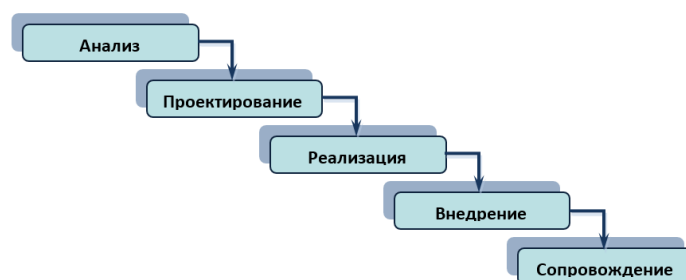


Рисунок 2.2 – Каскадная (водопадная, классическая) модель жизненного цикла

Данная модель применяется при разработке информационных систем, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования.

**Достоинства** модели:

- на каждой стадии формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в четкой последовательности стадии позволяют уверенно планировать сроки выполнения работ и соответствующие ресурсы (денежные, материальные и людские).

**Недостатки** модели:

- реальный процесс разработки КС редко полностью укладывается в такую жесткую схему. Особенно это относится к разработке нетиповых и новаторских систем;
- жизненный цикл основан на точной формулировке исходных требований к информационной системе. Реально в начале проекта требования заказчика определены лишь частично;
- основной недостаток – результаты разработки доступны заказчику только в конце проекта. В случае неточного изложения требований или их изменения в течение длительного периода создания ИС заказчик получает систему, не удовлетворяющую его потребностям.

### 2.1.3. V-образная (шарнирная) модель

V-образная модель была предложена для того, чтобы устранить недостатки каскадной модели, а название – V-образная, или шарнирная – появилось из-за ее специфического графического представления (рис. 2.3).

V-образная модель дала возможность значительно повысить качество ПО за счет своей ориентации на тестирование, а также во многом разрешила проблему соответствия созданного продукта выдвигаемым требованиям благодаря процедурам верификации и аттестации на ранних стадиях разработки (пунктирные линии на рисунке указывают на зависимость этапов планирования/постановки задачи и тестирования/приемки).

В целом V-образная модель является модификацией каскадной и обладает многими ее недостатками: и та и другая слабо приспособлены к возможным изменениям требований заказчика.



Рисунок 2.3 – V-диаграмма жизненного цикла

#### 2.1.4. Инкрементная модель (поступательная, многопроходная)

Инкрементная стратегия подразумевает разработку информационной системы с линейной последовательностью стадий, но в несколько инкрементов (версий), т.е. с запланированным улучшением продукта, рис. 2.4.

**Инкрементная разработка представляет собой процесс частичной реализации всей системы и постепенного наращивания функциональных возможностей.**

В начале работы над проектом определяются все основные требования к системе, после чего выполняется ее разработка в виде последовательности версий. При этом каждая версия является законченным и работоспособным продуктом. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т.д., пока не будет получена полная система.

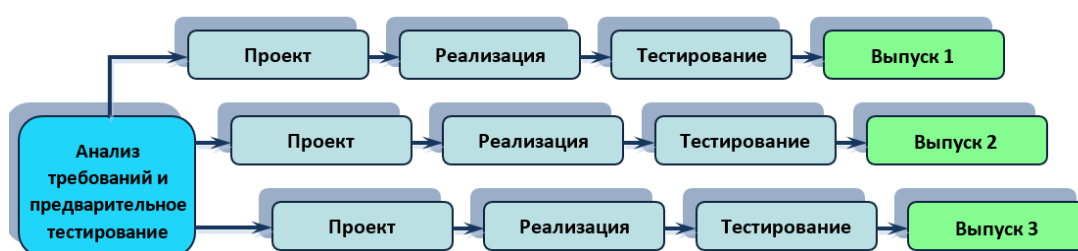


Рисунок 2.4 – Инкрементная модель жизненного цикла

Эта модель жизненного цикла применяется при разработке сложных и комплексных систем, для которых имеется четкое видение (как со стороны заказчика, так и со стороны разработчика) того, что собой должен представлять конечный результат (КС). Разработка версиями ведется в силу разного рода причин:

- отсутствия у заказчика возможности сразу профинансировать весь дорогостоящий проект;
- отсутствия у разработчика необходимых ресурсов для реализации сложного проекта в сжатые сроки;
- требований поэтапного внедрения и освоения продукта конечными пользователями. Внедрение всей системы сразу может вызвать у ее пользователей неприятие и только «затормозить» процесс перехода на новые технологии.

*Достоинства* и *недостатки* этой стратегии такие же, как и у классической. Но в отличие от классической стратегии заказчик может раньше увидеть результаты. Уже по результатам разработки и внедрения первой версии он может незначительно изменить требования к разработке, отказаться от нее или предложить разработку более совершенного продукта с заключением нового договора.

### **2.1.5. Спиральная (эволюционная, итерационная) модель**

Эта модель жизненного цикла применяется при разработке новаторских (нетиповых) систем. В начале работы над проектом у заказчика и разработчика нет четкого видения итогового продукта (требования не могут быть четко определены) или стопроцентной уверенности в успешной реализации проекта (риски очень велики). В связи с этим принимается решение разработки системы по частям с возможностью изменения требований или отказа от ее дальнейшего развития.

*Достоинства* модели:

- позволяет быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований;
- допускает изменение требований при разработке информационной системы, что характерно для большинства разработок, в том числе и типовых;
- обеспечивает большую гибкость в управлении проектом;
- позволяет получить более надежную и устойчивую систему. По мере развития системы ошибки и слабые места обнаруживаются и исправляются на каждой итерации;
- позволяет совершенствовать процесс разработки – анализ, проводимый в каждой итерации, позволяет проводить оценку того, что должно быть изменено в организации разработки, и улучшить ее на следующей итерации;
- уменьшаются риски заказчика. Заказчик может с минимальными для себя финансовыми потерями завершить развитие неперспективного проекта.

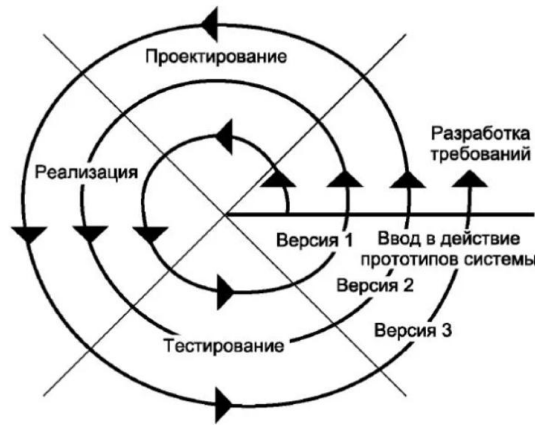


Рисунок 2.5 – Спиральная (эволюционная, итерационная) модель жизненного цикла

**Недостатки** модели:

- увеличивается неопределенность у разработчика в перспективах развития проекта. Этот недостаток вытекает из предыдущего достоинства модели;
- затруднены операции временного и ресурсного планирования всего проекта в целом. Для решения этой проблемы необходимо ввести временные ограничения на каждую из стадий жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа выполнена. План составляется на основе статистических данных, полученных в предыдущих проектах и личного опыта разработчиков.

### 2.1.6. Модифицированные модели жизненного цикла информационной системы

**Поэтапная модель с промежуточным контролем («водоворот»).**

Доработка каскадной модели с учетом взаимозависимости этапов и необходимости возврата на предыдущие ступени, что может быть вызвано, например, неполнотой требований или ошибками в формировании задания (рис. 2.6).



Рисунок 2.6 – Модифицированная каскадная модель (поэтапная модель жизненного цикла с промежуточным контролем, «водоворот»)

Результатом каждой фазы является один или несколько документов, которые утверждаются. Следующая фаза не должна начинаться до того, пока предыдущая не завершена.

Практическое использование данной модели выявило множество ее недостатков, главный из которых состоял в том, что она больше подходит для традиционных видов инженерной деятельности, чем для разработки ПО.

Одной из самых больших проблем оказалась ее «предрасположенность» к возможным несоответствиям полученного в результате продукта и требований, которые к нему предъявлялись. При этом полностью сформированный продукт появляется лишь на поздних этапах разработки, но так как работу на разных этапах обычно выполняли различные специалисты и проект передавался от одной группы к другой, то по принципу испорченного телефона оказывалось так, что на выходе получалось не совсем то, что предполагалось вначале.

### **2.1.7. Модель быстрой разработки приложений RAD**

Один из подходов к разработке ПО в рамках спиральной модели ЖЦ – получившая широкое распространение методология (технология) быстрой разработки приложений RAD (Rapid Application Development). Данная модель очень хорошо подходит к разработке учебных программ, т.к. включает в себя три составляющие:

- небольшую команду программистов (от 2 до 10 человек);
- короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
- повторяющийся цикл, при котором разработчики по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.

Команда разработчиков должна представлять собой группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ПО с использованием CASE-средств, способных хорошо взаимодействовать с конечными пользователями и трансформировать их предложения в рабочие прототипы. Жизненный цикл ПО по методологии RAD состоит из четырёх фаз:

- анализа и планирования требований;
- проектирования;
- построения;
- внедрения.

Технология неприменима для разработки операционных систем; сложных расчетных программ с большим объемом программного кода и сложными уникальными алгоритмами управления; приложений, в которых отсутствует ярко выраженная интерфейсная часть, наглядно определяющая логику работы системы (приложения реального времени), так как итерационный подход предполагает, что несколько первых версий не будут полностью соответствовать требованиям.

### **Основные принципы** технологии RAD:

- разработка приложений итерациями;
- необязательность полного завершения работ на каждом этапе ЖЦ;
- обязательное вовлечение пользователей на этапе разработки;
- использование прототипирования, позволяющего выяснить и удовлетворить все требования конечного пользователя;
- тестирование и развитие проекта одновременно с разработкой;
- грамотное руководство разработкой, четкое планирование и контроль выполнения работ.

#### **2.1.9. Спиральная модель «Win-Win»**

Спиральная модель «Win-Win» содержит в себе больше фаз, в которых внимание сконцентрировано на участии заказчика в процессе разработки. Это достигается путем добавления к начальной фазе каждого цикла так называемых действий Теории W (Theory W activities). Теория W – это принцип управления, при реализации которого особое значение придается ключевым организаторам совместного дела, выполняющим разработку системы (пользователь, заказчик, разработчик, наладчик, создатель интерфейсов и т.д.), которые станут «победителями», если проект окажется успешным.

В этом методе, основанном на постоянном согласовании, циклы состоят из следующих фаз или стадий:

- определение участников следующего уровня;
- определение условий, необходимых для одержания участниками победы;
- согласование «победных» условий;
- формулирование целей, ограничений и альтернативных вариантов следующего уровня;
- оценка альтернативных вариантов на уровне продукта и процесса, решение рисков;
- определение следующего уровня продукта и процесса, включая сегментацию;
- обоснование определений продукта и процесса;
- обзор и комментарии.

Важной стадией является последующее планирование следующего цикла и обновление плана жизненного цикла, включая разделение системы на подсистемы, разработка которых осуществляется в ходе выполнения параллельных циклов. Эта стадия может включать в себя план прекращения проекта, если продолжение работы является слишком рискованным или невозможным.



Рисунок 2.8 – Модель жизненного цикла ПО win-win

Спиральная модель «Win-Win» имеет следующие *преимущества*:

- более быстрая разработка ПО благодаря содействию, оказываемому участниками проекта;
- уменьшение стоимости программ благодаря уменьшению объема переработок и текущего сопровождения;
- более высокий уровень удовлетворения со стороны участников проекта, достигаемого до разработки самого продукта;
- более высокое качество ПО благодаря использованию компромиссных качественно-атрибутивных моделей на уровне архитектуры;
- исследование большого количества вариантов построения архитектуры на ранних этапах разработки.

## 2.2. Пример выполнения работы

Задание: система автоматизированного обслуживания туристов в турагентстве.

### 2.2.1. Анализ процессов в нотации BPMN

Рассмотрим возможную критичность подпроцессов, представленных на рис. 2.9.

Согласно определению **возможной критичности в простейшем случае, которая может быть рассмотрена на 3-х уровнях системы: аппаратном (техническое обеспечение), программном (программное обеспечение) и методологическом (модели, методы, результаты анализа данных)**, в рассматриваемой системе можно отметить отсутствие признаков требований к системам высокой готовности, систем обеспечения инфраструктурной критичности и т.п.

Автоматизации текущих процессов не накладывает обязательств по отношению к аппаратным платформам (например, их защите или дублированию); не накладывает обязательств к дублированию ПО, методологий и алгоритмов, лежащих в основе ПО. В связи с этим, будем учитывать локальную критичность процессов предприятия.

Это означает, что требуется обеспечить эффективное взаимодействие клиента с системой (критичность интерактивного взаимодействия) (красная стрелка 1); критичность по отношению к БД, формирование списка требований в системе (красные стрелки 2, 3). Эти точки локальных «критичных» процессов должны быть учтены при проектировании этой системы в виде: дополнительного тестирования интерфейсов с клиентом (красная стрелка 1); тестирования и защиты БД (красные стрелки 2, 3).

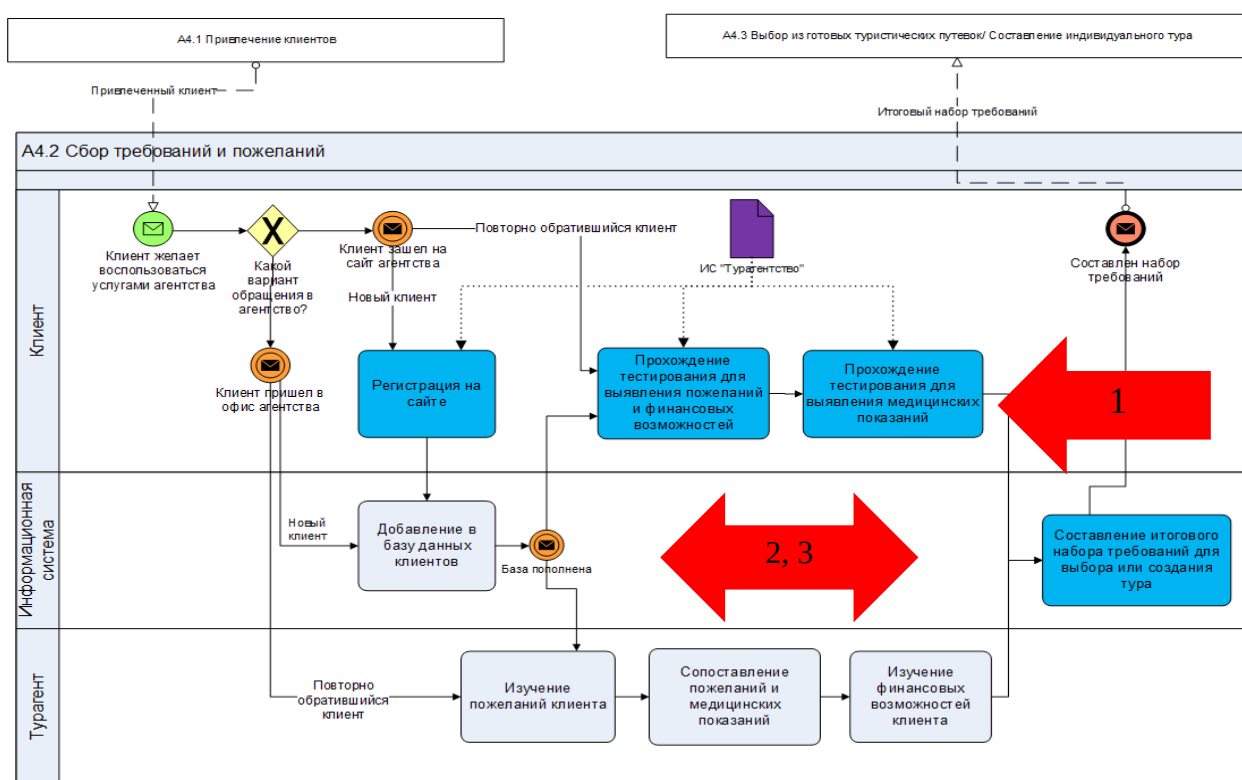


Рисунок 2.9 – Процессы автоматизированной регистрации туристов в турагентстве

### 2.2.2. Обоснование выбора ЖЦ для создания системы автоматизированного обслуживания туристов в турагентстве (САОТ)

Разрабатываемая система представляет собой интерактивную веб-ориентированную систему с базой данных. Проектирование такой системы возможно на основе гибридного ЖЦ, например, водопадно-спирального. Водопадная (последовательная часть ЖЦ будет применена для реализации подпроцессов: составлении ТЗ на САОТ, передачи заказчику после тестирования и т.п.); спиральная часть ЖЦ – при создании ПО, схематически это отражено на рис. 2.10.

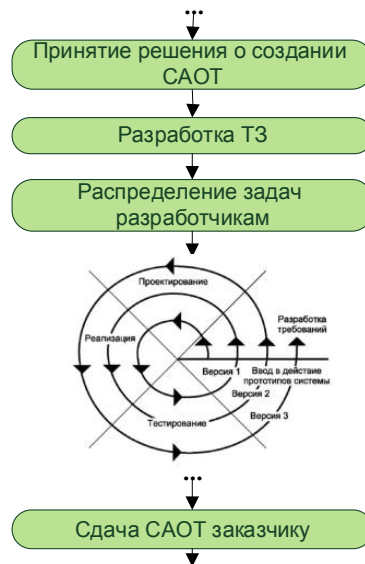


Рисунок 2.10 – Укрупненная схема гибридного ЖЦ при создании прикладной САОТ

2.11. Подробная схема гибридного ЖЦ для создания САОТ приведена на рис.

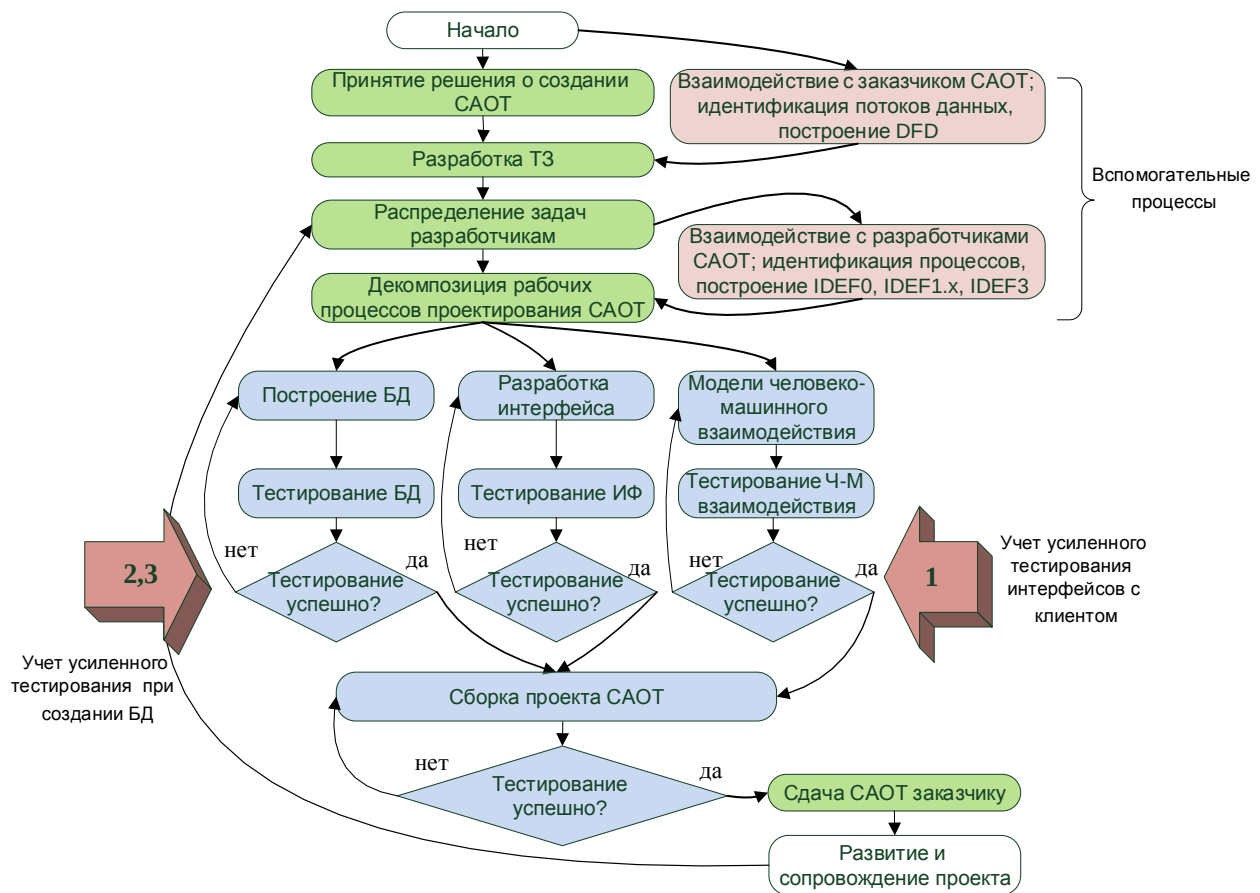


Рисунок 2.11 – Схема гибридного ЖЦ при создании прикладной САОТ

**Варианты заданий для выполнения лабораторной работы.**

0 – Система организационного управления предприятием выпуска микропроцессорной техники в условиях интенсивного изменения планов производства.

1 – ERP-система (Enterprise Resource Planning System – система планирования ресурсов предприятия) – корпоративная информационная система (КИС), предназначенная для автоматизации учёта и управления.

2 – Автоматизированная система управления критической инфраструктурой объекта с атаками вирусного типа.

3 – Система поддержки принятия решений по обеспечению безопасности беспилотного транспорта.

4 – Интеллектуальная система распознавания лиц и формирования оптимальных траекторий движения спортсменов в командных играх.

5 – Система формирования динамической библиотеки видео-продукции: выбор сюжетов, компоновка одного фильма из нескольких сюжетов, пополнение библиотеки, классификация.

6 – Система управления взаимоотношениями с клиентами – прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами), в частности для повышения уровня продаж (CRM- система).

7 – Интенсивно пополняемая web- ориентированная база данных интернет-магазина с активным противодействием конкурентов.

8 – Гетерогенная система управления концерном предприятий-смежников по выпуску сложной техники.

9 – Система управления IT-проектами по созданию игровых приложений.

### 3. Лабораторная работа № 2

## АНАЛИЗ МЕТРИК ДИВЕРСНОСТИ И РАСЧЕТ ПОКАЗАТЕЛЕЙ НАДЕЖНОСТИ МНОГОВЕРСИОННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

#### Цели и задачи работы.

Многоверсионное программное обеспечение (МВС) применяется для повышения надежности и отказоустойчивости критических вычислительных систем путем использования нескольких независимых версий программ. Основной принцип – снижение риска одновременных отказов за счет диверсности, то есть отличий между версиями.

Основные задачи лабораторной работы:

- Ознакомиться с понятием МВС и диверсности;
- Научиться вычислять метрики диверсности между версиями ПО;
- Рассчитать показатели надежности МВС на основе математических моделей;
- Провести анализ влияния диверсности на надежность сложных систем в различных предметных областях (по вариантам).

### 3.1. Теоретические сведения

#### 3.1.1. Определение многоверсионного программного обеспечения (МВС)

МВС – это система, в которой реализовано несколько версий одного и того же программного продукта. Они запускаются параллельно или поочередно для повышения устойчивости к ошибкам и сбоям.

Диверсность – степень различия между этими версиями, измеряемая с помощью специфических метрик. Чем выше диверсность, тем ниже вероятность одновременного отказа.

#### 3.1.2. Модель надежности многоверсионных систем

Модель (n,m) – много версий, где из n версий минимум m должны работать правильно для корректной работы всей системы. Надежность системы – это вероятность того, что это условие выполнится, рассчитывается по формуле

$$P = \sum_{k=m}^n C_k^n p^k (1-p)^{n-k},$$

где  $p$  – вероятность безотказной работы одной версии.

Формула надежности системы с учетом скорректированной вероятности исправной работы:

$$P = \sum_{k=m}^n C_k^n (p_{adj})^k (1-p_{adj})^{n-k},$$

где  $n$  – число версий,  $m$  – минимально необходимое число исправно работающих версий,  $p_{adj}$  – скорректированная вероятность исправной работы одной версии.

Важна скорректированная вероятность  $p_{adj}$ , зависящая от диверсности, вычисляемой как

$$p_{adj} = \min(p \times \text{div\_factor}, 1)$$

где  $\text{div\_factor}$  – коэффициент, связанный с средней дистанцией Хэмминга  $\text{avg\_dist}$ , например:

$$\text{div\_factor} = 0.5 + 0.5 \times \frac{\text{avg\_dist}}{\text{max\_dist}}$$

Теперь, когда мы рассматриваем график вероятности  $P$  как функцию диверсности, мы меняем  $\text{avg\_dist}$  от 0 до максимума, а значит меняем  $p_{adj}$ , и, следовательно, меняется  $P$ .

Это значит, что на оси  $x$  (диверсность/среднее расстояние) мы имеем изменяющееся число  $p_{adj}$ , а на оси  $y$  итоговую надежность  $P$ , которая по формуле биномиального распределения зависит от  $p_{adj}$ .

Таким образом, хотя формула для одной фиксированной версии и параметров даёт число, график строится при варьировании параметра диверсности, что делает его функцией и даёт линию/кривую, а не одиночное число.

Рассмотрим, что такое базовая и скорректированная вероятность надежности:

1. **Базовая вероятность** – это исходная оценка надежности одной версии без учета различий между версиями, заданная как фиксированное значение.

2. **Скорректированная вероятность** – та же вероятность, но с учетом влияния **средней диверсности (различий)** между версиями. Модель обычно уменьшает оценку надежности.

Разница между этими вероятностями оказывает, что версии в системе недостаточно дивергентны, и их ошибки вероятно частично совпадают. Если бы версии были сильно различны (высокая дистанция Хэмминга), то скорректированная вероятность была бы равна или даже выше базовой, отражая низкую корреляцию ошибок между версиями.

– Если версии недостаточно дивергентны (низкая дистанция Хэмминга, мало отличий), вероятные ошибки и отказы сильно коррелируют, поэтому скорректированная вероятность может быть ниже базовой - это свидетельствует о высоком риске одновременных отказов.

– При высокой диверсности (большой дистанции Хэмминга) корреляция ошибок низкая, ошибки одних версий не совпадают с другими, и скорректированная вероятность может быть равной или даже выше базовой, что отражает большую надежность.

– В итоге, чем выше диверсность, тем больше надежность всей системы, так как вероятность одновременного отказа всех версий снижается.

**Почему надежность одной версии выше, чем всей системы?**

– Версия обычно оценивается по отдельности, без учета возможностей резервирования.

– Общая надежность системы, которая требует, чтобы минимум  $m$  версий работали, обычно выше (при правильной настройке модели и наличии диверсности), поскольку система считается работоспособной даже при отказе некоторых версий.

– Отрицательная ситуация – когда версии имеют очень низкую диверсность, то вероятность совместного отказа возрастает, и надежность системы может оказаться ниже отдельной версии, что указывает на плохую диверсность.

### **Почему иногда бывает наоборот?**

– При низкой диверсности, ошибки в версиях коррелируют, и отказ одной версии повышает риск отказа всей системы.

– В случае высокой диверсности, вероятности отказов между версиями существенно не пересекаются, и система становится более отказоустойчивой – это и есть главный эффект диверсности.

### **3.1.3. Основные метрики диверсности**

1) Дистанция Хэмминга: количество позиций, в которых два бинарных представления различаются.

2) Корреляция ошибок: статистическая мера, показывающая взаимосвязь возникших ошибок в разных версиях.

3) Другие показатели, включая структурные и поведенческие различия.

**Математически расстояние Хэмминга** между двумя строками или векторами одинаковой длины  $X = (x_1, x_2, \dots, x_n)$  и  $Y = (y_1, y_2, \dots, y_n)$  определяется как количество позиций  $i$ , в которых символы  $x_i$  и  $y_i$  различны.

Формула

$$d_H(X, Y) = \sum_{i=1}^n \delta(x_i, y_i),$$

где

$$\delta(x_i, y_i) = \begin{cases} 0, & \text{если } x_i = y_i, \\ 1, & \text{если } x_i \neq y_i. \end{cases}$$

Таким образом, расстояние Хэмминга – это сумма индикаторных функций отличий по всем позициям строк.

Это число соответствует минимальному количеству замен символов, необходимых для превращения строки  $X$  в строку  $Y$ . Расстояние Хэмминга удовлетворяет аксиомам метрики: оно неотрицательно, симметрично: ( $d_H(X, Y) = d_H(Y, X)$ ), равно нулю, только если строки совпадают, и подчиняется неравенству треугольника.

Для двоичных строк оно может быть интерпретировано как вес суммы (покоординатной операции XOR по модулю 2) двух слов:

$$d_H(a, b) = w(a \oplus b),$$

где  $w(.)$  – количество единиц в векторе (вес).

Кроме рассмотренной, существуют разные виды метрик расстояния, которые применяются в задачах сравнения объектов, классификации и кластеризации. Вот основные из них:

**Евклидово расстояние (L2)** – классическая метрика, измеряет длину прямой между точками в пространстве:  $d(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$

**Манхэттенское расстояние (L1, расстояние городских кварталов)** – сумма абсолютных разностей координат:  $d(X, Y) = \sum_{i=1}^n |x_i - y_i|$

**Расстояние Чебышева (L $\infty$ )** – максимальное абсолютное отличие по любой координате:  $d(X, Y) = \max_i |x_i - y_i|$ .

**Расстояние Минковского** – обобщение L1, L2 и L $\infty$  с параметром  $p$ :  $d(X, Y) = (\sum_{i=1}^n |x_i - y_i|^p)^{\frac{1}{p}}$ .

В работе требуется использовать две метрики.

### 3.2. Ход работы

#### Шаг 1. Анализ исходных данных

Проанализировать вариант задания. Сформулировать обоснование особенностей версий, построить таблицу сравнения версий по 10 признакам. Сформировать данные о двух и более версиях ПО в бинарном или упрощённом виде для последующего анализа согласно варианту задания, пример:

Таблица 3.1 – Таблица сравнения версий по 10 признакам

| <i>№ Признака версии</i>                         | <i>Версия 1</i> | <i>Версия 2</i> | <i>Версия 3</i> |
|--------------------------------------------------|-----------------|-----------------|-----------------|
| Признак 1: наличие человеко-машинного интерфейса | 1 (есть)        | 0 (нет)         | 1 (есть)        |
| Признак 2: наличие системы ИИ                    | 0               | 1               | 1               |
| ...                                              | ...             | ...             | ...             |

#### Шаг 2. Расчет метрик диверсности

Рассчитать дистанцию Хэмминга между бинарными версиями. Рассчитать расстояние между данными еще на основе другой метрики.

#### Шаг 3. Расчет показателей надежности

Применить модель (n,m) для оценки надежности системы из нескольких версий.

#### Шаг 4. Моделирование и анализ

Проанализировать, как изменение числа версий и параметра m влияет на надежность для заданного варианта работы. Сделать выводы.

**Шаг 5. Моделирование и анализ для другой метрики, провести все те же шаги.**

Сделать выводы.

### 3.3. Пример фрагмента программного кода на Python

*Примечание: при копировании кода, возможно изменение отступов команд, что является ошибкой для python, требуется проверка и отладка.*

```
import matplotlib.pyplot as plt
from scipy.special import comb
# Возможно, потребуется установка других похожих библиотек:
scipySTAT или
# DATAscipy, если scipy базовая не заработает
import numpy as np

def hamming_distance(version1, version2):
    # Расчет дистанции Хэмминга между двумя бинарными строками
    assert len(version1) == len(version2), "Длины версий должны
совпадать"
    return sum(c1 != c2 for c1, c2 in zip(version1, version2))

def average_hamming_distance(versions):
    # Средняя дистанция Хэмминга по всем парам версий
    n = len(versions)
    total_dist = 0
    count = 0
    for i in range(n):
        for j in range(i + 1, n):
            total_dist += hamming_distance(versions[i], versions[j])
            count += 1
    return total_dist / count if count > 0 else 0

def adjusted_reliability(p, n, m, avg_dist, max_dist):
    # Расчет надежности с учётом средней диверсности версий.
    # Чем выше avg_dist, тем выше надёжность одной версии и системы
    в целом.
    # p – базовая вероятность безотказной работы версии
    # n – число версий
    # m – минимально необходимое число исправных версий
    # avg_dist – средняя дистанция Хэмминга между версиями
    # max_dist – максимальная дистанция (длина версии в битах)

    # Коэффициент коррекции надежности одной версии от диверсности,
    варьируется от 0.5 до 1.0
    div_factor = 0.5 + 0.5 * (avg_dist / max_dist)
    # Исправленная вероятность безотказной работы одной версии
    p_adj = min(p * div_factor, 1.0)
    # Расчет вероятности безотказной работы системы (n, m)
```

```

prob = 0
for k in range(m, n + 1):
    prob += comb(n, k) * (p_adj ** k) * ((1 - p_adj) ** (n - k))
return prob, p_adj
def visualize_versions(versions):
    # Визуализация бинарных версий с цветовой маркировкой битов:
    # зеленый - бит 1, красный - бит 0
    n = len(versions)
    length = len(versions[0])
    fig, ax = plt.subplots(n, 1, figsize=(length / 1.5, 1.5 * n))
    if n == 1:
        ax = [ax]
    for idx, version in enumerate(versions):
        colors = ['green' if c == '1' else 'red' for c in version]
        for i in range(length):
            ax[idx].text(i, 0.5, version[i], fontsize=14, ha='center', va='center',
color=colors[i])
        ax[idx].set_xlim(-1, length)
        ax[idx].set_ylim(0, 1)
        ax[idx].axis('off')
        ax[idx].set_title(f"Версия {idx + 1}")
    plt.tight_layout()
    plt.show()

def visualize_reliability(p, n, m, max_dist):
    # Визуализация зависимости надежности многоверсионной системы
от диверсности
    divs = np.linspace(0, max_dist, 100)
    reliabilities = []
    adjusted_ps = []
    for d in divs:
        rel, p_adj = adjusted_reliability(p, n, m, d, max_dist)
        reliabilities.append(rel)
        adjusted_ps.append(p_adj)

    fig, ax1 = plt.subplots(figsize=(10, 6))

    ax1.set_xlabel('Средняя дистанция Хэмминга (диверсность)')
    ax1.set_ylabel('Результирующая надежность системы', color='blue')
    ax1.plot(divs, reliabilities, color='blue', label='Надежность системы')
    ax1.tick_params(axis='y', labelcolor='blue')
    ax1.grid(True)
    ax2 = ax1.twinx()

```

```

    ax2.set_ylabel('Скорректированная вероятность безотказной работы
версии', color='green')
    ax2.plot(divs, adjusted_ps, color='green', linestyle='dashed',
label='Корректированное p')
    ax2.tick_params(axis='y', labelcolor='green')
    fig.suptitle(f'Зависимость надежности от диверсности для n={n},
m={m}')
    fig.legend(loc='upper left', bbox_to_anchor=(0.15, 0.85))
    plt.show()
if __name__ == "__main__":
    versions = [
        "110011001100",
        "110111000100",
        "100011100110",
    ]
    visualize_versions(versions)
    base_p = 0.95
    n_versions = len(versions)
    m_required = 2
    max_distance = len(versions[0])
    avg_dist = average_hamming_distance(versions)
    print(f"Средняя дистанция Хэмминга между версиями: {avg_dist:.2f}
из {max_distance}")

    system_reliability, adjusted_p = adjusted_reliability(base_p, n_versions,
m_required, avg_dist, max_distance)
    print(f"Корректированная вероятность безотказной работы версии:
{adjusted_p:.4f}")
    print(f"Вероятность безотказной работы системы (n={n_versions},
m={m_required}): {system_reliability:.4f}")
    visualize_reliability(base_p, n_versions, m_required, max_distance)

```

На рис. 3.1 - 3.3 приведены результаты работы программы.



Рисунок 3.1 – Исходные коды версий

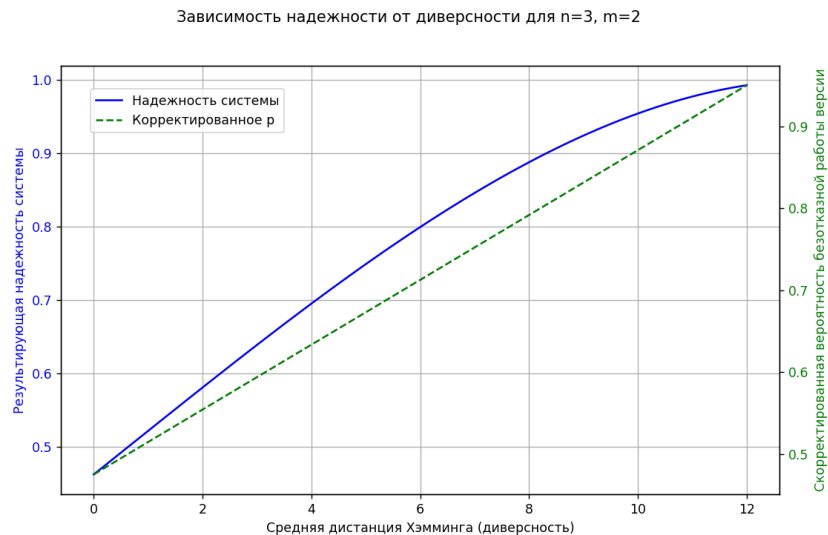


Рисунок 3.2 – Графики зависимости надежности от диверсности

```

Средняя дистанция Хэмминга между версиями: 3.33 из 12
Корректированная вероятность безотказной работы версии: 0.6069
Вероятность безотказной работы системы (n=3, m=2): 0.6580

Process finished with exit code 0

```

Рисунок 3.3 – Информация в консоли

### Анализ полученных результатов.

Рассчитанная дистанция Хэмминга показывает, насколько версии разнородны. В полученном примере она равна 3.33, что свидетельствует о заметной разнице. Вероятность надежности версии 65%, при этом она скорректирована (снижена) до 60.69% при параметрах  $n = 3$  и  $m = 2$ .

Часть 2 (шаг 5). Результаты моделирования для метрики по Жаккару.  
Для метрики Хэмминга:

$$p_{adj} = \min\left(p \times \left(0.5 + 0.5 \times \frac{d_H}{L}\right), 1\right) \text{ и } P = \sum_{k=m}^n \binom{n}{k} p_{adj}^k (1 - p_{adj})^{n-k}$$

Для метрики Жаккара (где  $d_J$  – нормированное расстояние от 0 до 1):

$$p_{adj} = \min(p \times (0.5 + 0.5 \times d_J), 1) \text{ и } P = \sum_{k=m}^n \binom{n}{k} p_{adj}^k (1 - p_{adj})^{n-k}$$

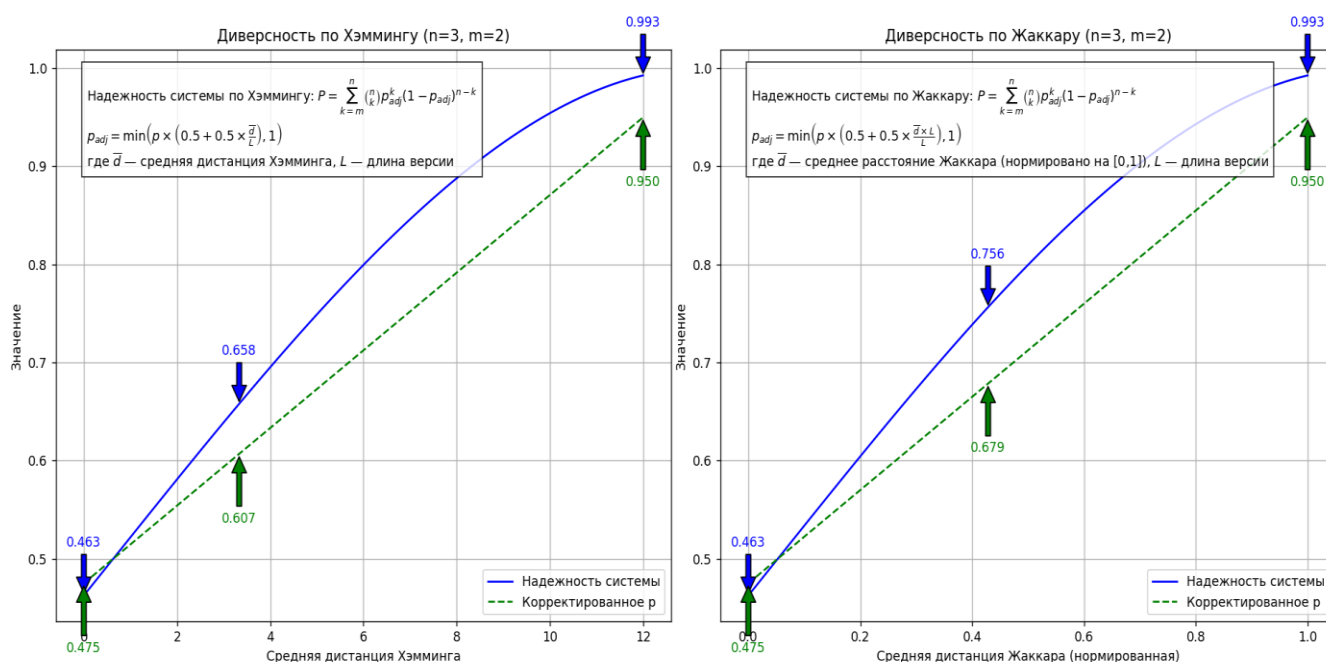


Рисунок 3.4 – Графики зависимости надежности от диверсности для двух мер

## РАСЧЕТЫ НАДЕЖНОСТИ

Средняя дистанция Хэмминга между версиями: 3.333 из 12

Среднее расстояние Жаккара между версиями: 0.429 (нормировано на [0,1])

-- Хэмминг --

Скорректированная вероятность безотказной работы версии ( $p_{adj}$ ): 0.6069  
 Вероятность безотказной работы системы (n=3, m=2): 0.6580

-- Жаккар --

Скорректированная вероятность безотказной работы версии ( $p_{adj}$ ): 0.6786  
 Вероятность безотказной работы системы (n=3, m=2): 0.7565

## ВЫВОДЫ

Средняя дистанция Хэмминга показывает среднее количество отличающихся бит между версиями.

Среднее расстояние Жаккара отражает степень непересечения множества позиций, где стоят единицы.

Было учтено влияние диверсности на вероятность безотказной работы версии и системы в целом.

Чем больше диверсность, тем выше диверсифицированность версий, что корректирует вероятность работы.

При данном уровне диверсности с учетом числа версий и минимально допустимого числа исправных, система показывает разумную надежность.

### Контрольные вопросы

1. Что такое диверсность в контексте многоверсионного программного обеспечения?
2. Какой жизненный цикл разработки ПО реализует многоверсионный проект?
3. Как вычислить дистанцию Хэмминга между версиями?
4. В чем суть модели (n,m) многоверсионных систем?
5. Как диверсность влияет на надежность ПО?
6. Какие современные методы повышения надежности можно применять?

### Варианты заданий для лабораторной работы

| №  | Тематика заданий в области критического компьютеринга | Описание задания                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1  | Жизнеобеспечение                                      | Анализ диверсности и надежности систем жизнеобеспечения                                                                                                                                                                                                                                                                                                                                                                                                      |
| 2  | Авиация                                               | Моделирование надежности ПО систем управления полетом                                                                                                                                                                                                                                                                                                                                                                                                        |
| 3  | Медицинское оборудование                              | Оценка отказоустойчивости ПО для диагностических приборов                                                                                                                                                                                                                                                                                                                                                                                                    |
| 4  | Финансовые системы                                    | Анализ надежности алгоритмов обработки транзакций                                                                                                                                                                                                                                                                                                                                                                                                            |
| 5  | Производственные управляющие системы                  | Расчет надежности многоверсионного ПО станков и роботов                                                                                                                                                                                                                                                                                                                                                                                                      |
| 6  | Интернет вещей                                        | Метрики диверсности версий прошивок для распределенных датчиков                                                                                                                                                                                                                                                                                                                                                                                              |
| 7  | Космические аппараты                                  | Надежность многоверсионных систем управления спутниками                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8  | Военное программирование                              | Анализ отказоустойчивости систем связи и навигации                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9  | Автомобильная промышленность                          | Расчет надежности ECU с использованием MBC. ECU (Electronic Control Unit) - это электронный блок управления, представляющий собой миниатюрный компьютер в автомобиле, который контролирует и управляет работой различных систем и подсистем автомобиля. Основная его функция - обработка данных, поступающих с множества датчиков, и формирование управляющих воздействий на исполнительные механизмы для оптимизации работы автомобиля, особенно двигателя. |
| 10 | Робототехника                                         | Моделирование отказов ПО для многоверсионных роботов                                                                                                                                                                                                                                                                                                                                                                                                         |
| 11 | Энергетика                                            | Надежность программных систем для управления электросетями                                                                                                                                                                                                                                                                                                                                                                                                   |

| №  | Тематика заданий в области критического компьютеринга | Описание задания                                                         |
|----|-------------------------------------------------------|--------------------------------------------------------------------------|
| 12 | Системы искусственного интеллекта                     | Анализ устойчивости и диверсности ML-моделей в критических задачах       |
| 13 | Телекоммуникации                                      | Оценка надежности многоверсионных протоколов передачи данных             |
| 14 | Обработка больших данных                              | Диверсность алгоритмов обработки данных для повышения отказоустойчивости |
| 15 | Кибербезопасность                                     | Анализ диверсности версий ПО для повышения устойчивости к атакам         |

## 4. Лабораторная работа № 3

### ОЦЕНКА ВЕРСИЙ СИСТЕМ КРИТИЧЕСКОГО ПРИМЕНЕНИЯ ПУТЕМ АНАЛИЗА МЕТРИК РАЗЛИЧИЯ

**Цель работы:** изучение принципов построения метрик различия применительно к сравнению версий компьютерных систем.

#### 4.1. Теоретические сведения

##### 4.1.1. Формализация обработки качественных признаков

Множество альтернативных вариантов систем (версий), систематизированных в морфологических таблицах, может быть отражено списком их качественных признаков (функций, свойств). Список признаков, определяющий вариант морфологического множества, представляет его признаковый образ [1].

Множество образов вариантов систем может быть представлено как матрица, имеющая  $q$  столбцов и  $p$  строк (порядка  $p \times q$ ), причем номеру столбца соответствует наименование системы  $S_j$  ( $j = 1, 2, \dots, q$ ), а номеру строки – название признака  $Z_i$  ( $i = 1, 2, \dots, p$ ).

Информационным содержанием матриц являются указания о присутствии или отсутствии каждого из учитываемых признаков в рассматриваемых системах. При этом если  $i$ -й признак присутствует в  $j$ -й системе, то на пересечении  $i$ -й строки и  $j$ -го столбца помещается "1", в противном случае – "0".

Любой  $j$ -й столбец матрицы назовем описанием  $j$ -й системы, любую  $i$ -ю строку – описанием  $i$ -го признака. В терминах теории множеств

$$S = S_i | j \in J, \quad (1)$$

где  $j$  – индексированное множество с элементами  $S_i$ ,  $S_i = S_j$ , если  $k = j$ ,  $J$  – индексное множество;  $J = \{ j | j - \text{целое число, } 1 \leq j \leq q \}$ .

Формула (1) читается так: семейство множеств  $S$ , состоящее из всех  $S_j$ , таких, у которых элементы  $j$  принадлежат множеству  $J$ .

Аналогично  $Z = \{ Z_i | i \in I \}$  есть индексированное множество, а  $I$  – индексное множество:  $I = \{ i | i - \text{целое число, } 1 \leq i \leq p \}$ .

Индексация позволяет различать множества, состоящие из одинаковых элементов.

Пример матрицы образов представлен в таблице 4.1.

Таблица 4.1 – Матрица образов версий как семейство множеств и их признаков\*

| <b>Версии →<br/>Признаки версий<br/>↓</b>                                | <b>Версии (варианты)<br/>системы:<br/><math>S_1</math></b> | <b><math>S_2</math></b>                                   | <b><math>S_3</math></b> | <b>...</b> | <b><math>S_q</math></b> |
|--------------------------------------------------------------------------|------------------------------------------------------------|-----------------------------------------------------------|-------------------------|------------|-------------------------|
| <b>Функция системы или<br/>название подсистемы:<br/><math>Z_1</math></b> | Функция $Z_1$<br>отсутствует в версии<br>системы $S_1$ : 0 | Функция $Z_1$ имеется<br>в версии<br>системы $S_2$ :<br>1 | 0                       | ...        | 1                       |
| $Z_2$                                                                    | 1                                                          | 1                                                         | 0                       | ...        | 1                       |
| $Z_3$                                                                    | 1                                                          | 1                                                         | 1                       | ...        | 0                       |
| ...                                                                      | ...                                                        | ...                                                       | ...                     | ...        | ...                     |
| $Z_p$                                                                    | 0                                                          | 0                                                         | 0                       | 0          | 0                       |

\*Примечание:  $Z_i$  могут представлять название отдельных функций систем и их версий; название подсистем или свойств, присущих версиям.

Рассмотрим основные меры, порождающие отношения на множестве исследуемых систем.

#### 4.1.2. Меры сходства и различия

Мерой сходства (близости) обычно называется величина  $C(S_j, S_k)$ , имеющая предел и возрастающая с возрастанием близости объектов. Под мерой сходства будем понимать неотрицательную вещественную функцию  $C(S_j, S_k)$ , обладающую следующими свойствами:

$$\begin{aligned} 0 \leq C(S_j, S_k) \leq 1 \text{ для } k \neq j; \\ C(S_j, S_k) = 1 \text{ для } k = j; \\ C(S_j, S_k) = C(S_k, S_j). \end{aligned}$$

Здесь  $S_j, S_k$  – множества значений признаков, описывающие сравниваемые объекты. Мера, коэквивалентная мере сходства, называется мерой различия  $D(S_j, S_k)$  и обладает свойствами метрики, если:

$$\begin{aligned} D(S_j, S_k) \geq 0 \text{ для всех } j, k \in J; \\ D(S_j, S_k) = 0 \text{ для } S_j = S_k; \\ D(S_j, S_k) = D(S_k, S_j); \\ D(S_j, S_k) \leq D(S_j, S_s) + D(S_k, S_s), \text{ где } s \in J. \end{aligned}$$

Этими свойствами обладает, в частности, континуум эквивалентных мер, представляемых формулой

$$C(S_j, S_k)_u = \frac{2m(S_j \cap S_k)}{(1+u)[m(S_j) + m(S_k)] - 2um(S_j \cap S_k)} \quad (2)$$

где  $-1 \leq u \leq \infty$ ;

$m(S_j)$  — обозначение числа элементов множества  $S_j$ .

### Правила расчёта мер сходства по формуле (2):

1. Для заданных множеств  $S_j, S_k$  вычисляются измерения множества (например, размеры, количество элементов) и их пересечения.

2. Параметр  $u$  изменяет способ нормализации и веса пересечения в знаменателе.

3. Подставляя конкретное значение  $u$  и  $S$ , получают разные меры сходства, например:

4.  $u=0$ : мера Чекановского - Серенсена.

5.  $u=1$ : мера Жаккара.

6.  $u=3$ : мера Сокала - Снита.

7.  $u = -1/2$ : мера Андреева.

8. Числитель остаётся постоянным:  $m(S_j \cap S_k)$ , отражая значение пересечения, а знаменатель масштабируется параметром  $u$  для усиления или ослабления влияния пересечения и размеров множеств.

Меры сходства и различия "изобретаются" по специальным правилам [2], а выбор конкретных мер зависит, в первую очередь, от суперзадачи – цели конкретного исследования, а также от шкалы измерений. В табл. 2 приведены наиболее распространенные меры сходства и различия для различных значений коэффициента  $u$ , предназначенные для обработки качественных и количественных признаков.

Вычисление меры сходства, например, по формуле Чекановского – Серенсена с учетом бинарной матрицы осуществляется по следующему выражению:

$$C(S_1, S_2) = \frac{2 \sum_{i=1}^p x_{i1} x_{i2}}{\sum_{i=1}^p x_{i1} + \sum_{i=1}^p x_{i2}},$$

где  $x_{i1}, x_{i2}$  – одно из двух значений  $\{0, 1\}$ .

#### 4.1.3. Правила вычисления количества элементов некоторых множеств, получаемых в результате операций над ними

Количество элементов множества  $S$  равно

$$m(S) = \sum_{i=1}^p x_i,$$

где  $p$  – общее число элементов множества  $S$ ;  $x_i$  – значение  $i$ -го элемента множества  $S$ , при этом  $\forall x_i \in S \rightarrow x_i = 1$ .

Количество элементов пересечения двух множеств  $S_1 \cap S_2$  равно

$$m(S_1 \cap S_2) = \sum_{i=1}^p x_{i1} x_{i2},$$

где  $x_{i1}, x_{i2}$  – соответственно значения  $i$ -го элемента для множеств  $S_1$  и  $S_2$ .

Таблица 4.2 – Формулы мер сходства и различия

| № | Название меры                                | Формула для вычисления                                                                      |
|---|----------------------------------------------|---------------------------------------------------------------------------------------------|
| 1 | Мера сходства Чекановского - Серенсена (u=0) | $C(S_j, S_k) = \frac{2 m(S_j \cap S_k)}{m(S_j) + m(S_k)}$                                   |
| 2 | Мера сходства Жаккара (u=1)                  | $C(S_j, S_k) = \frac{m(S_j \cap S_k)}{m(S_j \cup S_k)}$                                     |
| 3 | Мера сходства Сокала - Снита (u=3)           | $C(S_j, S_k) = \frac{m(S_j \cap S_k)}{2 m(S_j) + 2 m(S_k) - 3 m(S_j \cap S_k)}$             |
| 4 | Мера сходства Андреева (u=-1/2)              | $C_{-1/2}(S_j, S_k) = \frac{4 m(S_j \cap S_k)}{m(S_j) + m(S_k) + 2 m(S_j \cap S_k)}$        |
| 5 | Мера сходства Мульчинского                   | $(S_j, S_k) = \frac{2}{m(S_j \cap S_k)} \left( \frac{1}{m(S_j)} + \frac{1}{m(S_k)} \right)$ |
| 6 | Мера различия                                | $D(S_j, S_k) = m(S_j) + m(S_k) - 2 m(S_j \cap S_k)$                                         |
| 7 | Мера сходства Половинкина                    | $C(S_j, S_k) = 1 - \frac{m(S_j \cup S_k) - m(S_j \cap S_k)}{m(S_j \cup S_k)}$               |

**Описания формул мер сходства и различия:****1. Мера сходства Чекановского - Серенсена (u=0)**

$$C(S_j, S_k) = \frac{2 m(S_j \cap S_k)}{m(S_j) + m(S_k)}$$

Эта мера вычисляет сходство как удвоенный размер пересечения множеств, делённый на сумму их размеров. Такой подход сравнивает пересечение с усреднённым размером множеств, что делает её устойчивой к значительным различиям в размере.

**2. Мера сходства Жаккара (u=1)**

$$C(S_j, S_k) = \frac{m(S_j \cap S_k)}{m(S_j \cup S_k)}$$

Сравнивает размер пересечения с размером объединения. За счёт учёта уникальных элементов мера чувствительна к уникальному содержимому каждого множества.

**3. Мера сходства Сокала - Снита (u=3)**

$$C(S_j, S_k) = \frac{m(S_j \cap S_k)}{2 m(S_j) + 2 m(S_k) - 3 m(S_j \cap S_k)}$$

Формула придаёт больший вес пересечению, снижая влияние уникальных элементов, что делает меру более строгой по сравнению с мерой Жаккара.

#### 4. Мера сходства Андреева ( $\alpha=1/2$ )

$$C_{-1/2}(S_j, S_k) = \frac{4 m(S_j \cap S_k)}{m(S_j) + m(S_k) + 2 m(S_j \cap S_k)}$$

Это модификация меры Чекановского, длинной знаменатель учитывает пересечение, что "вознаграждает" множества за общие элементы, повышая значения сходства.

#### 5. Мера сходства Мульчинского

$$C(S_j, S_k) = \frac{2}{m(S_j \cap S_k)} \left( \frac{1}{m(S_j)} + \frac{1}{m(S_k)} \right)$$

Использует среднее гармоническое величин, обратных размерам множеств и пересечения. Такая мера сильнее штрафует за большую разницу в размерах сравнительных множеств.

#### 6. Мера различия

$$D(S_j, S_k) = m(S_j) + m(S_k) - 2 m(S_j \cap S_k)$$

Это метрика расстояния: чем меньше пересечение, тем выше отличие множеств.

#### 7. Мера сходства Половинкина

$$C(S_j, S_k) = 1 - \frac{m(S_j \cup S_k) - m(S_j \cap S_k)}{m(S_j \cup S_k)}$$

После преобразования эквивалентна мере Жаккара, представляет собой альтернативное выражение классического показателя сходства.

#### 4.1.4. Пример построения матриц отношений сходства и включения и оценки их мер

Этот алгоритм отличается для указанных двух мер лишь методом расчета значений матриц сходства и включения.

Шаг 1. Формируются два множества: множество исследуемых объектов  $J = \{ S_1, S_2, \dots, S_p \}$  и множество признаков  $Z = \{ Z_1, Z_2, \dots, Z_r \}$ . Каждый объект  $S_i$  описывается подмножеством признаков  $\{ Z_i \} \in Z$ , являющимся качественным признаковым образом. Все образы объектов систематизируются в матрицу образов, где представляются индексированными множествами (табл. 4.3).

Шаг 2. Генерируются все парные сочетания объектов, и для каждой пары описаний объектов  $S_i$  и  $S_j$  строится индексная матрица  $B = \| x_{ij} \|$ ;  $i = \overline{1, p}$ ;  $j = \overline{1, 2}$ ; где  $p$  – число строк матрицы образов, соответствующее числу рассматриваемых признаков  $m(Z)$ . На основе индексной матрицы рассчитываются меры сходства  $C(S_i, S_j)$  или включения  $W(S_i, S_j)$ .

Таблица 4.3 – Пример матрицы образов

|          | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|----------|-------|-------|-------|-------|-------|-------|-------|
| $Z_1$    | 1     | 1     | 0     | 1     | 0     | 0     | 1     |
| $Z_2$    | 0     | 1     | 0     | 1     | 0     | 1     | 1     |
| $Z_3$    | 0     | 1     | 1     | 0     | 1     | 1     | 0     |
| $Z_4$    | 1     | 1     | 1     | 0     | 1     | 1     | 0     |
| $Z_5$    | 1     | 0     | 1     | 0     | 1     | 1     | 1     |
| $Z_6$    | 0     | 0     | 1     | 0     | 1     | 0     | 1     |
| $Z_7$    | 1     | 1     | 0     | 1     | 1     | 0     | 1     |
| $Z_8$    | 0     | 1     | 0     | 0     | 0     | 0     | 1     |
| $Z_9$    | 1     | 1     | 1     | 0     | 0     | 0     | 0     |
| $Z_{10}$ | 1     | 0     | 1     | 0     | 0     | 1     | 0     |

Например, для пары объектов  $S_1$  и  $S_2$  меры сходства и включения имеют следующие значения:

$$\text{Мера сходства } C(S_1, S_2) = \frac{2 \sum_{i=1}^{10} x_{i1} x_{i2}}{\sum_{i=1}^{10} x_{i1} + \sum_{i=1}^{10} x_{i2}} = \frac{2 \cdot 4}{6 + 7} = 0,614.$$

Шаг 3. На основе рассчитанных на шаге 2 значений мер сходства и включения (рис. 4.2) строятся соответствующие матрицы размерностью  $q \times q$ , по которой делают заключение о сходстве версий.

**Матрица мер сходства**

|       | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| $S_1$ | 1     | 0,62  | 0,50  | 0,44  | 0,55  | 0,55  | 0,50  |
| $S_2$ | 0,62  | 1     | 0,46  | 0,60  | 0,50  | 0,50  | 0,62  |
| $S_3$ | 0,50  | 0,46  | 1     | 0     | 0,73  | 0,73  | 0,33  |
| $S_4$ | 0,44  | 0,60  | 0     | 1     | 0,75  | 0,25  | 0,67  |
| $S_5$ | 0,55  | 0,50  | 0,73  | 0,75  | 1     | 0,60  | 0,55  |
| $S_6$ | 0,55  | 0,50  | 0,73  | 0,25  | 0,60  | 1     | 0,36  |
| $S_7$ | 0,50  | 0,62  | 0,33  | 0,67  | 0,55  | 0,36  | 1     |

Рисунок 4.2 – Матрица мер сходства

### Задание на лабораторную работу.

1. Изучить теоретические сведения по способу оценки схожести и различия версий как в диверсных системах, так и в многоверсионных проектах.
2. Для варианта из лабораторной работы № 1 сформировать морфологическую таблицу 1 для чего определить возможные версии  $S_j$  (3-4 версии) и сформулировать признаки этих версий,  $Z_i$  (5-6 признаков).
3. В Excel построить программу расчета мер сходства и различия сформулированных версий для любых 4 мер из таблицы 2.
4. Построить матрицу мер сходства (таблица 3).
5. Сделать выводы о сформированных версиях.
6. Оформить и защитить отчет о выполнении лабораторной работы.

**Контрольные вопросы**

1. Привести примеры метрик диверсности.
2. В каких случаях актуальна диверсность ПО, а к каким не используется?
3. Пояснить смысл метрик сходства и различия.
4. Каким образом выбираются исходные данные для использования метрики различия?
5. Что такое мера сходства, мера различия?

## 5. Лабораторная работа № 4

### ИССЛЕДОВАНИЕ АСПЕКТОВ КОЛИЧЕСТВЕННОГО АНАЛИЗА КОМПЛЕКСА ТЕХНИЧЕСКИХ СРЕДСТВ КРИТИЧЕСКОГО ПРИМЕНЕНИЯ

#### Цель работы:

- Получить практические навыки формирования количественных оценок комплекса технических средств.
- Получить практические навыки в проведении количественного анализа вычислительных систем.

#### 5.1. Теоретические сведения

В основе анализа комплекса технических средств лежит определение соответствия объёмов работ и временных ограничений по количеству и производительности технических средств.

Основным ограничивающим временным фактором является цикл работ (Ц) обработки информации. Под этим термином понимается промежуток времени от начала поступления данных на обработку  $T_H$  до срока готовности задачи  $T_G$ :

$$Ц = T_G - T_H. \quad (1)$$

В свою очередь, цикл работ будет складываться из допустимых продолжительностей выполнения подготовительных операций  $T_P$  и времени решения задач на ПЭВМ (вычислений)  $T_B$ . Продолжительность решения задач на ПЭВМ зависит от реализации основных этапов и режимов обработки информации.

Композиционно время решения задачи решения разбивается на этапы

$$t = t_I + t_P + t_E + t_O, \quad (2)$$

где  $t_I$  – ввод (загрузка, input) программы и данных;  $t_P$  – процессорное время (processing), затрачиваемое на счёт непосредственно;  $t_E$  – время, затрачиваемое на обмен в процессе счёта с внешними (external) устройствами;  $t_O$  – формирование и вывод (output) результатов работы.

Время ввода определяется по элементарной формуле

$$t_I = \frac{M_I}{v_I}, \quad (3)$$

где  $M_I$  – объём массива вводимых данных;  $v_I$  – быстродействие устройства.

Процессорное время может рассчитываться двояко, в зависимости от степени известности информации об организации вычислительного процесса. В первом случае

$$t_P = \bar{t}_{op} \cdot k_{CT} \cdot M_I, \quad (4)$$

где  $\bar{t}_{op} = \sum_j t_j \cdot q_j$  – среднее время выполнения операции,  $k_{CT}$  – коэффициент соотношения операций, показывающий, какое количество операций приходится на байт введённой информации;  $t_j$  – время выполнения  $j$ -й операции;  $q_j$  – удельный вес  $j$ -й операции.

Для определения коэффициентов соотношения операций и удельных весов операций задачи разбиваются на типовые группы.

I. Задачи перегруппировки и упорядочивания информации, нахождения сумм, выборок, сечений по относительно простым критериям. Удельный вес логических операций достигает 70%, арифметических – до 30%. Коэффициент соотношения находится в пределах 50 – 100 операций на байт введённой информации.

II. Манипуляция несколькими массивами, часть которых находится на внешних запоминающих устройствах. Объём арифметических операций незначителен: нахождение сумм, парных произведений, операции с матрицами. Удельный вес логических операций достигает 90 – 95%, арифметических – до 5 – 10%,  $k_{CT} \in (1000 – 2000)$  операций на байт.

III. Задачи экстремального характера: численная реализация сложных математических методов, изобилует циклами, итерациями и т. п. Удельный вес арифметических операций достигает 90%, а удельный вес логических – всего около 10%,  $k_{CT} \in (1000 – 3000)$  операций на байт.

Во втором случае, если априори известен алгоритм решения задачи, то расчёты целесообразно вести по формуле

$$t_p = \sum_j t_j \cdot h_j, \quad (5)$$

где  $h_j$  – количество  $j$ -х операций применённых в реализации алгоритма.

Затраты времени на обмен с внешними устройствами рекомендуется определять по формуле

$$t_E = \frac{k_{rw}}{v_{rw}} \sum_j f_j \cdot V_j, \quad (7)$$

где  $k_{rw}$  – коэффициент, учитывающий контрольные операции, при проведении операций считывания или записи (например, для контрольного суммирования  $k_{rw} = 2$ );  $v_{rw}$  – скорость чтения или записи;  $f_j$  – количество обращений к внешним устройствам;  $V_j$  – объём считываемой или записываемой порции данных. Коэффициенты, находящиеся в (7) перед знаком суммы, могут быть, в случае разнообразия устройств ввода и вывода, использованы в качестве множителей под знаком.

Время вывода результата определяться быстродействием конечного устройства как отношение

$$t_o = \frac{M_o}{v_o} = \frac{M_I \cdot \kappa_{IO}}{v_o}, \quad (8)$$

где  $M_O$  – объём массива выводимых данных;  $v_O$  – быстродействие устройств;  $M_I$  – объём массива вводимых данных;  $k_{IO}$  – коэффициент соотношения выводимой информации по отношению к объёму входной.

Общее время решения последовательности задач составит

$$t_{\Sigma} = \sum_i t_i, \quad (9)$$

где  $t_i$  определяется (2) для каждой задачи.

Отсюда потребность в вычислительных устройствах составляет

$$N \geq \frac{U}{t_{\Sigma}}. \quad (10)$$

## 5.2. Ход работы

1. Для варианта задания (описания многоверсионной системы) провести расчеты по приведенным формулам.
2. Написать программу для расчёта параметров (возможно использование Excel).
4. Дополнить программу конструкциями для вывода графиков.
5. Оформить отчёт, защитить результаты выполнения лабораторной работы

## 5.3. Пояснения к выполнению лабораторной работы

**В работе ПРЕДПОЛАГАЮТСЯ численные значения параметров, исходя из варианта.**

**Требуется обоснование в виде графиков и анализа результатов. Пример приведен ниже.**

Исходные данные и предположения (в цифровых значениях)

1. Объём вводимых данных  $M_I = 5000$  байт.
2. Быстродействие устройства ввода  $v_I = 250$  байт/с.
3. Среднее время выполнения операции  $t_j = 0.0001$  с.
4. Коэффициент соотношения операций  $k_{CT} = 100$  операций на байт.
5. Удельные веса операций: логические 70%, арифметические 30%.
6. Количество операций по алгоритму  $h_j = 500000$ .
7. Коэффициент контроля операций чтения/записи  $k_{rw} = 2$ .
8. Скорость чтения/записи данных  $v_{rw} = 1000$  байт/с.
9. Количество обращений к устройствам  $f_j = 10$ .
10. Объём считываемых/записываемых данных  $V_j = 1000$  байт.
11. Объём выводимых данных  $M_O = 1000$  байт.
12. Быстродействие устройства вывода  $v_O = 300$  байт/с.
13. Коэффициент соотношения выводимой информации к входной  $k_{IO} = 0.2$ .

Расчёт времени основных этапов цикла:

1. Время ввода по формуле (3):

$$t_I = \frac{M_I}{v_I} = \frac{5000}{250} = 20 \text{ секунд}$$

2. Процессорное время

2.1 По формуле (4):

Среднее время одной операции суммарно:

$$\sum_i t_j q_j = t_j \cdot (0.7 + 0.3) = 0.0001 \text{ с}$$

Тогда

$$t_P = M_I \cdot k_{CT} \cdot \sum_i t_j q_j = 5000 \times 100 \times 0.0001 = 50 \text{ секунд}$$

2.2 Альтернативно по формуле (5) с известным количеством операций  $h_j$ :

$$t_P = \sum_i h_j t_j = 500000 \times 0.0001 = 50 \text{ секунд}$$

(совпадает с расчетом выше)

3. Время обмена с внешними устройствами  $t_E$  по формуле (7):

$$t_E = k_{rw} \cdot \frac{f_j \times V_j}{v_{rw}} = 2 \times \frac{10 \times 1000}{1000} = 20 \text{ секунд}$$

4. Время вывода результата по формуле (8):

$$t_O = \frac{M_O}{v_O} = \frac{1000}{300} \approx 3.33 \text{ секунды}$$

Итоговое время решения задачи по формуле (2):

$$t = t_I + t_P + t_E + t_O = 20 + 50 + 20 + 3.33 = 93.33 \text{ секунды}$$

Расчет общего времени решения последовательности из  $N = 10$  задач (по формуле 9):

$$\sum_i t_i = N \times t = 10 \times 93.33 = 933.3 \text{ секунды} \approx 15.5 \text{ минут}$$

Оценка потребности в вычислительных устройствах по формуле (10).

Пусть необходимое время решения  $t_{\Sigma} = \sum_i t_i$  где цикловое время  $t_{\text{ц}} = 1000$

секунд (примерно 17 минут).

Если общее время решения последовательности 933.3 секунд, то одного устройства достаточно, так как сумма  $t = 933.3 < t_{\text{ц}} \cdot N = 1000 \cdot 10$ .

Если бы время решения превышало это, нужно было бы увеличить число устройств.

Таким образом, для обработанной задачи с входными данными 5000 байт время ввода составило 20 секунд. Процессорное время при 100 операциях на байт и среднем времени операции 0.0001 секунды - 50 секунд. Время обмена с внешними устройствами - 20 секунд, вывод результата - 3.33 секунды. Общее время одной задачи - 93.33 секунды. Для последовательности из 10 задач общее

время решения - 933.3 секунды. При заданном цикловом времени 1000 секунд одного вычислительного устройства достаточно для своевременного решения задач.

### Примеры возможных графиков:



Рисунок 5.1 – Затраты времени по отдельным этапам с известным алгоритмом



Рисунок 5.2 – Затраты времени по отдельным этапам с неизвестным алгоритмом

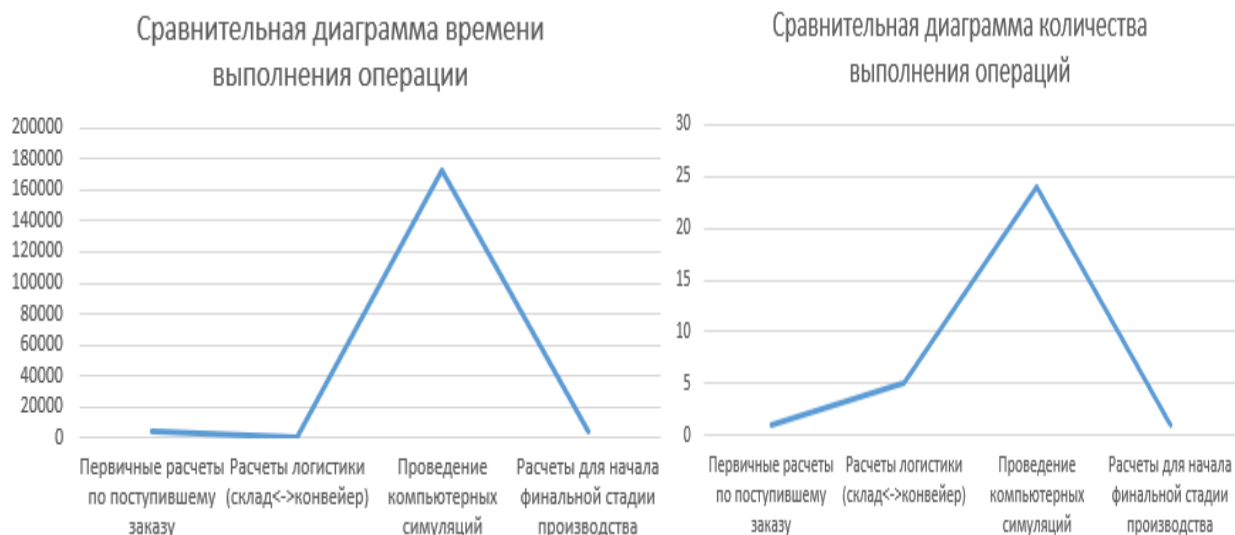


Рисунок 5.3 – Сравнительные диаграммы времени (а) и количества (б) выполняемых операций

Анализ рисунков показывает, что и в случае работы по заранее известному плану и без него, самым времязатратным является этап процессорного времени, что является нормой в нашем случае.

Это подтверждают также и рисунки 5.3, а, б, из которых видно, что как время выполнения одной операции «проведение компьютерных симуляций», так и их количество намного больше относительно других операций, поскольку при этом требуется большое количество просчетов для входных данных.

### Варианты заданий для выполнения лабораторной работы

0 – Система организационного управления предприятием выпуска микропроцессорной техники в условиях интенсивного изменения планов производства.

1 – ERP-система (Enterprise Resource Planning System – система планирования ресурсов предприятия) – корпоративная информационная система (КИС), предназначенная для автоматизации учёта и управления.

2 – Автоматизированная система управления критической инфраструктурой объекта с атаками вирусного типа.

3 – Система поддержки принятия решений по обеспечению безопасности беспилотного транспорта.

4 – Интеллектуальная система распознавания лиц и формирования оптимальных траекторий движения спортсменов в командных играх.

5 – Система формирования динамической библиотеки видеопродукции: выбор сюжетов, компоновка одного фильма из нескольких сюжетов, пополнение библиотеки, классификация.

6 – Система управления взаимоотношениями с клиентами – прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами), в частности для повышения уровня продаж (CRM-система).

7 – Интенсивно пополняемая web-ориентированная база данных интернет-магазина с активным противодействием конкурентов.

8 – Гетерогенная система управления концерном предприятий-смежников по выпуску сложной техники.

9 – Система управления IT-проектами по созданию игровых приложений.

### **Контрольные вопросы**

1. Что такое цикл работ обработки информации?
2. Для чего задачи, решаемые на ЭВМ, разбиваются на типовые группы
3. Какова роль соотношений между видами операций при расчёте процессорного времени?
4. Что такое “время решения последовательности задач”?

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Андрейчиков, А. В. Анализ, синтез, планирование решений в экономике : учеб. для студентов вузов, обучающихся по специальности "Прикладная информатика в экономике" / А. В. Андрейчиков, О. Н. Андрейчикова. – Изд. 2-е изд., доп. и перераб. – Москва : Финансы и статистика, 2004. – 463, [1] с. : ил., табл. : 21 см.; ISBN 5-279-02901-7. – Текст : непосредственный.
2. Андрейчиков А. В. Автоматизированное принятие решений в иерархических системах / А. В. Андрейчиков, О. Н. Андрейчикова, Ф. Ф. Джабер. – Текст : непосредственный // Программные продукты и системы. – 1993. – № 3. – С.23 – 29.
3. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем / А. М. Вендров. – М. : Финансы и статистика, 1998. – 176 с. ил.; 21. – (Прикл. информ. технологии); ISBN 5-279-01979-8. – Текст : непосредственный.
4. Документация по методологии Microsoft Solutions Framework. Режим доступа – <https://msdn.microsoft.com/en-us/library/jj161047.aspx>
5. Дроздов, А. Использование средств описания процессов при внедрении корпоративных информационных систем / А. Дроздов, А. Коптелов // Проблемы теории и практики управления. – 2006. – № 10. – С. 54-70. – EDN KWFOJT. – URL: <https://elibrary.ru/item.asp?id=12896483> (12.11.2025). – Текст : электронный.
6. Зыков, С. В. Основы проектирования корпоративных систем : монография / С. В. Зыков ; Высш. шк. экономики - Нац. исследовательский ун-т. – Москва : Изд. дом Высш. шк. экономики, 2012. – 430, [1] с. : ил. : 21 см.; ISBN 978-5-7598-0862-6. – Текст : непосредственный.
7. Информационные системы учебное пособие для студентов высших учебных заведений, обучающихся по направлению подготовки дипломированных специалистов "Информатика и вычислительная техника" / Ю. Избачков, В. Петров, А. Васильев, И. Телина. – 3-е изд. – Санкт-Петербург [и др.] : Питер, 2011. – 539 с. ил.; 24. – (Учебник для вузов); ISBN 978-5-49807-158-9. – Текст : непосредственный.
8. Марка, Д. А. Методология структурного анализа и проектирования : [Пер. с англ.] / Дэвид А. Марка, Клемент Л. МакГоуэн ; предисл. Д. Т. Росса. - [Москва] : Фирма "Мета Технологии", 1993. - 240 с. : ил.; 29 см.; ISBN 5-7395-0007-9. – Текст : непосредственный.
9. Орлов, С. А. Технологии разработки программного обеспечения : учебник / С. А. Орлов. – СПб. : Питер, 2002. – 464 с. – ISBN 5-94723-145-X. – Текст : непосредственный.
10. Абрамов, Г. В. Проектирование информационных систем : учебное пособие / Г. В. Абрамов, И. Е. Медведкова, Л. А. Коробова ; М-во образования и науки Российской Федерации, ФГБОУ ВПО "Воронежский ин-т инженерных

технологий". – Воронеж : ФГБОУ ВПО "ВГУИТ", 2012. – 171 с. : ил., табл. : 21 см.; ISBN 978-5-89448-953-7. – Текст : непосредственный.

11. Вдовин В. М. Теория систем и системный анализ учебник, для студентов экономических вузов, обучающихся по специальности "Прикладная информатика (в экономике)" / В. М. Вдовин, Л. Е. Суркова, В. А. Валентинов; Издат.-торг. корпорация "Дашков и К°". – Москва : Дашков и К°, 2010, [т.е. 2009]. – 637, [1] с. ил., табл.; 21. – ISBN 978-5-394-00076-8. – Текст : непосредственный.

12. Цуканова, Ольга Анатольевна. Методы и инструменты моделирования бизнес-процессов : учебное пособие / О. А. Цуканова, А. В. Олейниченко, Е. Е. Олейниченко ; Министерство науки и высшего образования Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего образования "Национальный исследовательский Университет ИТМО". – Санкт-Петербург : Страта, 2021. – 106 с. : ил., табл., цв. ил. : 21 см.; ISBN 978-5-907314-86-3. – Текст : непосредственный.

13. Непейвода, Н. Н. Стили и методы программирования : учебное пособие / Н. Н. Непейвода. – 4-е изд. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2025. – 295 с. – URL: <https://www.iprbookshop.ru/146400.html> (дата обращения: 12.11.2025). – Режим доступа: для авторизир. пользователей – ISBN 978-5-4497-0938-7. – Текст : электронный.

14. Флегонтов, А. В. Моделирование информационных систем. Unified Modeling Language / А. В. Флегонтов, И. Ю. Матюшичев. – 3-е изд., доп. – Санкт-Петербург : Лань, 2023. – 140 с. – URL: <https://e.lanbook.com/book/351815> (дата обращения: 12.11.2025). – Режим доступа: для авториз. пользователей. – ISBN 978-5-8114-4274-4. – Текст : электронный.

# **КРИТИЧЕСКИЙ КОМПЬЮТИНГ**

*Методические указания  
к лабораторным работам*

*Составители:* **Доронина** Юлия Валентиновна,  
**Пестрякова** Анна Вячеславовна

В авторской редакции

Изд. № 213/2025. Объем 2,75 п. л. Усл. печ. л. 2,56. Уч.-изд. л. 2,695.  
Издательский центр ФГАОУ ВО «Севастопольский государственный университет»